Piërre van de Laar · Jan Tretmans
Michael Borth *Editors*

Situation Awareness with Systems of Systems

Springer

Situation Awareness with Systems of Systems

Piërre van de Laar • Jan Tretmans • Michael Borth
Editors

Situation Awareness with Systems of Systems

Springer

Editors
Piërre van de Laar
Embedded Systems Institute
Eindhoven, Netherlands

Michael Borth
Embedded Systems Institute
Eindhoven, Netherlands

Jan Tretmans
Embedded Systems Institute
Eindhoven, Netherlands

ISBN 978-1-4614-6229-3 ISBN 978-1-4614-6230-9 (eBook)
DOI 10.1007/978-1-4614-6230-9
Springer New York Heidelberg Dordrecht London

Library of Congress Control Number: 2012956306

Foreword

Nothing less than a disruption in thinking about naval systems was taking place at Thales when we, together with ESI, conceived the POSEIDON project. Thales has a long track record in defense systems – using many proprietary solutions designed to perform reliably under extreme operational conditions of armed conflict anywhere in the world. For the future, we wanted to enter the market of maritime safety and security (MSS). Systems supporting MSS missions impose very different requirements and in fact open up the possibility to utilize open source and state-of-the-art technologies from the civil domain.

Thales had a challenge in using commercial off-the-shelf technologies as they were traditionally not qualified to meet the demanding requirements of our core business of building highly reliable defense systems: How to conceive and develop a different type of system targeted at a new market opportunity? This had to be achieved with our existing pool of highly talented technical professionals with mission critical defense systems in their blood.

As with all high-tech organizations, it all begins with a handful of key people with a vision who are capable of convincing decision makers to allocate budgets to new projects that will result in attractive and smart solutions. This was all set in place with a number of projects running to achieve our MSS ambition. Expectations, however, were very high in the sense that it was assumed that we could reach the level of deliverable products very quickly; after all it was "R&D business as usual." In our pragmatism, we missed the point somewhere in this palette of projects.

We were in need of a more out-of-the-box thinking that would come up with new concepts to pull the population at Thales across the line. This was where ESI, together with its partners, came in and POSEIDON was born. The Industry-as-Laboratory approach used by ESI was key in our decision to proceed. It matched our belief that product goals and research goals can go hand in hand. Given the right environment of quality staff, who understand, respect, and support each other's goals and are backed up by facilities and processes, they catalyze each other and can achieve great results. This is exactly why POSEIDON has been so successful. It gave birth to numerous product concepts that found their way into our naval systems portfolio and also resulted in many publications and dissertations.

Now after 5 years, you will find everything you may want to know about the results of POSEIDON in this book. I would like to add, with reference to a famous Gestalt law, known from psychology, that "the whole is more than the sum of its parts." Above all, POSEIDON has been a highly inspiring journey that substantially contributed to the mindset change now helping Thales to develop advanced naval systems for the future.

Delft, September 2012

Jimmy Troost
Director TRT-Delft
Thales Netherlands

Preface

It is with great pleasure that I welcome you to the final book on the Embedded Systems Institute project POSEIDON. The project was funded under the Dutch BSIK program "Embedded Systems." The project partners were the Embedded Systems Institute (ESI), Thales Netherlands, Noldus Information Technology, Delft University of Technology, Eindhoven University of Technology, University of Amsterdam, Tilburg University, and VU University Amsterdam. The project started in June 2007, ended in May 2012, and encompassed an overall volume of 84 fte.

As for all of ESI's large projects, POSEIDON has followed the by now well-known Industry-as-Laboratory paradigm, in which scientific research is performed in the context of an industrial case. For POSEIDON, the case was defined in the context of the new emerging market of support systems for maritime safety and security. The POSEIDON partners addressed a variety of research topics ranging from integration and testing to systems-of-systems, from visualization to security, from vessel trajectory segmentation to adapter generation, and from situation awareness to trustworthy information interoperability.

The POSEIDON project has been highly successful. Among the results we count the following highlights:

- An architectural framework for information-centric systems of systems and an integrated demonstrator, showing how the combination of many new technologies can be applied to offer improved system support to coast guard operators for a higher level of situation awareness.
- An extendable method to analyze and visualize the kinematic behavior of moving objects. This method offers powerful solutions for the construction of user-defined operational pictures in next-generation maritime systems.
- A highly efficient data reduction method resulting in vessel trajectories using only 2 % of the original amount of data.
- A formal definition of a semantic concept hierarchy of maritime information, enabling automatic reasoning on semantic level with maritime concepts, implemented in a knowledge base.

- A new method for trust management and distributed access control for use in a systems-of-systems environment in the absence of a central security authority.
- Concepts and techniques for systems integration and acceptance at runtime: systems join-and-leave, runtime acceptance testing, and system health diagnosis.
- Adaptor generation techniques for the quick realization of reliable connections between systems.
- A method for runtime anomaly detection by mining of semantic information about ship movements.
- Strong cooperation between universities resulting in a number of shared publications.
- Over 100 scientific and professional publications and PhD. and MSc. theses.

All partners in the project are satisfied with the results achieved in the POSEIDON project. Some of the results and insights obtained in POSEIDON will find their way in the Thales Netherlands product portfolio. Other achievements have found their way to the portfolio of projects that ESI is executing together with industrial and academic partners, including the successor project METIS, where new research topics are tackled that were instigated by POSEIDON.

I would like to thank all project participants for their commitment and contributions: as a team they have turned POSEIDON into a success! The support of Thales Netherlands and the Dutch Ministry of Economic Affairs (now EL&I) through AgentschapNL is gratefully acknowledged. We also thank Springer for their willingness to publish this book. With this book, we expect to share the important results achieved with a larger, worldwide audience, both in industry and academia.

Eindhoven, September 2012

Prof. dr. ir. Boudewijn Haverkort
Scientific Director and Chair
Embedded Systems Institute

Acknowledgements

Situation Awareness with Systems of Systems is a result of the POSEIDON project. The Industry-as-Laboratory project POSEIDON would not have happened without the technological and managerial vision of the Embedded Systems Institute (ESI) and the provision of a "laboratory" inside their company by Thales Netherlands. POSEIDON was partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

We gratefully acknowledge the cooperation with the employees of Thales Netherlands throughout the 5 years of the POSEIDON project. We are also grateful to the Dutch Coastguard and Marin for providing us with domain knowledge, data, and valuable insights. We thank the employees of our academic partners (Eindhoven University of Technology, VU University Amsterdam, University of Amsterdam, Tilburg University, and Delft University of Technology), Thales Netherlands, Noldus Information Technology, and ESI for writing and reviewing a variety of book chapters. All efforts together resulted in this book that shows the current state of the art in situation awareness with systems of systems.

Contents

Part I General

1 Introduction: Situation Awareness, Systems of Systems, and Maritime Safety and Security 3
Jan Tretmans and Piërre van de Laar

2 Improving Situation Awareness in the Maritime Domain 21
Maurice Glandrup

3 On the Architecture of Systems for Situation Awareness 39
Michael Borth

4 The POSEIDON Demonstrator 55
Piërre van de Laar

Part II Situation Awareness

5 Visualization of Vessel Traffic 73
Niels Willems, Roeland Scheepens, Huub van de Wetering, and Jarke J. van Wijk

6 Extending Track Analysis from Animals in the Lab to Moving Objects Anywhere 89
Wil van Dommelen, Piërre van de Laar, and Lucas P.J.J. Noldus

7 Recognizing Vessel Movements from Historical Data 105
Gerben de Vries and Maarten van Someren

8 Density-Based Anomaly Detection in the Maritime Domain 119
Jeroen Janssens, Eric Postma, and Jaap van den Herik

9 Analyzing Vessel Behavior Using Process Mining 133
Fabrizio M. Maggi, Arjan J. Mooij, and Wil M.P. van der Aalst

10 The Simple Event Model 149
Willem Robert van Hage and Davide Ceolin

Part III Systems of Systems

11 Specification and Generation of Adapters for System Integration 173
Arjan J. Mooij and Marc Voorhoeve

12 The POLIPO Security Framework 189
Daniel Trivellato, Sandro Etalle, Erik Luit, and Nicola Zannone

13 Assessing Trust for Determining the Reliability of Information 209
Davide Ceolin, Willem Robert van Hage, Guus Schreiber, and Wan Fokkink

14 Online Fault Localization and Health Monitoring for Software Systems 229
Éric Piel, Alberto Gonzalez-Sanchez, Hans-Gerhard Gross, and Arjan J.C. van Gemund

15 Prioritizing Tests for Fault Localization 247
Alberto Gonzalez-Sanchez, Éric Piel, Rui Abreu, Hans-Gerhard Gross, and Arjan J.C. van Gemund

A POSEIDON Project Partners 259

B POSEIDON Publications 261

Index 269

Part I
General

Chapter 1
Introduction: Situation Awareness, Systems of Systems, and Maritime Safety and Security

Jan Tretmans and Piërre van de Laar

1.1 Introduction

Situation awareness, i.e., being aware of the environmental situation by collecting and interpreting information, is a prerequisite for many organizations to make informed decisions and take appropriate actions. In many domains, such as air traffic control, chemical plant surveillance, combating emergency situations, and controlling maritime safety and security, computer-based support is thereby indispensable for gathering and processing all the relevant data. Such a computer system for supporting situation awareness is often implemented as a *system-of-systems*, i.e., as an evolving collection of distributed, heterogeneous, autonomous, cooperating systems, without a clearly identifiable centralized control.

This book presents and discusses various aspects, challenges, and solutions for developing systems-of-systems for situation awareness, with applications in the domain of maritime safety and security. This chapter introduces the book, provides an overview of the chapters it contains in Sect. 1.6, and introduces the core topics. First, the concept of situation awareness is elaborated in Sect. 1.2, which is followed by a discussion on computer support for situation awareness in Sect. 1.3. Section 1.4 discusses the characteristics of systems-of-systems. Situation awareness in the domain of maritime safety and security is further investigated in Sect. 1.5. Since the results presented in this book were obtained in the Dutch research project POSEIDON, Sect. 1.7 concludes this introductory chapter by putting the results in the context of this project.

Since the area of situation awareness with systems-of-systems is dynamic, lively, and broad, it is impossible to be complete and cover all relevant topics in full detail. This book is also not a blueprint for building systems-of-systems for situation

J. Tretmans (✉) • P. van de Laar
Embedded Systems Institute, Eindhoven, The Netherlands
e-mail: jan.tretmans@esi.nl; pierre.van.de.laar@esi.nl

P. van de Laar et al. (eds.), *Situation Awareness with Systems of Systems*,
DOI 10.1007/978-1-4614-6230-9_1,

awareness. Yet, the book does discuss many challenges when building such systems and proposes solutions in many areas, including the construction of a demonstrator in Chap. 4. The maritime domain, more specifically maintaining safety and security in the Dutch part of the North Sea, is chosen as the primary application area, but many, if not all of the issues discussed in this book are easily transferable to other domains of situation awareness and to other kinds of systems-of-systems.

This book intends to give an accessible overview of the results of the POSEIDON project for anybody interested, for academics as well as for technical professionals working in the area of situation awareness and/or systems-of-systems. It is not the intention to give a full scientific treatment of the topics covered; where necessary references to other publications, such as journal and conference papers, are made. A list of all POSEIDON publications is contained in Appendix B. The different chapters are independent, so that sequential reading is not necessary.

1.2 Situation Awareness

Collecting, aggregating, and interpreting information in order to know what is happening in the environment and to be aware of the situation in the surroundings, i.e., *situation awareness*, is a prerequisite for many animals, humans, and organizations to make informed decisions and take appropriate actions. A rabbit observes its environment, it looks around, listens, and smells to identify a potentially dangerous situation, such as a fox approaching, to be able to react in time.

A car driver observes, interprets, and tries to predict the behavior of other cars in the vicinity. He, or she, adapts his own behavior, and combines his observations with previous experience and knowledge about cars in similar situations, while incorporating additional information from traffic signs, traffic information on the radio, and instructions from his navigation device, in order to prevent accidents, avoid traffic jams, and safely reach his destination.

An organization such as a traffic control center must be constantly aware of the situation on the roads in its designated area, in order to optimize traffic throughput, minimize congestion, maintain safety, enforce traffic rules, respond to emergency situations and accidents, deal with road blocks and reconstruction works, and minimize environmental pollution. The traffic center observes and monitors the traffic using different sources of information, e.g., cameras, detection loops, visual observation, intelligent road sensors, information obtained from satellites, and, if possible, messages sent from cars themselves. In addition, external information sources that are only indirectly linked to the current traffic are important for appropriate traffic control, such as current and predicted weather conditions, historical information about traffic streams and rush hour patterns, holiday periods, planned reconstruction work, special or voluminous transports, events that attract a lot of people and cars such as a football match and a rock concert, and information from neighboring traffic control areas.

There are many organizations for which knowledge about what is happening in the environment is of prime importance as a starting point for making decisions and taking actions. Examples are air traffic control, chemical plant surveillance, monitoring large and complex machines, responding to emergency situations and natural disasters, monitoring and controlling safety and security at sea including tsunami warnings, knowing positions, movements, and threats in military operations, and crowd surveillance such as knowing how people move during a soccer match, a demonstration, or a concert.

Situation awareness involves acquiring information about the environment, about who is doing what and where, and then interpreting this information for a particular goal [1]. Apart from directly observing the environment, additional, indirect sources of information can be used to help with the interpretation and understanding of the environment, e.g., historical information, extra information about actors in the environment, or public information in the news or available on the web.

Vast amounts of information can be produced by different sources. Often these sources will agree, but sometimes they may provide inconsistent or contradicting information. Selecting, aggregating, filtering, combining, and interpreting information, reasoning about the acquired information, searching for correlations, assessing the trust and reliability of information, and trying to predict how the environment will evolve, are all part of creating situation awareness.

In many domains a main goal of situation awareness is to detect abnormal or unusual events that can lead to dangerous, threatening, or undesired situations, e.g., a traffic jam, a tsunami threat, a potentially explosive situation in a chemical plant, a hostile missile approaching with high speed, or squashed people in a crowd. Perceiving and alerting to such anomalous situations in the vast amounts of information is then important, while filtering out normal situations as much as possible.

1.3 Systems Supporting Situation Awareness

In most domains, computer-based support is indispensable for attaining good situation awareness. A support system for situation awareness helps with gathering, processing, and interpreting the vast amounts of relevant data. Typically, such a system presents its output to a human operator, e.g., to an operator surveilling traffic in a traffic control center. With the help of such a system the operator will get a better overview of what is happening, and, consequently, can make better decisions and take more effective actions.

Some situation awareness systems are able to perform actions autonomously. Most of such systems are either simple so that appropriate actions are straightforward, or time-critical so that human intervention would cost too much time. Our focus, however, is on systems that only support decision making by presenting a view of the current situation. Taking appropriate actions is then left to the human surveillance operator.

Support systems for situation awareness face several challenges. Whereas at first sight such systems look like straightforward information processing systems, i.e., gathering input data, processing these data, and presenting the information to the (human) user, a more precise analysis shows that there is more involved. We mention a couple of challenges.

First, the data sources provide data in large quantities. This means that filtering, focusing, compression, and selection of relevant data are needed in such a way that no important information is lost. Reduction of data quantity is necessary to make incoming data more easily processable, to make it presentable to a human user, and to enable storage of data to gradually build a set of historical data.

Availability of historical data allows to recognize patterns, to compare current data with what happened in the past, and to use data as training set for learning purposes. But it adds a second challenge of managing these historical data: keeping the amounts of data under control, recognizing and removing obsolete data, alignment of historical data with changing situations, and keeping the integrity and consistency of historical data.

A third challenge concerns the heterogeneity and independence of data sources, ranging from (intelligent) sensors, radar, and satellite links, to databases and the (semantic) web. This implies that there are differences in format, syntax, semantics, and protocols, which must be aligned. Format and syntax transformations must deal with syntactic interoperability, and protocol converters and adapters are needed to bridge the gaps between various protocols. Semantic differences, such as using the same term for different concepts (consider all the different meanings of the sentence "The girl saw the man with the glasses."), or using different terms for the same concept (such as the use of 'car', 'vehicle', 'voiture', or 'auto' to denote the same concept) require semantic and ontology alignment. Information fusion is needed to make it possible to combine different pieces of aligned information into larger chunks. Semantic reasoning shall be applied to add knowledge and understanding: aggregating the pieces of information into meaningful new information and deducing higher level knowledge, such as patterns, clusters, and classifications of situations.

An additional challenge in this reasoning is that trust, reliability, and also privacy and confidentiality of information have to be taken into account. Since the information comes from different sources, which are probably not equally reliable, they may provide mutually inconsistent or contradicting information. This leads to notions of trust and uncertainty in information that a system for situation awareness must deal with. Moreover, due to different privacy and confidentiality rules, it can be that not everybody has access to the same information.

Finally, all information and deduced knowledge must be presented to the human operator in such a way that it is easily accessible, manageable, and tractable. Sophisticated visualization techniques are necessary to present the information, both as an overview picture of the environmental situation, and in detail, e.g., to indicate anomalies and explain why a situation is considered abnormal.

1.4 Systems of Systems

A computer system for supporting situation awareness in complex domains is not a monolithic, coherent system. Such a system needs to perform many tasks, it gathers information from different sources at distinct locations, and it interacts with various stakeholders and other systems. Many of the components that perform these tasks or that serve as sources are actually complex, independent systems themselves, which are not under the full control of the situation awareness system. Such a system in which the constituent components are autonomous, complex systems themselves, is called a *system-of-systems*.

A system-of-systems (SoS; sometimes called collaborative system, or federation of systems) is a large-scale, non-monolithic, distributed, heterogeneous, complex system, built from multiple interacting sub-systems, which are complex, autonomous, independently operating systems themselves. There is no central control, and there is no single owner or responsible for the entire system-of-systems. Yet, by collaborating in a system-of-systems, functionalities can be provided that its constituent systems alone would never be able to provide [9].

Research and development in the area of systems-of-systems started in the late 1990s. It was triggered by the growing connectivity between systems and the recognition that connections and collaborations between systems would enable many new applications and opportunities, but that they would also generate many new challenges that surpass the feasibility of traditional system engineering. These challenges have various dimensions involving technological, political, and organizational aspects. Different institutions, universities, government agencies, as well as commercial companies work on them, and also within the European Union research programmes they form a key area [2].

Examples of systems-of-systems are found in traffic management where various systems, including in-car devices, collaborate to optimize traffic flow in urban areas; smart cities where systems for traffic management, public transportation, energy management, etc. work together to optimize sustainability; smart buildings, where surveillance, access control, fire emergency, heating, climate control, and lighting interact; cross-company integrated business process management; and many systems in the domain of situation awareness.

The main characteristic of a system-of-systems, as opposed to a classical system consisting of components, is the autonomy and operational independence of the constituent systems, and thus the lack of central control. This has a couple of important consequences which challenge the design, validation, testing, deployment, and maintenance of systems-of-systems.

A first consequence is the evolving nature of systems-of-systems and the necessity to perform activities like testing, acceptance, and reconfiguration at runtime, i.e., online. Since each constituent system runs independently from the others, it can autonomously be started, stopped, removed, replaced, updated, or degraded. This means that the configuration of the system-of-systems changes and evolves dynamically without central control. Other systems and the entire

system-of-systems must be able to cope with such reconfigurations and must adapt to them, e.g., searching for a substitute system that delivers a service that can replace the service of a leaving system. But also the other way around, the system-of-systems must adapt itself and can remove or disconnect an individual system, e.g., if the quality of its service degrades too much. The necessity to perform dynamic, runtime reconfigurations is strengthened by the requirement of typical application areas of systems-of-systems, e.g., systems for situation awareness, that the systems are always up and running.

A derived consequence is that in systems-of-systems that perform runtime reconfigurations, several validation and quality checks must also be performed dynamically. Examples are runtime monitoring of systems and their quality of service, runtime testing to decide whether a new system can join the system-of-systems, and runtime fault diagnosis to pinpoint the malfunctioning system in case a failure occurs. Runtime testing, however, may lead to additional complexity by causing side-effects through undesired interactions between the operational system-of-systems and the tested system, e.g., during a test of the fire alarm system it is not always desirable that also the entire automatic sprinkler system is activated. Runtime verification and validation activities must continuously check and maintain the quality and reliability of the entire system-of-systems, in particular also during the reconfigurations.

A second consequence of autonomy is that the constituent systems in a system-of-systems were designed independently, i.e., they were not specifically designed to work together. Consequently, their combined behavior may lead to emergent behavior, i.e., behavior that is not fully predictable from knowing the behaviors of the constituent systems, thus leading to uncertainty about the overall behavior.

In addition, if systems have to interact with other systems that are not known in advance, interfaces must be flexible enough to adapt to such interactions, or special connectors or adapters have to be made that can bridge both the syntax and semantic differences. It is a challenge to design systems that are flexible enough to operate, adapt, and connect to other systems in the dynamic, evolving context of systems-of-systems.

A third point that follows from the lack of central control together with dynamic reconfigurations and tests, is the difficulty to precisely know the global state of the entire system-of-systems. This involves the configuration, i.e., knowing which constituent systems are available at a given moment, the quality of the constituent systems, i.e., knowing which systems are healthy and operating correctly, as well as the quality, reliability, and validity of the information being processed by the constituent systems: an unhealthy system may produce unreliable information.

The entire system-of-systems, as well as the constituent systems, must be able to cope with the uncertainty caused by the lack of knowledge about the global state. The consequence is that a system-of-systems, in addition to dealing with its primary information, must also communicate and reason with meta-information about its own configuration, its own health and the health of its constituent systems, and the quality and reliability of the primary information. This means that a system-of-systems must reflect on its own operation. One might say that this constitutes a

'meta-situation awareness system' for system awareness: like a situation awareness system for road traffic monitors and controls traffic streams and warns for anomalies on the roads, the meta-situation awareness system monitors and controls information streams and anomalies in the system-of-systems.

A final issue in systems-of-systems concerns security, privacy, and confidentiality. The autonomy of the constituent systems implies that they will all have their own policies with respect to sharing and protecting sensitive information. There is no central authority arranging all security issues. This requires special policies and methods to communicate allowances to specify who is allowed to have which information at what occasions. Special care must be taken that also during reconfigurations and runtime testing no confidential information leaks away.

Compared with traditional systems and system development, systems-of-systems have to deal with blurring boundaries, both in space and in time. In space, a starting point for traditional system engineering is the distinction between a system and its environment. The above discussion shows that for a system-of-systems the boundary between what belongs to the system-of-systems and what belongs to its environment, is not sharp. In time, the boundaries between the traditional development phases, such as design, building, validation, testing, deployment, operation, maintenance, and decommissioning, diminish. After some time of continuous operation, while systems are leaving and joining, a completely new system-of-systems may have emerged, consisting of completely new constituent systems, but still performing the same tasks.

1.5 Situation Awareness for Maritime Safety and Security

Our seas have many functions and are used in many ways, for several purposes, and involving various stakeholders. A challenging and important application area for situation awareness is *Maritime Safety and Security* (MSS). In maritime safety and security, the goal is to keep the seas safe and secure, in particular, the coastal regions that are under control and responsibility of a specific country (Fig. 1.1).

Keeping the seas safe and secure involves many aspects. First of all, the seas serve as one of the most important transportation infrastructures. Many ships use the seas and they shall adhere to sea-traffic rules that must be monitored. Shipping lanes must be marked and maintained, traffic in and out of harbors must be controlled, collisions shall be avoided, and in case of emergency assistance shall be provided. Second, the seas are important for food production. Fishing must be monitored, illegal fishing must be prevented and detected, and fish farms shall be regulated and guarded. Third, the seas often constitute the border between countries, implying that border control, customs, smuggling, illegal immigration, and general defense are issues at sea. The coastal area being a part of the country implies that law enforcement is a fourth issue. This includes, for example, combating piracy, terrorism, and drug trafficking, and protection of assets such as pipelines, historical ship wrecks, and war graves. Fifth, the seas constitute an important ecological system, susceptible

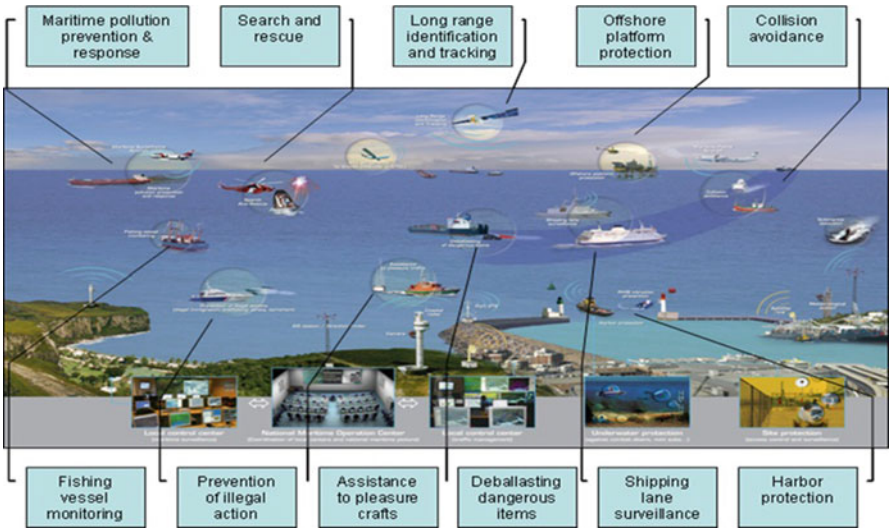


Fig. 1.1 Examples of maritime safety and security (©Thales Nederland B.V.)

to contamination, so pollution monitoring, prevention, and control are important tasks at sea. A sixth aspect of the usage of the seas is the provision of energy through off-shore oil and gas platforms, wind parks, pipelines, and, perhaps in the near future, algae farms. Also these shall be regulated, protected, monitored, and accidents inhibited. A final aspect are safety threats through sea mines, ship wrecks, and lost cargo, and emergency situations ranging from search and rescue to neutralization of oil spills that must be dealt with.

The various activities taking place at sea entail risks, threats, and potential dangers that may jeopardize safety and security. Maintaining maritime safety and security is typically a task that is coordinated by the coast guard, working together with harbor authorities, (water) police, customs, navy, rescue-teams, etc. The first step is attaining situation awareness, i.e., knowing what is happening at sea. Considering the large numbers of ships, the many activities at sea, and the large areas that have to be covered, computer-based support herewith is indispensable. A system for maritime situation awareness will support and guide the surveillance officers and operators in a coast guard control center with monitoring and controlling what is happening at sea, so that better decisions can be made and appropriate actions can be taken.

An important task for maritime safety and security is anomaly detection. A situation awareness system must support the surveillance operators in identifying and focusing on suspicious or abnormal situations, while filtering out normal situations as much as possible. Abnormal situations may indicate undesired or unlawful activities, risks, or threats. Examples include ships violating traffic rules or sailing outside shipping lanes, ships being too close to oil platforms, wind parks, or the coast, and ships being (too) close to each other. The latter may indicate a

near-collision, it may point to handing-over of illegal goods at sea such as drugs, but if one of the vessels is a tugboat or a pilot boat then it probably is completely normal behavior. In general, any vessel that is somewhere where it does not belong, or that makes strange maneuvers, is an anomaly. Of course, this depends on the type of ship, e.g., a fishing ship sailing on fishing grounds is not abnormal, but a passenger ship is. Other examples of abnormal behavior are a ship that drifts too much or seems to be out of control; a vessel providing inconsistent information such as claiming to be an oil tanker while making a 180°-turn within a minute; a small ship approaching the coast with high speed somewhere where there is no harbor which might be an indication that smuggled goods are dropped on the shore; and two ships decreasing speed at the same place outside of traffic lanes just a short period after each other, which might indicate that one ship dropped a packet that is picked up by the second.

Challenges for anomaly detection in the maritime domain are long time frames and external influences. For some abnormal behaviors observations must cover a long time frame, e.g., a potential collision or strange behavior of an oil tanker stretches over several hours, and also the "drop a packet–pick a packet" scenario described above may require observations over several hours before it is clear that an anomaly occurred. External influences, like weather conditions, play an important role when determining whether some situation is abnormal or not, e.g., strange maneuvers of a ship outside the shipping lane may be completely normal during stormy weather conditions.

1.5.1 Vessel Tracking

A core functionality for anomaly detection in the maritime domain is the monitoring and tracking of vessels: where are vessels and what are they doing. Nowadays a main source of information for vessel tracking is AIS – the Automatic Identification System. All vessels, except the smallest ones, are required to be equipped with an AIS transceiver, which broadcasts messages with status data about the vessel and its movements according to a protocol standardized by the International Telecommunications Union (ITU) [13]. Depending on what the ship is doing, it sends AIS messages every 2–10 seconds when underway, and every 3 minutes when at anchor.

There are different kinds of AIS messages. One type contains kinematic information about the ship such as its current position, speed, heading, turn rate, and navigational status ('under way using engine', 'at anchor', 'moored', etc.). Another type of message contains information about the ship itself and its journey such as its name, internationally recognized identifiers (MMSI number – Maritime Mobile Service Identity number[1] and IMO number – International Maritime Organization

[1]The MMSI numbers used in the examples in this book are fictitious and do not relate to existing ships.

number), call sign, dimensions, draught, the type of ship (oil tanker, passenger ship, cargo, fishing, dredger, pilot boat, etc.), cargo, destination, and expected time of arrival.

AIS messages can be received by neighboring vessels to prevent collisions, and by coast guards and similar organizations for vessel tracking and monitoring. AIS is the main source of information for a maritime situation awareness system. Many web sites provide AIS information.[2]

Other sources of information in the maritime domain are radar, satellite, and visual observations, data from harbors and customs, and the web where many web sites, both paid and free, are found that provide useful information for the maritime domain. Examples are weather and news sites, geographic information,[3] Lloyds,[4] and The Paris Memorandum of Understanding on Port State Control (Paris MOU[5]), which contains a lot of information about ships.

1.6 Overview of the Chapters

Many of the challenges described in the previous sections present themselves in maritime situation awareness systems. This book, in its subsequent chapters, elaborates these challenges and discusses solutions. This section introduces these chapters. Although examples in the chapters are taken from the maritime domain, most of the presented topics are more widely applicable to any system-of-systems or domain of situation awareness.

The chapters are divided into three parts: the chapters in Part II discuss situation awareness, whereas Part III focuses on the systems-of-systems aspects. Before starting with these two parts, Part I investigates some general topics. Chapter 2: *Improving Situation Awareness in the Maritime Domain*, further investigates the application domain of maritime situation awareness systems. The role and activities of an organization for maritime safety and security, the kind of support, alerts, and visualizations that may help the operators, the complexity of their decisions, and the various stakeholders with their conflicting interests are all discussed.

Building a system to support attaining situation awareness, whether in the maritime domain or elsewhere, requires the consideration of various, partly conflicting, functional and non-functional concerns. Chapter 3: *On the Architecture of Systems for Situation Awareness*, discusses such issues by reflecting on the architecture of systems-of-systems for situation awareness. The chapter discusses

[2]Examples of web sites providing AIS information, are www.vesseltracker.com, www.shipais.com, www.vesselfinder.com, www.marinetraffic.com, www.aishub.net, and www.shipspotting.com.

[3]www.geonames.org

[4]www.lloydslistintelligence.com

[5]www.parismou.org

the architecture starting from general principles, considering both a functional view on the information processing that results from domain analysis, and a system architect's view on properties required to deliver that functionality.

Many of the ideas presented in this book have been implemented, and these implementations have been integrated into a demonstrator that presents an elaborate scenario in the maritime safety and security domain. Chapter 4: *The* POSEIDON *Demonstrator*, describes the demonstrator, including screen shots of its operation, reflections on the building process, and the value of such a prototype for stakeholder communication and validation of the research.

1.6.1 Overview of Part II: Situation Awareness

A system supporting situation awareness must present its output in a useful and manageable way to a human user. Using textual output is not an option given the large amounts of vessel information. Powerful visualization techniques are necessary to enable the operator to quickly understand and interpret the current situation. Chapter 5: *Visualization of Vessel Traffic*, presents advanced visualization methods based on density maps. They enable the operator to attain an overview of movement patterns over a period of time, as well as to zoom in on particular situations. Current behavior of vessels can be visually combined with historical vessel movements, presented in a density map, to detect abnormal behavior, i.e., outliers. Filtering on vessel attributes allows to focus on particular ships or situations, e.g., on all passenger ships sailing on some place where there are normally (in the historical data) no passenger ships.

As discussed in Sect. 1.5.1, an important function in situation awareness in the maritime domain is the tracking of vessels and the analysis of their movements. There are many other domains where tracking of objects and the analysis of behavior are important tasks, e.g., traffic monitoring and control, transportation of system parts in a warehouse, crowd monitoring for public safety and security, migrating animals such as whales, reindeer, and birds, and animals moving in a confined area such as a cage or aquarium. Chapter 6: *Extending Track Analysis from Animals in the Lab to Moving Objects Anywhere*, compares vessel tracking with animal tracking. A specialized tool for video tracking of animal behavior in a cage in a laboratory setting is analyzed and adapted for use in the maritime situation awareness domain. The goal of Chap. 6 is to increase insight in the specificities of both domains of object tracking, and to gather requirements for a universal tracking tool which is applicable to multiple domains.

The main source of data for a maritime safety and security system is AIS data (Automatic Identification System); see Sect. 1.5.1. There are many ships, each of them sending AIS messages every few seconds, even if the ship follows a straight, predictable course. Chapter 7: *Recognizing Vessel Movements from Historical Data*, presents in three steps how AIS data is prepared and used for the recognition of vessel movements and behavior analysis. First, AIS data is compressed into track

segments using a technique called piecewise linear segmentation. This enables higher level reasoning about ship trajectories, and reduces the quantity of AIS data with more than 95 % without sacrificing the quality of subsequent behavioral analyses. Second, the level of similarity between different trajectories is quantified using a distance function. Knowing which trajectories are similar to each other enables clustering of similar trajectories and the recognition of movement patterns. Third, the movement patterns are combined with other knowledge about ships and their context, such as the ship type or the geographical location of its position, because such knowledge may influence what is normal: a movement pattern that is normal for a ship of type 'fishing boat' may be abnormal for an oil tanker, and a speed which is normal for open waters can be too fast when we know that the ship's position corresponds to the entrance of a harbor.

The idea of detecting abnormal vessel behavior by calculating similarities and differences between behaviors using distance functions is further elaborated in Chap. 8: *Density-Based Anomaly Detection in the Maritime Domain*. Vessel behavior is then classified as an outlier if it has a large distance to other behaviors. Chapter 8 introduces a method coined Stochastic Outlier Selection, that automatically identifies outliers.

Chapters 7 and 8 use statistical comparison of current behavior with historical behaviors using distance functions to define what normal behavior is. This leads to an implicit (and circular) definition: normal is that what everyone does. An alternative approach is to explicitly define normal behavior via a set of rules. A violation of the rules is then an anomaly. A typical example is fixing a set of traffic rules that all ships, or cars, must satisfy.

Chapter 9: *Analyzing Vessel Behavior using Process Mining*, uses rule-based anomaly detection. A graph-based language is introduced, founded on linear temporal logic, in which rules specifying ship behavior can be expressed, for example "whenever a ship is moored, then eventually in the future it will be under way using engine". Satisfaction of such rules is checked at runtime. Behavior rules can be explicitly expressed, but they can also be learned from historical data, thus providing a combination of rule-based and history-based anomaly detection. Chapter 9 also presents a template-based method for learning behavior rules.

Situation awareness involves knowing about the events happening in the environment. Consequently, the concept of an 'event' is important, and a system supporting situation awareness must be able to handle events. Events are observed, modeled, and defined, they must be stored, manipulated, and related to each other, and they must allow various kinds of (formal) reasoning. Therefore, Chap. 10 *The Simple Event Model*, introduces an ontology-based event model, that is used to model all kinds of events and their related concepts like actors, places, times, and their types. Events can be modeled at various levels of abstraction: a ship moving at a particular place and time, corresponding to one AIS message, is an event, but also a ship trajectory, the hijacking of a tanker in the Strait of Malacca, and an armada sailing from Wellington to Amsterdam can be modeled as an event. In the context of situation awareness, where information comes from different sources, events must be flexible, they must deal with partial, duplicate, contradicting, or uncertain

information, and they must enable enrichment with new information and additional aspects, that are obtained via observations, from the web, or through semantic reasoning.

1.6.2 Overview of Part III: Systems-of-Systems

One of the challenges in the development of systems-of-systems is the fast, runtime integration of autonomous components or systems, which were originally not designed to interact with each other. In such cases a dedicated adapter may be developed that bridges the differences and incompatibilities between the systems. Chapter 11: *Specification and Generation of Adapters for System Integration*, discusses two methods to generate such an adapter semi-automatically from a model of the interface behavior of the systems. The first method uses techniques from controller synthesis; the second one builds on incremental view maintenance in databases. The methods provide a generic and systematic approach for the construction of adapters. This is illustrated with an example in the maritime domain: the connection of a system providing AIS messages to Google Earth, in order to display these messages.

Chapter 12: *The POLIPO Security Framework*, discusses security issues in systems-of-systems. Since systems-of-systems are dynamic coalitions of autonomous systems, a central security policy cannot be implemented. Each of the constituent systems will have its own policy. To cope with this situation, Chap. 12 introduces the POLIPO security framework that protects the information exchanged among the systems in a system-of-systems, while preserving autonomy and interoperability of the systems. It uses context-aware access control and trust management to protect information from unauthorized access, while ontology-based services maintain autonomy and interoperability.

Apart from security, i.e., the question who is allowed to know what, the distributed, autonomous, and heterogeneous nature of systems-of-systems also raises the question of trust, i.e., which information can be counted on. In particular, if different sources of information provide inconsistent or contradicting opinions about the same ship or about the same event, it is important to know which information can be trusted. Chapter 13: *Assessing Trust for Determining the Reliability of Information*, discusses how trust can be assessed and quantified, and how combinations of different opinions increase the level of trust if they agree, and decrease it if they contradict each other.

Systems-of-systems change and evolve dynamically in ways not designed and anticipated in advance. Consequently, during and after each change the quality of the newly integrated system has to be checked again without disturbing or interfering too much with the normal operations of the system-of-systems. This requires runtime monitoring and testing techniques. Moreover, if a failure occurs, runtime diagnosis techniques must be able to localize and isolate the faulty system. Chapter 14: *Online Fault Localization and Health Monitoring for Software Systems*,

discusses the detection of failures and the localization of the faults that led to the failures by adapting existing design-time techniques to the dynamic context. In particular, the technique of spectrum-based fault localization is combined with health monitoring and extended to runtime fault localization.

Since runtime fault localization shall as little as possible disturb the normal operations of the system-of-systems, it shall be effective and fast. Chapter 15: *Prioritizing Tests for Fault Localization*, shows that current test selection and prioritization techniques mainly optimize the fast detection of failures, but not their fast localization. Therefore, Chap. 15 presents techniques for selecting and prioritizing test cases such that fault localization is optimized, i.e., the time to localize the fault is minimized.

1.7 POSEIDON

POSEIDON[6] was a collaborative, industrial-academic Dutch research project, managed by the Embedded Systems Institute (ESI). The goal of the project was to develop new concepts, methodologies, and prototype components for situation awareness systems-of-systems, and to apply them in the domain of maritime safety and security. In POSEIDON, researchers and engineers from the companies Thales Netherlands and Noldus Information Technology, and from ESI, worked closely together with researchers from five universities: Eindhoven University of Technology, Delft University of Technology, VU University Amsterdam, University of Amsterdam, and Tilburg University (Figs. 1.2 and 1.3).

Fig. 1.2 The POSEIDON team at the final POSEIDON symposium

[6] www.esi.nl/poseidon

Fig. 1.3 Partners in POSEIDON

The academic partners contributed their state-of-the-art knowledge and expertise in their specialized research areas. ESI, in addition to project management, contributed its expertise in multi-disciplinary model-based systems architecting and engineering. The main industrial partner Thales Netherlands, also referred to as the 'carrying industrial partner' in project terminology, provided the application domain of maritime safety and security, and contributed its knowledge about the maritime domain and about maritime and naval systems. Noldus Information Technology provided its expertise and experience in tooling for track analysis.

The academic-industrial cooperation in POSEIDON took place in a setting referred to as Industry-as-Laboratory [4, 10, 12, 17]. This means that the actual industrial setting is used as a laboratory, akin to a physical or chemical laboratory, where new theories, ideas, and hypotheses, mostly coming from the academic partners in the project, are tested, evaluated, and further developed. This setting facilitates the transfer of knowledge from academia to industry, and it provides direct feedback about the applicability and usefulness of newly developed academic theories, which may again lead to new academic research questions. For POSEIDON,

the laboratory was provided by Thales Netherlands,[7] which is the Dutch branch of the international Thales Group. Thales Netherlands specializes in designing and producing professional electronics for defense and security applications, such as radar and communication systems, with emphasis on the maritime domain. Thus, the Industry-as-Laboratory setting enabled the project to employ the latest industrial insights in the maritime domain.

POSEIDON has achieved various results, both academic and industrial. A large number of scientific publications appeared in journals and in proceedings of conferences and workshops, several PhD. theses were defended, articles were published in popular magazines and professional journals [5, 7, 8, 11, 14–16, 18, 21], a spin-off company, NexusZ.com, was founded, and a couple of (international) workshops were organized [3, 6, 19, 20]; see Appendix B for a complete list of publications.

The scientific and technical innovative results include the following:

- Multi-objective visualization and methods for the construction of user-defined operational pictures of kinematic behavior of moving objects (Chap. 5);
- Compression of behavioral data through segmentation enabling efficient similarity calculation, comparison, and clustering of ship trajectories, used as the basis for anomaly detection (Chap. 7);
- Stochastic outlier selection as a method for the detection of outliers that outperforms existing methods on reference data sets (Chap. 8);
- The use of process mining techniques for rule-based, runtime anomaly detection (Chap. 9);
- The ontology-based, flexible Simple Event Model enabling manipulation, semantic reasoning, storage in a knowledge base, and connection to foreign ontologies of all kinds of events, at different levels of abstraction, and with potentially partial, inconsistent, duplicate, or uncertain information (Chap. 10);
- Two complementary methods for semi-automatic generation of adapters that can bridge the communication differences between heterogeneous systems (Chap. 11);
- A method for security management and distributed access control for use in systems-of-systems without central security authority (Chap. 12);
- The application of trust calculations to ship data in case of different opinions about ships (Chap. 13);
- Runtime monitoring and diagnosis techniques, which enable the detection and localization of faults in an operational system, and which prioritize additional tests with respect to diagnostic effort (Chaps. 14 and 15).

In addition, the architectural challenges and principles for designing information-centric systems-of-systems, such as situation awareness systems, were investigated (Chap. 3).

[7] www.thalesgroup.com/nlhome

A demonstrator, an integrated concept validation platform with many of the investigated aspects implemented, was built (Chap. 4).The demonstrator, showing several scenarios in maritime safety and security, is a basis for further experimental research, and has been shown to various stakeholders in Thales Netherlands, as well as to other interested parties. It serves both dissemination of results and triggering of reflection and feedback.

A comparison of different kinds of object tracking tools, in particular for animals and vessels, led to important requirements for tracking tools in general (Chap. 6). These requirements are, in turn, input for the European research project *Pronto*,[8] where one of the goals is to build one tool for tracking objects in different domains.

Several of the techniques developed in POSEIDON are now considered by Thales Netherlands for future, intelligent, maritime situation awareness systems. In general, POSEIDON gave lots of inspiration to Thales Netherlands to extend its competences, and to the academic and research partners to direct their research.

Despite POSEIDON being successful, a lot of interesting challenges remain. The successor project METIS,[9] again with Thales Netherlands as 'carrying industrial partner' and led by ESI, continues the research on systems-of-systems for situation awareness in the maritime domain. METIS focuses on techniques for semantic alignment of heterogeneous information sources, analysis of the quality of information and reasoning with uncertainty, and visualization of large amounts of complex and uncertain information, while building on the successes of POSEIDON.

Acknowledgements This research has been carried out as a part of the POSEIDON project at Thales under the responsibilities of the Embedded Systems Institute (ESI). This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

Special thanks go to Thales Netherlands for organizing the Industry-as-Laboratory project POSEIDON in their company. We gratefully acknowledge the cooperation with many people at Thales Netherlands during the POSEIDON project.

References

1. Endsley MR (1995) Toward a theory of situation awareness in dynamic systems. Hum Factors 37(1):32–64
2. European Union DG INFSO G3 (2009) Report of a workshop on systems of systems – contribution to ICT work programme 2011+, 21 Sept 2009. ftp://ftp.cordis.europa.eu/pub/fp7/ict/docs/esd/workshop-report-v1-0_en.pdf
3. Gross H-G, Lormans M, Tretmans J (eds) (2009) Software integration and evolution @ Runtime – SINTER '09; an 2009 ESEC/FSE workshop. ACM. http://doi.acm.org/10.1145/1596495
4. Hamberg R, Verriet J (eds) (2012) Automation in warehouse development. Springer, London
5. Intermediair (2009) Wetenschapsbeeld. VNU Media: Amsterdam, The Netherlands 40:34. http://www.intermediair.nl

[8] www.ict-pronto.org

[9] www.esi.nl/metis

6. Janssens J, Postma E, Hellemons J (eds) (2011) International workshop on maritime anomaly detection (MAD 2011), Tilburg Center for Cognition and Communication, Tilburg University, The Netherlands. http://mad.uvt.nl/mad/mad2011-proceedings.pdf
7. Keulen, J-P (2010) Vaart in kaart. Kijk, 2013 Sanoma Media Netherlands B.V.: Hoofddorp, The Netherlands 24(2):64–65 http://www.kijkmagazine.nl/
8. Magilsen, I (2011) Poseidon Ontsluiert Geheimen op Zee. Matrix 18(1):46–47. http://w3.tue.nl/fileadmin/csc/matrix/Matrix_2011-1.pdf
9. Maier, MW (1998) Architecting principles for systems-of-systems. Syst Eng 1(4):267–284
10. Mathijssen R (ed) (2009) Trader: reliability of high-volume consumer products. Embedded Systems Institute, Eindhoven
11. NewScientist (2009) Visions of data, Sept 2009. http://www.newscientist.com/gallery/dn17746-visions-of-data
12. Potts C (1993) Software-engineering research revisited. IEEE Softw 10(5):19–28
13. Radiocommunication Sector of International Telecommunication Union, Geneva (2010) Technical characteristics for an automatic identification system using time-division multiple access in the VHF maritime mobile band, Apr 2010. http://www.itu.int/rec/R-REC-M.1371-4-201004-I/en
14. Thales (2009) Poseidon Verhoogt Veiligheid op Zee. In: Login to Thales-Magazine of Thales Nederland B.V., July/Aug 2009, pp 8–9
15. Theunissen X (2009) Poseidon Signaleert Afwijkend Gedrag Schepen. De Technologiekrant 19:11
16. Theunissen X (2009) Zeeverkeersinformatie. De Ingenieur 14/15:HTS 6–7
17. van de Laar P (2010) Observations from the industry-as-laboratory research project Darwin. In: 8th annual conference on systems engineering research, Hoboken, 17–19 Mar 2010, pp 658–667
18. van de Ven M (2010) Veilig en Alert Heersen over de Zee. Cursor 52(32):12. http://web.tue.nl/cursor/internet/jaargang52/_pdf/cursor32.pdf
19. van Erp M, van Hage WR, Hollink L, Jameson A, Troncy R (eds) (2011) International workshop on detection, representation, and exploitation of events in the semantic web (DeRiVE 2011), Bonn, Germany. http://ceur-ws.org/Vol-779
20. van Someren M et al. (ed) (2010) Workshop on modelling moving objects in the Netherlands. University of Amsterdam, The Netherlands. http://staff.science.uva.nl/~maarten/MMO-webpage.htm
21. Willems N (2011) Shipping routes north sea. Geo-info 8(6):9. http://www.geo-info.nl/download/?id=17685718

Chapter 2
Improving Situation Awareness in the Maritime Domain

Maurice Glandrup

2.1 Introduction

Organizations in the maritime safety and security domain have to cope with complex situations at sea, harbors, and rivers to maintain acceptable levels for safety and security. For safety, traffic rules apply at sea. Yet in areas with heavy traffic, it is difficult to tell which vessels adhere to traffic rules and which ones do not. For security, suspicious activities such as drugs smuggling need to be detected. In the past years it has become clear that there now are more means to detect vessels, because more vessels have equipment to broadcast their current position or are detected by sensor systems that are placed along the coast. At the same time, the rising problem is that it has become difficult to handle complex situations because of the amount of vessel movements in an area. In addition to the more complex view on vessel movement, more information on vessels is available in databases and on the Internet. To identify a vessel as suspect, information about the crew, cargo and owner can be used as a discriminator. Today's maritime safety and security systems do not provide the mechanisms to handle the amount and diversity of information that is available. It is the task of operators to combine all information and elicit suspicious vessels. The risk is that operators have an information overload which causes them to miss important and relevant events, or to notice them too late. In this chapter we discuss several scenarios that indicate difficulties that maritime safety and security organizations face while building situation awareness.

M. Glandrup (✉)
Thales Netherlands, Hengelo Ov., The Netherlands

Current affiliation: NexusZ.com, Hengelo Ov., The Netherlands
e-mail: maurice.glandrup@nexusz.com

P. van de Laar et al. (eds.), *Situation Awareness with Systems of Systems*,
DOI 10.1007/978-1-4614-6230-9_2,

2.1.1 Situation Awareness

Maritime safety and security organizations build situation awareness of the area they control and monitor. Situation awareness, see also Chap. 1, Sect. 1.2, is defined in [4] as:

> The perception of environmental elements with respect to time and/or space, the comprehension of their meaning, and the projection of their status after some variable has changed, such as time.

In this definition examples of environmental elements in the maritime domain are vessels and oil-rigs. Also, areas with stormy whether or with an oil spill are environmental elements.

Situation awareness involves building the awareness in an area and associating information with objects in that area. If an incident happens, all information is available and decisions can be taken to cope with it. Situation awareness applies to several domains, see Sect. 1.2, and the characteristics that these domains have in common are:

- The high diversity of information,
- The large amount of information, and
- Real-time streaming (high flow rate) of information.

The complexity of the information stream is further increased since there are usually more information sources to be considered. There is also criticality associated to the domain. A poor decision because of bad situation awareness may result in a catastrophic impact.

Building and having up-to-date situation awareness where there is a complex information stream is almost impossible to do for decision makers without the help of a system. The purpose of such systems is foremost to structure the information such that operators do not suffer from an information overload.

2.1.2 State-of-the-Art in Maritime Safety and Security Systems for Situation Awareness

Maritime safety and security organizations become increasingly aware of the heaviness of the traffic in an area. In this book, the traffic along the Dutch coast is used as an example case. The North Sea is one of the areas with the heaviest traffic in the world. This increases the complexity of information streams to handle. The complexity in handling traffic is further increased by the following:

- Smaller vessels have to report their position nowadays which results in an increase in the amount of vessels and in information on the position of these vessels;
- More accurate information on the type of vessel and its cargo is available;

- A trend that can be observed in the past years is that vessels become larger, which implies that if such a large vessel is involved in an incident, the (catastrophic) impact will also be larger.

The monitor and control systems that these organizations currently use cannot cope with the amount of information that is available. At this moment, the majority of information sources must be interpreted by operators, or is simply ignored.

2.1.3 Challenges in the Maritime Safety and Security Domain

Maritime safety and security organizations face at least the following two challenges in building situation awareness.

- The first challenge is gathering information on vessels in general, and more specifically on vessels that act strange, or can potentially harm others because of course and speed. Sensor data must be combined with information from closed or public information sources to elicit suspicious vessels. The amount of extra information from these sources can easily result in an overload. It is often quite cumbersome to relate information from information sources with sensor data. Today's maritime safety and security systems do not provide sophisticated interaction mechanisms to deal with the diversity of information. Here, examples of closed information are cargo manifests and the organization's own information of a vessel. Public information sources include web pages on the owner of a vessel and social pages of crews.
- The second challenge is identifying critical situations. It is not trivial to see what is critical in the normal. The increased complexity by the number of moving objects and the amount of information that can be associated with an object, makes it difficult to notice and act on incidents. It also makes it more difficult to keep performing routine work while incidents happen. There exists a huge variety in how incidents happen. Incidents can be unintended or intended,[1] may involve various (types of) vessels, can happen in any weather condition, etc. The maritime safety and security systems that are available at this moment are quite limited in alerting operators on critical situations.

Building accurate situation awareness is becoming more and more the result of multiple organizations that work together. There is collaboration with affiliated organizations (national and international) to trace vessels that are potentially dangerous. This also puts additional requirements on information exchange between organizations. Not all information can be shared because of confidentiality constraints. It may even be that information is not shared unless there is an incident where it is essential to have additional information because of increased risks.

[1]Unintended incidents are safety incidents. Examples of intended incidents are smuggling and terrorist acts.

These challenges and the limited capabilities of maritime safety and security systems give maritime safety and security organizations that work on building situation awareness a difficult task. It forces them to intensively study information they receive and watch vessel movements. This results in strain and possible fatigueness. The resulting loss of attention may lead to missing of events. These challenges were discussed with experts in the field, and they are further refined throughout the chapters of this book.

Maritime safety and security organizations are often government organizations that have to deal with equal or shrinking budgets while their tasks become more complicated. To cope with this, organizations need to streamline their activities and become more efficient. The goal of this chapter is to point out what type of characteristics a maritime safety and security system should have to help streamlining activities and make operators more efficient.

In the next section we explain the difficulties that organizations face to build situation awareness. In Sect. 2.3 we make a case that it is essential to cross-link information from several information sources to build good situation awareness. Section 2.4 discusses several scenarios that can be seen as illustrative for the type of problems organizations face. Section 2.5 discusses how maritime safety and security organizations' tasks can be alleviated and where a system can be of assistance.

2.2 Building Situation Awareness

Maritime Safety and Security (MSS) organizations have to make a considerable effort to build situation awareness of an area by using the current generation of monitoring and control systems. Building situation awareness of the Rotterdam area, let alone the Dutch coast is quite complex. In 2007 on a daily basis there were about 1,500 vessels that transmit their position by using the Automatic Identification System (AIS; see Sect. 1.5.1). AIS transmits a number of times per minute data of a vessel that contains the position of the vessel, its destination, etc. Of these vessels the majority crosses or is in the neighborhood of the Rotterdam area. On a yearly basis there are more than 500,000 vessel movements that are reported by AIS. Larger vessels are obliged to transmit their position by using AIS transponders. However, small vessels at sea often do not carry AIS transponders, so the amount of traffic is in reality larger. From the budget that is assigned to the Dutch coast guard and from experts we know that the number of vessel movements is relatively stable, and that the vessels themselves get larger. The number of vessels that report their position by using the AIS is also growing. By law, in coming years smaller vessels also have to carry AIS transponders.

Besides moving objects, there are also a number of static objects in the North sea. There are over 300 oil and gas platforms, and a dozen windmill parks.

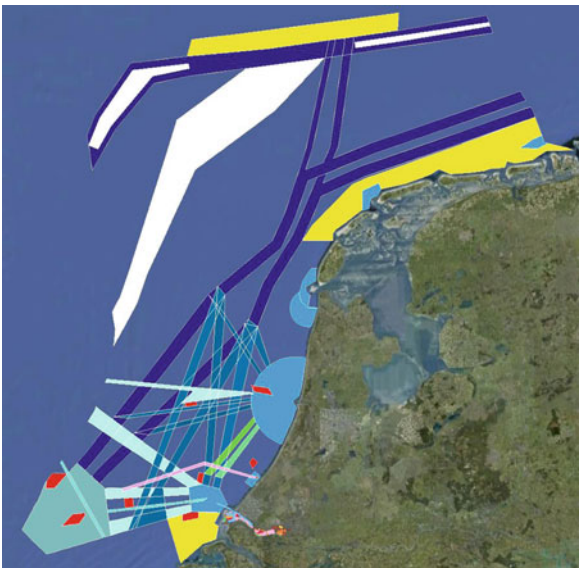


Fig. 2.1 Sea lanes in front of the Dutch coast, shown as a layer in Google Earth (©2011 Google)

2.2.1 Traffic Separation Scheme Along the Dutch Coast

To control the traffic along the coast, a number of laws are applicable. The goal of these laws is to keep the amount of incidents and their impact as low as possible. For example, traffic with dangerous cargo, such as chemicals and oil, must stay clear from nature preservation areas such as the Waddenzee. The Dutch government organization Rijkswaterstaat[2] defined a so-called Traffic Separation Scheme. The Traffic Separation Scheme aims at regulating the traffic at sea in so-called sea lanes. Figure 2.1 shows the Traffic Separation Scheme before the Dutch coast.

In Fig. 2.1, the Dutch coast is clearly recognizable. The colored areas in the figure visualize the Traffic Separation Scheme. The red areas denote anchor areas. These are used by vessels to hold a position before they can go to the harbor to release their cargo. The yellow areas mark restricted areas. Large cargo ships, tankers, etc. are not supposed to be in these areas. The other, mostly blue shaded, areas represent clear ways and sea lanes that vessels use. In the remainder of this section the term sea lane is used. Note that the colors of the areas and sea lanes are only used to differentiate the different areas and sea lanes.

Sailing between the lanes is not prohibited, but may result in dangerous situations. Needless to say that the amount of vessel movement and the sailing between

[2] http://www.rijkswaterstaat.nl/

the lanes is not beneficial for building good situation awareness. Larger vessels that follow the sea lanes often follow a plan and their movement can be predicted. Smaller vessels usually just go from A to B which can be confusing for operators.

2.2.2 Necessity of Building Situation Awareness

One year of AIS data before the Dutch coast was analyzed by POSEIDON. In this data set, several examples where found where vessels:

- Cross or closely pass wind-mill parks or oil-rigs that are present in the North-sea,
- Start to drift,[3]
- Suddenly deviate from their course so collisions with another object (vessel or oil-rig) may occur, or may not occur in case of evasive maneuverings,
- Sail against the traffic,
- Pass restricted areas (such as nature preservation areas) at too close distance while carrying dangerous cargo.

These are all examples that show that a close monitoring and control of traffic at sea is essential. To do this, situation awareness must be built.

Situation awareness that MSS organizations currently build is shown in Fig. 2.2. This figure shows the vessel movements in the North-sea around the Waddenzee area [3]. The purple lines and areas mark the waterway system that exists in the Dutch waters. The arrows in the map indicate the sailing direction of vessels. The labeled boat-shaped figures mark vessels, the labels of these figure are vessel names, and the squares mark the oil-rigs in the area.

It takes some experience to know the vessel movement in an area. If vessel movement is captured at all, it is at this moment not used by MSS organizations for play-back purposes to learn from historical vessel movement. MSS organizations need to learn patterns in vessel movement, behavior that can be associated to types of vessels, etc. Being knowledgeable on patterns of vessels and behavior that can be associated to types of vessels helps in identifying and preventing incidents. Organizations also need to be knowledgeable on differences in behavior of vessels that are caused by the environment, for example:

- Behavior of maintenance boats in wind parks,
- Coast guard ships leaving ports when heavy weather is expected,
- Seasonal fishing patterns,
- Reaction of commercial shipping to heavy weather,
- Pilot boats delivering a pilot to an incoming ship.

[3]Drifting is when a vessel does not have engine power, cannot be steered and is going with the waves and wind

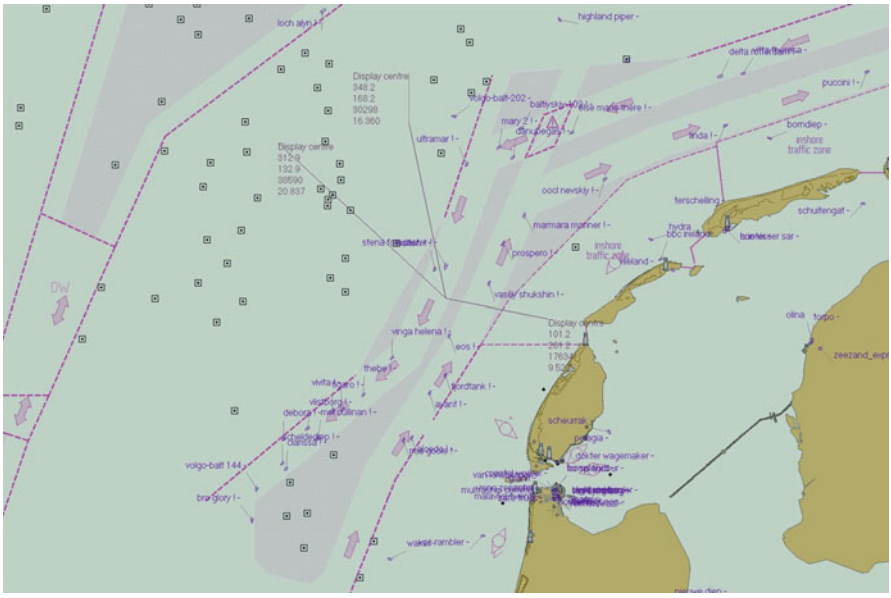


Fig. 2.2 Screen shot of the Netherlands Coastguard system in the Operations Centre in Den Helder [3]

The difficulties operators of these organizations face is that beside interpreting the geo-spatial information of vessel movement, they nowadays also have to combine this information with information about the owner, flag, and even news items in which a vessel is mentioned. Using only geo-spatial information usually does not reveal vessels that have a higher risk profile or that are involved in illegal activities such as smuggling, environment pollution, piracy. With the availability of AIS, improved in-house information systems and the Internet more information on vessels is available to determine the risk profile of a vessel. However, combining different information sources is a time consuming task and can result in an information overload. Another approach can be to simply visit every vessel at sea and investigate it. This is a very time consuming approach. On average, a vessel visit with investigation takes a few hours. Since not all vessels can be visited at sea, it comes to the experience of individual operators to find suspect vessels.

2.2.3 Dealing with a Reduced Sensor Network to Build Situation Awareness

To monitor large areas such as the Dutch coast, a relatively large number of sensors is needed. Along the Dutch coast there are several radar stations and AIS base stations that detect and pick up data from vessels. This data is combined and

transmitted to the center of an MSS organization. Since the number of sensors is relatively large, it is possible that one or more sensors produce data with less quality than normal. The result of data quality loss is usually that an area cannot be monitored or can be monitored less well. A reduction in the quality can imply that a sensor is not functioning, or that the weather conditions in the area prohibit a good transmission of signals. To increase the quality, a measure can be to sent out a ship or an airplane for investigation, or to deploy a mobile station. Here, mobile means that base stations are situated on coast guard vessels, or airplanes. In other words, the network of AIS and radar stations is extensible and reconfigurable.

2.3 Combining Information

Operators need to interpret and reason on a lot of information to make the correct decision. The military domain already recognized this problem and defined a process to cope with it. This process is coined the OODA loop: Observe, Orient, Decide and Act [2].

2.3.1 Observe, Orient, Decide and Act

Organizations that provide monitoring and control services often adopted the OODA loop. The OODA loop guides operators on the decision making process, and on using available information. The OODA loop has its origin in military command and control systems, but was also adapted for civil systems. Figure 2.3 shows the process.

The process has the following steps:

- Observe: know what is happening.
- Orient: understand the meaning of what was observed.
- Decide: weighing the options available and picking one.
- Act: carrying out the decision.

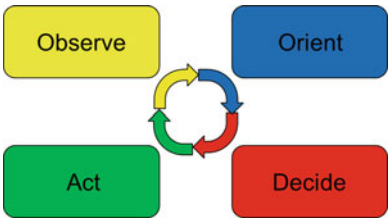


Fig. 2.3 The OODA loop

Following the OODA loop in itself is not possible if information is not combined. Combining information from multiple sources implies that operators need to interpret and reason upon the information.

2.3.2 *Interpreting and Reasoning on a Situation*

What operators do, and what the OODA loop describes is that operators use data from sensors and domain knowledge to interpret and reason on a situation. This happens at the beginning of the loop, and starts again after a decision has been made and an action has been taken. Figure 2.4 illustrates this.

In Fig. 2.4, the triangles represent (on a high level) the knowledge and information sources where operators have access to, and his abilities. The arrows indicate how the sources are used by operators. The blue triangle represents domain knowledge. Here, domain knowledge can be the experience of operators, or the information that an operator retrieves from the organization's MSS systems. Operators need to be creative to apply the domain knowledge to a situation. The red triangle represents the reasoning ability of an operator to make this mapping. The green triangle represents the data of a situation that is picked up by sensors and that must be interpreted by operators, or that operators retrieve from the Internet or other sources. The green triangle also represents the result of the interpretation, as is explained below.

When an operator starts interpreting or reasoning on a situation, sensor data is likely to be the most informative input. After having added more domain knowledge and having reasoned on the situation, the sensor data becomes less important, and the domain knowledge and reasoning ability of operators more important. Figure 2.5 illustrates this reasoning, combining and abstracting of information.

In Fig. 2.5, in the background the three above mentioned triangles can be recognized. The arrows indicate the flow in which operators reason and combine. The arrows form a spiral that goes from bottom to top. The spiral crosses several levels of reasoning and combining. These levels are separated by horizontal lines.

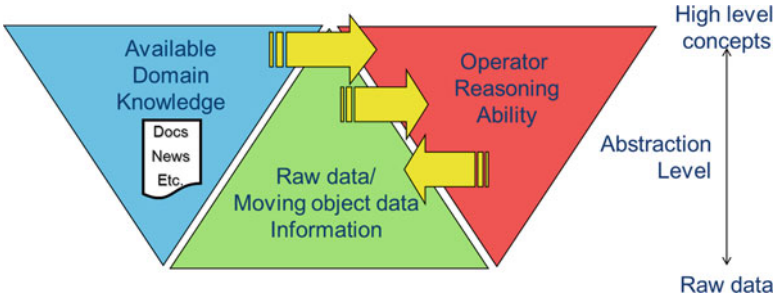


Fig. 2.4 Combining raw data, knowledge and reasoning ability

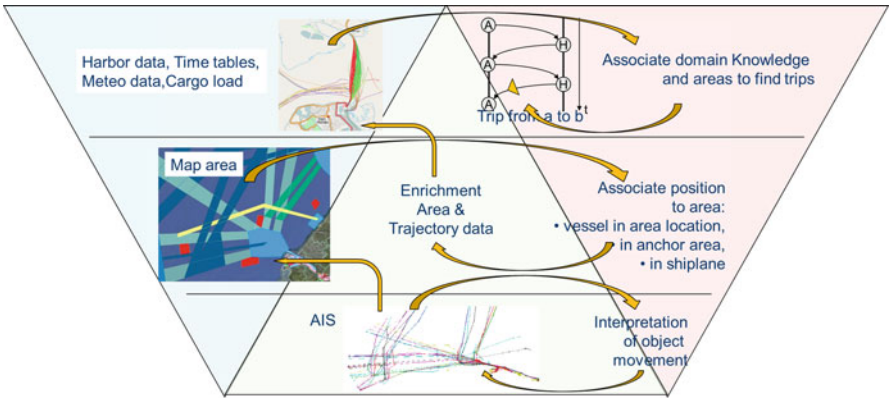


Fig. 2.5 Abstractions in reasoning

To solve a question three levels of reasoning and combining were needed. Other questions may take fewer or more reasoning levels. The question that is solved in the figure is: "is the position and movement, and timing of a vessel according to a scheme". Here, scheme means that the vessel is using the sea lanes that it is supposed to use, and that it is moving at an expected moment of the day. At the lowest level, operators interpret the AIS data that is received by sensors and that is visualized by the MSS system of the coast guard. The operator searches for behavior in the movement of objects that cannot be understood without further investigation. To make the object movement better interpretable, the image is enriched with map data of different forms. This map data can include data about sea lanes, ship wreckages, weather information, seasonal effects on fishing behavior, etc. By doing so, it becomes easier for operators to associate the position of objects in an area. Through the association, a meaning can be linked to the position. For example, the operator knows the vessel is in an anchor area. The object movement can be further explained by arrival and departure data from harbors, time tables of daily trips, maintenance schedules of oil-rigs, meteo data, the cargo of a vessel or by radioing the vessel. By using this information the operator can deduce that the vessel needs to take another, less economic route to its destination because of its dangerous cargo, or that a vessel follows a time-table.

Figure 2.5 also shows that the type of information that an operator has to handle has large diversity and can vary per scenario. To illustrate the reasoning and combining of information together with the OODA loop, the following example is used:

Observe: Operators are observing the traffic before the Dutch coast. This is done by watching visualizations of AIS messages on screen, and listening to radio traffic and reading Internet sites with blog data of the area.

Orient: A vessel that is sailing in a sea lane suddenly deviates from the lane. This in itself is not abnormal behavior. However, it is now headed for an oil-rig.

According to the AIS messages that are sent by the vessel, it is a so-called Special craft. Special crafts are a rather broad class of maritime investigation vessels or maintenance vessels. In this case, it can mean that the vessel is a maintenance vessel that is going to do repair activities, or that is delivering goods to the oil-rig. If the vessel is not doing maintenance activities, then it should not be in the neighborhood of the oil-rig. The vessel movement is projected on a map of the area. Vessels that deviate from a sea lane can be found more easily this way. Also, vessels heading for static objects such as oil-rigs can be identified more easily. The additional information in the AIS messages of the vessel gives the operator additional information to reason on and to further enrich the data. In this particular case, the operator starts to search if the vessel is a maintenance vessel of some sort.

Decide: In general, vessels must keep a distance of about half a nautical mile from static objects or other vessels. This is to prevent incidents. For example, if the vessel becomes "not under command" for some reason or suddenly deviates from its course, it may hit an oil-rig and cause an environmental disaster. If the vessel is a maintenance vessel, it should be mentioned on a short list. The operator decides to check if this is the case. By using time-table type-of data in combination with information on maintenance vessels in the area, information is cross-linked and the operator can determine whether the vessel is a maintenance vessel. Since the vessel is not in the list of vessels that is the result of the cross-link, more information is needed. There are multiple ways of getting this information.

Act: The suspicious vessel is not listed as a maintenance vessel and the operator decides to radio the vessel to ask its intentions and to find more information on the vessel. The operator asks the vessel and enriches the result of the conversation with the available information.

Observe: The result of the radio call is that the vessel reports they are hired by the owner of the oil-rig to do under water maintenance. So the operator starts to gather additional information on the vessel itself.

Orient: A quick search in the coast guard's MSS system learns that the vessel is a special craft that is equipped with under water repair equipment for oil-rigs. The found characteristics of the vessel is combined with the available information.

Decide: The operator is satisfied and decides that further checking is not needed.

2.4 Typical Scenarios

MSS organizations have their specific tasks. The type of incidents that are watched for differ per organization. In this section we discuss typical incidents such as detecting traffic rule violations, performing search and rescue actions, and finding vessels that act suspiciously. Where applicable, in the scenarios we point out the crucial/vital information that leads to a decision.

2.4.1 Strange Behavior

Strange behavior is mostly related to kinematic data (course, speed, position) of vessels. The movement data is picked up by sensors. Operators are looking for unwanted, suspicious, or illegal movements of vessels and interactions between vessels. Characteristics to identify incidents are:

- Speed of movement:
  - Absolute speed is larger than 25 knots in open sea
  - Loitering or hovering in an area
  - Sudden increase or decrease of the speed
  - Type of vessel in relation to its speed
- Direction of movement:
  - Making 180° or 360° turns
  - Sailing against traffic
  - Sailing through anchor areas
  - Unusually large number of course changes
- Vessels outside the historic behavior
- Unknown origin and/or destination of vessels
- Vessels that are drifting:
  Vessels at drift are vessels that are not under control. This may occur because the engine breaks down or the bridge is not under command. Drifting vessels may cause a lot of problems if they collide with other vessels or obstacles at sea such as oil-rigs.

Watching for the above type of behavior is a way for operators to focus on vessels that are more likely to be involved in incidents.

2.4.2 Search and Rescue Operations

Another type of activities in which operators have to assist are search and rescue actions. To aid rescuers, operators search for background information on vessels that are involved in the incident. We illustrate this type of action by using a scenario in which a vessel is involved in a clandestine operation and gets into trouble in front of the Dutch coast. Note that the scenario itself is fiction, but consists of elements that have happened in practice.

An Italian navy vessel is patrolling in the Somalia area. The vessel is patrolling in the area as part of the NATO operation Atalanta. The vessel observes through its sensors that two vessels have an interaction. The Italian navy vessel requests information from other vessels in the operation Atalanta if more is known about the vessel and if an interception is needed.

The request is handled by a vessel of the Danish navy and it passes general information about the questioned vessel to the Italian vessel. Details about the cargo are not shared. Additionally, the Danish vessel announces that the vessel is not to be intercepted. Although not shared, this is because the Danish navy and the Danish police are involved in a joint operation where they are monitoring clandestine operations. The Danish police has infiltrated and wants to see the course in which the clandestine operation evolves. The vessel that is involved in the clandestine operation is headed for the Danish capital Copenhagen. The vessel has to pass the Dutch coast.

Before the Dutch coast the vessel that is involved in the clandestine operation gets into trouble. It has had a collision with another vessel in stormy weather. The incident happens in a sea lane. Both vessels are in need and send a distress signal. Operators of the Dutch coast guard are monitoring the sea. They receive the distress signal and try to establish communication with the vessels to get more details on the situation on board. The AIS data that is picked up by the Dutch coast guard contains the identity of the vessels. Other information that is transmitted through AIS and that is relevant in this case is the type of vessel, whether the cargo is dangerous or not, the size of the vessel, and its destination. The coast guard asks vessels in the neighborhood to be aware of the incident and that it may result in dangerous situations. Since the vessels are powerless, they start to drift. The coast guard sends out a helicopter to get more information on the incident. The result of these actions is that a search and rescue operation must be set up.[4]

Since the coast guard wants to have as much information as possible to better structure the search and rescue, it asks affiliated organizations if they have more information on the vessel that is important for a search and rescue activity. The Dutch coast guard passes a request to Northwood in the UK from which the operation Atalanta is controlled and monitored to learn if more is known about the vessel. Northwood passes all information it has on the vessel that is involved in the clandestine operation. Through this information the Dutch coast guard finds out that the cargo contains Anthrax, which was meant to be distributed to terrorist cells in Europe via Denmark. The rescuers must use protective clothes and other vessels must keep a safe distance of at least 1,000 m. Vessels that are within the 1,000 m or are on a projected course in which they cross the Search and Rescue area must be notified to not enter the area or to leave the area. The coast guard approaches all relevant vessels individually to take the necessary actions.

2.4.3 Drug Trafficking

Drugs trafficking in general is difficult to detect. Vessels involved in such activities try to behave as normal as possible to avoid drawing attention. Also in Dutch waters,

[4]Note that the OODA loop is used here.

drug trafficking takes place. Getting the drugs into the country can happen in several ways. The vessel with the drugs can just sail into the harbor and release its cargo. Harbor authorities should be aware of this type of incident and act on it. Operators of the coast guard can be supported by finding background data of the vessel and determine the chances of it being involved in illegal activities. Other ways are to have rendezvous at open sea, package droppings (a vessel drops a package in sea that is picked up by another vessel), or very quickly go to shore, drop a package and get out again before authorities can respond.

The following scenario illustrates a rendezvous of vessels. Note that the scenario itself is fictional, but consists of elements that have happened in practice.

It is commonly know that fishing trailers are at sea for larger periods of time, and go to distant seas to catch fish. In this case a relatively small fishing trailer heads to the horn of Africa to pick up a drugs transport and bring it to Europe.

The vessel approaches the Dutch coast from the south. North-west of Texel, 5 miles out of the coast, the trailer deviates from the regular sea lane and decreases its speed to come to a full stop. At that time several vessels rendezvous with the trailer to transfer the drugs. After some time they depart again. The trailer carries on and the sailing vessels head back to the harbor to bring in the drugs.

Is is difficult to detect rendezvous at open sea. There are a number of reasons: the incident happens in a relatively short time frame, vessels do stop at sea, the incident may happen just out of reach of vessels that can respond. Catching the drugs traffickers on shore is usually difficult since it is mostly done at night, and the dunes offer sufficient protection.

2.4.4 Smuggling of Weapon Technology

Finding out whether a vessel is suspicious or not is not trivial. Of course, if the behavior of the vessel is strange, then operators may notice this by just monitoring an area of the sea. It is also relatively easy to identify vessels whose AIS information is not correct, such as the destination of a vessel. This is not sufficient if a vessel behaves normally.

That a vessel can be suspicious while its behavior and information on crew, cargo, and owner are fine can be best illustrated by a news article that was published in June 2010 in the New York Times [1]. In this article it is illustrated how countries such as Iran are establishing a network of vessel owner companies that maintain a continuously changing fleet of vessels. In every transition of owner, the vessels change their identification. This means that not only the name of the vessel changes, but also unique identifiers that are passed in AIS messages such as the MMSI and IMO number (cf. Sect. 1.5.1). By doing this, vessels can be used to transport sensitive cargo such as nuclear weapons and products for countries that are not supposed to have access to them.

To illustrate the above case of finding background information, we use the example of the Indian vessel company WMA. Note that this scenario itself is fictional, but consists of elements that have happened in practice.

WMA is a medium sized company that has several tankers, container and cargo ships in its fleet. A vessel of WMA is appearing before the coast of Rotterdam. There are no issues with the approach the vessel takes to go the harbor, or with its AIS information. Also, the flag of the vessel company (India) does not raise alarm bells. The vessel enters the harbor, releases its goods, takes on new good and leaves for a next destination.

A few months after this event, Iran is involved in an attack in which a fast attack patrol craft was used. The type of craft can carry torpedoes and is therefore forbidden technology for Iran to possess. Iran was able to obtain this type of fast attack patrol craft through vessels of WMA. A vessel of WMA loaded this type of patrol craft a few months ago in the harbor of Rotterdam. WMA is affiliated to an Iran vessel company. WMA is smuggling for Iran, since also the identity of the vessel that was used in the smuggling is relatively new and can be traced to other companies.

2.4.5 Terrorist Actions

The context of the following scenario is a hypothetical situation in which the Dutch coast is confronted with "Non-Cooperative Maritime Actors", i.e. terrorists. Operators in the coast guard center notice that a part of the Dutch coast has less sensor coverage than normal. In this area AIS information is less reliable. An investigation is started to determine if the behavior is explainable through natural causes, such as bad weather areas.

To cope with the reduced sensor coverage an airplane is sent to investigate the situation at sea. Also, a coast guard vessel that carries its own AIS base station is sent out. The vessel relays the received AIS data to the coast guard center.

The reduced sensor coverage takes longer than expected. Therefore, a maintenance engineer is sent to the AIS base station that produces less reliable results. Upon arrival at the base station, the engineer notices that someone tampered with the station.

The tampering of the base station makes this incident a terrorist action. Although the coast guard responded adequately in coping with the reduction in sensor coverage, the area was unobserved for some time. Deploying the vessel and airplane simply takes time.

2.5 Challenges for System Support

Finding incidents in large areas is not a trivial task. Smugglers, drugs traffickers, and terrorists are aware of what organizations can and cannot find, and are creative in hiding themselves. MSS organizations, therefore, have to look for details in the movement of vessels and other information that can be associated to vessels. This usually means combining several information sources, and reasoning on the information to reach a decision.

It is clear that if MSS organizations are adequately supported by systems that provide the right information at the right time, they are able to find more incidents and work more efficiently. The challenges of such a system can be summarized as follows:

- Alert operators when needed –
  It is commonly known that observing a situation for some time is a tedious task and that it may be difficult for operators to keep full attention. Consequently, an effect is that operators may not continuously monitor a situation. This may result in a late response to, or even completely missing of an event. Because of this, operators need a system that operates in the background and that alerts them when needed. In the meantime the system is responsive so operators can use it to investigate vessels and find their background data.

  In Sect. 2.4.1 we discussed a number of violations to traffic rules. These violations can be categorized as geo-spatial violations and information violations. Geo-spatial violations concern speeding and sailing against the traffic. Information violations concern wrong specification of cargo, or a wrong destination. In case of geo-spatial violations, operators would be helped with an alert when a vessel is sailing against the traffic.
- Provide a historic overview on vessel movement to operators –
  One of the largest problems for operators is to have a clear view of how vessels move over time. This movement may change over time. For example, in case of bad weather, vessels may be forced to go for anchor at unexpected places, or may deviate from their course. Also, the season may have effect on traffic. Fishing, for example, is not allowed during the whole year. It may also be that temporary situations have impact on traffic. For example a construction site at sea results in a change in the movement pattern in that area. Having an overview of vessel movement in the past in combination with the knowledge that construction activities are deployed in an area, allows operators to compare and decide whether actions are needed.

  Operators have special attention for areas where vessels cross each other. The reason for this is that these areas are more likely to have incidents because there is more activity in that area. Typically, these are the areas where sea lanes join or cross each other. When observing vessel movements in an area, however, there appear to be more areas than the most obvious ones. Knowing the areas that, from a historical perspective, have vessel crossings, improves the understanding of a situation.

- Provide interaction to the system, so operators can loop back over time to check vessels –
  For operators it is difficult to notice whether vessels have some kind of interaction. A number of interaction patterns can be identified, for example:
  - Follow pattern: Vessels follow each other at close distance;
  - Rendezvous pattern: Vessels meet each other at open sea;
  - Package dropping pattern: Vessels that stop or slow down, speed up again, in combination with vessels that do the same, where the stop or slow down is in the same area.

  These patterns are typical for smuggling. They are also difficult to detect since the event evolves over some time and the moment of transfer may be short. In fact, it may be difficult to detect this type of behavior at the moment of occurrence. It would therefore help operators if they can loop back in time and replay the movement of vessels to analyze the behavior they show.
- Support the operator in up-keeping the quality of data when a sensor or other system part fails –
  We discussed the combination of information by operators. The information itself is coming from several sources. The sources can be sensor systems, information systems, or information services. Most often these sources come from different vendors or rely on infrastructures that are not controlled by the organizations that use them. The latter applies to most web-based services. The availability of information sources is not guaranteed. This means that the system as well as the operator himself must be aware that a source may not be available. A system should therefore be able to handle unavailability of information sources and integrate an information source when it is added to the system or when it comes online.

This chapter introduced various challenges for improving situation awareness in the domain of maritime safety and security. The next chapters of this book will elaborate these challenges and work on solutions.

Acknowledgements This research has been carried out as a part of the POSEIDON project at Thales under the responsibilities of the Embedded Systems Institute (ESI). This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

We would like to thank Piërre van de Laar, Jan Tretmans, Pierre America, and Frans Reckers for their useful feedback on an earlier version of this chapter.

References

1. Becker J (2010) Web of shell companies veils trade by iran's ships. The New Yourk Times, 7 June 2010. http://www.nytimes.com/2010/06/08/world/middleeast/08sanctions.html
2. Boyd JR (1995) The essence of winning and losing, 28 June 1995. http://www.danford.net/boyd/essence.htm

3. Nederlandse Kustwacht (2009) De nederlandse kustwacht. http://www.kustwacht.nl/sites/default/files/StencilKWjeugdnw_0.pdf
4. Wikipedia (2012) Situation awareness. http://en.wikipedia.org/wiki/Situation_awareness

Chapter 3
On the Architecture of Systems for Situation Awareness

Michael Borth

3.1 Introduction

The core of systems for maritime safety and security centers on one task: generate situation awareness from various sources of information and observations provided by many systems that are open to cooperation, but operationally independent. At first glance, this task may be seen as a tree-like structure of information processing steps that starts with many providers of data or information (the roots), provides information processing that aggregates, interprets, and turns many individual items into understanding, thus generating the sought after situation awareness picture (the stem), which allows to reach various application goals (the leaves together with the fruit). The realization of such a system is a challenge, but – again, at first glance – only in regard to individual data processing steps. The system's architecture seems straightforward.

However, the real-world challenges and constraints that such systems face are far from simple: Timing in complex real-time interactions with feedback circles and humans in the loop, uncertainty of observations, and the flexible nature of a system-of-systems configuration do require elaborate architectural concepts. This chapter introduces these concepts and the architectural reasoning behind them.

3.2 The Domain Experts' View on Information Processing

We start our investigation of the architecture with a domain experts' view on the information processing necessary for application goals in maritime safety and security. In this domain, which is introduced in more detail in Chap. 2, operators

M. Borth (✉)
Embedded Systems Institute, Eindhoven, The Netherlands
e-mail: michael.borth@esi.nl

P. van de Laar et al. (eds.), *Situation Awareness with Systems of Systems*,
DOI 10.1007/978-1-4614-6230-9_3,

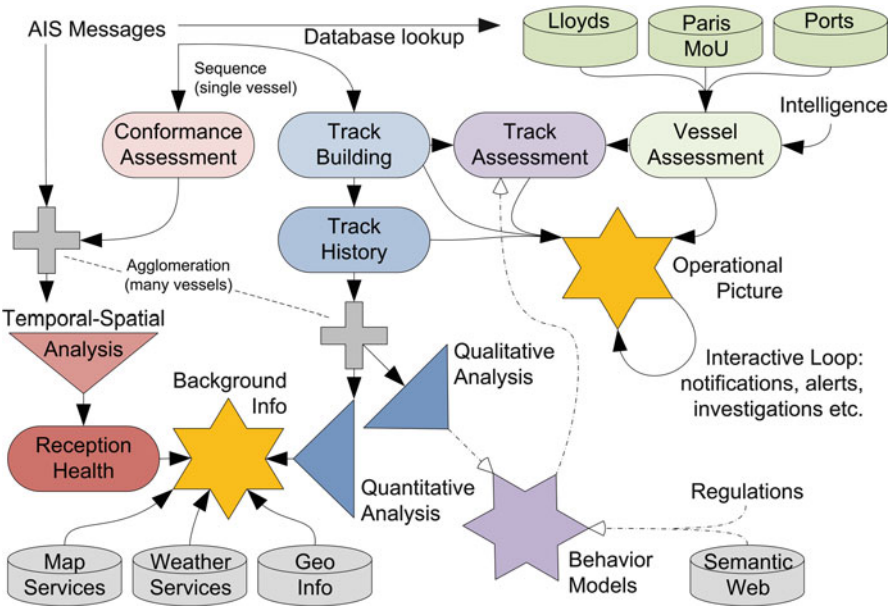


Fig. 3.1 Information processing for maritime situation awareness (Overview)

use many available information sources: AIS messages of vessels, online databases of port authorities, Lloyds [5, 6], Paris MoU [13], or others, information services, e.g., for maps or weather data, intelligence provided by third parties, plus the open Internet including the semantic web. Any subsequent step stems from such information sources, as Fig. 3.1 on information processing activities shows.

The real-time online activities of track and vessel assessment are at the heart of the operation; they produce assessments used to judge vessels and their behavior, trigger alerts or notifications. These assessments are based on the analysis of observations, including comparisons with expectations and models of behavior generated via qualitative analyses of past actions. Many of these comparisons depend on content and context, e.g., one might compare a track to behavior models of different ship types given weather conditions and local circumstances. Such and similar considerations span processing steps and data of different timescales, introduce human operators in the system, leading to interactivity and loops, and altogether result in the complex information flows that challenge the system's architect.

3.3 The Architect's View on Information Processing

The understanding of the domain experts' view on the information processing translates into a high-level view on functionality that the system needs to perform for its operators on the given tasks. A system architect's view on the

information processing, however, factors in many additional aspects, e.g., performance requirements, available resources, goals regarding other systems and long time-spans, like re-usability and evolvability. Especially, the system-of-systems (SoS) gestalt that our system takes impacts many of these aspects [7]. Here, we focus on those SoS aspects and their effects which turn the information processing into a challenge that requires an architecture beyond the tree-like structure assumed at first glance: origin and provenance of data, information channels, the timing of information flows, interaction between SoS components and humans-in-the-loop, the handling of uncertainty, the health of the SoS, and the flexibility of the SoS configuration. These interrelated challenges can be summarized as follows.

Many systems of the distributed SoS provide information, but there is no single entity that has the task, capacity, or right to access all data. Therefore, information available at one part of the system may not be available to another part. Even if data is passed on, it might come in with a delay, or be incomplete due to channel effects. This hinders recognition of coincidence and causality during information fusion, changing results and affecting subsequent actions of operators or subsystems, as many tasks and processing steps are sensitive to context or content.

All observations that feed the system and many of its processing steps are inherently uncertain: Sensors have technical limitations; environmental effects like weather may impact accuracy and range; information sources may omit to pass on sensible data or even falsify their transmissions to achieve their own agendas. Furthermore, situation awareness systems use algorithms to give meaning to data that measure similarities between observations or models, thus introducing soft assessments into the information processing. The effects of uncertainty in data are similar to those described above as it also impacts results and subsequent steps.

The information processing of situation awareness systems depends on the performance and reliability of many contributing parts, in short the health of the SoS and its information. As the notion of performance and quality-of-service depends on a given task, such health considerations are application dependent and dynamic. In turn, they impact the information processing, e.g., if data is known to be unreliable or individual tasks find their prerequisites not given.

As the systems in an SoS are operationally independent (see Chap. 1), they may join and leave the SoS, thus changing the configuration. Equally, parts of the SoS may change the configuration they use as basis for their own operations, e.g., by switching from one input to another to improve performance. Consequently, no part of the SoS may rely on a specific SoS configuration. Instead, all operations must become independent of these details. Meanwhile, locally available data about the configuration and its abilities is beneficial to many tasks and should be used.

Altogether, these challenges make it necessary to adapt and add to the idea of a tree-like information architecture: As missing, un-timely, uncertain, or false data together with context-sensitivity disturb the straightforward generation of the situation awareness picture, mechanisms set to rectify this require structures that provide feedback or context. As recognition and analysis steps impact subsequent steps, but interact in different and changing time-frames and on different abstraction-levels, a flexible decoupling must allow for that, requiring time-wise adaptive information

processing and memory structures. As the SoS configuration is dynamic and impacts performance and available functionality, insights on those aspects become beneficial to many information processing steps, requiring system-level functionality with additional information flows. We introduce the architectural concepts that we chose to fulfill these requirements in the remainder of this section.

3.3.1 Information Flows

To address many of the challenges described above, i.e., handling of uncertainty, effects of origin, provenance and channels of information, health of the SoS, and flexibility of the SoS configuration, we set up the concept of information flows as the building units of our architecture. Such flows express the notion that information is moved and processed within the system, and that the ways in which this happens is what requires the architect's attention (as opposed to content of data or functions computed with the data alone). The term's association regarding moving liquids is intentional: There are sources of information which equal wells, end-uses that are like sinks, channels that transport as pipes do (effects of capacity and potential loss of content included), fusion steps that might generate stronger streams, but also might find that the contents involved do not mix, and processing that changes content and form according to their objectives using available inputs.

Structure-wise, such information flows are compositional. Complex flows are built from simpler flows with the fundamental forms of source, channel, and processing step as basic units. These fundamental flow units are directional steps that transport information towards an output. The outcome, i.e., the content of the output, depends on the specifics and the type of the information flow: The outcome of a source is the data generated here with a possible impact of intentional tampering or non-intentional interference. The former may be seen as a function given the data's content; its implication can be modeled as trust in the source. The latter is a matter of quality-of-service that may be seen as a probabilistic function that changes the outcome randomly. The outcome of a channel is solely dependent on its single input and its quality-of-service that may again be seen as a probabilistic function. The outcome of a processing step is defined by the functional processing of the input(s), e.g., a filter, a merge operation, or the computation of new attributes. The processing may depend on parameters, to be set internally or dependent on (further) input. Additionally, quality-of-service affects the outcome as well, similar to the explanations above.

Certain properties and aspects are shared by all flow units and must be retained during composition. Foremost, we warrant a principle of locality for all flow operations such that they are only subject to inputs and parameters directly linked to them. Equally important is a combined handling of the information that is inside the flow and meta-information about the flow, e.g., quality insights and provenance data. Meta-level information is applicable as input and output: Parts of the system might use the meta-information to adapt their own processing steps, e.g., by addressing

quality issues or disregarding anything from a specific source, but also compute new data for the meta-level from their own insights or operations. In this parallel handling of insights about the data flow itself, we realize a secondary information flow that complements the one the core functionalities of the system work upon. To stay in the allegory of liquids, we regard the pipes as principal parts of the system that inform us about origin, quality, composition, temperature, pressure, etc. of the liquid.

The understanding of an information-centric system-of-systems as a composition of flows that adheres to the described principles and realizes a joint handling of all types of information in a way that they may impact each other has an important consequence: The system-of-systems' gestalt matters, not only in the realization, where it is cause for effects, but also during architecting and runtime. During runtime, it enables mechanisms and processing steps that address the concerns listed above. This is described in more detail in Sect. 3.4. During architecting, however, it enables the explicit investigation of such effects in alternate system-of-systems configurations, especially with model-based design techniques. The combination of both of these aspects in one consistent approach is where we see the biggest advantage, surpassing alternate solutions that exist for individual tasks, as, e.g., type-enhanced data flows in web services or data exchange models that also inform about metadata, but help little with our other tasks.

3.3.2 Transport Mechanisms, Time-Aspects, and Memory

The information flow principle and its analogy of transporting liquids through pipes partly explains the movement of information through the system. Especially in the 'the root section' of our system, where sensors provide input that is fused directly, we see how such transportation mechanisms work: Many sensors, e.g., AIS receiver stations, collect and provide input, AIS messages in this example. They push their output – the input into the system – forward, just like a well would do as long its pressure is sufficient to move the liquids along. Such a push mechanism is easy to implement in information systems, e.g., by a time-triggered heartbeat that 'pumps' whatever is there at specified intervals, or by a queue mechanism that triggers the transportation of content the moment a certain threshold is reached. In our work, we are bound to existing data distribution services (DDS) for this purpose.

Investigations, identification of relevant observations, cross-referencing, and interactions with operators to provide for their needs are too interactive and too dependent on situations to be realized per push mechanisms. In essence, we need the option to integrate local loops of queries and answers that achieve two goals: First, they must enable the dynamic ad-hoc investigations that operators require outside the scope of foreseen (and thus possibly pre-computed) tasks. Second, all information processing and its outcome must be enabled to impact other tasks and insights the system provides, as situation awareness often requires the interweaving of all relevant pieces of information. Furthermore, many of the more cognitive

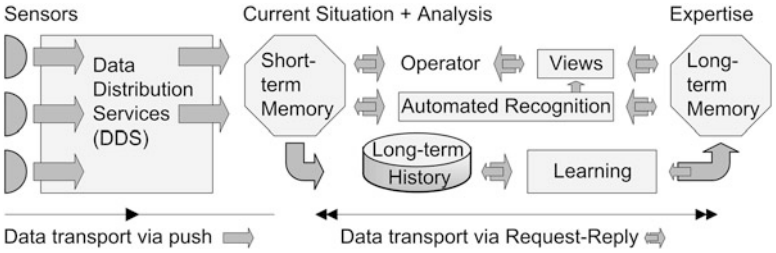


Fig. 3.2 Multi-stage information architecture

steps within the generation of situation awareness, e.g., recognition of maneuvers or assessments via qualitative analysis (see Fig. 3.1), need a history of data in order to learn from it the models on which they operate. This, too, cannot be realized by the simple push principle. Instead, we use a multi-stage architecture that adds a short-term and a long-term memory, which are set to couple higher order information processing with views on longer time-frames and dynamic behavior. This architecture is sketched in Fig. 3.2.

3.4 Architectural Reasoning

Within the architect's view on information processing, we listed various non-functional concerns that challenge systems-of-systems for situation awareness. The architectural concepts on information flows, transport mechanisms, time and memory aspects introduced above are set to address these challenges. The five parts of this section detail the architectural reasoning behind this contemplation.

3.4.1 System-of-Systems Aspects

If we evaluate systems-of-systems from the point of view of those responsible for the engineering of the overarching system, we realize that clear advantages exist, but they come at a high cost: The independencies between stakeholders and their systems lead to spread efforts and responsibilities, but lead to a lack of overview and direction as well, as neither operation nor development is centrally controlled.

As a major consequence, we require architectures and mechanisms that enable collaboration across boundaries of systems that were not designed in concert. This includes an inherent resilience against ill-effects of configuration changes, means for the runtime integration, test, and acceptance of systems that may remain a 'black box', and adaptations to different communication and interaction protocols. Our approaches to the latter items that are concerned with the SoS formation and

interoperability are detailed in Chaps. 14, 15, and 11. The need for an inherent resilience against ill-effects of configuration changes, however, is an architectural item that is addressed using the information flow principle.

First and foremost, the flows carry the meta-information that identifies the provenance of data streaming through the system as well as the channels it took. This is crucial for the interpretation of incoming information (see Chap. 13), its protection (see below), but also for aspects that relate directly to the dynamics in a SoS' configuration: Configuration change may alter the observable theater or the type of observations, and impact the quality of data, the processes, and protocols. Consequently, one might need to re-test or even adapt parts of the system. Given flows that contain the metadata detailing the cause of such effects, any system component is enabled to react accordingly, e.g., by reinstating runtime integration procedures. Information processing, especially with regard to the data quality, often benefits from such meta-information, too. One might disregard configuration insights on higher abstraction levels, but that is a choice of the respective stakeholders, not one to be taken for them. Consequently, we do not use any mechanism that fully hides changes to the underlying configuration from system components, e.g., structural design patterns like Bridge [2]. Instead, we de-couple system parts as far as necessary via computations on the short-term memory that abstract away from binding details with full knowledge of relevant SoS aspects.

3.4.2 Handling Uncertainty

As we laid out at the beginning of Sect. 3.3, observations that feed the system and many of its processing steps are limited in regard to their trustworthiness, accuracy, unambiguous interpretation, or other factors that introduce uncertainty. Furthermore, any situation awareness system that evaluates or predicts actions in an operational theater must by its very nature consider different possible behaviors of the involved actors, since it is uncertain about the truth.

The handling of the thus evoked uncertainty benefits from a consistent and coherent method and mathematical calculus, as this allows the re-use of corresponding components or implementations, furthers common and thus shared interpretations, and avoids transformations, which are computationally expensive and may introduce imprecision themselves. The system-of-systems gestalt of situation awareness systems, however, denies the assumption of such an integrated implementation. Instead, the architecture must provide the means for local investigations of the effects of uncertainty within any system component that is concerned with it. This translates to the requirement that any such component learns about the uncertainties that accompany its inputs and that it is enabled to provide similar information about its own output. The architectural concept of information flows ensures this with the provision of metadata transport.

The content of that metadata may take many forms. In our work, we found that we could express any notion of uncertainty and its origins via conditional probabilities. These may stem from expert assessments, measurements (e.g., of mean-time between failures or accuracy), or calculations, as, e.g., Pearl describes in the first chapters of [9]. We regard conditional probabilities as the most simple form of metadata that fits our requirements, thus providing the common ground that unifies the necessary interpretations of uncertainty. Furthermore, they are the foundation for the use of Bayesian networks [8], a methodology of choice for real-time reasoning on uncertain information [3]. With Bayesian networks, one can undertake local uncertainty computations within components efficiently – thus offering a suitable approach for any system within the system-of-systems – but ultimately, the individual systems' stakeholders remain free in their decision how to handle uncertainty within their scope of responsibility.

From the system-of-systems' point of view, there is a strong advantage if at least key functions are all implemented with Bayesian networks to compute the uncertainties involved: Using object-oriented networks [4] with causal modeling techniques [10] at the interfaces between system parts allows to establish a more global reasoning about the uncertainties within the information, but also about the content of the situation awareness itself. Such a global reasoning can be realized either in a distributed fashion, if all information processing units contribute and the information flows between them is bi-directional, or by a separate system that processes the information provided by the other parts on-demand. Altogether, we find that the information flow concept is well suited to enable the handling of uncertainty in systems-of-systems for awareness tasks.

3.4.3 System Health

Any complex system is prone to faults, defects, or unintended behavior. In a system-of-systems setting, undesired emergent behavior and ill-effects of failures become even more likely, as they cannot be countered in development given the lack of control over the implementation of the individual systems. Consequently, we require runtime diagnosis and other mechanisms to ensure the health of the overall system, especially as malfunctioning or serious performance issues will often have entangled effects, which are difficult to handle.

Individual systems that contribute to the SoS might have self-diagnosis capabilities. If so, we benefit from mechanisms that transport any produced data on system health to other system parts without relying on a centralized health system (which might still exist on behalf of the SoS builder). The information flow mechanics offer this, transporting health related metadata to any interested party. On the SoS-level, however, health considerations cannot rely solely on health data from individual systems, as such functionality will often be absent and, even if present, will likely fail to account for system interoperability and overarching effects. Given the SoS configuration dynamics due to the operational independence

of contributing systems and the 'black box' nature any individual system might have towards the SoS, it becomes in fact necessary to compute system-level health solely from observations that can be made about systems and their performance.

We use such an observation-based approach combined with spectrum-based runtime diagnosis (Chap. 14). The use of metadata, especially the trace of a data stream, i.e., the list of all contributing system parts, is mandatory to enable this. We can ensure the availability of such metadata via the information flows together with a short-term memory that stores it for later use. Alternatively, a system-level health component might also act upon centralized data, provided that the SoS tracks its configuration accordingly.

Interestingly, there is strong discrepancy in architectural means and goals between the health investigations and the situation awareness tasks, which both require the short-term memory to fulfill their data needs: The former must access configuration data here, whereas the latter will often use this level to abstract away from such consideration to become (more) independent of the SoS' layout and may actually benefit from 'getting rid of such clutter' for lean computations. Our architecture compromises here, as necessary metadata stays with the flow but not with individual data items.

Another item of interest in system health considerations is the value that most kinds of feedbacks offer: Individual systems will benefit from notifications that their performance is low or that their output is doubtful, as such information might trigger or direct diagnostic efforts. Such feedback might happen between system parts directly connected via the information flows, indicating again an advantage of bi-directional flows of metadata. It could be more far reaching as well. We investigate this further in the next section.

3.4.4 Information Health

The technical health of a system often influences its performance and thus also its contribution to the information-centric operations of a system-of-systems that determine its services and their application. However, systems close to breakdown might still offer the one required functionality and a fully working system might still fail in its application context, especially due to environment effects, e.g., as heavy weather disturbs many sensors in the maritime domain. An SoS-level health consideration on either any individual system or the SoS in total is thus not defined by the absence of faults and breakdowns, but the level of contribution to the overall SoS performance in regard to the application goals. Such a notion of health, however, might depend on the current use of the SoS, the situation, and the environment. Moreover, contribution to the global SoS application goals is often hard to determine, as user satisfaction, a dominant factor here, is difficult to measure. In contrast, automated investigations of an SoS' health require a notion of health that may be computed as locally and as objectively as possible.

The first requirement, locality, combines practical issues, i.e., the need to compute health statements in dynamic and even partly unknown configurations. With locality interpreted as an area of sufficient mutual effects between a system and other systems, i.e., the set of systems whose outputs depend strongly on the input from the investigated system – and thus its health – we see that such computations may transcend the direct neighborhood of any system. They will, however, be limited to a set of inter-reliant systems that can be identified, e.g., via a sensitivity analysis. The second requirement, objectivity, requests an option to compare, communicate, and use health information in a standard fashion without a dependency on subjective user input, which will often be missing anyhow.

In our work, which was only investigative on this item, we opted to combine the needed handling of uncertainty with the objectives here. By understanding the SoS as a set of information-centric operations that transform data or information into data or information of higher quality (and thus more useful to the SoS goals), we arrive at the notion of information health. Such health can be understood as the absence of doubt with regard to interpretations of observations, data, or situations. Doubt is expressed in the amount of uncertainty associated with such information. We quantify it with entropy measures (as introduced in [12]) that also allow to compute information gain during individual processing steps. A healthy system is thus one that transforms the state of not knowing (maximum entropy) to one of situation awareness, i.e., the absence of doubt and uncertainty on the situation.

The computation of information health requires both the memory structures and the communicative means that our architecture offers, but nothing more: The availability of history enables the measurement of in- or decrease of information health and bi-directional information flows transport the necessary metadata. Such health investigations provide insights into the inner workings of awareness systems, which can direct optimizations to where they have the most impact.

3.4.5 Information Protection and Access Control

An important part of information-centric awareness systems for security or safety applications is the protection of sensitive or private information. Both acceptance of such systems and cooperation within them relies on this cross-cutting aspect. Information protection is, however, quite orthogonal in aims and means to the rest of the system, as it is not about sharing and fusing information, but on separation and access control that works within the limitations of a need-to-know.

In monolithic systems, as well as in distributed systems under a common operational control and responsibility, there is a central authority able to enforce mechanisms to protect sensitive information. In cooperative systems-of-systems as the one we consider, such an authority is lacking – but the coalition that works together on the application goals must still have the means to enact protocols set to protect information while sharing it. How this comes to pass is detailed in Chap. 12, but the importance of this topic warrants a closer look at the integration of the needed mechanisms with the architecture.

Providing sensitive information to another party requires trust that this party will act according to the needs and wishes of the provider, that is to say according to its policies. Given the dynamic configurations of SoS, this results in the architectural problem that every party must know the policies to observe without the existence of a central authority. The technique labeled sticky policies [14], where policies are attached to items they are concerned with, can ensure that. Our architectural concept of information flows provides the necessary mechanism to transport the needed meta-information, i.e., policies and provenance, efficiently.

This approach works well for any direct use of the information provided, even if a flow was not preconceived, as a policy might detail possible uses as well as protective measures, e.g., the initiation of trust negotiations. Restrictions to pass on data to parties that lack certain credentials or are not expected to conform to the allowed usage of information might also be forwarded.

Secondary use of information, i.e., use of information that was computed from sensitive information, is more complex. We encounter such information especially in processing steps that act on a short-term memory to fuse information or to raise the level of abstraction or understanding. The party executing such a processing step needs to provide a new policy to the SoS, which shall take the policies into account that were provided with the information in use. This is challenging in regard to the business logic, but not for the architectural means. Still, we have to acknowledge that this task and the required level of trust supersedes common practice in many application domains. Consequently, parties within a coalition that forms an SoS might be unwilling to permit secondary uses of any information they provide due to their lack of understanding and control, thus breaking the concept of cooperation. Luckily, the maritime application scenarios we investigated were seldom of such a sensitive nature.

3.5 Mapping of Core Functionalities Towards System Parts

A major task for system architects is to bring together two different notions: the functional view of the system that is often generated in a top-down fashion from the user's requirements, and realization views describing components, etc. that is understood to stem from a bottom-up approach. These notions meet in the middle, typically realized in a mapping of functionality to components. For the purpose of this chapter, a component view detailing the realization of our system – which is for demonstration purposes only, anyhow – is out of scope. Still, the relations of core functionalities and key architectural concepts merit discussion.

Based upon the domain experts' view on information processing depicted in Fig. 3.1, we distinguish the following services and functionalities.

First, information and data sources: The reception of AIS messages by system components and the various online databases plus the mechanisms used to access them are mostly defined by the state of affairs. As laid out in Sect. 3.3, we use push mechanisms for any incoming data that is not subject to queries or reasoning.

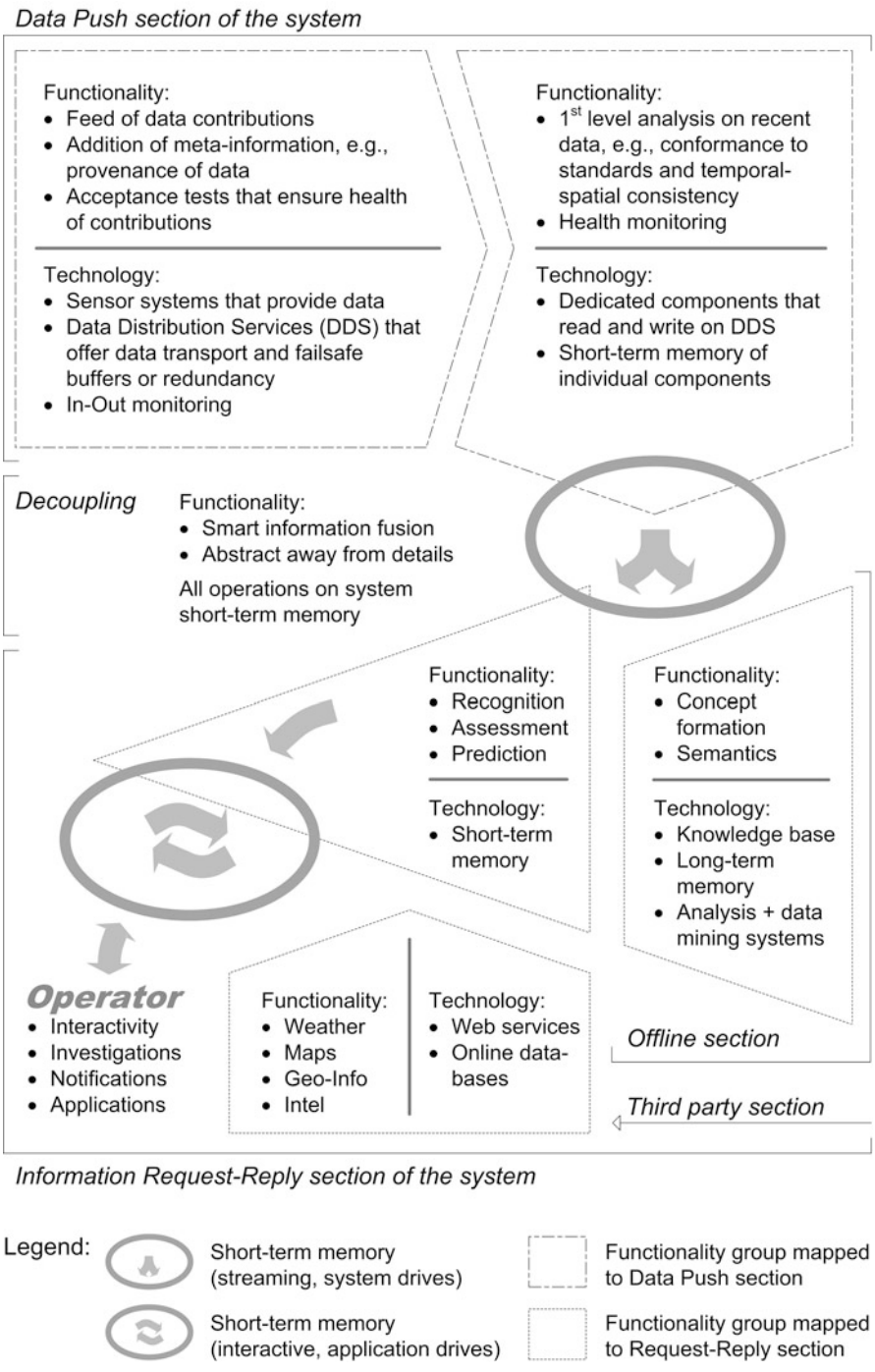


Fig. 3.3 Conceptual architecture of POSEIDON

Second, services that provide information which enriches available data or puts it in context: All specific information queries, including weather data and map services, plus online computations and assessments are within the scope of system parts able to fuse data both across the boundaries of a single information source and a single moment in time. Such parts of the system basically always function upon a short-term memory. This allows the computation of higher representations, turning raw data into interpretations, i.e., in our application, track building and assessment (track and vessel, plus the background of the operational picture).

Third, services which compute, store, and provide high-level concepts and domain knowledge: Analyses that form interpretable models on situations, actors, their actions and behaviors may be done offline, fully outside the operational SoS. They will often use data collected from the SoS, e.g., to learn such knowledge models from observations collected over a longer period of time. Given a closed loop between short-term and long-term aspects of the system, one may integrate adaptive mechanisms in these services and computations, so that the knowledge is constantly updated to new insights.

Fourth, interactivity: The means that allow to query the system according to user interests are best realized on the information in short-term memory, as actuality and interpretation of joined information flows are available here.

Altogether, this forms a conceptional view of the architecture in regard to the core functionality as shown in Fig. 3.3. A specific realization might consolidate an arbitrary number of features, combine or separate any number of functional units on or between individual components given the system's configuration and the preferences of its stakeholders, and add means and mechanisms that are required to ensure reliability, due process, or other relevant concerns. Our own efforts on this are summarized in Chap. 4.

3.6 Open to Consideration

In this chapter, we introduced our information-centric architecture for maritime situation awareness system-of-systems. We envisioned a realistic approach that integrates many techniques to address the challenges of the application domain and the needed realization. This approach led to the realization of the demonstrator system that we describe in Chap. 4, thus validating many of the decisions we took. There are, however, many choices and alternatives open to consideration – and thus worth mentioning here at the end of this chapter.

Part of the realism of our approach is that the system is not set to understand everything which happens in its area of operation. Instead, it aims to recognize vessels, events, and behaviors that are well within the scope of what operators consider to be normal – and thus not worthy of their attention and investigation – or abnormal, in which case alerts, notifications, and elaborations focus the operator's attention on more unusual and possibly dangerous situations. This is a difficult task, but it can be achieved. As our experiments on maritime data indicate, the diversity of

situations that need to be handled this way is within the scope of the used techniques and the integration of domain expertise and domain models generated from data by machine learning allows to detect such events and to report on them.

Nonetheless, our approach draws many of its strengths from the human operators it supports. By accepting the limits of computer-based recognition and reasoning, it focuses on the tasks where it performs well, but relies on available expertise and manpower otherwise. Other system architects may find this to be an invalid prerequisite and require a very high automation, omitting interactivity for strong reasoning capabilities to be provided by advanced AI techniques.

Another item is that our approach shields the operator from many of the system-of-systems dynamics that may greatly influence the situation picture it provides. This keeps SoS operations and safety and security applications well separated. While this provides focus, it also denies the option of active re-configuration to achieve application goals by operational means. Furthermore, it limits the operator's understanding of the sources and reliability of information that is presented to him or her. On both accounts, an alternate approach might be more sensible, provided it does not become a burden due to the added complexity. One related and fundamental choice that should be considered regarding the system-of-systems dynamics is the level of automation of re-configurations. Looking at the different, but in some aspect comparable domain of the Internet, we see that reachability among autonomous systems in computer networks is achieved without human involvement. Here, the Border Gateway Protocol for routing decisions directs data flows with a very high robustness, even though many obstacles similar to those we address exist, e.g., the uncertain availability of nodes in the network that mirrors the join-and-leave of systems in a system-of-systems [11]. Such a strong automation might be possible in our domain, too, even though the tasks in our domain are much more diverse, resulting in more complex dynamics as well.

Furthermore, one might consider the level of adaptivity of the system's parts that are based on learning, modeled knowledge, and reasoning. These techniques may be used in variants that use constant feedback to improve their performance by fine-tuning their findings. This requires the resources to provide such feedback as well as trust into a system that is constantly changing and thus prone to content drift, barely allowing for verification and certification.

As such rather fundamental choices have a strong impact on the architecture of a system, it is safe to say that there is no single architecture that fits all tasks. The Embedded Systems Institute and its partners will address some of these considerations in future work, e.g., [1], and we hope that many more will join us in their investigation.

Acknowledgements This research has been carried out as a part of the POSEIDON project at Thales under the responsibilities of the Embedded Systems Institute (ESI). This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

References

1. Embedded Systems Institute. The Metis project. http://www.esi.nl/metis
2. Gamma E, Helm R, Johnson R, Vlissides J (1995) Design patterns: elements of reusable object-oriented software. Addison-Wesley Longman, Boston
3. Heckerman D, Mamdani A, Wellman MP (1995) Real-world applications of Bayesian networks. Commun ACM 38(3):24–26
4. Koller D, Pfeffer A (1997) Object-oriented Bayesian networks. In: Proceedings of the 13th annual conference on uncertainty in artificial intelligence – UAI'97, Providence, Rhode Island, pp 302–313
5. Lloyd's List. http://www.lloydslist.com
6. Lloyd's List Intelligence. http://www.lloydslistintelligence.com
7. Maier MW (1998) Architecting principles for systems-of-systems. Syst Eng 1(4):267–284
8. Pearl J (1985) Bayesian networks: a model of self-activated memory for evidential reasoning. In: Proceedings of the 7th conference of the cognitive science society, University of California, Irvine, pp 329–334
9. Pearl J (1988) Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann, San Francisco
10. Pearl J (2009) Causality: models, reasoning, and inference, 2nd edn. Cambridge University Press, New York
11. Quoitin B, Uhlig S (2005) Modeling the routing of an autonomous system with C-BGP. IEEE Netw 19(6):12–19
12. Shannon CE, Weaver W (1949) The mathematical theory of communication. University of Illinois Press, Champaign
13. The Paris Memorandum of Understanding on Port State Control. http://www.parismou.org
14. Trivellato D, Spiessens F, Zannone N, Etalle S (2009) POLIPO: policies & OntoLogies for interoperability, portability, and autonomy. In: 10th IEEE international symposium on policies for distributed systems and networks (POLICY'09). IEEE Computer Society, IEEE Press Piscataway, NJ, USA, pp 110–113

Chapter 4
The POSEIDON Demonstrator

Piërre van de Laar

4.1 Introduction

The POSEIDON project was carried out using the Industry-as-Laboratory paradigm under the responsibilities of the Embedded Systems Institute (ESI). The Embedded Systems Institute has ample experience with the Industry-as-Laboratory paradigm, as exemplified by the Boderc [4], Tangram [8], Ideals [9], Trader [7], Darwin [6], Falcon [3] and Condor [1] projects. Yet, in one aspect POSEIDON differed from all previous projects. Whereas, in the past all carrying industrial partners were active in the project's application domain, Thales Netherlands, the carrying industrial partner of the POSEIDON project, was not. Thales Netherlands wanted to become active in the new, emerging application domain of support systems for maritime safety and security. POSEIDON brought together a broad range of advanced scientific disciplines with the vision and expertise in naval systems from Thales Netherlands. POSEIDON had a technology-push character. It had to demonstrate the possibilities for an integrated architecture serving maritime safety and security. In this chapter, we describe the experience with realizing and using the POSEIDON demonstrator. Note that the demonstrator included many but not all POSEIDON results. For example, Linked Open Piracy [2], a data set containing all piracy attack reports issued on the web by the International Chamber of Commerces International Maritime Bureau, was not integrated since world-wide piracy was not considered relevant for operators at the North Sea.

In the next Sect. 4.2, we describe how we built and deployed the demonstrator. We describe how we constructed a coherent set of inputs to operate the demonstrator

P. van de Laar (✉)
Embedded Systems Institute, Eindhoven, The Netherlands
e-mail: pierre.van.de.laar@esi.nl

P. van de Laar et al. (eds.), *Situation Awareness with Systems of Systems*,
DOI 10.1007/978-1-4614-6230-9_4,

in Sect. 4.3. In Sect. 4.4 screen shots show the demonstrator in action and the results we obtained. Our lessons learned are presented in Sect. 4.5. We end with the conclusion in Sect. 4.6.

4.2 Building the POSEIDON Demonstrator

The demonstrator was targeted to implement the architecture as described in Chap. 3. An architectural view on the flow of information, as depicted in Fig. 4.1, highlights a few architectural constraints and decisions relevant for this chapter. First, data is both pushed and pulled into the system. In particular, the sensors and operators push data into the system, while both the information available on the Internet and results of analysis are pulled into the system. Note that the data can be pulled both periodically, e.g., for incrementally updating the situational picture, and aperiodically, e.g., resulting from a specific query by the operator. Second, the access of stored data is protected by means of access control. Finally, all results of analysis are stored. Hence, all analysis results are available to everyone, with the appropriate access rights, enabling not only visualization but also further analysis.

4.2.1 Component View

The demonstrator contains a wide variety of components as depicted in Fig. 4.2. We will now briefly describe these components.

- **AIS replayer:** A simulator to drive an AIS receiver.
- **AIS receiver:** Sensor that receives AIS messages broadcast by ships and pushes these messages into the system.
- **OpenSplice DDS**[1]**:** A Data Distribution Service for real-time systems.

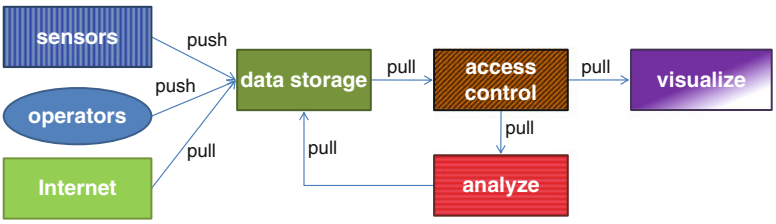


Fig. 4.1 Architectural view on the flow of information

[1] http://www.prismtech.com/opensplice

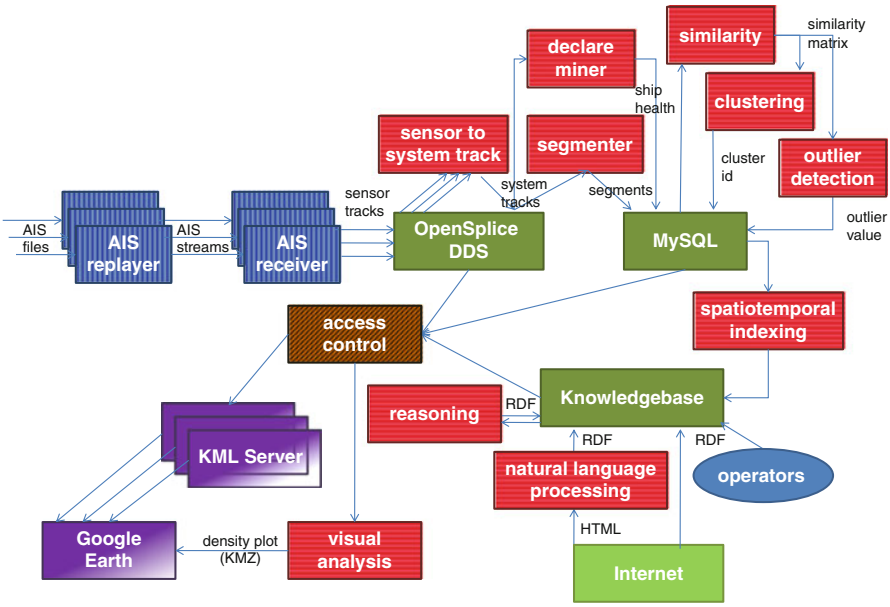


Fig. 4.2 Components in the demonstrator. The same color patterns are used as in Fig. 4.1 to reflect the position of each component in the architecture

- **Sensor to system tracks:** Analyzer that combines the data of all sensors into a consistent view on the objects in the environment of the system.
- **Segmenter:** Analyzer that compresses the system tracks into segments; see Chap. 7 for details.
- **Declare miner:** Analyzer that monitors and compares the current behavior of ships with their desired behavior; see Chap. 9 for details.
- **MySQL:** Data storage designed for relational data. The data storage can be queried using the Structured Query Language.
- **Similarity:** Analyzer that computes the similarity matrix between different ship tracks; see Chap. 7 for details.
- **Clustering:** Analyzer that clusters ships based on the similarity of their tracks.
- **Outlier detection:** Analyzer that detects ships of which the tracks are unlike any other; see Chap. 8 for details.
- **Spatiotemporal indexing:** Analyzer that inserts the segments in the knowledge base to enable reasoning over them.
- **Knowledge base:** Data storage designed for knowledge management. The data storage stores RDF triples and can be queried using Prolog.
- **Operators:** Persons working with the system can push intelligence and pieces of their domain knowledge into the knowledge base.
- **Internet:** Knowledge from external sources connected to the Internet can be pulled into the knowledge base, either directly or after processing.

- **Natural language processing:** Analyzer that transforms human readable text into knowledge in a machine readable structure.
- **Reasoning:** Engine that constructs new knowledge using reasoning on the available knowledge; see Chap. 10 for some examples.
- **Access control:** Access point that ensures that the data is only made available to the appropriate services and persons given the current situation; see Chap. 12 for details.
- **Visual analysis:** Analyzer that visualizes the behavior of ships; see Chap. 5 for details and examples.
- **KML Server:** Server of a situational picture: a geographic information layer that shows a particular view on the current situation based on the information in the system.
- **Google Earth**[2]**:** A virtual globe, map and geographical information program.

When comparing the components in the demonstrator, i.e., Fig. 4.2, with the architectural view on the flow of information, i.e., Fig. 4.1, you might notice that we have deviated from the envisioned architecture. We deviated from the architectural decision that data is always protected by access control for two reasons. First, the number of persons with access control knowledge was limited in the project and they had to perform their own research as well. Second, the performance penalty of access control was considerable. Although the performance could be improved, it was not considered as research topic in the POSEIDON project.

We also deviated from the architectural decision that all data is made available to everyone for pragmatic reasons. In particular, data that was not (envisioned to be) used by others was not stored. For example, the similarity matrix was directly passed on in Matlab to the only two interested components, and performance benefited from the elimination of database accesses to write and read the matrix.

4.2.2 Development and Deployment

Many components of the demonstrator were developed during the POSEIDON project. A few of these components were developed in cooperation, but most were developed by a single partner. Many components were manually developed using a variety of programming languages, like Java, Prolog, and Matlab, yet some components[3] were generated as described in Chap. 11. The source code of

[2] http://earth.google.com

[3] In fact, not only some components but also the communication with the database of some components, i.e., some of the arrows in Fig. 4.2, were generated.

all components was centrally archived using Subversion.[4] Furthermore, all binary dependencies were handled using Maven[5] and Nexus.[6]

The hardware configurations for the demonstrator were not the same at Thales Netherlands, the Embedded Systems Institute, and the universities. For example, at Thales the hardware configuration consisted of a mid-range server and two laptops, whereas at the Embedded Systems Institute the hardware configuration consisted of one high-end server and a single laptop. Consequently, the components had to be deployed on different hardware configurations. The components thus had to have configuration parameters. The values of these configuration parameters were determined by the particular hardware configuration, and influenced both the compilation and the runtime communication.

4.3 Inputs for the POSEIDON Demonstrator

To be able to operate the POSEIDON demonstrator we needed a coherent set of inputs. In this section we describe how we constructed these inputs.

A support system for maritime safety and security not only gets streaming data from sensors, such as radar and AIS receivers, but can also request data from databases and websites, such as ParisMoU[7] and GeoNames.[8] Both kinds of data are time-dependent: sensors measure the evolving world and the content of databases and websites are regularly updated. For reproducibility of the demonstration and the scientific experiments, we decided to access, process, and store the relevant content of databases and websites once, and use the stored information during all demonstrations and scientific experiments. Furthermore, we decided to use a large, fixed data set of sensory data: All AIS messages received during 2007 along the Dutch coast. This real-world data set was made available by MARIN.[9] This data set is not publicly available, but similar data sets can be created by storing the streaming data from one or more AIS receivers or from websites such as AIS Hub,[10] MarineTraffic.com,[11] and vesseltracker.com.[12] Based on this data set the researchers could investigate the behaviors of ships. Figure 5.5 shows for example the behavior during storm and normal weather conditions.

[4] http://subversion.tigris.org
[5] http://maven.apache.org
[6] http://nexus.sonatype.org
[7] http://www.parismou.org
[8] http://www.geonames.org
[9] http://www.marin.nl
[10] http://www.aishub.net/
[11] http://www.marinetraffic.com
[12] http://www.vesseltracker.com

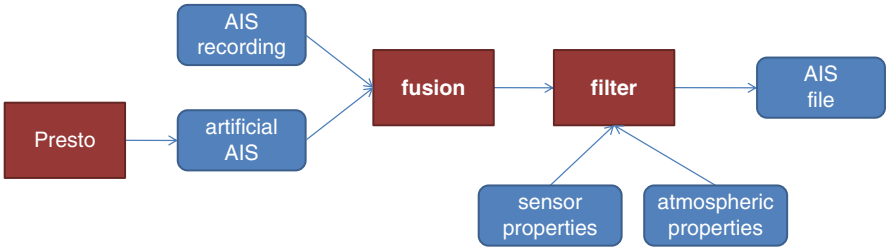


Fig. 4.3 The offline process to create sensory data for each AIS receiver

Although multiple AIS receivers were involved in the data collection process, the provided data set is agnostic to the existence of AIS receivers. In particular, the provided data set contains no information about the AIS receivers that received a particular AIS message. Furthermore, each AIS message occurs only once independent of the number of AIS receivers that received it. For test and demonstration purposes, we regularly needed streaming data from multiple AIS receivers. Since the data set did not contain information about the AIS receivers, we artificially created AIS receiver data by filtering AIS messages based on the distance between the AIS receiver and the ship and the atmospheric conditions. Furthermore, we also needed instances of undesired, suspicious, and illegal behavior. Yet, the data set contained only a limited set of these instances, since fortunately no pirate activities and only a limited number of collisions happened in the North Sea in 2007. Consequently, we had to artificially add behavior to test and demonstrate our algorithms. For this purpose we used Presto [5]. Figure 4.3 summarizes the offline process that we applied to create the sensory data of each AIS receiver for the demonstrator.

4.4 Integrated Results

In this section we present a number of screen shots showing the results of the POSEIDON project, in general, and the POSEIDON demonstrator, in particular. These screen shots illustrate that the results of the POSEIDON project are not only compatible with each other but also reinforce one another in the maritime safety and security domain. Note that all ship names and MMSI & IMO numbers either are fictive or have been removed to protect the privacy of the ships involved.

The right side of Fig. 4.4 highlights the exchange of credentials necessary for access control as described in Chap. 12. The left side of Fig. 4.4 shows ship tracks of about 1 h on the North Sea. The ship tracks are based on segments, see Chap. 7, colored based on the AIS receiver that actually received the AIS messages, and displayed using the adapter techniques described in Chap. 11. In Fig. 4.4, it is clearly visible that AIS messages of ships are received by multiple receivers, both in parallel and sequential, resulting in ship tracks that change color.

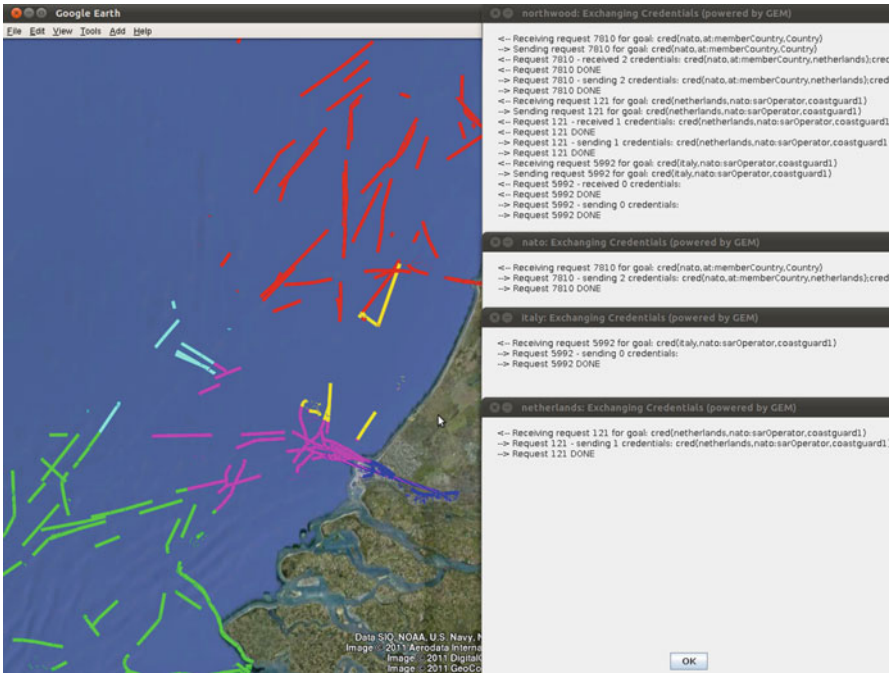


Fig. 4.4 Screen shot of the demonstrator showing the exchange of credentials next to ship tracks received by multiple AIS receivers as a layer in Google Earth (©2011 Google)

Figure 4.5 illustrates a warning for a ship with reduced health. The lower part shows the process mining techniques as described in Chap. 9 that generated the warning. The upper part of Fig. 4.5 shows a view on the current maritime situation: the segments of the ships over a density map. How the density maps and segments are obtained is described in Chaps. 5 and 7, respectively.

Figure 4.6 shows the multiple geographic information layers available in the demonstrator. Many of these layers are developed using the adapter techniques as described in Chap. 11. The layers differ among others in how vessels are grouped, as one can see on the right. Vessels can for example be grouped based on their behavior, i.e., "not under command", "under way sailing", …, "under way using engine", and on their type, i.e., "tanker", "cargo", …, "tug". Currently only two layers are visible: "Segments Ship Health" and "Harbors". The former shows the segments, see Chap. 7, colored based on the ship health, as measured by the techniques described in Chap. 9. The latter shows the different harbors in the Rotterdam area. The harbors are colored based on their kind, such as liquid bulk, dry bulk, general cargo, food, and passenger terminals. This information is obtained from the Port of Rotterdam Authority[13] and stored in the knowledge base. Figure 4.6 also shows in

[13] http://www.portofrotterdam.com/en/Port/port-maps/Pages/branches.aspx

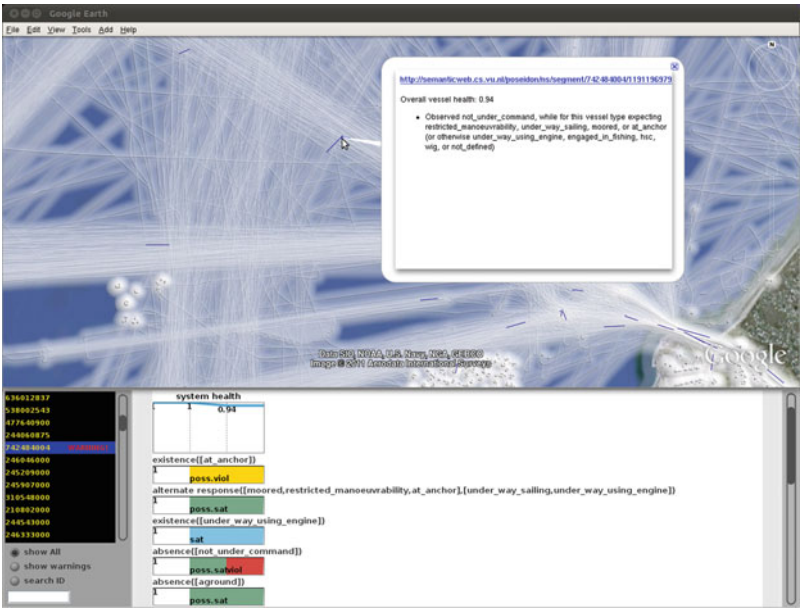


Fig. 4.5 Screen shot showing a warning for a ship with reduced health

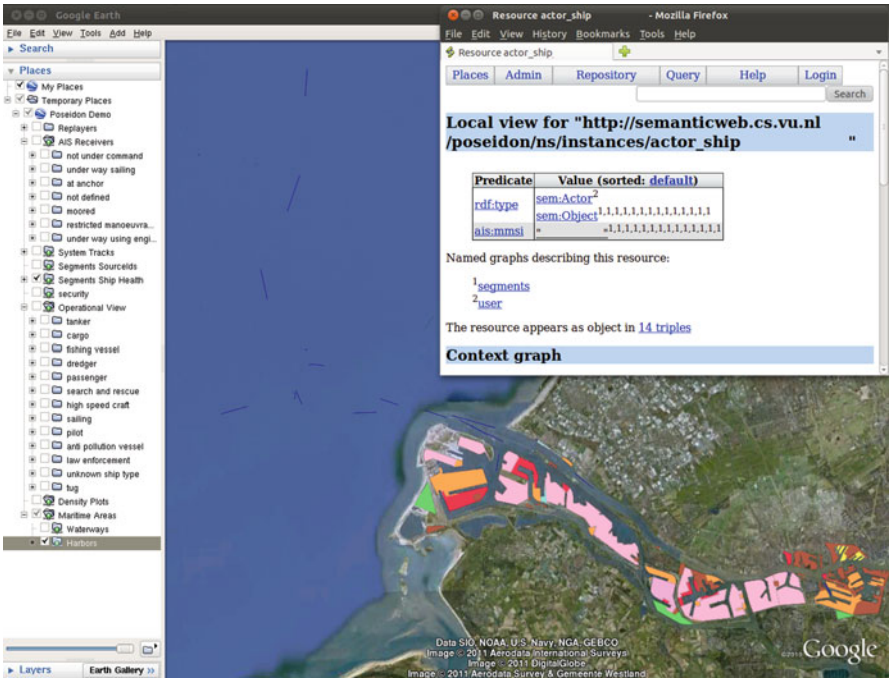


Fig. 4.6 Screen shot showing multiple layers and information obtained from the knowledge base

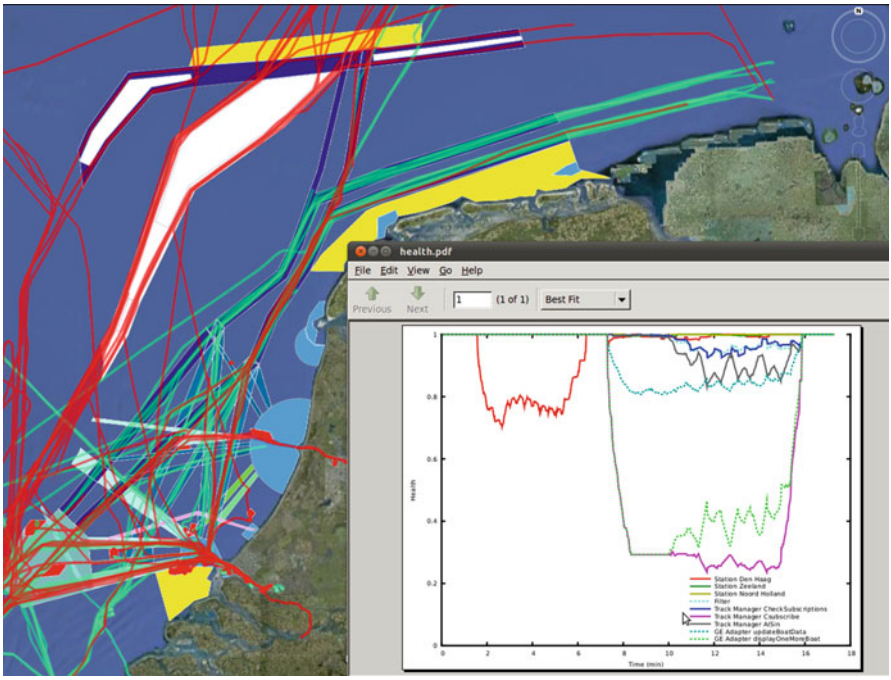


Fig. 4.7 Screen shot showing the demonstrator in action, using multiple layers in Google Earth (©2011 Google), behind a graph showing the health of the demonstrator's constituent components

the upper left corner the amount of information of a particular vessel available in the knowledge base. This information is stored using the Simple Event Model as described in Chap. 10.

Figure 4.7 highlights the health monitoring and runtime fault localization capabilities of the system, as described in Chaps. 14 and 15. The screen shot was made after two artificial errors were subsequently and temporarily injected. Furthermore, test selection was activated to improve the error-localization a few minutes after injecting the second error. The health figure in the lower right corner clearly reflects that two errors were temporarily injected at different locations after one another. Additionally, the impact of test selection on error-localization is clearly visible since only after activation two different locations become separated. Figure 4.7 also shows the density plots, as described in Chap. 5 of heavy special cargo vessels and heavy tankers in green and red, respectively, on top of the different kind of areas as specified by Rijkswaterstaat. Many vessels travel along the blue shipping lanes, but also more than a few violate the traffic rules and sail outside the assigned shipping lanes.

Figure 4.8 shows the analysis of the historic behavior of a ship. The screen shot shows ship tracks where the current location of each ship is indicated by its MMSI number and an icon reflecting its kind. Furthermore, it shows the different

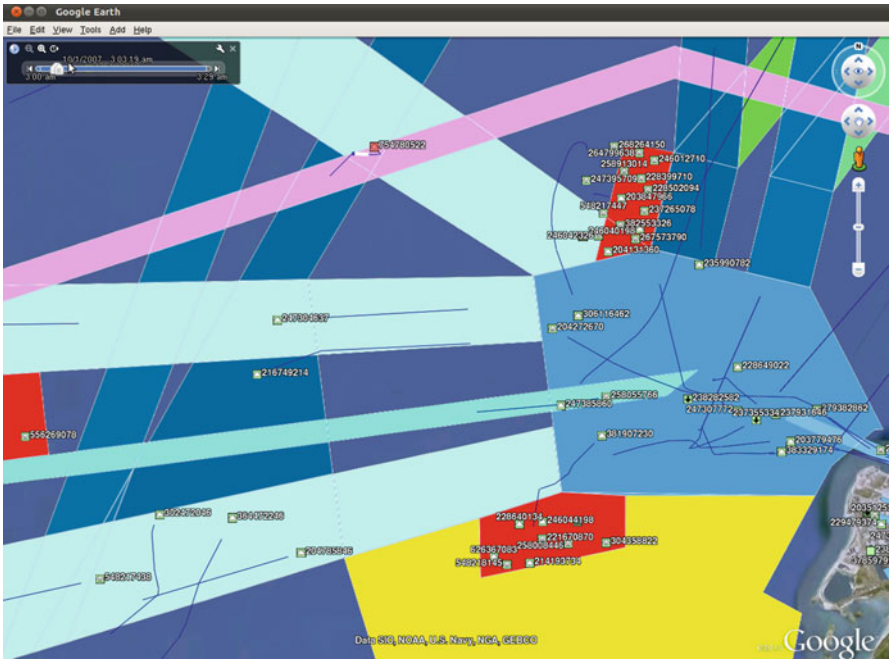


Fig. 4.8 Screen shot showing the analysis of a ship's historic behavior

kind of areas as specified by Rijkswaterstaat. Figure 4.8 clearly shows that many ships travel along the blue shipping lanes, and even more ships lay still in the red anchorage areas. The attention of the operator was drawn to a particular ship by coloring its icon red. Chapters 8 and 9 describe examples of anomaly detection algorithms that can be applied to color the icons. Since the anomalous ship, with MMSI 754780522, is at the intersection of two shipping lanes (in the upper middle part of the screen shot), the operator decided to investigate further. In particular, the operator investigated the historic behavior of this ship by playing back its movements. The screen shot shows that the window of animation of the ship's behavior is between 3:00 a.m. and 3:29 a.m. Currently, we see a thick white segment as part of the ship track that reflects the position of the ship at 3:03:19 a.m.

4.5 Lessons Learned

In this section, we reflect on the process to create the POSEIDON demonstrator and on the demonstrations for the various stakeholders. In each subsection, we present a specific lesson-learned while creating and using the POSEIDON demonstrator.

4.5.1 Applying Best Practices for Integration

The emphasis in the POSEIDON project was on the development of new information processing methods, not on technology and process choices for project realization and demonstration. In the later stages of the project this has caused quite a few well-known integration problems [8]. A more unified choice for implementation technologies for the demonstrator and applying best practices from system and software engineering could have saved much integration efforts. We are aware that in particular cases too rigidly applying rules and guidelines can hamper research, but typically research is also supported by such rules and guidelines. Based on our POSEIDON experience, we recommend that at least:

1. All components have test-suites since component errors not only complicate integration but also threaten the validity of the research results.
2. The dependencies of components are explicitly managed since these dependencies are needed in every context to compile and execute the components.
3. All components have an installation procedure since installation typically happens once and knowledge about installation details is often lost and forgotten.

4.5.2 Reproducibility and Relevance

The quality of research results is largely determined by their reproducibility. When the results cannot be reproduced, as was for example the case with cold fusion,[14] the results are rejected by the scientific community. The relevance of research results is among others determined by the compatibility with other related results and the applicability within relevant application domains. For example, it is well-known that CPU intensive results cannot be applied in the embedded systems domain. The Industry-as-Laboratory paradigm in general and the demonstrator in particular require that the research results obtained in an academic context are reproduced, validated, integrated, and applied in an industrial context. In the POSEIDON project most research results obtained by the academic partners were reproduced at Thales Netherlands and the Embedded Systems Institute. Although all academic partners were aware from the beginning of the project that their results had to be reproduced and they were guided by the architecture as described in Chap. 3, almost all academic partners still largely underestimated the effort needed to reproduce their results in another context. The effort spent to reproduce results of the POSEIDON project mostly targeted non-fundamental, implementation-related problems that could have been prevented as was already discussed in Sect. 4.5.1. Only minor effort was needed to deal with interface problems related to the interaction between the

[14] See for example http://en.wikipedia.org/wiki/Cold_fusion for more information on cold fusion and its rejection by the scientific community.

results. Although a demonstrator does not come for free, it has significant academic and industrial value. A demonstrator proves that the results are compatible with each other and applicable in a particular application domain. Furthermore, it ensures the scientific quality of the results since they are reproduced in another context.

4.5.3 Scenarios and Data Sets

All researchers needed scenarios that highlighted the added value of their algorithms and results. Finding relevant scenarios in the domain that highlight specific algorithms is far from trivial. And even scenarios are not enough to be able to calibrate, test, and demonstrate the algorithms. All researchers also needed appropriate data sets with a large number of sensor data samples and sufficient density of examples relevant for their scenario. We learned that real-world data sets can initially be too complex for researchers to calibrate and test their algorithms. The effort and time needed to clean and simplify real-world data sets can be considerable. Furthermore, we learned that it can take an extensive period before a large data set is acquired. Even worse, coherent data sets of different sensors and sources, such as AIS, radar, unmanned aerial vehicles, and Internet, are needed for information fusion, yet they are typically not available. And since the collection of such coherent data sets is expensive, researchers often have to prove the added value of the combination of sensors and sources by combining incoherent data sets, i.e., data sets that are not captured simultaneously in the same area, to justify the collection of coherent data sets. Finally, often a data set contains too few examples relevant for a scenario, since for example captains of ships do not wish to collide. In that case the data set has to be extended either with examples from other data sets, e.g., by adding collisions of vessels in Singapore to the Rotterdam data set, or with artificial examples, e.g. made by Presto [5].

4.5.4 Simulation Environment

While using the demonstrator, we noticed two limitations of our simulation environment. One, we did not have full control over the synchronization between the replayers. Two, we did not have a central clock. Consequently, repeatability of simulation runs was not perfect and running the simulation faster than real-time resulted in violations of speed rules and confused many anomaly detection algorithms.

Conceptually, the observed limitations can easily be solved. Instead of simulating the AIS receivers independently, they must be simulated dependently. This can be realized by executing the process to create sensory data as depicted in Fig. 4.3 not

offline but online. In other words, the simulation should start at the independent sources of the AIS messages, i.e., the vessels, instead of the dependent AIS receivers.

4.5.5 Dependencies

The POSEIDON project has created a coherent chain of methods for information processing that are mutually interdependent. Some of these methods use known, proven technologies, but most methods are innovative and advance the state of art. Since the results of innovation are not guaranteed, the POSEIDON project was at risk: One failing or non-performing method could break the complete chain. We learned to minimize dependencies, limit innovation in methods where many others depend on, such as infrastructural methods, and to pay sufficient attention to the creation of fallback solutions for each method.

4.5.6 Using Adapters

During test and integration we visualized the output of each processing step using adapters, as described in Chap. 11. Whereas the investment effort was minimal since the adapters were largely generated, the benefits were huge. These adapters enabled us to visually inspect the output of each processing step and thus to quickly pinpoint failing components. The integrators all agree that the adapters reduced the integration time and increased the quality of the POSEIDON demonstrator. We consider visualizing the output of each processing step using adapters a best practice.

4.5.7 Feedback

The demonstrator of the POSEIDON project was a convenient instrument to present many project results. The demonstrator could be used for both technical and non-technical stakeholders. Seeing algorithms working together and augmenting each other in relevant scenarios removes all doubts about compatibility and applicability in the application domain. Furthermore, since the results of the POSEIDON project were clearly positioned and highlighted using demonstrable scenarios in the domain of maritime safety and security, all stakeholders were able to provide valuable feedback directly linked to these scenarios, their domain knowledge, their experience, and their current way of working.

4.6 Conclusion

The POSEIDON project was a prominent example of an Industry-as-Laboratory project, applied to early technology research up to proof-of-concept with further industrial innovation potential. A broad range of advanced scientific disciplines was brought together in close cooperation with each other and with Thales Netherlands as carrying industrial partner. An important result of the POSEIDON project was a demonstrator of the possibilities for an integrated architecture serving maritime safety and security.

In this chapter we discussed the POSEIDON demonstrator. We described how we built, deployed, and operated the demonstrator. In addition, we showed screen shots of the demonstrator that highlighted the results that we obtained. Furthermore, we presented our lessons-learned during the creation and usage of the POSEIDON demonstrator. An important advantage of a demonstrator is that it proves that all results are compatible with each other and are applicable in the application domain. In addition, since the results of individual researchers are reproduced in another context, the results achieved a stronger scientific foundation than regular academic results. Finally, many results of the POSEIDON project could be easily presented to both technical and non-technical stakeholders by the demonstrator. Since the results and the relations between them were clearly highlighted in relevant scenarios for the application domain, all stakeholders were able to provide valuable feedback directly linked to these scenarios, their experience, and their current way-of-working.

Acknowledgements This research has been carried out as a part of the POSEIDON project at Thales under the responsibilities of the Embedded Systems Institute (ESI). This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

We would like to thank all industrial and academic partners of the POSEIDON project for their efforts to realize the POSEIDON demonstrator. We would like to thank Hester van Ouwerkerk, Teade Punter, Jan Tretmans, Michael Borth, and Dave Watts for their useful feedback on an earlier version of this chapter.

References

1. Doornbos R, van Loo S (eds) (2012) From scientific instrument to industrial machine. Coping with architectural stress in embedded systems. SpringerBriefs in electrical and computer engineering. Springer: Dordrecht/London
2. van Hage WR, Malaisé V, van Erp M, Schreiber G (2011) Linked open piracy. K-CAP 2011 Poster Session. http://www.few.vu.nl/~wrvhage/pdf/kcapPoster.pdf
3. Hamberg R, Verriet J (eds) (2012) Automation in warehouse development. Springer, London
4. Heemels M, Muller G (eds) (2006) Boderc: model-based design of high-tech systems. Embedded Systems Institute, Eindhoven
5. Janssens JHM, Hiemstra H, Postma EO (2010) Creating artificial vessel trajectories with Presto. In: 22nd Benelux Conference on Artificial Intelligence (BNAIC 2010), Luxembourg
6. van de Laar P, Punter T (eds) (2011) Views on evolvability of embedded systems. Embedded systems. Springer, Dordrecht/New York

7. Mathijssen R (ed) (2009) Trader: reliability of high-volume consumer products. Embedded Systems Institute, Eindhoven
8. Tretmans J (ed) (2007) Tangram: model-based integration and testing of complex high-tech systems. Embedded Systems Institute, Eindhoven
9. van Engelen R, Voeten J (eds) (2007) Ideals: evolvability of software-intensive high-tech systems. Embedded Systems Institute, Eindhoven

Part II
Situation Awareness

Chapter 5
Visualization of Vessel Traffic

Niels Willems, Roeland Scheepens, Huub van de Wetering, and Jarke J. van Wijk

5.1 Introduction

When something is moving, people would like to know why; they want to be aware of what is happening in the situation around them: *situation awareness*. This principle of situation awareness also applies to vessel traffic, since there are several reasons why certain vessel movements take place. Operators and analysts know, for instance, that a ferry always sails between two harbors. What if at some moment in time it does not? Then a possible threat, called an anomaly, occurs, since the ferry may be hijacked. A surveillance operator should be triggered to investigate this ferry in more detail to find out why the movement changed. One way we take to come to such conclusions about anomalies is that an operator needs to know about normal behavior. In our approach we focus on showing distributions of movements, which serves as a tool to describe and capture normal behavior.

Object movements can be easily tracked with sensors and stored as data files. Each object is tracked with one or more sensors, resulting in a sequence of timestamped records with measurement values for each of the sensors: *a trajectory*. A trajectory requires the records to contain at least the following attributes: a timestamp, a position, and an identification. For given movement data a user would like to find reasons for certain movements in the trajectories. Often it is difficult to formally define what movement patterns a user is looking for, because they are often

N. Willems (✉)
Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands

Current affiliation: SynerScope BV, Eindhoven, The Netherlands
e-mail: niels.willems@synerscope.com

R. Scheepens • H. van de Wetering • J.J. van Wijk
Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands
e-mail: R.J.Scheepens@tue.nl; H.v.d.Wetering@tue.nl; J.J.v.Wijk@tue.nl

P. van de Laar et al. (eds.), *Situation Awareness with Systems of Systems*,
DOI 10.1007/978-1-4614-6230-9_5,

complex. Therefore, we propose to use visualizations, so that we can use the human visual system, which is capable to identify inexact patterns.

In our visualization research we are driven by two factors, which are typical movement data in general and specific for vessel traffic data. The first factor is that users need to be able to interactively analyze large amounts of trajectories. Over the last couple of years, the tendency is that movement data are steadily increasing in size. This is due to the availability of cheap location trackers, such as Global Positioning System (GPS) devices, which makes it easy to track movements for many objects. The challenge of these large data sets is that we should visualize them fast, for interactive usage, and choose a suitable presentation, since the number of records is comparable or larger than the amount of pixels of a monitor. For the presentation we have chosen to use density, which shows a distribution by summing up a characteristic of trajectories in the neighborhood of some point at the screen. Each and every trajectory contributes to the distribution, therefore density is robust to large data sizes. Density computations are rather expensive, but by using modern graphics hardware we are able to execute these computations in real-time.

The second factor in our research concerns the fact that vessel traffic movements are captured with different sensors and as a result, a user knows many attributes in a trajectory at any moment in time, such as time, position, identification, draught, destination and so forth: *a multivariate trajectory*. For object tracking we expect this trend to proceed, since for many types of objects we could add more sensors or join additional information from other data sources [10] to be used in the analysis. The multivariate aspect is interesting, since often similar behavior is captured with similar attribute values, however it is not known what the best way is to show these patterns in the data. To cope with this the user may be supported by using a rapid prototyping environment. The above two factors are not specific for vessel traffic, but apply for many other kind of objects that move around, for instance, cars, airplanes, animals, and people in a secured area, such as an airport. Current technology, both academic and industrial, is still in its pioneering phase when it comes to the analysis of data with these two characteristics.

We present in this chapter a showcase of some highlights of our research [8], which is conducted in a prototype driven approach: a concept is developed based on user requirements, implemented in a software prototype, and evaluated with users [11]. This chapter is structured as follows: In Sect. 5.2 we describe our density approaches. We describe a number of practical use cases in Sect. 5.3 and we end this chapter with conclusions and suggestions for future work.

5.2 Density Approaches

We present a number of density visualizations to display large amounts of multivariate trajectories. We distinguish between a density field, which is a distribution represented with a grid with values, and density maps, which are the pictures of

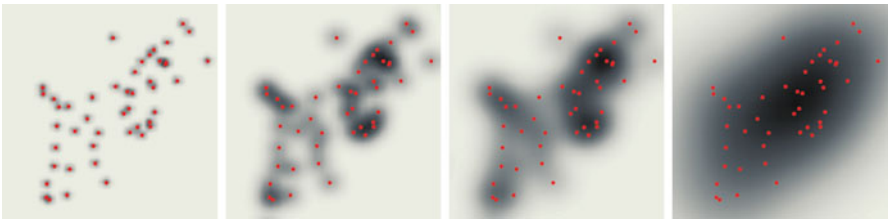


Fig. 5.1 Kernel density estimation for a set of points shown as solid points. From left to right the bandwidth is increased (0.05, 0.15, 0.25, 0.75) which results in a transition from details for a small bandwidth to an overview for a large bandwidth

density fields, for instance the values of a density field shown with a color map. A value at some point of a density field shows the summed up contributions of a set of trajectories in the neighborhood of that point. This procedure is called smoothing and shows a distribution of the trajectories as we describe in Sect. 5.2.1 in more detail. The difference between the density visualization approaches is the usage of these density fields allowing to show different classes of movements. The user can manipulate a density field or compute a number of density fields and combine them in a single image, a density map. In Sect. 5.2.2 we discuss the different visualization approaches.

5.2.1 Density Fields for Trajectories

A density field shows a distribution of smoothed trajectories, which aims at showing normal behavior to an operator. Smoothing is a well-known technique for points, called kernel density estimation [7], where data points are smoothed to see spatial trends by means of their distribution as shown in Fig. 5.1. This process is also known as convolution in image analysis and signal processing. Kernel density estimation averages a value at a point weighted with its neighborhood using a so called kernel, which is a function that gives these weights in the neighborhood. Often the weights are chosen in such a way that points that are further away contribute less. The kernel has a bandwidth parameter, which is related to the kernel size; the larger the kernel the smoother the data. All averaged values are summed up in a density field. This method excels to represent many points, since it preserves the number of occurrences of points at a certain location. Density solves a problem that is called overdrawing, which would happen if these points would have been drawn with a simple black dot. For a large number of points we would end up with a black area, even if the distribution of points is not uniform.

Trajectories are lists of records of which the positions can be connected, since the movement is continuous over time. Therefore, we should smooth line segments instead of points to smooth the movement that is not captured in the data. There are

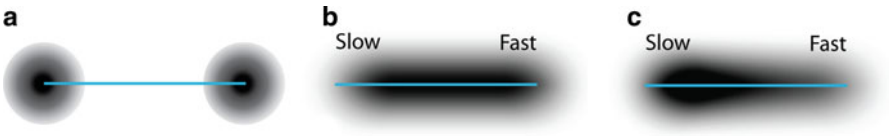


Fig. 5.2 Convolution of a trajectory with a slow starting point and a fast ending point. (**a**) Point convolution. Line convolution at (**b**) constant speed with the average velocity give in the end points and (**c**) accelerated by interpolating using the velocities given in the end points

several approaches to do this as shown in Fig. 5.2. We can treat the end points of the line segment as points and smooth them (Fig. 5.2a), however undesirable holes will occur in the smoothing of long line segments. Hurter et al. [1] used this approach to analyze air traffic. In the second approach, we may know the speed of the object in the end points of the trajectory segment. We can move a kernel along a line segment with the average speed given in the end points to obtain a distribution without holes (Fig. 5.2b). Lampe et al. [3] used a similar approach. In the third approach, we propose to use the speed in both end points to show speed variations by moving the kernel with the speed according to the speed of the object (Fig. 5.2c), resulting in a time-weighted density. The physical analog of this process occurs when you draw lines with a fat pen that continuously releases ink. In places where the pen moves fast there is little ink and in slow places the ink accumulates. The ink level is in our case represented with density values and the pen tip is the kernel, which has non-uniform weights, typically in a Gaussian shape. This type of density has as advantage that we can accurately represent the acceleration of an object with only two points, while in the second approach (Fig. 5.2b) we need to put many intermediate points, and even then there is no smooth transition between two line segments.

Often trajectory data consist of many records, and hence many line segments. Each line segment contributes to the density with an area relative to the length of the segment and the kernel size, which makes it expensive to compute these contributions. Due to the independence of the computation between various line segments, it is possible to use parallelism in the implementation. In graphics we often use graphics cards or a Graphics Processing Unit (GPU), which are available for a standard personal computer and can compute programs in parallel due to massive parallelism ($>$100 cores). We have implemented this approach on graphics hardware as well. By including velocity, the quality of our density is higher compared to competitors, such as Hurter et al. [1] and Lampe et al. [3], but to do so we have slightly sacrificed the performance. We use our approach to generate *density fields* for trajectories, which in turn are used in a number of different ways as described in the next section.

5.2.2 Density Visualizations

For trajectories we introduce three different usages of the density fields, as proposed in the previous section. The approaches are called *vessel density*, *density maps*, and *composite density maps*. Their architectures are shown in Fig. 5.3. Note that at a meta level, three parts called "Data", "Density model", and "Visualization" are reoccurring in each architecture. These parts are in essence the same. In the data part the trajectory data are organized, in the density model part the density fields are computed, and in the visualization part the density fields are displayed as an image in a density map.

The first approach is called *vessel density* [9], which shows two important features for vessel traffic: anchor zones (stopping areas) and sea lanes (common routes). These features popup in a density field, however they appear for different kernel sizes. Figure 5.1 illustrates this principle: for different kernel sizes different features are revealed; a small kernel shows details, while a large kernel shows an overview of the locations with many points in the neighborhood. The stopping areas are visible for a small kernel size and the routes are visible for a large kernel size. This inspired us to show two density fields with all trajectories simultaneously to see both features together. After computing the two density fields as shown in Fig. 5.3a, we use two different visual cues, namely color and gray values, that can be composed in a single image and are still easy to distinguish. The color is used to show the large kernel density field using a color map, for instance a yellow-to-red color map with yellow for low density values and red for high density values. The gray values are used for showing the sum of both density fields as a 3D terrain map, which is illuminated with a light source. The reflected light intensity corresponds with the gray values. As a result we see global structures of trajectories in the colors, such as sea lanes, and details in the gray values, such as anchor zones. We demonstrate this approach with real data in the use case in Sect. 5.3.1.

The second approach is a generalization of vessel density, called *density map* [5,6] as shown in Fig. 5.3b. Common attribute values in the trajectories may indicate similar behavior, which helps to give reasons why certain movements have occurred. For instance, for ship types we can see that passenger ships take different routes than cargo ships, since dangerous cargo ships must keep distance from the coast. To find common attribute values, filtering on the attributes of the data is used to create subsets of trajectories. This results in multiple density fields that are combined in a single image. Per density field the kernel size (amount of smoothing) and kernel weight (a scale factor for the density) can be adjusted. The operator can choose to combine the values of the density fields with density aggregation and make a density map from this single density field (solid lines in Fig. 5.3b). The other route the operator can choose is to create images from the individual density fields and combine the colors of these images together in a single density map (image composition) (solid and dashed lines in Fig. 5.3b). For both density aggregation and image composition a number of variations are shown in Fig. 5.3b. This approach is demonstrated with real-world data in the use cases in Sects. 5.3.2 and 5.3.3.

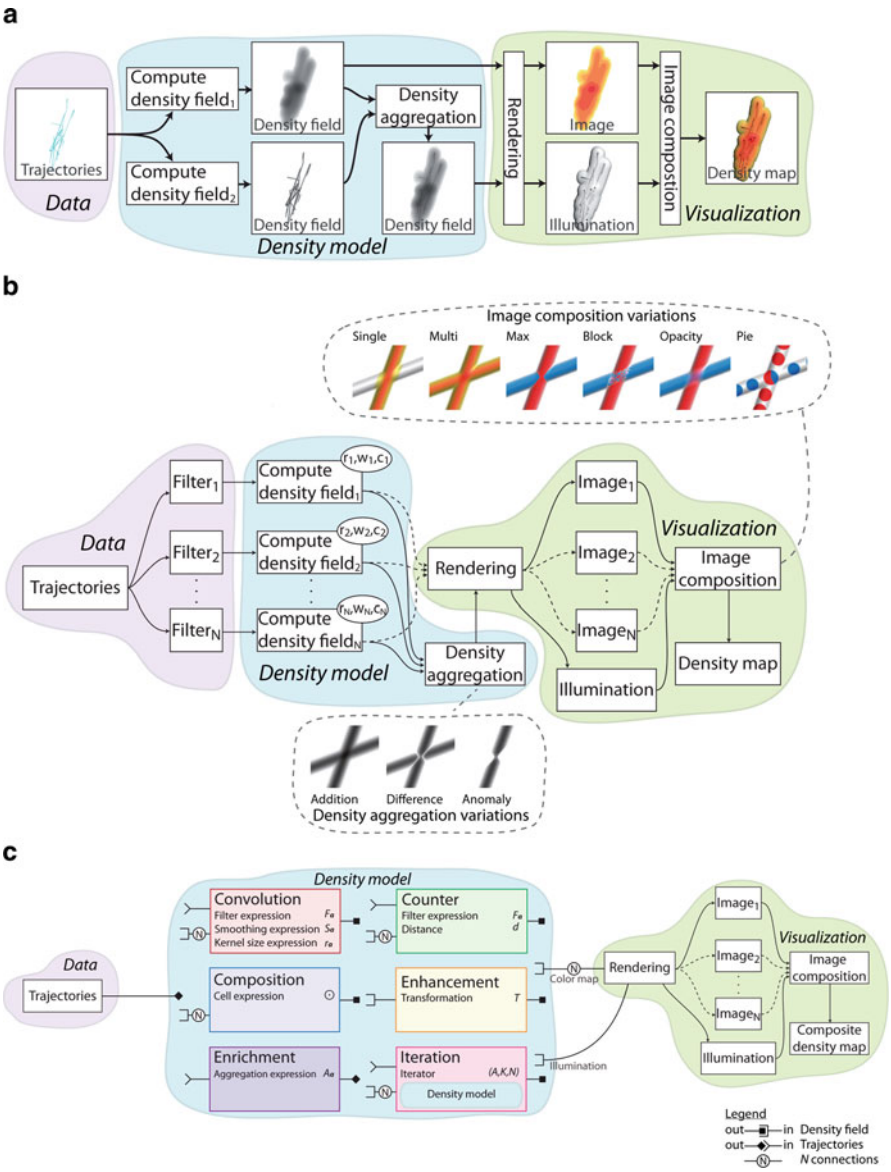


Fig. 5.3 Architectures of (**a**) vessel density, (**b**) density maps, and (**c**) composite density maps

The third and last approach allows the user to define the way how density fields are computed with a number of predefined blocks: *composite density maps* [4] as displayed in Fig. 5.3c. Composite density maps are a generalization of density maps, since often filtering is not sufficient to express specific movement features

based on experience and domain knowledge. There are six types of blocks for creating, composing, and enhancing density fields and trajectories: *Convolution*, *Composition*, *Counter*, *Enhancement*, *Enrichment*, and *Iteration*. These blocks can be connected in any way the user wants, as long as the input and output types (e.g., density fields or trajectories) between the connections are the same. The blocks are configurable with expressions that depend on the attributes in the data. The density fields resulting from the blocks can be connected to the visual cues of color and gray values. Examples of composite density maps with real-world data are shown in Sects. 5.3.4 and 5.3.5.

5.3 Examples Using Density Approaches

In Sect. 5.2 we have given a brief overview of the different approaches we propose to get insights in trajectory data. This section shows the viability of the different solutions using maritime use cases with real-world vessel traffic data obtained from the Automatic Identification System (AIS) [2].

5.3.1 Vessel Density: Weather Conditions

The base case is an investigation of shipping movement during calm weather as shown in Fig. 5.4 on a country-size scale, with all shipping movements in front of the Dutch coast. We see that three North-South sea lanes appear in the orange colored regions. Anchor zones pop up as highlighted dots in the individual trajectories as shown in the insets. These dots appear as a result of the density field computation; since the kernel is moved slowly in these areas, the density values become high. Furthermore, some maintenance vessels move slowly in a small area, typically around an oil platform or a wind mill park, which can be observed by intense individual trajectories. Lastly, in the South the ferry between Vlissingen and Breskens yields for regular traffic which can be observed by intense dots just before the sea lane.

In Fig. 5.5a we show a close up of the entrance of Rotterdam harbor during good weather, where the two anchor zones catch the eye. Shipping movements differ during a gale. In Fig. 5.5b, a North-West gale of force 8 on the scale of Beaufort appears during the day. In general, there are fewer movements and sea lanes are less used compared to Fig. 5.5a. Captains try to keep course, but from the twisty individual trajectories we see that they drift slightly. Furthermore, anchor zones are less often used, and vessels slowly sail along the wind towards the anchor zones to avoid rolling, as can be observed by intense left-top to right-bottom trajectories.

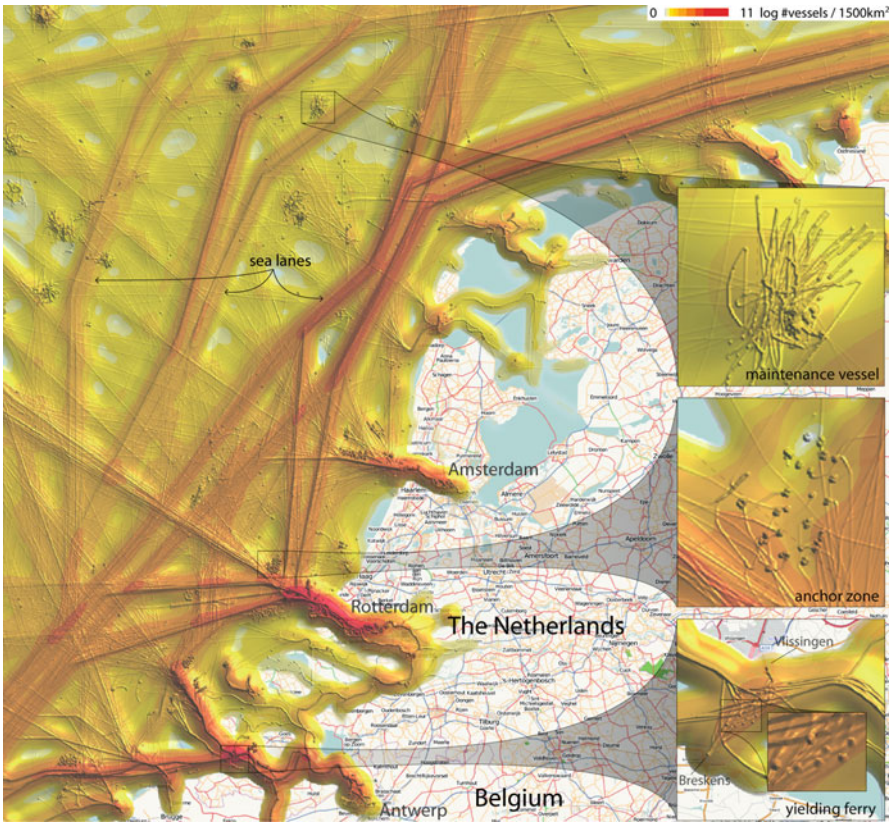


Fig. 5.4 Vessel density of the first week of June 2007 of vessel traffic in front of the Dutch coast. This map represents 3.5 GB of sensor data consisting of approximately 1,500 vessels

5.3.2 Density Map: Temporal Aggregation

In vessel density, the order in which movements take place is lost, since all trajectories are convolved with the same smoothing kernel. By using the time as an attribute the user can distinguish between various moments over time. With the density maps, the user can vary the kernel radius and kernel weight (a scalar multiplied with density value) for each density field. Figure 5.6a shows a single day of vessel traffic in front of Rotterdam harbor with four subsets of 6 h, resulting in four density fields. During the day, the kernel radius is decreasing, while the weights are increasing, and by using the Addition as density aggregation variant (see density aggregation variants in Fig. 5.3b) the various moments in time are distinguishable. The resulting density map consists of a rendering in both color and gray values of a single aggregated density field using the Multi variant for image composition (see image composition variants in Fig. 5.3b).

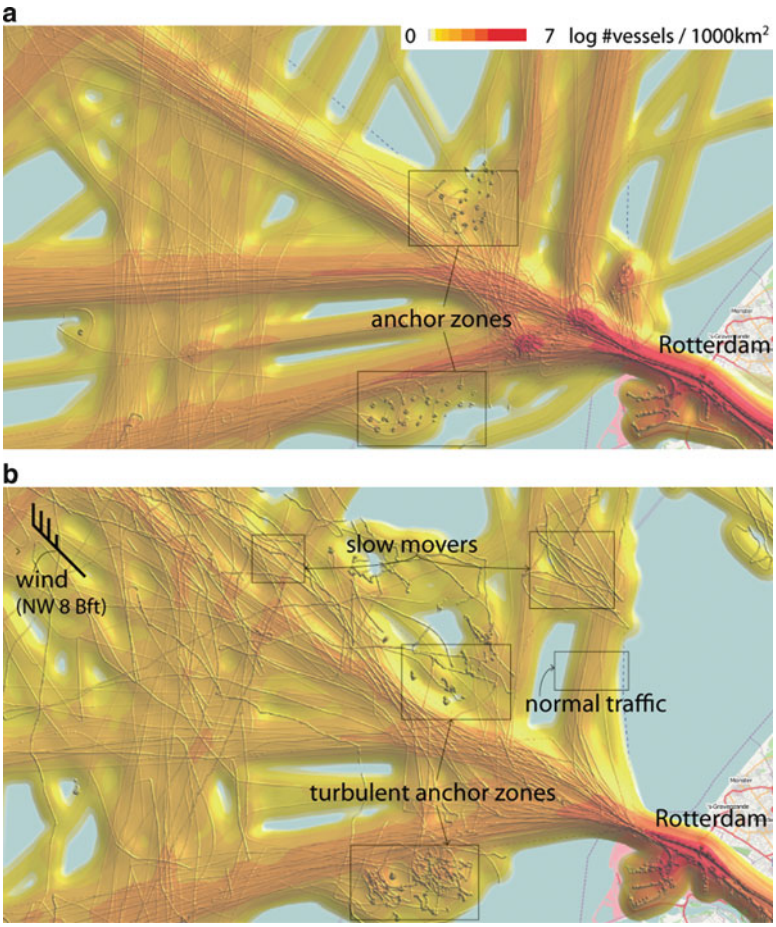


Fig. 5.5 Vessel density of 1 day of vessel traffic at the entrance of Rotterdam: (**a**) a day in June 2007 with smooth weather, (**b**) a day in November 2007 with stormy weather

In Fig. 5.6a we see that the subset in the evening, shown with small and dark trajectories are highlighted, while the others serve as a context. We can also show variations over time using the Max variant for image composition as displayed in Fig. 5.6b. This has as an advantage that the operator can choose intuitive colors, like the chosen daylight colors, which makes it more easy to interpret the picture. For instance, in the encircled area it is instantly clear in the yellow-blue version that a ship went to this location during the night hours and stayed there during the day.

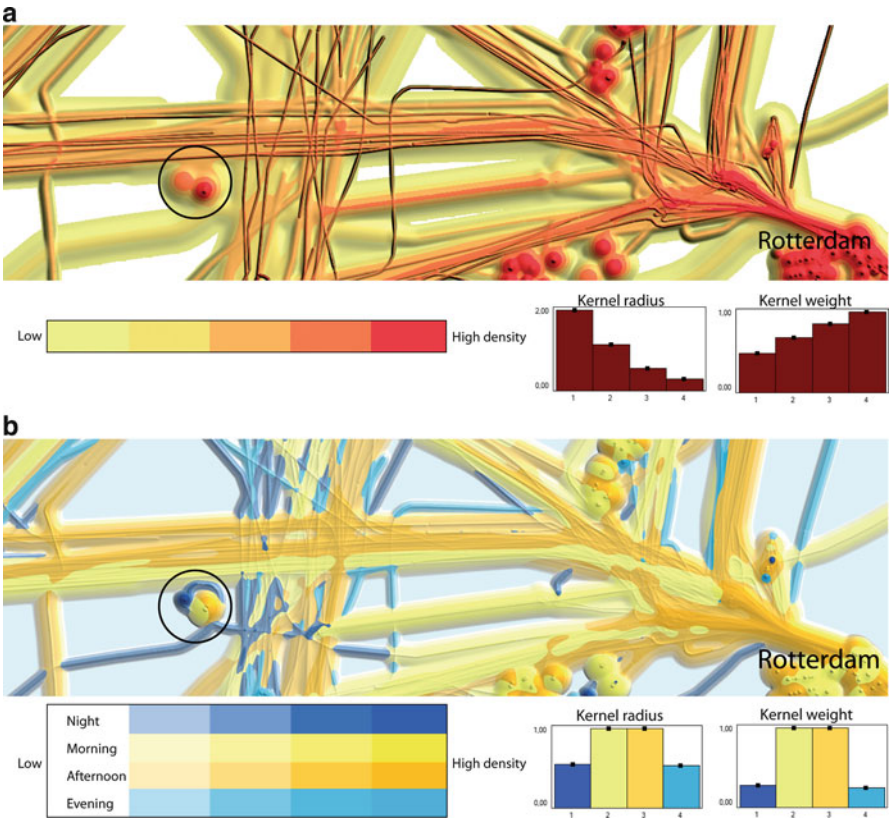


Fig. 5.6 An aggregation of four density maps each covering 6 h of a day starting at midnight, using (**a**) the Addition variant for density aggregation and (**b**) the Max variant for image composition. In both pictures the kernel radius and kernel weight is adjusted for each density field as shown in the bar charts

5.3.3 Density Map: Anomaly Detection

Density maps can be used to show anomalously behaving vessels. A density field of a large amount of vessels can represent the nautical history in an area, indicating, for instance, which movements are usual. By comparing other trajectories with this density field, the user can determine abnormal behavior in areas where the density field values are low. Figure 5.7 shows the result of the Anomaly variant for density aggregation of two density fields: one with 6 days of data between Amsterdam and Scheveningen indicating normal movements and one with the traffic of the last 2 h. For the latter one, the kernel radius is decreasing backwards in time, i.e., the head is the most current position. The resulting density field is shown with the Single variant for image composition and displays potential anomalies in color from white (none) and green (low) to red (high) in the context of both the live and historical data

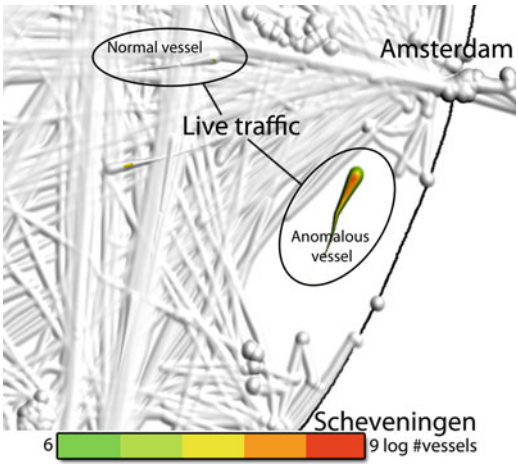


Fig. 5.7 Anomaly detection using a density map with one density field containing 6 days of vessel traffic representing normal movements and one density field containing live traffic with a trail of 2 h. Using the anomaly variant for density aggregation live anomalous ships are highlight in areas where normally no vessel occurs. A vessel sailing between Amsterdam and Scheveningen is marked as anomalous, since normally no single vessel sails in this area

shown in the shading. This example shows how density-based anomaly detection can be used in a real-time system.

5.3.4 Composite Density Map: Drifters

A dangerous situation may arise when a vessel has an engine failure, becomes uncontrollable, and starts drifting. With composite density maps we can visualize potential drifters. A vessel is said to be drifting when it is moving slowly (i.e., for velocity $v(t)$ between 3 and 5 knot) and its course $c(t)$ and actual orientation $h(t)$ have a significant difference (i.e., more than 30°) as depicted in Fig. 5.8a. These specific values may be tuned by operators using user interface components.

To visualize potential drifters, the user can create a block diagram as shown in Fig. 5.8b. We use one convolution block to compute the drifters. The input of this block is filtered with a filter expression $F_\alpha \equiv 3 \leq v(t) \leq 5$ to select only slow movers (see convolution block in Fig. 5.3c). The density values ρ are scaled with the difference between the course $c(t)$ and actual orientation $h(t)$ at some moment in time t. This can be captured in the following smoothing expression $S_\alpha(\rho, t) = \rho|c(t) - h(t)|$ for the kernel weight to configure the convolution block. The resulting density field is connected to the color. For contextual information, we also create a density field for the entire set of trajectories and connect it to the visual cue with gray values. This reveals a number of potentially dangerous situations. In Fig. 5.8c, we show an overview of the drifters we find with this block diagram and zoom in on several particular cases in Fig. 5.8d–g.

In Fig. 5.8d we can see a vessel with drifting patterns around its turns. Closer investigation, by selecting the specific vessel with the mouse, shows that this is a research vessel which is expected to make sharp turns, causing a ship to

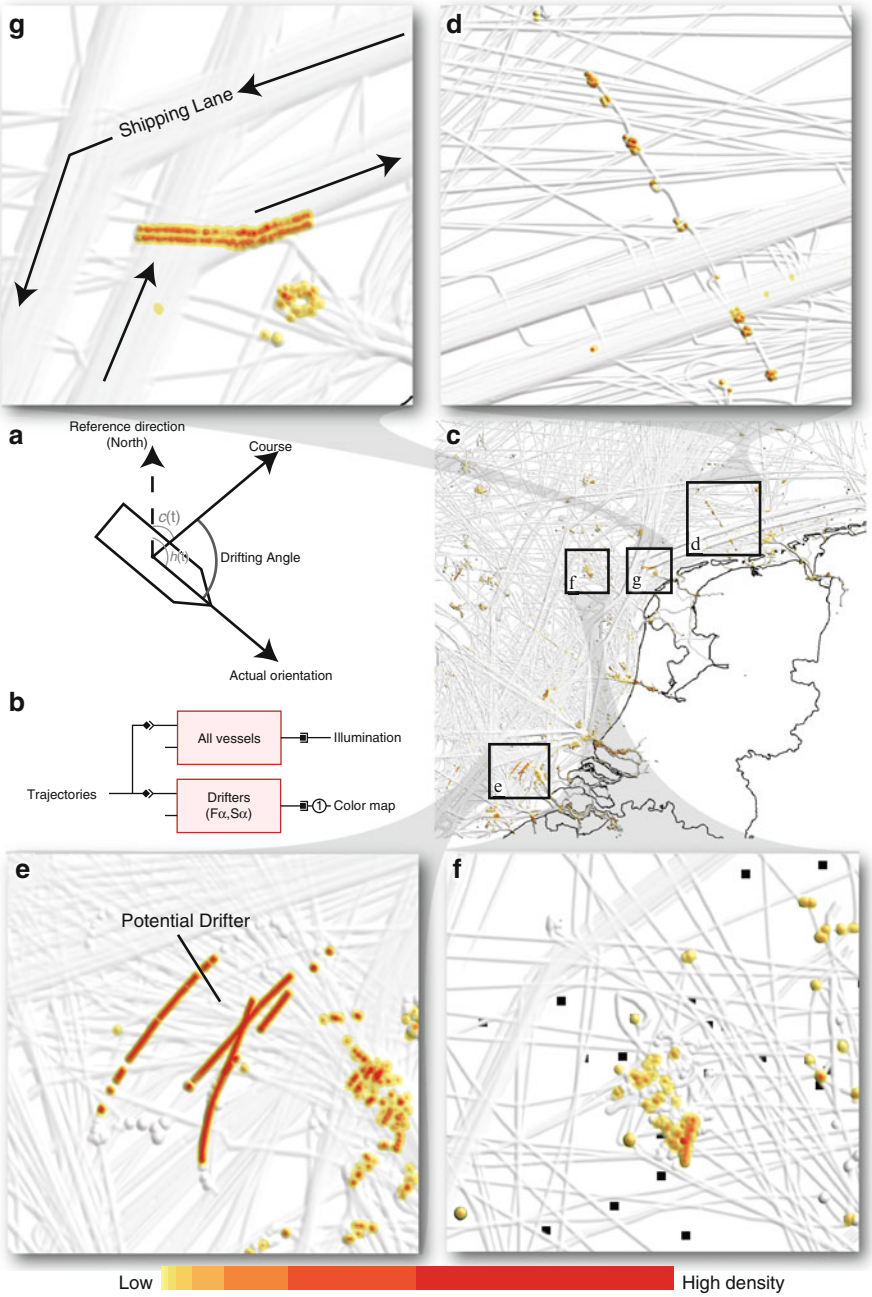


Fig. 5.8 (**a**) The course $c(t)$ and actual orientation $h(t)$ of a vessel. (**b**) The block diagram for drifters. (**c**) An overview of all drifter movements shown in color with all data shown in the illuminated height field. In (**d**) to (**g**) several close ups of unusual drifter patterns are shown

drift, since ships do not have enough resistance in the water. In Fig. 5.8e, we see potential drifters in the areas shown in red. By selecting the ships, we notice that these trajectories are caused by a single cargo vessel that originated from the North, loitered in the area for approximately 2 days and then proceeded into the harbor of Antwerp. It is possible that the vessel was too early and had to wait 2 days before it could head to its destination harbor. This vessel has most likely been drifting on several occasions while it was waiting. The vessel appears to alternate between drifting in a South-West direction and actively moving in a North-East direction. In Fig. 5.8f cargo vessels move between ocean platforms, which are visualized as black squares. These vessels are most likely resupplying ocean platforms and making short stops and sharp turns. In Fig. 5.8g we see two parallel drifter trajectories. Since these trajectories occur within a busy area, this may be a dangerous situation. Closer investigation, by selecting the vessel, reveals that these drifter patterns are caused by a research vessel.

5.3.5 Composite Density Map: Sea Lanes

We separate the sea lanes and other frequently used routes from other movements to get an overview of normal shipping traffic (see Fig. 5.9). Sea lanes are defined by maritime authorities in busy areas to guide large vessels. In the North Sea, several sea lanes are in effect, but they are not mandatory for all vessels. For extracting the sea lanes from the data, we partition the data into eight subsets, sectors η, each representing a course range. These sectors are needed to capture crossing traffic. For each sector, we compute a density field and sum the resulting fields into the final field shown in Fig. 5.9b. The partial lanes are masked with a thresholded smoothed count field to obtain only busy lanes. We filter out all movements with a course that differs from the average course by at least some constant using an average course field. The 'Average course' field is computed using the cell-wise division of two density fields: a density field 'Convolved course' using the smoothing expression $S_\alpha(\rho,t) = \rho c(t)$ and a density field called 'Convolved duration' with $S_\alpha(\rho,t) = \max(\rho,\varepsilon)$, for a small $\varepsilon > 0$ to avoid divisions by zero. The sea lanes appear to disintegrate in the marked areas in Fig. 5.9b. This happens where vessel directions start diverging or where there are simply not enough trajectories to extract a reliable route as shown in the encircled area for a little used sea lane. This example shows that we can compute complex features in a structured way, without any need for further programming, allowing for the required rapid prototyping.

5.4 Conclusion and Future Work

We have shown three different approaches to visualize multivariate trajectories with density. These approaches help us to understand why movements have occurred. The *vessel density* approach shows basic maritime features, while *density maps*

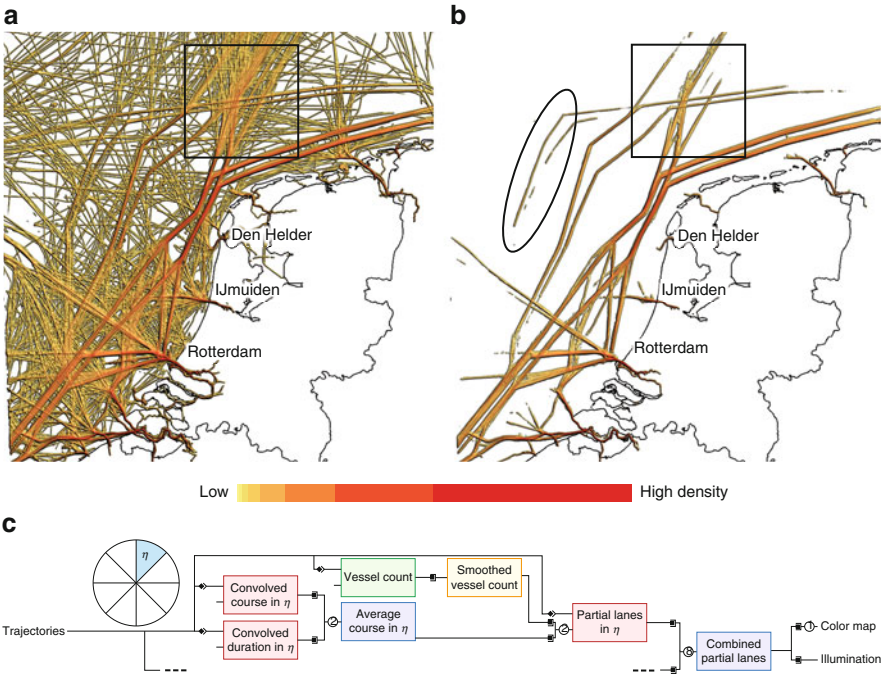


Fig. 5.9 A density map (**a**) containing all movements, (**b**) containing only movements over sea lanes. (**c**) The block diagram for extracting shipping lanes

can incorporate more attributes, to show trajectories with common attribute values. Finally, with *composite density maps* operators are able to define their own density computation with a block diagram using their domain knowledge.

Generally, in the visual analysis of moving objects there is still an open problem to be solved. There is still no simple drawing technique to show the dynamics of movements to explain how objects have moved, both individual and multiple objects. The next step in our work is to show uncertainty in the acquisition of the trajectories, since the data is often obtained from sources that are not fully reliable: the transmission can be poor due to weather conditions, some sources are always slightly unreliable, like some web sources about ships, or the objects simply turn off their AIS signal to hide themselves and get the possibility to make illegal moves. Users need to be informed about the uncertainty in the data to come to well-founded decisions based on the visualized situation.

Acknowledgements This research has been carried out as a part of the POSEIDON project at Thales under the responsibilities of the Embedded Systems Institute (ESI). This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

We thank our industrial partners Thales and Noldus IT, and the management and research fellows at the Embedded Systems Institute. We thank our domain experts at the Dutch Maritime

Research Institute (MARIN). We thank the Fraunhofer Institute IAIS and the partners in the POSEIDON project in the preparation of this chapter. Maps in the figures are provided by OpenStreetMaps and National Geospatial-Intelligence Agency.

References

1. Hurter C, Tissoires B, Conversy S (2010) Accumulation as a tool for efficient visualization of geographical and temporal data. In: Proceedings of the AGILE workshop geospatial visual analytics: focus on time. Guimarães
2. International Telecommunications Union (2001) Technical characteristics for a universal shipborne automatic identification system using time division multiple access in the VHF maritime mobile band, Recommendation ITU-R M.1371-1
3. Lampe OD, Hauser H (2011) Interactive visualization of streaming data with kernel density estimation. In: IEEE Pacific visualization symposium, Hong-Kong, pp 171–178
4. Scheepens R, Willems N, van de Wetering H, Andrienko G, Andrienko N, van Wijk JJ (2011) Composite density maps for multivariate trajectories. IEEE Trans Vis Comput Graph 17(12):2518–2527
5. Scheepens R, Willems N, van de Wetering H, van Wijk JJ (2011) Interactive visualization of multivariate trajectory data with density maps. In: IEEE Pacific visualization symposium, Hong-Kong (PacificVis 2011), pp 147–154
6. Scheepens R, Willems N, van de Wetering H, van Wijk JJ (2012) Interactive density maps for moving objects. IEEE Comput Graph Appl 32(1):56–66
7. Silverman BW (1992) Density estimation for statistics and data analysis. Monographs on statistics and applied probability no. 26. Chapman & Hall, London
8. Willems N (2011) Visualization of vessel traffic. PhD thesis, Eindhoven University of Technology. http://alexandria.tue.nl/extra2/719764.pdf
9. Willems N, van de Wetering H, Wijk JJ (2009) Visualization of vessel movements. Comput Graph Forum 28(3):959–966. Proceedings of EuroVis 2009
10. Willems N, van Hage WR, de Vries G, Janssens JHM, Malaisé V (2010) An integrated approach for visual analysis of a multisource moving objects knowledge base. Int J Geogr Inf Sci 24(10):1543–1558
11. Willems N, van de Wetering H, Wijk JJ (2011) Evaluation of the visibility of vessel movement features in trajectory visualizations. Comput Graph Forum 30(3):801–810. Proceedings of EuroVis 2011

Chapter 6
Extending Track Analysis from Animals in the Lab to Moving Objects Anywhere

Wil van Dommelen, Piërre van de Laar, and Lucas P.J.J. Noldus

6.1 Introduction

Tracking and analysis of vessel movements are important tasks in the domain of maritime safety and security. Tracking answers the question which vessel is where and when, and track analysis may answer the question why. Also in many other domains tracking and analysis of object movements is a core activity. Examples include car traffic management and congestion control, air traffic management, products and parts being transported in a warehouse and in an assembly plant, people in a supermarket and retirement home, crowd monitoring for public safety and security in restricted areas such as airports and soccer stadiums, animals seasonally moving around the globe, such as whales, reindeer, and migrating birds, and animals moving in a confined area such as a cage and an aquarium.

Noldus Information Technology with headquarters in Wageningen, the Netherlands is a medium sized private company that develops and sells tooling for human and animal behavioral researchers. One of the products is EthoVision XT,[1] a tool that automatically records and analyzes tracks of animals in an enclosure, such as a cage, petri dish or aquarium, when observed using top-view video. Because tracking with a single video camera is limited to small-scale indoor applications, Noldus

[1] www.noldus.com/ethovision

W. van Dommelen (✉) • L.P.J.J. Noldus
Noldus Information Technology BV, Nieuwe Kanaal 5, P.O. Box 268,
6700 AG Wageningen, The Netherlands
e-mail: W.van.Dommelen@Noldus.nl; L.Noldus@Noldus.nl

P. van de Laar
Embedded Systems Institute, Eindhoven, The Netherlands
e-mail: pierre.van.de.laar@esi.nl

P. van de Laar et al. (eds.), *Situation Awareness with Systems of Systems*,
DOI 10.1007/978-1-4614-6230-9_6,

wishes to extend its portfolio. In particular, Noldus wishes to track and analyze objects anywhere, both indoor and outdoor.

In this chapter we describe a case study in which two different tracking domains, viz. animal tracking and vessel tracking, were compared to learn the diversity in tracking and analysis requirements. The case study investigated whether EthoVision XT, a tool for video tracking and analysis of animals in a laboratory setting, can be adapted and applied to the tracking and analysis of vessels at sea. More specifically, the case study used EthoVision XT to detect vessels speeding in zones reserved for anchored vessels where speeding is not allowed.

This chapter is structured as follows: In Sect. 6.2 we describe track analysis in general and EthoVision XT in particular. Our case study that finds speeders in anchoring zones is described in Sect. 6.3. Section 6.4 describes the differences that we observed between tracking animals in the laboratory and vessels at sea. We end this chapter with a summary and future work in Sect. 6.5.

6.2 Track Analysis

In this section we start with describing track analysis in general. Next, we focus on EthoVision XT, a Noldus product for video-based track analysis. We end with the similarities between animal and vessel tracking that justify the investigation whether EthoVision XT could be extended and applied beyond the laboratory.

6.2.1 Track Analysis in Any Domain

Track analysis is used in many different domains. In the domain of animal tracking in the laboratory, the tracks of one or more animals in a confined area, such as a cage, are analyzed. The animals are continuously tracked, for instance by a video camera, for a period of a few minutes up to several days. Typical animals are rodents (rats, mice), but fish, farm animals, and insects are no exception. The animals are tracked as part of scientific experiments in which hypotheses are tested. A hypothesis usually compares the behavior of a study group to a control group. The behavior of the animals is derived from the quantified spatio-temporal information obtained from the track analysis, such as distances, speeds, path shapes, and frequencies of zone transitions. An experiment can, for example, investigate whether a particular genetic modification or treatment with a particular drug results in enhanced memory as demonstrated by the ability of the animal to find a hidden platform in a swim tank.

In the domain of vessel tracking at sea, the tracks of all vessels in view are analyzed to increase the safety and security at sea. The vessels are tracked 24 h a day, 7 days a week, and 365 days a year, for instance by radar and using the Automatic Identification System (AIS). Vessels that expose undesired or atypical behavior are contacted via radio or even physically visited to understand their intent and, if necessary, correct their behavior. The behavior of vessels is derived

from the quantified spatio-temporal information obtained from the track analysis. For example, a vessel sailing against traffic in a sea lane is recognized based on its deviating course, and smuggling vessels may be recognized based on their simultaneous stop at the same spot in the middle of the sea.

In the domain of animal or people tracking on the globe, the tracks of one or more subjects freely traveling the world are analyzed. Animals can be continuously tracked, for instance using GPS, for a period of up to a few years. Such animals are typically tracked to increase the understanding of their habitat use, dispersal, or migration behavior. For example, animal tracking can show a change in their migration behavior due to the growth of a harbor. Farmers can find changes in the cow behavior in the herd, helping early detection of diseased animals.

In shops and public places such as stations and libraries, the tracks that people make as observed by (security) cameras quantify the efficiency of the building's usage and reveal opportunities to improve for instance throughput or sales. Tracking public transport vehicles in a city reveals planning inefficiencies. Coaches use the tracks that athletes make to improve their performance.

Summarizing, track analysis enables behavioral researchers to quantify and understand behavior and changes therein by relating quantified spatio-temporal information to behavior. Behavioral researchers are interested in the implications of this behavior, e.g., the drug reduces epileptic seizures or a vessel is smuggling, and typically not in the activities needed to obtain these implications, such as data acquisition, programming, and quantification.

6.2.2 EthoVision XT

EthoVision XT is an automated PC-based video tracking and motion analysis system. Figure 6.1 shows EthoVision XT in action. It has been under development since the nineties and is highly attuned to the domain of animal tracking in the laboratory. EthoVision XT handles most of the data acquisition, programming, and quantification needed for this domain. The user of EthoVision XT only has to point a fixed camera to a confined area, such as a cage, and let one or more animals move about in it. Furthermore, EthoVision XT supports proved methods and standardized tests, as used in e.g., neuroscience, biology, and toxicology, to ensure the validity of the experimental setup, the reproducibility of results, and the accuracy of the statistics. For example, up to 16 confined areas containing multiple animals can be tracked simultaneously and independently. The system has become an industry standard, in use at approximately 1,950 laboratories around the world.

During data acquisition, EthoVision XT samples a video stream at a fixed rate of, e.g., 25 frames per second, extracts the center of gravity from the detected object, and writes that to a track file. The maximum area under observation is constrained by the size of the object: with a standard video resolution the area cannot be larger than 200 times the size of the object.

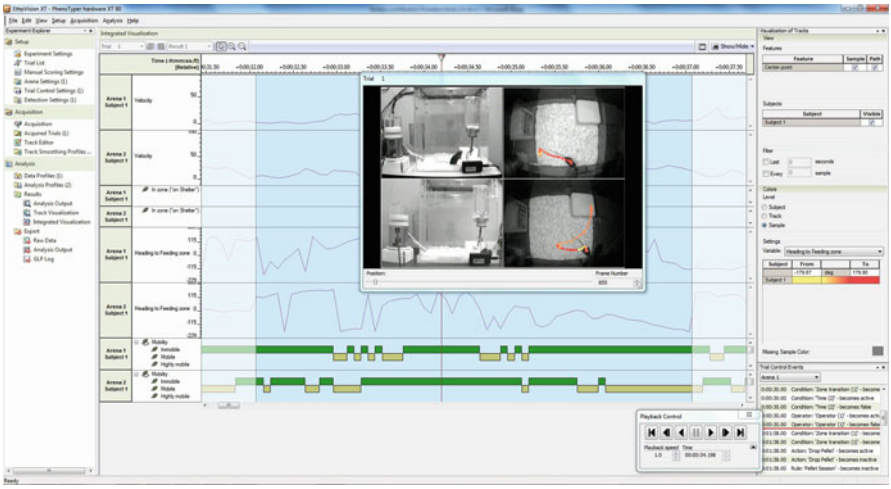


Fig. 6.1 Screen shot of EthoVision XT as used in a neuroscience experiment

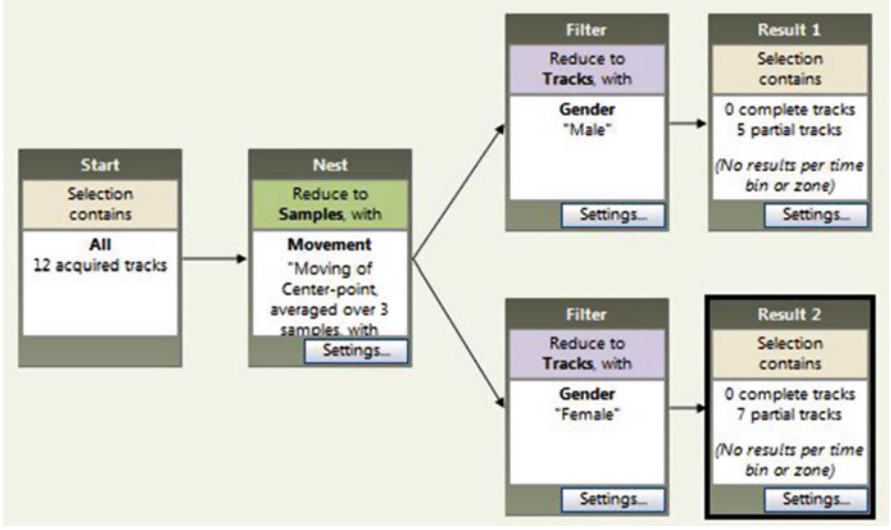


Fig. 6.2 Screen shot of data filtering using the graphical block editor

Researchers can use the visual editors of EthoVision XT to specify the area definition, trial control, data acquisition, selection, and processing steps, see Fig. 6.2. For example, tracks can be smoothed in several ways to remove body wobble, noise and outliers, and data can be filtered based on dependent variables (such as 'in zone') and independent variables (such as 'gender' or 'treatment').

EthoVision XT translates live motion into quantified spatio-temporal information, such as:

- Distance moved by each animal;
- Latency time between action and response;
- Current, minimal, and maximal velocity of each animal;
- Duration of the time spent by each animal in a particular zone;
- Relative speed between animals (moving to/from); and
- Frequency and other statistics of behaviors such as rearing, grooming, sniffing, eating, and drinking.

Finally, EthoVision XT can visualize the recorded tracks. Besides visualizing the actual positions of animals, also the recent history can be shown. In this case, all animals get a 'tail' showing their recent activities. See Fig. 6.5 for such an example. Note that playback of recorded track data can be up to 20 times faster than real time.

6.2.3 Similarities Between Animal Tracking and Vessel Tracking

In principle there is no difference between tracks made by ships and tracks made by animals: speeds, locations, turn angles, etc. are all treated with the same mathematical formulas resulting in the same statistical variables. From a practical point of view there are differences, but they can be resolved (as is described in Sect. 6.4).

For most researchers that use EthoVision XT, behavior is a means to an end, i.e. behavior is a read-out of the underlying process that is the subject of study. Most researchers take a sufficiently large number of tracks and draw inferences from that. EthoVision XT will then deliver a quantitative assessment of behavior, e.g. to demonstrate significant statistical differences between groups of animals, in order to determine the effect of a changed variable, given that all other environment influencing variables are kept constant. In short, the researcher controls the environment to test a hypothesis.

It is easy to imagine EthoVision XT to be used by, for instance, coast guard personnel to track and analyze vessels but this was never the target audience. EthoVision XT was designed for observation and analysis but not for control. Coast guard personnel would want to focus on individual ships and their specific detailed behavior in order to effect control. There are other commercial tools better suited to this task. In EthoVision XT, specific situations of interest can be viewed but the detection mechanism for these situations is suboptimal. For instance: false coloring can be used to highlight situations but detailed information of a particular track will have to be looked up manually on other screens. The coast guard user would probably be better served with being presented only the situation of interest, keeping all other data hidden.

In the case of Maritime Safety and Security the researcher is certainly not in charge of the situation but he can control the environment to some extent if the situation takes sufficient time. As said before, EthoVision XT is aimed at

experimental research and testing of scientific hypotheses. Such hypotheses can be used to quantify normal behavior, thus allowing the creation of algorithms to detect abnormalities. Examples of this could be to quantify the influence of weather on ship behavior, circadian, seasonal or yearly trends, quantify similarities and differences in different ports, properties of zones or ships (nationality, destinations, cargo vessel, ferries) or specific intentional behavior ('ships with destination X will sail only in that direction', or 'traveling ships will move slowly in anchoring zones').

6.3 Case Study: Speeders in Anchoring Zones

The purpose of EthoVision XT is to describe a behavioral process or to test a research hypothesis. To determine whether EthoVision XT could be adapted and applied to the tracking and analysis of vessels in the maritime domain the following hypothesis was defined: 'If ships cross anchoring zones, they have a reduced speed and stay well clear of other, anchored ships'. The case study is designed to show how a prototyped EthoVision XT can be used to find quantitative support for the level of truth in such a hypothesis. This case study only demonstrates a tiny amount of the functionality in EthoVision XT and therefore functions as no more than a proof-of-concept.

In the rest of this chapter, we will discuss the preparation and analysis of the case study. The preparation consists of two steps: the construction of the data set and the calibration of the real life domain (coordinates on a nautical map) to the EthoVision XT domain (coordinate system of a video image). The analysis consists of three steps: the recognition of anchoring zones, the detection of offending ships which move through the anchoring zones, and the investigation of their speed and the minimum distance they keep from other ships while in the anchoring zone.

6.3.1 The IJmuiden Data Set

For the case study the area just outside IJmuiden in the Netherlands was chosen. Ships wait in an anchoring zone within this area to enter the canal to Amsterdam. The size of the area was chosen such that EthoVision XT could nicely play back the recorded tracks. Considering that a fast moving commercial ship moves at around 17 knots, i.e., 31.5 km/h, the sampling frequency of the data is once per 30s, EthoVision XT plays back at a rate of 25 samples per second, and we want ships to take 15 s of play back time to move along the area's diagonal, then the diagonal should be approximately $31.5\,\mathrm{km/h} \times 1/3{,}600\,\mathrm{h/s} \times 30\mathrm{s/sample} \times 25\,\mathrm{sample/s} \times 15\,\mathrm{s} = 98.4\,\mathrm{km}$. See Fig. 6.3 for the exact details of this area. For the case study, the ship tracks in this area during a period of 7 days were collected using AIS. During this period 1,052 different ships were observed in this area. Since EthoVision XT expects time-equidistant and synchronized data samples, every 30s the position

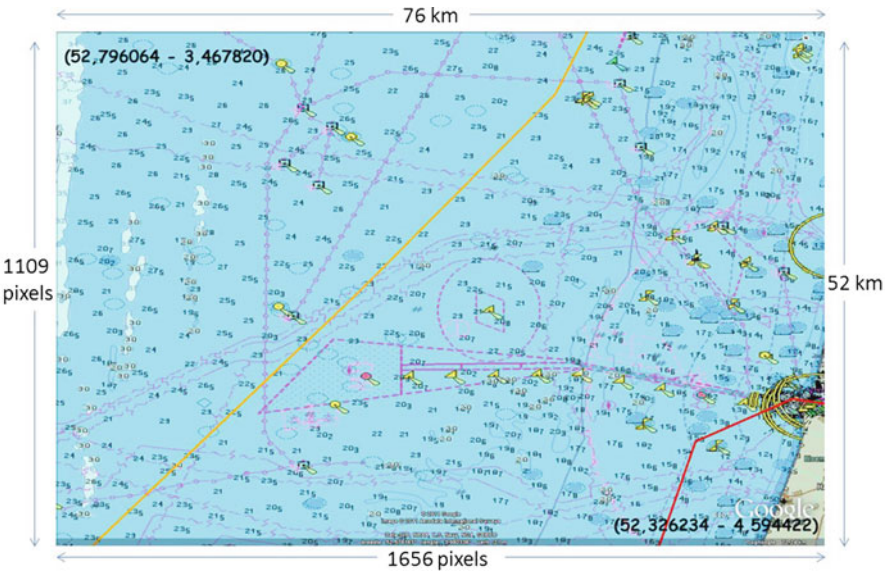


Fig. 6.3 Nautical map containing the anchoring area near IJmuiden together with its coordinates (longitude – latitude), actual distance (in kilometers) and distance in EthoVision XT (in pixels)

of each ship was determined using (linear) interpolation of the AIS ship tracks. The value of 30 s was chosen as compromise between time resolution and mass-storage requirements. Next, the recorded positions were ordered on a per ship basis, resulting in track files that consist of data arranged in four columns: one column for the time stamp, two columns for the position in longitude and latitude, and a column for the 'Destination' as reported by the ship via AIS.

6.3.2 Calibration

The provided background map has a resolution of 1,656 by 1,109 pixels. To be simply able to draw the ship tracks over the background map, this resolution was also chosen to represent the ship tracks. The horizontal distance calibration is thus 76 km/1,656 pixels = 45.9 m/pixel and the vertical distance calibration is 52 km/1,109 pixels = 46.9 m/pixel: an aspect ratio of 1.02. This non-linearity introduces an error of maximally 2%, which is acceptable in this case study. On average one pixel in the background map represents approximately 46.4 m in reality. Note that the aforementioned fast moving commercial ship that moves at around 17 knots, i.e., 8.7 m/s, thus moves 141 pixel/s on the screen: a smooth and clearly visible motion.

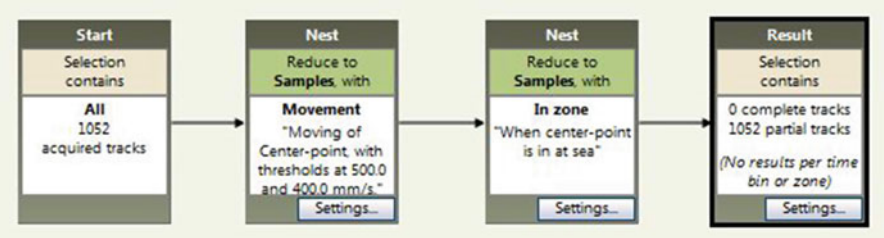


Fig. 6.4 Query to visualize stationary ships at sea to obtain the anchoring zones

6.3.3 Finding Anchoring Zones

To find the relevant anchoring zones, a query was made using the graphical editor. Between the 'Start' block, containing all 1,052 ship tracks, and the 'Result' block, visualizing these tracks, two data reduction blocks were inserted; see also Fig. 6.4. The first reduction block detects movement, i.e., speed above 500 m/h, and causes a difference in the visualization of the ships depending on the presence of movement: Non-moving, stationary ships appear in red, while moving ships appear in black. The second reduction block separates ships based on their location, i.e., presence in a zone. This separation was needed to distinguish stationary ships in anchoring zones from those in ports. Therefore a border, some 500 m at sea and parallel to the coast, was graphically drawn to create two zones: 'At sea' and 'Coastal zone'.

Using this query, the anchoring zones were visually detected by drawing zones around the stationary, red ships at sea. In this case study only a single zone was discovered, as is visualized in Fig. 6.5. Neither the case study nor EthoVision XT require a single zone. In fact, when multiple anchoring zones would have been present, we would have drawn multiple disparate zones, and introduced a cumulative zone that refers to all these disparate zones. This cumulative zone, instead of the single zone, would have appeared in all queries defined later on in the case study.

6.3.4 Detecting Crossing Ships

To detect ships that cross the anchoring zone, another query was made using the graphical editor in EthoVision XT. Between the 'Start' block, containing all 1,052 ship tracks, and the 'Result' block, visualizing these tracks, one reduction block was inserted; see also Fig. 6.6. The reduction block separates ships based on their (temporary) presence in the anchoring zone. From the 1,052 ships, 58 ships actually cross the anchoring zone in the investigated, 7 days period. EthoVision XT can give statistics for these 58 ships about the duration of the crossing and for the minimum and maximum velocity in the anchoring zone. For example, Fig. 6.7 shows the ships that cross the anchoring zone, sorted on their minimum velocity in the anchoring zone.

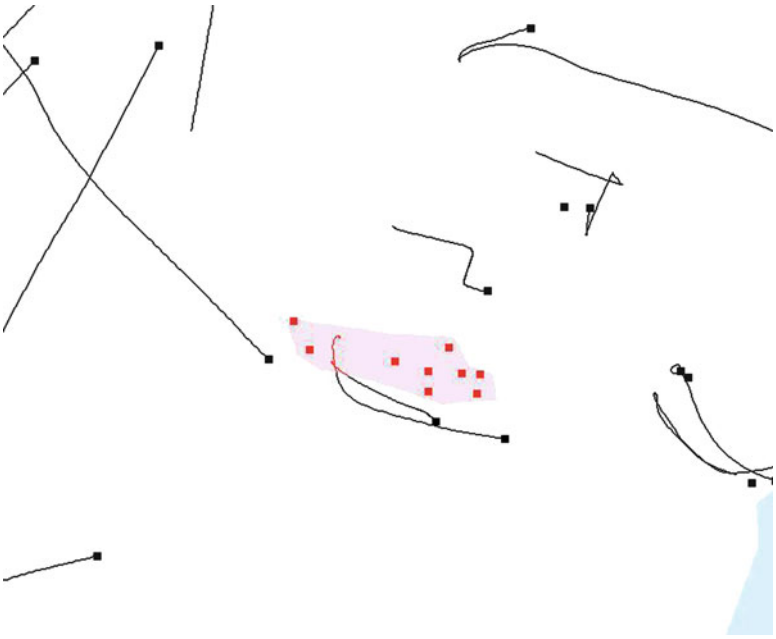


Fig. 6.5 Visualization of vessels in an area containing a (*gray*) anchor zone. Anchored and sailing vessels are indicated by *red blocks* and *black blocks with lines*, respectively

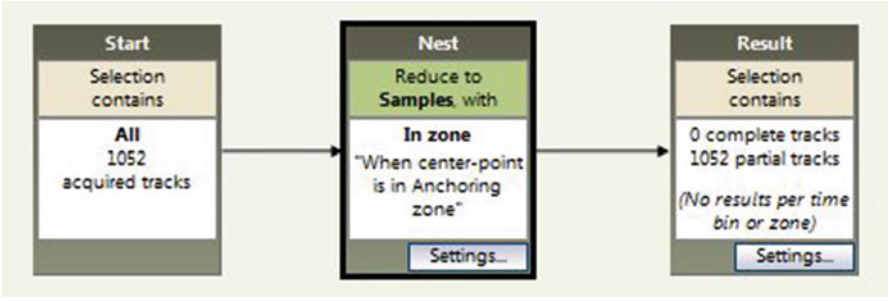


Fig. 6.6 Query to detect ships that cross the anchoring zone

6.3.5 Do Crossing Ships Keep Distance?

The 58 ships that cross the anchoring zone are further analyzed. This new analysis takes all pairs of these 58 ships to focus on the 'distance between objects' within the anchoring zone. We used the same block as before for data selection and selected two variables: 'Distance between objects' with statistics 'Minimum' and 'Velocity' with statistic 'Minimum'. Of the 58 ships, it turns out that 5 ships came closer to

Independent Variable	Velocity		
Destination	Maximum	Mean	Minimum
	mm/s	mm/s	mm/s
HAMBURG (HAMBURG) "	162.3349	162.3349	162.3349
IJM_NS (IJM_NS) "	152.3373	151.0748	150.1640
IJMUIDEN (IJMUIDEN)"	125.4097	124.2033	123.1452
AMSTERDAM (AMSTERDAR	115.6746	115.5946	115.5146
VELSEN (VELSEN) "	110.3650	109.3978	108.4852
IJMUIDEN (IJMUIDEN)"	117.9272	105.0810	94.2457
FALKENBERG (FALKENBJ	96.9943	95.7900	93.9948
DREDGINGAREA (DREDGN	90.9844	88.8343	86.9531
@@@@@@@@@@@@@@@@@@@@B	85.3949	85.3949	85.3949
	86.4802	85.6565	84.8329
AMSTERDAM (AMSTERDAR	88.9667	82.4848	79.7463
DORDRECHT (DORDRECHV	87.3373	82.4549	78.3133
STENUNGSUND (STENUNR	84.6907	78.8146	73.4947
AMSTERDAM (AMSTERDAR	68.6105	68.6105	68.6105
IJMUIDEN (IJMUIDEN)"	75.5890	70.8841	67.6445
RENKUM (RENKUM) "	74.5630	68.0251	62.7375
DONGES (DONGES) "	78.0129	68.0806	58.8690
STETTIN (STETTIN) "	62.6270	59.5147	57.5016
ROTTERDAM (ROTTERDAR	55.4432	54.1342	52.7936
AMSTERDAM (AMSTERDAR	100.9298	72.7081	45.7589
CUXHAVEN (CUXHAVEN)"	46.7026	46.2378	45.5374
SCHIEDAM (SCHIEDAM)"	81.7009	61.9770	41.7598
DREDGING OPS. (DREDR	72.7846	55.1282	37.8659
IJMUIDEN-Q7 GUARD (N	37.9301	21.8531	0.4463
AMSTERDAM (AMSTERDAR	45.3702	3.2404	0.0549
	121.5393	8.2561	0.0483
	83.1010	3.0926	0.0191
AMSTERDAM VIA KIEL *	121.7497	2.0567	0.0106
@@@@@@@@@@@@@@@@@@@@B	88.9949	4.4384	0.0101
AMSTERDAM (AMSTERDAR	105.5841	1.3319	0.0097
	67.7614	2.1223	0.0076
AMSTERDAMM (AMSTERDF	96.1216	2.6667	0.0064
AMSTERDAM (AMSTERDAR	125.0332	2.2032	0.0063
	106.9873	4.3162	0.0062
AMSTERDAM (AMSTERDAR	94.8665	7.4462	0.0055
IJMUIDEN (IJMUIDEN)"	76.8609	4.9464	0.0053
	74.9185	2.6279	0.0051

Fig. 6.7 Destination and velocity of ships that cross the anchoring zone. The ships are sorted on their minimum velocity in that zone (from high to low), where 1 mm/s corresponds to 0.12 knots

another ship than 500 m with a velocity in excess of 8.5 km/h. If so desired, these 5 ships can be examined in minute detail; see also Fig. 6.8. Furthermore, the tracks of these 5 ships can be given another color, to set them out against all other ships.

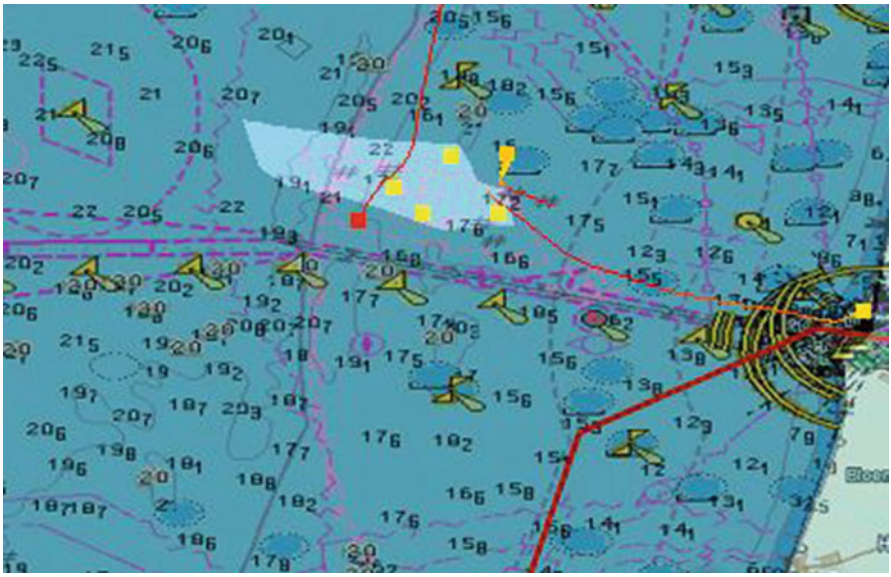


Fig. 6.8 Ship (in *red*) that crosses the anchoring zone at a too high speed and too close to other, anchored ships (in *yellow*)

6.4 Observed Differences Between Animal and Vessel Tracking

The case study showed a few important differences between animal tracking in a laboratory and vessel tracking at sea. The maritime domain, as opposed to the laboratory, requires taking into account (1) the curvature of the earth, (2) objects that temporarily disappear from view, (3) a volatile and dynamic set of objects, (4) changing roles of vessels at sea, and (5) irregularly arriving, non-time-equidistant data samples. The following sections describe these topics in detail.

6.4.1 Curved Area

The areas involved in tracking animals in a laboratory rarely exceed a few square meters. Including the earth's curvature for these small areas would only complicate the track analysis. Consequently, EthoVision XT assumes a linear relation between the number of pixels and the distance they represent. Yet, the earth's curvature can often not be ignored when tracking vessels at sea. Fortunately, the curved area can be projected onto a square using map projections[2] such as equal-area,

[2]See http://en.wikipedia.org/wiki/Map_projection for more information on map projections.

conformal, or equidistant projections. In fact, when the projection is consistently used, i.e., for tracks and background maps, visualization is independent of this projection and thus of the curvature of the area of interest. However, any analysis must include the actual projection, since the calculation of distances and angles needs to include the appropriate formulas for this projection. Because EthoVision XT could not easily be changed to include these appropriate formulas, the case study was limited to a relatively small area. Due to this limitation, the systematic error introduced by the earth's curvature is small in comparison to the other errors in the case study, such as, for instance, the GPS inaccuracy.

6.4.2 Known Location

A fact of animal tracking in a laboratory is that the location of the animals is always known. The animals are restricted to a confined area that is completely covered by the (video) tracking system. Of course, the confined area might contain so-called hidden zones, typically representing shelters. These hidden zones are intentionally added and their sizes and positions are known and constant. When animals are within a hidden zone, their location is known and the associated accuracy depends on the size of that hidden zone. The analysis of EthoVision XT knows the concept of hidden zones, even with multiple exits, to ensure correct variable calculation. For example, an animal which enters a hidden zone at location (x_1, y_1) at time t_1 and leaves at location (x_2, y_2) at time t_2 will get a correct velocity calculation at t_2.

The IJmuiden data set contained many instances of ships temporarily leaving from view at an arbitrary location only to return to view after an arbitrary time period at another arbitrary point. This temporarily leaving from view happened so often that it must be considered normal, not exceptional. Hence in the domain of maritime safety and security, the locations of tracked objects are not always known. Implementing this adaptation has consequences for the functionality of EthoVision XT: Missing samples for instance can no longer to be treated as a measure for track quality.

6.4.3 Fixed Number of Objects

A fact of animal tracking in a laboratory is that the animals are restricted to a confined area. Since animals cannot leave or enter this area, the number of objects tracked by the (video) tracking system is fixed. On the contrary, in the domain of maritime safety and security, the ships are not restricted to the area of interest. For example, the Dutch coast guard is interested in the North Sea, but ships are

free to leave this area and many regularly do so. To support the maritime domain, EthoVision XT has to drop the constraint of tracking a fixed number of objects. Yet, it would require a rewrite of EthoVision XT.

6.4.4 Fixed Roles

EthoVision XT is optimized for proved methods and standardized tests in the domain of animal tracking in a laboratory. This means that no variation is allowed in the roles that are assigned to the tracked animals. A 'treated' or 'female' animal retains this role throughout the experiment. In the IJmuiden data set, we observed that ships regularly change their attributes such as 'destination' and 'cargo' over time, and even while at sea. The only way to currently allow ship tracking in EthoVision XT would be to consider a changed attribute as being a change in object. The effect would be that ships appear and disappear. This concept may work well enough but it seems contrived and artificial.

6.4.5 Arrival Time of Samples

The biggest engineering problem encountered was the irregularly arriving (maritime) track samples. EthoVision XT is above all a video tracking system. In a video stream, there is the assurance that samples arrive an exact amount of time after each other (e.g. 40 ms). Better still, if multiple separate objects are detected in the video image they will likely have a different location but they all have the same point in time. So the distance between two moving objects requires no interpolation over time. EthoVision XT takes this alignment of time points and their time-equidistance as basis for its calculations: it is a valid assumption for a video analysis system and it greatly simplifies the calculations, resulting in a significant performance gain. In the IJmuiden data set, this assurance is not guaranteed. In fact, the AIS standard [1] explicitly states that the reporting interval varies with navigational status (i.e., anchored, moored, or under way), speed, and changing of course. In the case study, interpolation was used to guarantee time-equidistant samples. This was tried using pre-processing, thus offering a steady stream of time-equidistant geographic track samples. In making the pre-processor, we found two main errors that demonstrate that this workaround is undesired. First, the interpolated data arrives too late for any kind of real-time tracking. Second, the interpolation introduces rounding errors in time-related variables such as velocity. These rounding errors might negatively influence analysis. See for example Fig. 6.9 that shows the impact of interpolation on velocity.

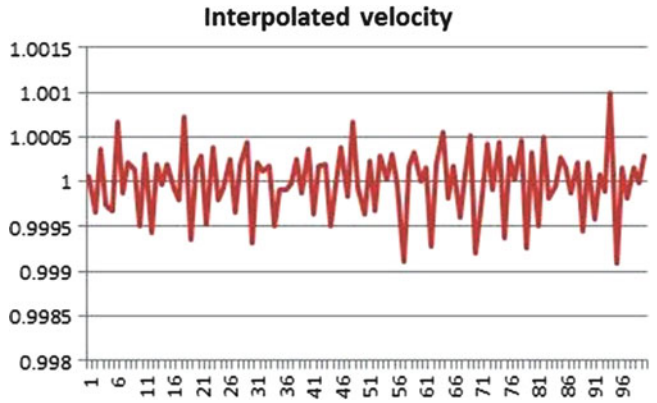


Fig. 6.9 Calculated velocity ratio between two real-world samples, using interpolated time-equidistant (artificial) samples positions. Instead of a constant speed, and thus a ratio of 1, a fluctuating speed with a maximal relative error of approximately 1% is observed between the two real-world samples due to rounding errors

6.5 Summary and Future Work

The portfolio of Noldus includes EthoVision XT, which is the most widely applied video tracking software that tracks and analyzes the behavior, movement, and activity of any animal in a laboratory setting. Noldus wishes to extend its portfolio. Among others, Noldus wishes to track and analyze objects anywhere, both indoor and outdoor. Therefore, we performed a case study to investigate the differences in track analysis of animals in the laboratory and objects moving around the globe. In particular, we investigated whether EthoVision XT could be extended to analyze tracks in the maritime domain to find vessels speeding in zones reserved for anchored vessels. EthoVision XT was successfully extended in this case study. In particular, a number of speeders in an anchoring zone were detected. During the case study, we discovered that EthoVision XT was not the perfect starting point to make a tool for tracking objects anywhere. The following fundamental differences, which would be costly to change, were discovered:

- Due to the earth's curvature, larger areas on the globe are obviously not flat.
- Outside the laboratory, tracked objects often disappear from view.
- Outside the laboratory, the number of objects to be tracked simultaneously typically varies over time.
- In the maritime domain, the attributes of tracked objects change, e.g., the destination of a ship can change from 'Rotterdam' to 'Singapore'.
- Whereas video tracking ensures periodic and synchronized detection of multiple objects, in the maritime domain track samples typically arrive irregularly and asynchronously.

Instead of extending the existing software, Noldus decided to architect a new prototype tool from scratch. This tool is under development within the scope of the European FP7 projects PRONTO and E-TRACK. Significant advances were made by exploiting the lessons learned from the case study. This new tool will also benefit from Noldus' extensive knowledge about tracking and analyzing of animals at laboratory scale that could be transferred to the domain of tracking and analyzing objects anywhere. Several innovations are being implemented: support for GPS coordinate streams, default implicit calibration, use of digital maps (e.g. a city street plan) with interactive zoom and pan capability, greater independence from the number of objects to track, and graphical annotation of maps to create regions of interest for spatial event detection. Most importantly, the basic algorithms for all distance calculations are based on a curved area and use an efficient implementation of the haversine formula instead of being based on a flat surface. The tool was further extended to cope with large numbers of moving objects, including ships, fire engines, buses, and people. Latest developments include functions that compute movement characteristics, e.g., velocity, periods of movement, distances, and path shape such as 'sharp corners' and abrupt stops. All this will result in a new Noldus product prototype, referred to as 'AnyTrack'.

While developing the new tool for tracking and analyzing objects anywhere, many lessons were learned. We mention two lessons learned that are also relevant for the domain of maritime safety and security. First, we experienced scalability issues in the popular and commonly available geographical libraries and tools. We observed a lack of performance in Google Maps and OpenLayers when large numbers of tracked objects or sample points were present. Compression as described in Chap. 7 is at most a partial solution since it only reduces the number of sample points. Second, any tool for tracking and analyzing objects anywhere must support a large variety of different kinds of background maps. Background maps vary from a building floor plan for indoor position data, to OpenStreetMap or Google Maps for outdoor position data, and from harbor plans to nautical charts for the maritime domain.

Acknowledgements This research has been carried out as a part of the POSEIDON project at Thales under the responsibilities of the Embedded Systems Institute (ESI). This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

We would like to thank Jan Tretmans, Arjan Mooij, and Maarten van Someren for their useful feedback on an earlier version of this chapter.

Reference

1. Radiocommunication Sector of International Telecommunication Union, Geneva (2010) Technical characteristics for an automatic identification system using time-division multiple access in the VHF maritime mobile band, April 2010. http://www.itu.int/rec/R-REC-M.1371-4-201004-I/en

Chapter 7
Recognizing Vessel Movements from Historical Data

Gerben de Vries and Maarten van Someren

7.1 Introduction

An important task of a Maritime Safety and Security (MSS) system is to acquire, store and analyze data and information from various sources and enable users to analyze and operate on this information. Our main sources of data are the Automatic Identification System (AIS) and the Internet. AIS data consists of transponders on vessels that transmit information about the vessel. This information is publicly available and is received by stations on land. The AIS data include an identification number, position, direction and speed, and it may include other information such as type of the vessel, country of registration and cargo type. The AIS data can be viewed online at several web sites. The main other source of information is the Internet, which includes historical information about vessels and geographical information. A realistic maritime safety and security system will also include radar, image and video data. The solutions that we developed for AIS data can be adapted to this setting. The output of a maritime safety and security system includes visualizations of vessel movements, which is addressed in Chap. 5, and recognizing events, objects and anomalies, which are addressed in Chaps. 8 and 9.

In a maritime safety and security system there are a number of steps between the AIS data that is transmitted by a vessel and the visualization and recognition of anomalies. Apart from relatively simple data transmission, the AIS data must be integrated (because data are received asynchronously by multiple stations) and aggregated to a higher level of abstraction than streams of AIS data. This is useful because some interesting events are not instantaneous but take place over a longer time and larger space. Examples of events are "leaving harbor" and "anchoring". To describe such events additional high level concepts are needed, for example about geographical locations ("anchoring area", "deep water lane") or types of vessels

G. de Vries (✉) • M. van Someren
University of Amsterdam, Informatics Institute, Amsterdam, The Netherlands
e-mail: G.K.D.deVries@uva.nl; M.W.vanSomeren@uva.nl

P. van de Laar et al. (eds.), *Situation Awareness with Systems of Systems*,
DOI 10.1007/978-1-4614-6230-9_7,

("passenger vessel", "fishing boat", "tanker"). Recognizing abstract events is useful for visualization and for detecting anomalies, such as illegal or dangerous events or anomalous objects that have an unusual combination of properties and behavior.

In more technical terms, the problem becomes to recognize abstract patterns or "concepts" in streams of AIS data. The approach that we take to finding abstract descriptions of objects is to learn from historical data. The main alternative approach is manual modelling, in which human expertise is used to construct classifiers manually. We preferred the approach based on historical data for practical reasons, because no abstract concepts and not many human experts were available, but there was a large set of logged AIS data. An additional argument is that anomalies are inherently irregular and therefore it is not easy to anticipate which concepts are useful for recognizing them. At the same time, the size of the volume of historical AIS data and the rate at which they are sensed in the online setting make abstraction essential. In the context of a complete maritime safety and security system, several additional issues should be taken into account. Before entering into our analysis, the signal passes through a number of components. Errors are introduced by transmission by an AIS transponder, atmospheric conditions, reception at the land stations, data integration, processes in systems that add data at a later stage or that impose security protocols, etc. The consequence of this is that certain data do not correspond to vessel movements but are generated by the system that processes the raw signals. This is also true for some anomalies and visualizations.

Besides the basic movement data (location and direction), additional information can be obtained from other sources such as online databases and other resources on the Internet. In fact, the AIS data provided by a vessel include additional information. It can be useful to be able to predict such additional information from data because the additional information provided by a vessel or database may be incomplete or incorrect. For example, the destination that was found on the web page of the owner may be different from its predicted destination based on historical data, which may be useful to know for the users of the maritime safety and security system.

In this chapter we focus on three specific technical problems that need to be solved in a maritime safety and security system. One is data compression. The AIS data that is generated by vessels is very intensive. Data become available with a high frequency from many vessels. This results in a large volume of historical data which makes it practically difficult to use these data. The solution is presented in Sect. 7.2. Section 7.3 addresses the second problem: to find a measure of similarity between movements that can be used to find clusters of similar movements, movements that are very different from all other movements or movements that are associated with a property of a vessel. Such a measure is the basis of methods for prediction of movements of vessels and of properties of vessels or of movements. The third problem is how to combine movement analysis from AIS data with other, mostly symbolic, information that is obtained from the Internet or from human experts. This is the topic of Sect. 7.4.

7.2 Compression of Movement Data

The AIS data that are sent out by a vessel make up a series of ⟨Time, Latitude, Longitude⟩ tuples. Latitude and Longitude denote the position of the vessel and Time the time at which the message was sent. A trajectory consists of a sequence of such tuples. Data are sent by different vessels at different rates. The AIS data include additional information about the vessel. Not all vessels send out all information. There is a substantial amount of noise in the data, due to errors in the transponder, the transmission and fusion of the data and to incorrect vessel information. The amount of AIS data is large because of the high rate at which AIS data are collected. For example 1 week monitoring of vessels before the Dutch west coast results in approximately 4 GB of data. It is therefore necessary, not so much for online incoming data but for historical data, to compress the large volume to a much smaller volume without losing important information. Compression exploits the fact that vessels usually move in a rather regular way and either move for quite some time or stop. The compressed trajectories can be used for visualization, clustering, classification of vessels, and predictions. In fact, the ultimate criterion for the quality of compression is in the extent to which the data can be compressed without damaging the use of the trajectory data for further processing.

The method that we use for compression is a version of Piecewise Linear Segmentation (PLS). The basic algorithm was invented several times in different disciplines, see [4, 5]. Beginning at the starting point of a trajectory, PLS draws a line between the starting point and the last point, the candidate end point. It checks if the distance between the trajectory and the line is below the threshold. If this is true, PLS is done and the line is kept as the approximation. If the threshold is exceeded then start and candidate end become the preliminary start and end of a partial linear approximation. The same procedure is then applied recursively to each candidate segment. This is illustrated in Fig. 7.1. The dotted line shows the trajectory of the vessel. The horizontal line connects the two points $p1$ and $p6$ on the trajectory. It is a candidate linear approximation of the actual trajectory $p1, \ldots, p6$, but it turns out that the distance between the point $p4$ and the linear approximation (E) is larger than the threshold (ε). Therefore the approximation by a single line is replaced by two lines and the point $p4$, which has the maximal distance, is included in the approximation. As a result, the trajectory $p1, p4, p6$ replaces the line and

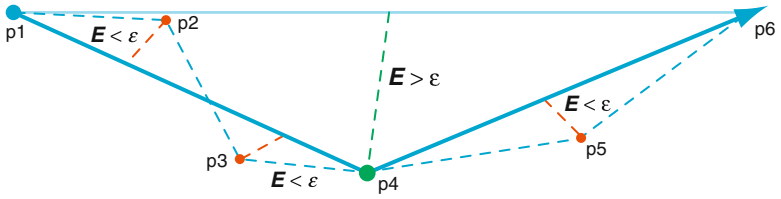


Fig. 7.1 Illustration of piecewise linear segmentation

becomes the new approximation. The procedure is now applied recursively to the two parts of the new approximation. The largest distances between the resulting approximation and the actual trajectory is below the threshold and therefore the two line approximation is kept. The points $p2$,$p3$ and $p5$ are removed, which leads to a 50% data reduction. Correction for the curvature of the earth is applied to make the distance measure independent of the location.

The basic PLS method is extended in two ways: time is added as a dimension and stops are retained. The original PLS method only considers the information on locations (Latitude and Longitude) and not the speed at which they are traversed. The PLS method for trajectories can be used with different distance functions. The basic case simply uses Euclidean distance. Time can be added into the distance function to create distances between vessels that move at different speeds or at irregular speed, using 3-dimensional Euclidean distance. This needs a weight for the time dimension that reflects how important time is in the distance relative to location.

In the case of vessel movements, it is important to retain stops as points in the piecewise linear segmentation, even if a vessel stops and then later continues in exactly the same direction, or if it moves slowly, for example when anchored. This is achieved by first applying PLS to speed only and only after that with the combined distance measure. The purpose of the first step is to find the stops, as the points where the speed differs most from other speeds. These points are then retained in the approximation of the trajectory. The resulting algorithm we call Two-Stage Piecewise Linear Segmentation (TwoStage-PLS) [9].

Experiments on AIS data show good results for TwoStage-PLS. For example, using separate thresholds of 2.5 knots for speed and 50 m for displacement gives a compression to 2% of the original amount of data while 97% of the actual stops are retained. Because of the large volume of data this actually enables the use of historical data at a scale that would otherwise have been impossible. Because large volumes of data are needed to obtain reliable predictions and interpretations, this compression technology is of key importance. We also created a variant of the algorithm that compresses AIS data from new incoming vessels “online” to make predictions about these partial trajectories faster and better comparable to historical data.

7.3 Measuring Similarity Between Movements

There are several possible approaches to the tasks of recognizing properties and movements of vessels and recognizing unusual trajectories. The approach that we take is based on similarity between vessels and vessel movements, in particular using historical information of vessels and vessel movement data. Predictions of properties are based on similarities between new and old trajectories and vessels. This relies on the use of one or more measures of similarity, in our case between movements of vessels. These measures should be able to take into account the

location, direction, speed, time but also other information, like the type of vessel (cargo, tanker, passenger carrier, fishing vessel) or other properties (size, origin). This flexibility is useful if a user is interested in different views of the data. A user may only be interested in (similarities regarding) the locations and not the time, or only the time, or the type of vessel and the locations, etc. The main alternative approaches are a probabilistic (Bayesian) approach that also uses historical data and a manual modelling approach. We decided to use the similarity–based approach for the following reasons:

- Representation of trajectories of moving objects as variables is a non-trivial problem because of the dynamics in time and location. (We do not have a fixed set of times or locations.) This complicates the use of the Bayesian approach. Our approach needs only similarity data and therefore does not require a data model based on vectors.
- The similarity–based approach is likely to be more efficient than the probability–based approach, at least at learning time and probably at recognition time, especially because of the large number of dimensions. The probability–based approach requires fitting distributions on all variables.
- The similarity–based approach makes it relatively easy to incorporate prior knowledge from ontologies such as labeled regions and properties of locations or properties of vessels.
- Especially detecting unusual events or objects is probably difficult for the manual modelling approach. There are not many experts and there is a very wide variety of unusual events that occur very rarely.

As can be seen in Sect. 7.3.1, the results confirm that our choice was a good one but experiments with other approaches are needed to support claims about what is the best approach.

The data for which we define our similarity measure consist of compressed "piecewise linear" trajectories of vessels and additional information about the vessel obtained from AIS data. In this section we consider only the trajectory data consisting of location and time. Section 7.4 extends this with additional geographical information. The similarity measure should meet several requirements:

- It should produce intuitive results when used for discovering patterns, in our case clusters,
- It should be effective in predicting properties of vessels,
- It should be able to handle trajectories of different lengths,
- It should be flexible enough to include different views of the data (in the sense of different subsets of properties or locations).

Our solution consists of a measure that is based on finding the optimal alignment of trajectories, using a distance function between (aligned) points as the basis for measuring the distance. This is a kind of string edit distance, similar to [1]. An edit distance alignment between two sequences builds pairs of matching points in the sequences. Figure 7.2 illustrates an edit distance alignment. Note that not all points are paired. The edit distance is calculated as a function of the distances between

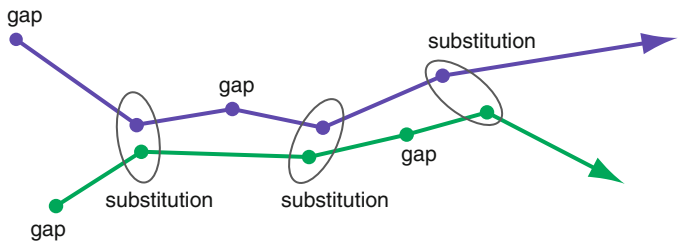


Fig. 7.2 Illustration of edit distance alignment

matched pairs of points, "substitutions", and "gaps", points that have no match in the other trajectory. The alignment is constructed so that the edit distance is minimized. To calculate the edit–distance the distances between matching points are summed and a penalty is added for each gap. The size of this penalty is a parameter of the algorithm. This distance can be converted to a similarity by taking the inverse.

7.3.1 Results

We evaluated the similarity measure in a clustering task and in a prediction task. For the clustering task we manually grouped 714 trajectories into 8 clusters. We then used the similarity measure in a standard clustering algorithm: kernel *k*-means [7]. We evaluated the method by matching the resulting clusters with the handmade clusters and we found that 85% of the trajectories are assigned to the same cluster in both cases. For details of this experiment see [3].

Figure 7.3a visualizes the eight manually labeled clusters representing different classes of trajectories. For example, cluster 1 contains vessels that pass the harbor area, cluster 3 contains vessels that leave the harbor and go north or west. Clusters 7 and 8 contain the trajectories of a ferry that moves back and forth. The clusters of trajectories can be given verbal descriptions, can be introduced as new domain knowledge and new vessel movements can be classified as belonging to one of these movement patterns.

Figure 7.3b shows a clustering results using the above similarity measure and clustering algorithm. Since kernel *k*-means starts with random initializations, different outcomes are possible, the shown clustering is one possible result. The biggest differences with the gold standard are that cluster 1 is separated into two clusters, and clusters 7 and 8 are combined into one cluster.

For the prediction task trajectory data was used to predict the vessel type as given by the AIS data. Using data from 1,900 trajectories it was possible to predict the vessel type with 72% accuracy on this dataset using the kernel–based support vector machine classification algorithm [6]. We found that some types were difficult to separate, in particular tankers from cargo vessels and pilot vessels from tugs.

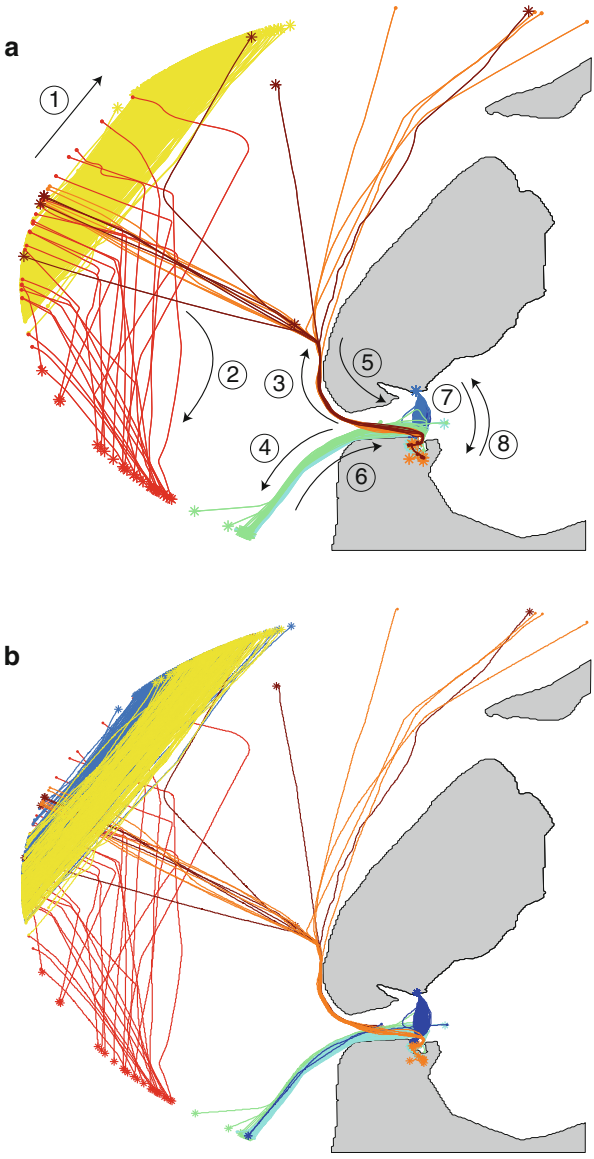


Fig. 7.3 The gold standard clusters are shown in (**a**). A generated clustering is shown in (**b**)

If we disregard these distinctions then 95% separation was achieved. Predicting the type of a vessel is in itself not a very useful task but the results of such predictions can be compared with the information provided by a vessel and then used to detect inconsistencies. This can then be used in trust analysis, as described in Chap. 13.

The previous experiments used AIS data of trajectories collected at parts of the Dutch North Sea coastal area. Although this enables complex analyses, the information that is lost by compression can affect the results of using the data.

To evaluate if this is the case, we repeated the experiments from this section with an uncompressed version of the data. There was no negative effect of PLS compression on the results of clustering or prediction and in some cases there even was a small positive effect in terms of clustering reproduction and classification accuracy. The computation time was reduced by a factor of 100, enabling clustering of 1,000 trajectories in minutes instead of hours. More details of this experiment can be found in [3]. To evaluate our solution relative to other algorithms we made a comparison with several alternative algorithms, in particular dynamic time warping and various algorithms from computational geometry. Results (more details are in [3]) show that the other algorithms were significantly slower and/or the results on the clustering and prediction tasks were worse or the same as for our method.

7.3.2 Discussion

The experiments show that the approach that we took, consisting of Piecewise Linear Segmentation with "stop retention", similarity measures based on edit–distance and using these in clustering and similarity–based prediction works well for clustering vessel trajectories and predicting vessel types. Most likely this works well because vessel trajectories naturally have a piecewise linear shape. Vessels tend to travel in an approximately straight line between obstacles. At the obstacle they change course to the next point. Their speed is more or less constant except when entering or leaving the harbor. Exceptions are vessels like law enforcement vessels, pilots, tugs, and fishing boats, that have rather irregular behavior. In our data there were hardly any fishing vessels. If sufficient fishing vessels are in the data and they show similar behavior then clusters of these vessels will be constructed. Our choice for a similarity–based approach and for kernel–based algorithms seems to be at least a good one.

7.4 Knowledge-Based Similarity

An attractive possibility in the area of maritime safety and security and in many other areas is to use explicit knowledge in combination with a large volume of data collected from sensors to improve similarity measures, and thereby clustering and prediction task performance. The similarity between vessel movements often depends on domain knowledge. For example, two movements at different locations, with different direction and different trajectories may be very similar if it is known that both are mooring at a passenger terminal, anchoring in an anchoring area or approaching the same harbor via different sea lanes. Two similar trajectories may be considered more different if we know that one is fishing and another is just passing through.

Therefore it is useful to be able to incorporate knowledge about the domain in the similarity measure. Our approach is to use the same method as before, based on edit distance. In the raw similarity for trajectories the distances between points and the penalties are calculated from the (Euclidean) distance between locations and between times. We incorporate information about vessels and locations into the distance function by defining a distance function that also includes the "semantic distance". For example, two trajectories that end at two different anchoring areas may now have a small distance because the *type* of location is now the same.

Clearly, which information is included in the distance function depends on the purpose of its application. In particular it depends on what are considered relevant dimensions of the distance. In a maritime safety and security system it must be possible to configure the distance function interactively depending on the task. This requires understanding the data, the task and the system. For example, whether details of the trajectories or the exact mooring are relevant for predicting vessel movements, properties of vessels or for recognizing anomalies will depend on the task. The user should therefore be able to control this. We evaluate our solution by comparing clusters and predictions that were constructed with and without geographical information to see if using the geographical information can actually improve the similarity and thereby the clusters and predictions.

Our approach is to use definitions of concepts from an ontology or knowledge base to calculate "semantic distance" and add this to the "raw" trajectory distance in the similarity function. This allows more information to be used in similarity and then for prediction, clustering and anomaly detection, resulting in better predictions and more informative clusters. The clusters themselves can be given a label and then they can be added to the ontology.

7.4.1 Geographical Domain Knowledge

Our geographical domain knowledge comes in the form of two simple ontologies. One ontology contains the definition of anchorages, clearways and other areas at sea. This was obtained from hydrographic maps from Rijkswaterstaat (a Dutch governmental organization). The other ontology has definitions for different types of harbors, such as liquid bulk and general cargo (containers). This was constructed manually from the harbor branches map of the Port of Rotterdam Authority; see also Fig. 4.6 in Chap. 4. A location is associated with a type and a geographical region which is represented as a polygon. The types of locations follow the Geonames[1] ontology and extend it where necessary. We added more specific types to the Geonames types, for example subtypes of *harbor*, e.g. *dry bulk harbor* were added. This gives additional properties of locations that we call geolabels. These properties are part of the Simple Event Model that is described in Chap. 10.

[1] http://www.geonames.org

A more detailed description of this domain knowledge can be found in [8]. Here we show how information about types of regions can be used to construct richer similarity measures and we investigate if these produce better clusters or predictions.

7.4.2 Trajectory Similarity

To use geographical domain knowledge we adapt the similarity measure to incorporate geographical information. In this case each point in a trajectory is associated with one or more geographical labels. The similarity between sequences of sets of geolabels is defined using additional "semantic" measures for the distance between (matching) points in the compressed trajectory. The semantic distance measure that we use is based on the number of properties that two locations have in common. This is normalized by dividing it by the maximal number of properties that could be shared by the locations. The properties are derived from the ontology. The properties of events correspond to those that are presented in Chap. 10. This makes it possible to include predicted properties in the ontology. The points where a vessel stops and starts are aligned separately and the distance between these points becomes a separate component in the distance. The results are combined in a weighted sum. The weights are determined by hand.

7.4.3 Evaluation

We compare the quality of the predictions of vessel type of three versions: one that uses only "raw" trajectories, see Sect. 7.3, one that uses only the manually defined domain knowledge about locations and not "raw" information and one that uses both. We found that the method that combines both gives a 75% percent accuracy. This is a 9% improvement over a solution that uses *only* the domain knowledge in the distance measure and a 3% improvement over a solution that uses *only* the raw trajectories.

Although the quality of clusters is more difficult to evaluate, we ran a parallel experiment with the three similarity measures used in a clustering algorithm (kernel k-means). Inspection of the resulting clusters shows that the combined distance measure on average produces clusters that are intuitively better than the versions that use either raw trajectories or knowledge-based distance. An illustration of a discovered cluster is given in Fig. 7.4. The behavior discovered in A and B is that of small cargo vessels that directly traverse the harbor of Rotterdam sailing straight to the main land behind. Because the vessels are small they do not need to take the deep water lane and are allowed to directly enter Rotterdam, without a pilot or tug. For cluster C only raw trajectory information is used. In this cluster some vessels

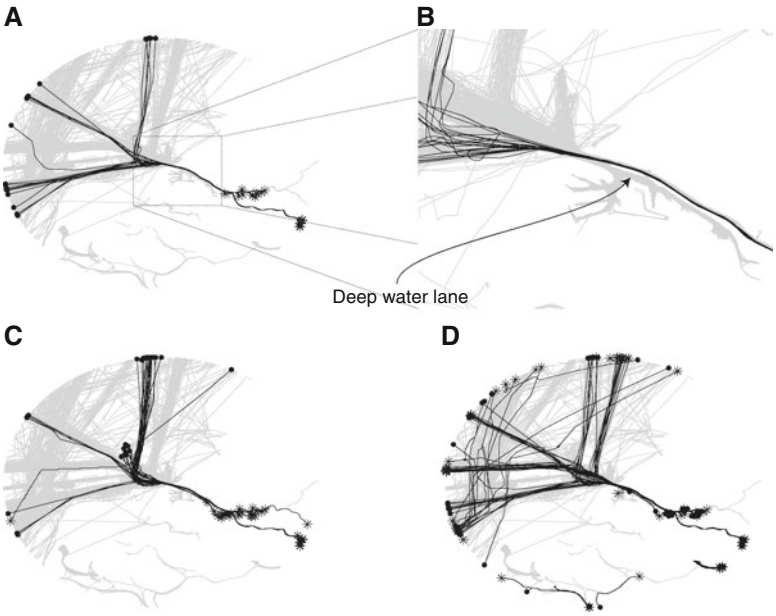


Fig. 7.4 Illustration of clustering using domain knowledge. *A* and *B* show a cluster discovered in the combined setting, where *B* zooms in on a part of *A*. Cluster *C* is the most comparable cluster discovered using only the raw trajectories and *D* is obtained using only domain knowledge

travel through the deep water lane and some do not, and some vessels come from anchorages and some do not. These behaviors are difficult to keep apart without domain knowledge, since the trajectory shapes are actually very similar. Cluster D is based on using only domain knowledge. In this case it is more difficult to keep trajectories with different directions apart, resulting in a very "messy" cluster. These three clusters illustrate the effect of including domain knowledge in similarity.

In the port area trajectories are restricted by physical obstacles and traffic rules. This means that trajectories will on average be more similar than at the open sea. The average similarity of clusters will then be larger (and the average "distance" smaller). Possibly even better results can be obtained by imposing a cut at the entrance of the port area to separate movements at sea from those in port. This is an example of how results can be improved by focusing the geographical context or other information that is included in the similarity.

The knowledge-based similarity measure consists of an edit distance alignment for trajectories and one for sequences of sets of geolabels, combined with information about the start and end of a trajectory. The results show that edit distance combined with gegraphical knowledge from an ontology gives better clusters of trajectories and prediction of vessel type. In both cases the results were better than without geographical knowledge, for details see [8].

7.5 Conclusion and Discussion

We discussed solutions to three related subproblems for the design and construction of a maritime safety and security system: the use of piecewise linear segmentation for the large amount of data, the use of edit distance as the basis of a similarity measure for vessel movements and the use of geographical and semantic distance in the similarity measure to obtain "knowledge intensive" similarity. These similarities are then used for prediction of properties of vessels and clustering. We evaluated the resulting methods on a collection of AIS data and found that these solutions are effective and efficient.

The solutions that we proposed were evaluated by "in vitro" experiments. We expect that some issues need to be addressed before the methods can be put to work in practice. We review some of these issues. One important question is which, and how much, data are needed for reliable predictions (and clusters). One issue is the amount of data. In general more data will produce better clusters and better predictions. However, the data that are used must be relevant. For example, the conditions change. New constructions are built at sea, new types of vessels appear and others disappear, traffic rules are changed and that makes old data less relevant. In general adding more properties of vessels, more areas from which data are collected increases the dimensionality of the problem. This means that more data is needed for reliable results. It can thus be effective to remove properties. For example, a trajectory can be unusual if we look only at the vessel type "passenger vessel". If all types of vessels are included then it may not be unusual. In the same way, a cluster of trajectories may be found if we only look at passenger vessels, but it may not be found if all vessels are considered. At the same time, a minimal number of data is needed to find clusters or unusual trajectories. It is therefore necessary to optimize the amount of relevant data. This makes it necessary to focus on subspaces that are defined by subsets of locations or properties. Focusing on such subspaces should be supported by visualization and anomaly detection as described in Chaps. 5 and 8. An interesting open research question is if a method can be given that guides or automates the selection of the best "view". This requires a combination of expert knowledge about which data are relevant and understanding of statistical principles to optimally benefit from the data and avoid spurious results.

The approach and methods that we used are specific versions of data mining methods [7] designed for trajectory data. If other type of information is available about events, locations or objects involved in the events then more standard data mining methods can be used and integrated to obtain better predictions or better clusters. Such data can be obtained from the Internet, by extracting information from text messages, or from structured databases. For these tasks standard methods are applicable but especially using geographical information and time information are still open issues.

The solutions use historical data that consist of AIS data, but other data can be added. For example additional properties of vessels and locations can directly be added in the similarity and used with the existing methods and implementation.

Our solution is evaluated on trajectory data obtained via AIS but similar data can also be obtained by analysis of radar data or other devices that produce trajectory data.

Application of our approach in a practical setting requires organization of the data acquisition and analysis process. The main issues are the volume of data that becomes available even if only AIS data is used as a source, the data mining and domain expertise that is needed to configure distance functions and interpret the results. We would not be surprised if in the future data mining and statistical expertise, and high-performance computing will become standard at maritime facilities but at the moment a better scenario seems to have experts do data analysis offline. We do not think that it is useful to aim at online updating of the models that are built from historical data. This process is not yet well-enough understood to allow automation or to perform this task without expertise.

We also experimented with recognizing unusual "outlier" trajectories as discussed in Chap. 8. Unlike the statistical approach in Chap. 8 we used the similarity measures to directly find trajectories that have a large distance to all clusters. These experiments also gave good results which are published in [2]. Beside the optimization of the "view", interesting topics for further work are improvement of the online recognition and anomaly detection. Another interesting extension is to apply the same approach to similarities between pairs of vessels that are close together. This would enable clustering joint movements, making predictions about this and recognizing unusual joint movements.

The methods for trajectory compression, the similarity measures for trajectories and the use of knowledge from an ontology are likely to be key ingredients for a maritime safety and security system. The same methods are more generally applicable to other types of moving objects such as airplanes, land vehicles and persons. Whether they are equally effective in other applications is a subject of further work.

Acknowledgements This research has been carried out as a part of the POSEIDON project at Thales under the responsibilities of the Embedded Systems Institute (ESI). This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

The authors wish to thank Willem van Hage and Véronique Malaisé for providing the geographical ontology, and Pierre van de Laar, Jan Tretmans, Richard Doornbos and Wil van Dommelen for their comments on a earlier version of this chapter.

References

1. Chen L, Ng R (2004) On the marriage of lp-norms and edit distance. In: VLDB '04: proceedings of the thirtieth international conference on very large data bases. VLDB Endowment, Toronto, Canada, pp 792–803
2. de Vries GKD (2012) Kernel methods for vessel trajectories. Ph.D. thesis, University of Amsterdam. http://dare.uva.nl/document/357350

3. de Vries G, van Someren M (2010) Clustering vessel trajectories with alignment kernels under trajectory compression. In: Balcázar JL, Bonchi F, Gionis A, Sebag M (eds) European conference on machine learning and knowledge discovery in databases (ECML PKDD 2010). Lecture notes in computer science, vol 6321. Springer, Dallas, TX pp 296–311
4. Douglas DH, Peucker TK (1973) Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica 10(2):112–122
5. Gudmundsson J, Katajainen J, Merrick D, Ong C, Wolle T (2009) Compressing spatio-temporal trajectories. Comput Geom 42(9):825–841
6. Schölkopf B, Smola AJ (2001) Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT Press, Cambridge
7. Shawe-Taylor J, Cristianini N (2004) Kernel methods for pattern analysis. Cambridge University Press, New York
8. Vries GKD, van Hage WR, van Someren M (2010) Comparing vessel trajectories using geographical domain knowledge and alignments. In: Fan W, Hsu W, Webb GI, Liu B, Zhang C, Gunopulos D, Wu X (eds) The 10th IEEE international conference on data mining workshops (ICDMW 2010). IEEE Computer Society, Los Alamitos, pp 209–216
9. Willems N, van Hage WR, de Vries G, Janssens JHM, Malaisé V (2010) An integrated approach for visual analysis of a multisource moving objects knowledge base. Int J Geogr Inf Sci 24(10):1543–1558

Chapter 8
Density-Based Anomaly Detection in the Maritime Domain

Jeroen Janssens, Eric Postma, and Jaap van den Herik

8.1 Detecting Anomalous Events in the Maritime Domain

Human operators monitoring maritime safety and security typically watch a large graphical display on which all vessel movements in the coastal region are plotted (see Chap. 2). Any unexpected deviation from normality should be detected by the operators. Such deviations from normality in the real world are generally referred to as "anomalities". Despite visual aids, anomalies may go unnoticed by human operators due to two cognitive limitations. The first cognitive limitation is that human observers are bad at maintaining vigilance for a sustained period of time [13]. The second cognitive limitation is that humans may be blind to visual changes due to attentional limitations [9].

Computers do not suffer from these limitations. Maintaining vigilance and monitoring large volumes of data are the hallmarks of computers. Of course, in comparison with human operators, computers fall short in understanding the "gist" of maritime situations. The situation awareness of maritime patterns by experienced operators relies largely on knowledge and familiarity with vessels, sea lanes, rules and regulations, the weather, and so forth. An important lesson from the early days of artificial intelligence is that such common sense or expert knowledge is very difficult to program into computers. Simply specifying all maritime knowledge in terms of rules leads to a system that has difficulty dealing with the uncertainties of the real world. These uncertainties arise, for instance, from incomplete or wrong information, noisy sensor readings, or weather forecasts. Given these considerations, the best way to proceed is to let the computer take over the tasks requiring vigilance and cognitive processing power and to leave the interpretation of the situation largely to the operator. The existence of uncertainties in the maritime

J. Janssens (✉) • E. Postma • J. van den Herik
Tilburg center for Cognition and Communication (TiCC),
Tilburg University, Tilburg, The Netherlands
e-mail: jeroen@jeroenjanssens.com; eric.postma@gmail.com; jaapvandenherik@gmail.com

P. van de Laar et al. (eds.), *Situation Awareness with Systems of Systems*,
DOI 10.1007/978-1-4614-6230-9_8,

domain dictates the use of probabilistic methods, which are now commonplace in artificial intelligence [2]. We focus on a particular class of probabilistic methods for anomaly detection, called the density-based methods [10].

The outline of the rest of this Chapter is as follows. Section 8.2 describes how outlier-detection tasks are represented in density-based methods. Then, in Sect. 8.3 an overview is given of existing density-based outlier detection methods. Section 8.4 presents the SOS outlier-detection method. The outlier-detection performances achieved by the SOS method are reported in Sect. 8.5. Finally, Sect. 8.6 concludes with the statement that the SOS method provides an outlier-detection method that can be successfully applied in a wide variety of domains.

8.2 Representation Space

In so-called density-based statistical methods, maritime objects (e.g., vessels) and events (e.g., vessel turns) are generally represented as points in a (potentially high-dimensional) representation space. The dissimilarity of objects or events is represented by distance. Anomalies may manifest themselves as points that are distant from all other points, so-called "outliers". To sketch a more concrete picture of statistical density-based methods, we consider, as an example, a straightforward two-dimensional representation space were the axes, represent the *features speed over ground* and *rate of turn* of vessels. Figure 8.1 shows such a representation space.

Let us suppose that all but one vessels form a cluster. In other words: all but one vessel have approximately the same speed over ground and rate of turn. The odd-one-out vessel is separated from the cluster by a considerable distance, indicating that the speed over ground and/or rate-of-turn of this vessel differs considerably from that of the other vessels. Figure 8.1 displays such a situation. The clustered points are the inliers (open circles). The outlier (asterisk) is separated from the cluster. In this particular example, the vessel associated with the asterisk is an

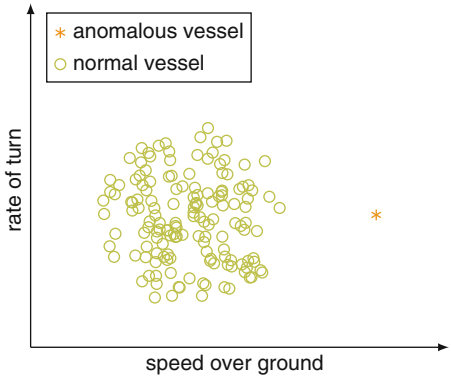


Fig. 8.1 Example of a two-dimensional representation space where the points (*open circles* and *asterisk*) represent combinations of the speed over ground (*horizontal axis*) and rate-of-turn (*vertical axis*) of vessels

outlier because it has an exceptionally large speed over ground. Its rate of turn is not anomalous, because many other vessels share approximately the same value on this feature. In realistic cases, points are anomalous on more than one feature and the detection of outliers requires taking into account multiple features, rather than just one.

The automatic detection of the odd-one-out vessel typically relies on a measure of distance, for the obvious reason that an outlier is by definition distant from all other points. The large range of outlier detection methods differ in their measurement and weighting of distance.

It is important to note that the definition of the features is crucial to the success of statistical outlier detection methods. Domain experts should be involved in the choice of the features that define the representation space. The features can be elementary, such as, the speed over ground and the rate of turn, or they can be abstractions that are known to be relevant for outlier detection, e.g., the degree to which a vessel is on a collision course with another vessel. Generally, domain experts have a good intuition about the types of information relevant to the task at hand. This intuition guides the choice of features. A useful representation for outlier detection in the maritime domain is described in Chap. 7. The number of features determines the dimensionality of the representation space and should be large enough to include the relevant information, but not too large because this hampers the ability to learn from the data [2].

8.3 Density-Based Outlier Detection Methods

This section reviews existing outlier detection methods that operate on points in representation spaces.

8.3.1 Traditional Statistical Outlier Detection

Traditional statistical outlier detection methods assume that points are normally distributed (i.e., the density of data points has a bell shape) and compute the average (center of the bell, μ) and standard deviation (half-width of the bell, σ) [1]. Figure 8.2 shows an example of normally distributed points on a line (i.e., a one-dimensional representation space). The bell-shaped curve represents the density of points at each position. The height of the curve is proportional to the number of points with that value of x, i.e., the feature of interest, e.g., the speed of a vessel. (x has average value μ and standard deviation σ.) The inset of Fig. 8.2 shows an enlarged view of the tail of the curve where at $x = 3\sigma$ inliers (open circles) are separated from outliers (asterisks). Statistical text books often define a point as an outlier when its distance to the average μ is more than m standard deviations. Figure 8.2 illustrates an example in which all points within a distance of 3 standard

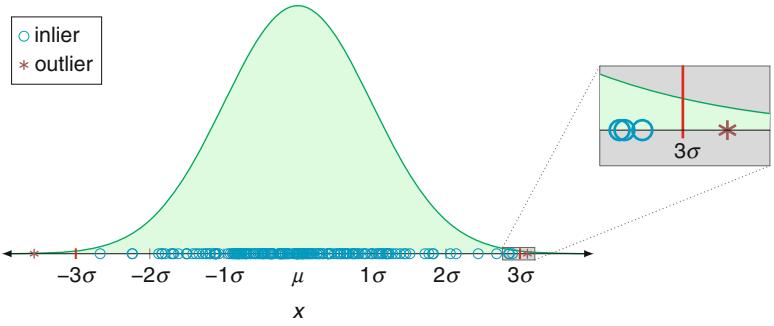


Fig. 8.2 Illustration of normally distributed points on a *line*. The bell-shaped *curve* represents the density of points at each position. The height of the *curve* is proportional to the number of points with that value of x, i.e., the feature of interest, e.g., the speed of a vessel. (x has average value μ and standard deviation σ.) The inset shows an enlarged view of the tail of the *curve* where at $x = 3\sigma$ inliers (*open circles*) are separated from outliers (*asterisks*)

deviations of the mean ($\mu \pm 3\sigma$) are considered to be inliers (represented by the open circles), those at larger distances are identified as outliers (represented by asterisks). This traditional outlier detection method is at the core of a large variety of statistical outlier detection methods. In application domains where the normality assumption holds, it offers an effective means to detect anomalies.

The main limitation of the traditional outlier detection methods is the assumption of normality, i.e., they assume that the distribution of points has a bell shape. In the maritime and many other real-world domains, data points are rarely normally distributed. Often, data points are distributed heterogeneously over space. For a single feature, the density of points does not form a single bell shape, but either multiple separated bell shapes, or totally different shapes. In two- (and higher) dimensional representation spaces, heterogeneous distributions are characterized by regions with many points (dense regions) that are interspersed with regions with few or no points (sparse regions). In terms of our example, dense regions correspond to vessels with frequently occurring speed over ground - rate of turn combinations and sparse regions correspond to the rare or no occurrence of vessels with associated speed over ground - rate of turn combinations.

8.3.2 Modern Statistical Outlier Detection: The LOF Method

Modern statistical outlier detection methods do not impose normality and deal with density variations by taking the local density into account. The most prominent density-based outlier detection method is the *Local Outlier Factor* (LOF) method [3], which originates from the domain of Knowledge Discovery and Data

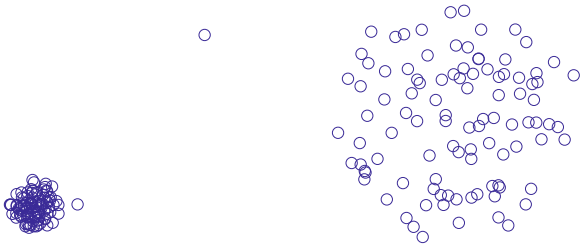


Fig. 8.3 Illustrations of two clusters of points, one with a high density (*lower left corner*) and one with a low density (*upper right corner*). Both clusters have a single outlier, but their distances from the clusters differ. LOF is capable of identifying both outliers

Mining [4]. The essence of the LOF method is that it compares the local densities of neighboring points. The local density of a point is a measure of the number of nearby points, i.e., the number of points in a predefined fixed-size spatial neighborhood. In a two-dimensional representation space, the neighborhood of a point is typically defined as a circular region around the point. A point located within a sparse region has a small local density, whereas a point located in a dense region has a high local density. The LOF method computes for each point p, an outlier value, called the Local Outlier Factor. This outlier value is obtained by dividing the averaged local densities of the points in the neighborhood (spatial vicinity) of point p by the local density of point p itself. If LOF has a value smaller or (approximately) equal to 1, the local density of point p is larger or (approximately) equal to the averaged local densities of the points in its neighborhood and the point is considered to be an inlier. Alternatively, if LOF has a value that is (much) larger than 1, the density in the neighborhood of point p is much higher than the density of the point itself, indicating that point p is an outlier.

The main advantage of the LOF method is that it can detect outliers in heterogeneous distributions of points. Returning to our two-dimensional maritime example, we consider the case of two spatially separated clusters of points representing two types of vessels, type A and type B, shown in Fig. 8.3. The speed over ground and rate of turn values of type A vessels have a small variation, whereas the speed over ground and rate of turn values of type B vessels have a large variation. As a result, the type A and B vessels give rise to clusters with high and low densities, respectively. For a type A vessel to be considered an outlier it has to be separated a certain minimal distance d_A from the type A cluster. Similarly, for a type B vessel to be considered an outlier it has to be located a certain minimal distance d_B from the type B cluster. The value of d_A is smaller than the value of d_B, because type A vessels have smaller variations in their rate of turn and speed over ground values than type B vessels. Where a purely distance-based outlier detection method would fail to take such density variations into account, the LOF method is able to identify outliers of both types.

Despite its widespread use, the LOF method suffers from two related drawbacks. The first drawback is that the LOF value is difficult to interpret. In order for a point to be an outlier, the LOF value should be much larger than 1, but how much larger depends on the problem at hand. Complex real-world domains, such as the maritime domain, are characterized by heterogeneous point distributions with unknown densities, and therefore pose a problem for the interpretation of the output of the LOF method. The second drawback is that the LOF method has no clear probabilistic foundation. As a result, LOF values cannot be interpreted in terms of probabilities. Operators assessing anomalies in maritime safety and security, would be much helped if they could assess the probability of a point being an outlier. For instance, when confronted with multiple outliers, probabilities allow them to weigh the costs of action (e.g., intercepting a vessel) against the costs of a false detection.

In recent years, a large number of density-based variants of the LOF method have been proposed. We mention three examples: the Nearest Neighbor Data Description (NNDD) method [12], the Local Correlation Integral (LOCI) method [8], and Least-Squares Outlier Detection (LSOD) method [6]. These three methods attempt to improve upon LOF in several respects, but they all suffer from the aforementioned two limitations. In the following section, we present our Stochastic Outlier Selection method, a density-based outlier detection method that does not suffer from these two limitations and we evaluate its performance by comparing it to the performances of LOF, NNDD, LOCI, and LSOD.

8.4 The Stochastic Outlier Selection Method

The Stochastic Outlier Selection (SOS) method [5] relies on three principles: (1) dissimilarity representation, (2) soft neighborhoods, and (3) outlier probabilities. The following three subsections describe these principles in detail.

8.4.1 Dissimilarity Representation

The SOS method relies on dissimilarities between points. Dissimilarities are proportional to the distances between pairs of points. The representation space is sometimes called a similarity space, because two vessels with similar speed over ground and rate of turn values are represented by nearby points, and two vessels with dissimilar values are represented by distant points. In representation space, vicinity translates into similarity, and distance into dissimilarity.

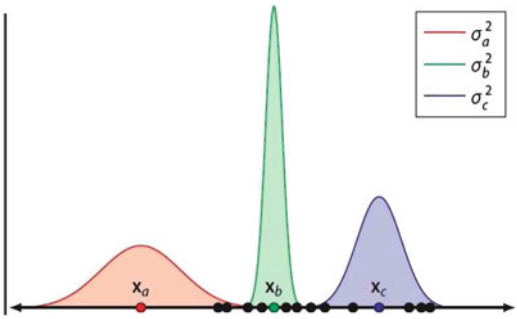


Fig. 8.4 Illustration of the bell shaped functions (soft neighborhoods) associated with three points, the *circles* labeled x_a, x_b, and x_c. The widths of the neighborhoods are determined by the local density, i.e., the number of neighboring points. The *solid circles* represent other points for which the soft neighborhoods are not drawn

8.4.2 Soft Neighborhoods

The SOS method does not treat all similarities equally. Inspired by insights from cognitive psychology [11], the similarity between two points separated by a distance d is given by a bell-shaped function centered at $d = 0$. In the domain of cognitive psychology, these points may represent, for instance, faces and the similarity space may be defined by two or more facial features (e.g., length of nose, size of mouth). The maximum similarity (top of the bell-shaped function) is obtained when two points are the same ($d = 0$, i.e., same lengths of nose and sizes of mouth). With growing distance between both points ($d > 0$, different lengths of nose and sizes of mouth), the similarity falls off towards zero (tail of the bell-shaped function). According to Shepard, the bell-shaped function is a universal law that relates distance to similarity [11]. In the cognitive psychology domain, the function returns the probability that two points (faces) fall in a region of representation space that are treated equally in terms of similarity judgment ("same face", "different face"). Shepard's similarity function is not restricted to faces, it applies to a wide variety of mental representation [11].

The bell-shaped function used in the SOS method can be interpreted as a soft version of the "hard" neighborhood used in the LOF and related methods. In a hard neighborhood, neighbor-ship changes at the circular neighborhood boundary from "neighbor" to "no neighbor". In the soft neighborhood of the SOS method, neighboring points have a neighbor-ship value N_{val} that varies from a maximum value for $d = 0$ ($N_{SOS} = 1$, top of the bell-shaped function) towards zero values of neighbor-ship for very large values of d ($N_{SOS} \to 0$, tail of the function). In the SOS method, the widths of the soft neighborhoods centered at each point are automatically set to values to ensure that all points have the same number of neighboring points. Figure 8.4 illustrates this for a one-dimensional representation space, i.e., a line. For three points, x_a, x_b, and x_c, the associated bell-shaped soft neighborhoods are drawn. The widths of the neighborhoods depend on the local density of points. If the local density is large, the neighborhood is small, whereas if the local density is small the neighborhood is large. Through the automatic scaling of the neighborhood, the SOS method deals effectively with density variations in the data. Hence, it can deal with heterogeneous densities.

8.4.3 Outlier Probabilities

To determine the probability of a point being an outlier, the SOS method examines for each point to what extent it is part of the soft neighborhood of all other points. If a point is highly dissimilar from all other points, it is located in the tails of all the associated soft neighborhoods. Being located in a tail implies a very low neighborhood-ship value, N_{SOS}, that is near to zero. In the formal definition of the SOS method, the neighborhood-ship values are expressed by the term $(1 - N_{SOS})$, where being located in a tail translates to a value that is near to one. The outlier probability of the i-th point indexed, $P_{outlier}(i)$ is proportional to the product of all these neighborhood-ship terms and is formally defined as:

$$P_{outlier}(i) = \prod_{j=1, j \neq i}^{K} (1 - N_{SOS}(j)), \tag{8.1}$$

where K is the total number of points and $N_{SOS}(j)$ is the neighbor-ship value of the j-th point.

8.5 Performance of the SOS Method

We evaluated the performance of the SOS method by comparing it to the performance of state-of-the-art outlier detection methods. Two such comparative evaluations were performed: one qualitative evaluation on artificial datasets and one quantitative evaluation on realistic datasets. In all evaluations, the parameters of the outlier detection methods were optimized to yield the best performance.

8.5.1 Evaluation on Three Artificial Datasets

To get some insights into the performances of the SOS method in comparison to the other outlier-detection methods LOF, NNOD, LOCI, and LSOD, we defined three different artificial two-dimensional datasets: *Banana*, *Densities*, and *Ring*. The Banana dataset consists of a banana-shaped cluster of points. The Densities dataset consists of two separated circular clusters of points with different densities, and the Ring dataset contains points arranged in a ring-shaped form. Applying an outlier-detection method to the Banana dataset tests if distance from a cluster of points affects the outlier value appropriately. Applying it to the Densities dataset tests if the method takes the different densities into account. Finally, applying the method to the Ring dataset tests if points inside and outside the ring are evaluated similarly. For the Banana dataset, the outlier values assigned to points should vary with distance from the shape of the banana and become gradually larger with

increasing distance from the points. For the Densities dataset, the rate of change from blue to red should be slower for the low-density cluster than for the high-density cluster. Finally, for the Ring dataset, outliers within the ring should be treated similarly to outliers outside the ring.

Figure 8.5 displays the representation spaces of the three datasets (the three columns) with the color-coded outlier values superimposed. The white dots are the points forming the datasets. The top row shows for SOS the outlier values (probabilities) assigned to each representation-space location. The colors range from dark blue (inliers; smallest outlier value or probability $P_{outlier} = 0$) to dark red/brown (outliers; largest outlier value or probability $P_{outlier} = 1$). On the Banana dataset, the SOS method assigns outlier values that vary smoothly with the shape of the banana and become gradually larger with increasing distance from the points. For the Densities dataset, the rate of change from blue to red is appropriately slower for the lower left cluster (which has a low density) than for the upper right cluster (which has a high density). For the Ring dataset, outliers within the ring are treated similarly to outliers outside the ring.

The bottom four rows of the figure illustrate how other state-of-the-art methods assign outlier values to locations in similarity space. For the Banana dataset, the outlier values generated by LOCI and LSOD fail to follow the banana shape of the points. For the Densities dataset, the other methods yield quite different outlier-value assignments. Finally, for the Ring dataset, all other outlier detection methods fail to treat interior and exterior ring locations equally in terms of outlier value assignment.

These qualitative evaluations show that the different methods behave differently on different data distributions. We now turn to a quantitative assessment of their performances on realistic datasets.

8.5.2 Performance Evaluation on Realistic Datasets

Our quantitative evaluation aims to assess the outlier detection performance of the SOS method in comparison with its main alternatives. In practical outlier-detection tasks, a wide variety of heterogeneous point distributions may arise. To ensure generality of our comparative evaluation, we decided to select 18 datasets, each from a completely different realistic application domain. We evaluated the outlier-detection performance in terms of what we call the "weighted AUC", a performance measure that takes into account the detection rate and the false positive rate and expresses the outlier-detection performance on a scale ranging from 0 (worst performance) to 1 (best performance). Figure 8.6 displays a plot of the results of the comparative evaluation. For each outlier-detection method, it shows the weighted AUC (vertical axis) achieved on each dataset (horizontal axis). The curves connect the performances of a single method. The SOS method achieves the best performance overall, because the curve associated with the SOS method (purple curve with diamond markers) is almost always on top.

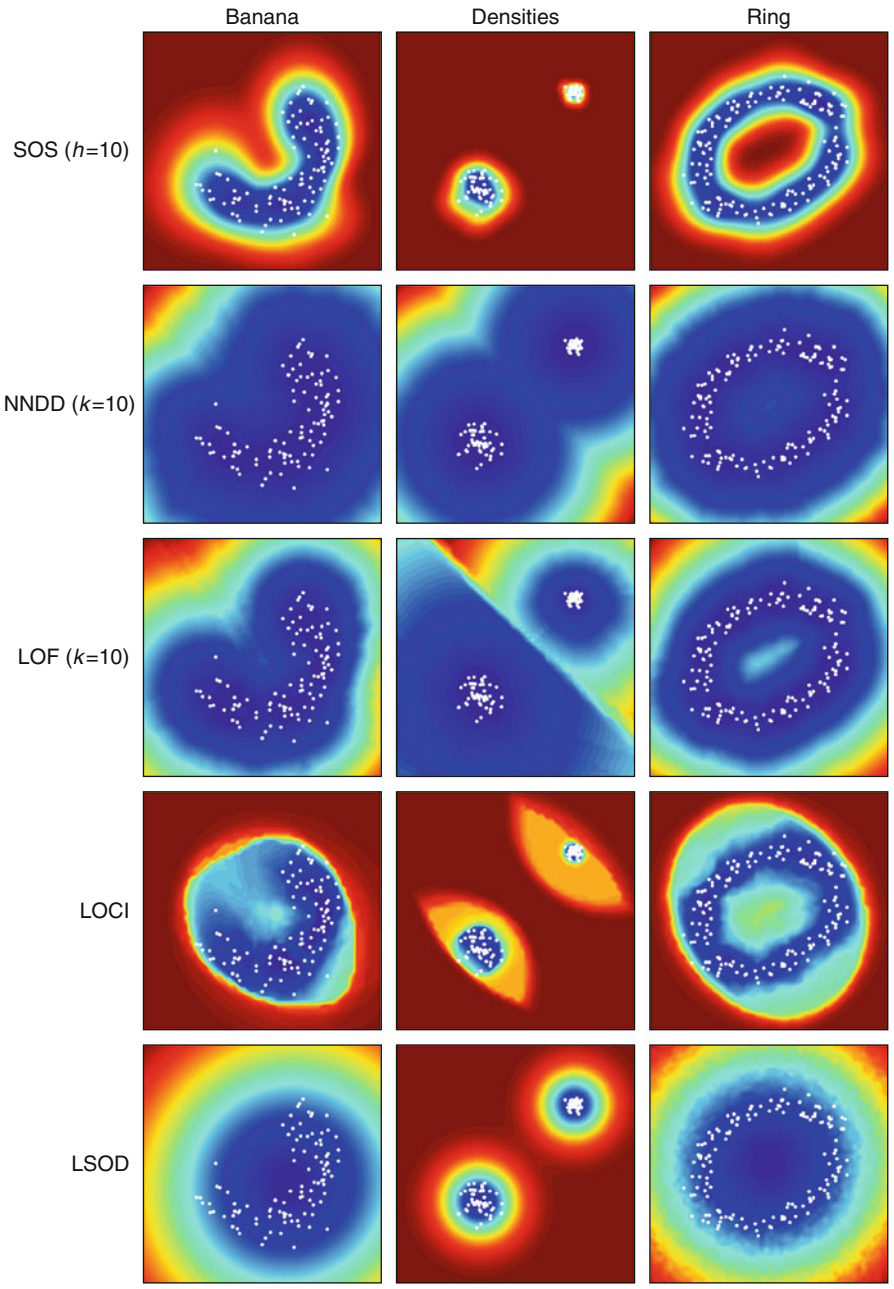


Fig. 8.5 Qualitative (visual) evaluation of outlier scores assigned to three datasets (columns) by the SOS method (*top row*) and four other state-of-the-art outlier detection methods (*bottom four rows*). Each *square* shows a two-dimensional representation space containing points (*white dots*). All other locations are colored according to the outlier value generated for that location. Outlier values are color-coded and range from *dark blue* (inliers) to *dark red/brown* (outliers)

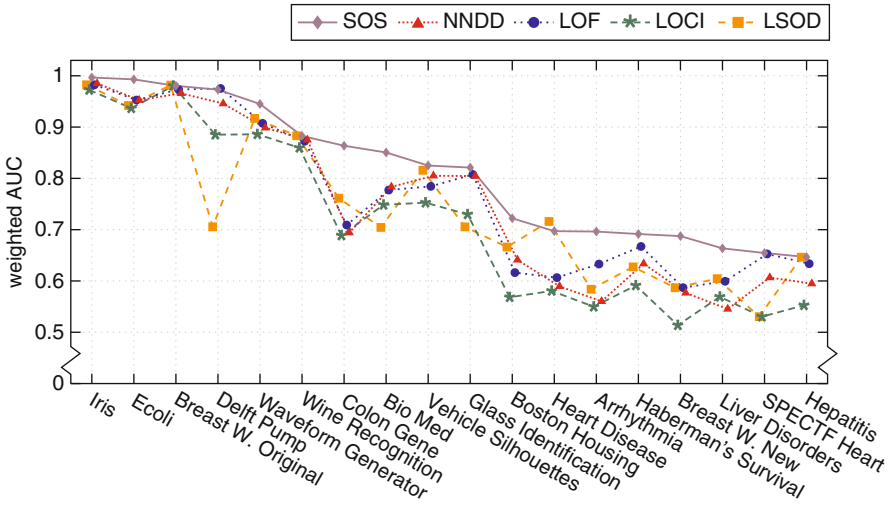


Fig. 8.6 Comparative evaluation of the SOS method and four competitive methods (NNDD, LOF, LOCI, and LSOD) on 18 realistic datasets. The outlier-detection performance is expressed in terms of the weighted AUC which ranges from 0 (worst performance) to 1 (best performance)

	Outlier-selection algorithm				
	SOS	NNDD	LOF	LOCI	LSOD
Average AUC	**0.811**	0.748	0.763	0.716	0.742
Average rank	**1.250**	3.444	2.833	4.639	2.833

Fig. 8.7 Numerical summary of the comparative evaluation of the SOS method and four competitive methods. The average AUC is the weighted AUC averaged over all 18 datasets. The average rank is obtained using a statistical method that determines the ranking of the methods on the basis of their performances. Smaller ranks correspond to better performing methods

A numerical summary of the results is presented in Fig. 8.7. The row labeled "Average AUC" lists the weighted AUC averaged over all 18 datasets. The row labeled "Average rank" specifies the ranks of the methods as obtained from a statistical method [7] that determines the ranking of the methods on the basis of their performances. The statistical method is necessary because we compare average performances of outlier-detection methods and we would like to assess the probability that differences in performance are due to chance. Smaller ranks correspond to better performing methods. The SOS method outperforms all other methods in terms of average AUC and achieves the highest rank.

8.6 Discussion and Conclusion

The SOS method has been shown to provide the best overall performance on our selection of 18 realistic datasets, as compared to state-of-the-art outlier detection methods. In addition to this result, the SOS method has an important advantage over existing density-based methods. It provides easily interpretable outlier values that correspond to probabilities. When confronted with many (potential) outliers, operators working in the maritime domain (or any other realistic domain) may prioritize the outliers using their associated probabilities, by dealing with the most probable outlier first.

It is a well-known fact in machine learning that there is no single best method for a given dataset or application domain. Similarly, we do not claim that the SOS method is the best method of choice for all domains. We have observed that the SOS method often, but not always, outperforms competitive methods.

Although we have succeeded in developing a density-based outlier-detection algorithm that performs well in comparison to state-of-the-art algorithms, a more extensive evaluation in the maritime domain has still to be performed. Provided that the maritime representation space is defined in cooperation with domain experts, we are confident that the SOS method will be successful in detecting outliers. We conclude that the SOS method provides an outlier-detection method that can be successfully applied in a wide variety of domains.

Acknowledgements This research has been carried out as a part of the POSEIDON project at Thales under the responsibilities of the Embedded Systems Institute (ESI). This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

References

1. Barnett V, Lewis T (1994) Outliers in statistical data. Wiley series in probability and mathematical statistics, 3rd edn. Wiley, Chichester
2. Bishop CM (2006) Pattern recognition and machine learning. Series in information science and statistics. Springer, New York
3. Breunig MM, Kriegel HP, Ng RT, Sander J (2000) LOF: identifying density-based local outliers. ACM SIGMOD Rec. Dallas, TX 29(2):93–104
4. Fayyad UM, Piatetsky-Shapiro G, Smyth P (1996) Knowledge discovery and data mining: towards a unifying framework. Knowl Discov Data Min 82–88
5. Janssens JHM, Huszar F, Postma EO, van den Herik HJ (2012) Stochastic outlier selection. Technical report TiCC TR 2012-001, Tilburg University, Tilburg Center for Cognition and Communication, Tilburg, The Netherlands
6. Kanamori T, Hido S, Sugiyama M (2009) A least-squares approach to direct importance estimation. J Mach Learn Res 10:1391–1445
7. Nemenyi P (1963) Distribution-free multiple comparisons. Ph.D. thesis, Princeton
8. Papadimitriou S, Kitagawa H, Gibbons PB, Faloutsos C (2003) LOCI: fast outlier detection using the local correlation integral. In: Proceedings of the 19th international conference on data engineering, Bangalore, India, pp 315–326

9. Rensink RA, O'Regan JK, Clark JJ (1997) To see or not to see: the need for attention to perceive changes in scenes. Psychol Sci 8:368–373
10. Riveiro M, Falkman G, Ziemke T (2008) Improving maritime anomaly detection and situation awareness through interactive visualization. In: 11th international conference on information fusion, 2008. IEEE, Piscataway, pp 1–8
11. Shepard RN (1987) Toward a universal law of generalization for psychological science. Science 237(4820):1317–1323
12. Tax DMJ (2001) One-class classification: concept-learning in the absence of counter-examples. Ph.D. thesis, Delft University of Technology, Delft, The Netherlands
13. Warm JS, Parasuraman R, Matthews G (2008) To see or not to see: the need for attention to perceive changes in scenes. Hum Factors 50:433–441

Chapter 9
Analyzing Vessel Behavior Using Process Mining

Fabrizio M. Maggi, Arjan J. Mooij, and Wil M.P. van der Aalst

9.1 Introduction

Maritime safety and security aims at preventing accidents and activities such as illegal immigration, terrorist attacks, smuggling, piracy, and illegal pollution. Electronic sensors such as radars and AIS (Automatic Identification System, [5]) receivers are used for collecting data about the vessels in a certain geographical area. When using AIS receivers, every vessel periodically broadcasts messages that report information such as vessel identifier, vessel type (e.g., passenger ship, cargo ship and military vessel), position, speed, destination, ship dimensions, and navigational state (e.g., *moored*, *under way using engine* and *not under command*). To detect suspicious vessel behavior, innovative methodologies are needed to analyze these large amounts of data.

In this chapter, we investigate the use of *process mining* [16] for analyzing the behavior of vessels, and in particular for detecting anomalies, i.e., deviations from the normal behavior. Process mining is usually applied in the context of business process management to analyze business processes based on event logs. An event log (see the example in Fig. 9.1) is a list of events, and each event refers to a process instance identifier (in Fig. 9.1, a process instance corresponds to a specific vessel identifier) and to an activity. The process instances are considered to be independent, and all events belonging to the same process instance are ordered. An event log can also contain a timestamp specifying when an event has occurred.

F.M. Maggi (✉) • W.M.P. van der Aalst
Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands
e-mail: F.M.Maggi@tue.nl; W.M.P.v.d.Aalst@tue.nl

A.J. Mooij
Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands

Current affiliation: Embedded Systems Institute, Eindhoven, The Netherlands
e-mail: arjan.mooij@esi.nl

P. van de Laar et al. (eds.), *Situation Awareness with Systems of Systems*,
DOI 10.1007/978-1-4614-6230-9_9,

Vessel identifier	Timestamp	Activity
35654423	30-12-2010, 11:02	at anchor
35654423	31-12-2010, 10:06	under way using engine
35654423	01-01-2011, 15:12	moored
35654423	06-01-2011, 11:18	under way using engine
35654423	07-01-2011, 14:24	restricted manoeuvrability
35654485	30-12-2010, 11:32	moored
35654485	30-12-2010, 12:12	under way using engine
35654485	30-12-2010, 14:16	restricted manoeuvrability
35654485	05-01-2011, 11:22	moored
35654485	08-01-2011, 12:05	under way using engine
35654521	30-12-2010, 14:32	moored
35654521	30-12-2010, 15:06	under way using engine
35654521	30-12-2010, 16:34	at anchor
35654521	06-01-2011, 09:18	under way sailing
35654521	06-01-2011, 12:18	constrained by her draught
35654521	06-01-2011, 13:06	under way using engine
35654521	08-01-2011, 11:43	under way sailing
35654521	09-01-2011, 09:55	moored
35654521	15-01-2011, 10:45	under way sailing

Fig. 9.1 Fragments of AIS event logs for three vessels based on changes in navigational state

In the context of process mining, the number of activities is generally assumed to be limited. Hence, in contrast to the techniques for anomaly detection described in Chap. 8, it is not natural to apply process mining by using as activities the exact geographical positions of vessels. A possibility would be to translate the positions into a number of areas. However, as a running example we focus on the navigational state of the vessels instead, resulting in event logs like the one shown in Fig. 9.1. The navigational state does not change frequently, and hence we only consider for each vessel the *changes* in the navigational state; i.e., the event *under way using engine* marks the moment that a vessel changes its navigational state to *under way using engine*.

To analyze the vessel behavior, we apply the two-phase approach from Fig. 9.2. In the first phase, *Discovery*, we use historical AIS data to extract a *reference model* of the normal behavior of vessels. To avoid being too specific (making a reference model per individual vessel), or being too general (making one reference model for all vessels), we aim for one reference model per vessel type (that can be derived from AIS messages). The discovery techniques from this chapter can do this automatically. The discovered models can be validated and adapted by domain experts to completely fit their needs.

In the second phase, *Monitoring*, we use monitoring techniques to compare the reference model with current data. The purpose is to assess whether the current data complies with the reference model and to provide diagnostics about any anomalies.

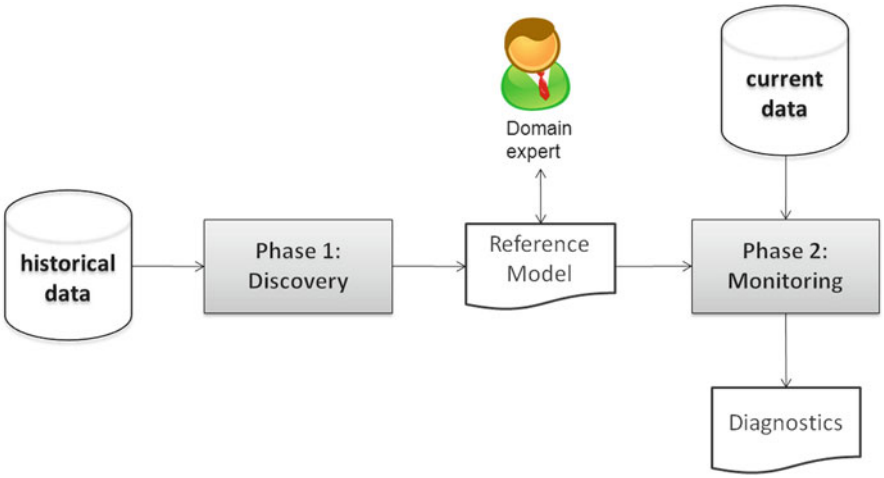


Fig. 9.2 Phased approach for analyzing vessel behavior

Traditional process mining techniques, especially those for discovery, use procedural models, which explicitly show all behaviors. For unstructured processes, this often leads to large and complex models [4, 6]. As an alternative, in this chapter we focus on constraint-based models, which describe the behavior using a (compact) set of constraints. An example of a single constraint is "whenever event *moored* occurs, eventually event *under way using engine* occurs".

Overview Section 9.2 introduces basic process mining concepts and terminology. Section 9.3 continues by introducing the *Declare* language to describe constraint-based process models. Such constraint-based models are used in Sect. 9.4 for offline discovery and in Sect. 9.5 for online monitoring. Finally, Sect. 9.6 concludes the chapter.

9.2 Process Mining

In this section, we introduce the field of process mining. Process mining [16] is a relatively young discipline developed in the context of business process management. It sits between computational intelligence and data mining on the one hand, and process modeling and analysis on the other hand. The idea of process mining is to discover, monitor and improve processes by extracting knowledge from data readily available in today's systems.

A process is a set of activities that are performed in coordination in an organizational and technical environment and jointly orchestrated to realize a goal [18]. Examples of processes include: invoice processing, insurance claim handling, online purchasing of flight tickets, mortgage application processing.

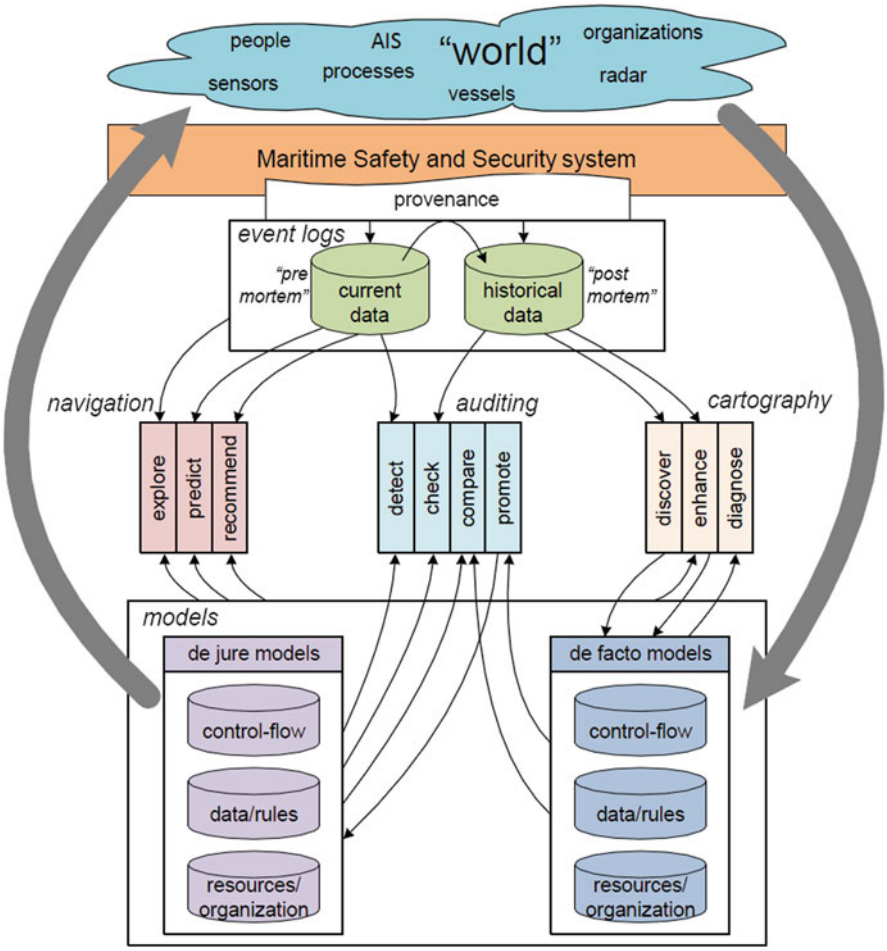


Fig. 9.3 Overview of the process mining spectrum for maritime safety and security systems

Broadly defined, processes can range from relatively simple activities such as publishing and maintaining information on a public web site to more sophisticated processes such as handling transactions in a large bank.

The starting point for process mining is an *event log*. All techniques for process mining assume that it is possible to sequentially record events. Each event refers to a *process instance* (i.e., a single execution of the process) and an *activity* (i.e., a well-defined step in the process). Event logs may store additional information about events, such as the *timestamp* of the event, the *resource* (i.e., person or device) executing or initiating the activity, or any other *data elements* recorded with the event (e.g., the size of an order).

Figure 9.3 (based on the reference framework from [16]) gives an overview of the process mining spectrum in a maritime context. The event logs are partitioned into

two kinds: *pre mortem* and *post mortem*. Pre mortem logs refer to current process instances that are ongoing; post mortem logs refer to historical process instances that have completed.

The framework also distinguishes two types of models: *de jure* and *de facto*. A de jure model is normative, i.e., it specifies how things should be done or handled. A de facto model is descriptive and its goal is not to steer or control reality; instead, de facto models aim at capturing reality.

The two large arrows in Fig. 9.3 illustrate that de facto models are derived from reality (right downward arrow) and that de jure models aim at influencing reality (left upward arrow). The framework identifies ten process mining related activities, which can be grouped into three categories: *cartography*, *auditing*, and *navigation*:

1. *Discover.* This activity is concerned with the extraction of (process) models from event logs.
2. *Enhance.* When existing process models (either discovered or hand-made) can be connected to events logs, it is possible to enhance these models. This connection can be used to repair models or to extend them.
3. *Diagnose.* This activity does not directly use event logs and focuses on classical model-based process analysis, e.g., process models can be checked for the absence of deadlocks, or process models can be simulated to estimate the effect of various redesigns on average cycle times.
4. *Detect.* This activity compares de jure models with current pre mortem data with the goal to detect deviations at runtime.
5. *Check.* Post mortem data can be cross-checked with de jure models. The goal of this activity is to pinpoint deviations and quantify the level of compliance.
6. *Compare.* De facto models can be compared with de jure models to see in what way reality deviates from what was planned or expected. No event log is used directly, but the de facto model may have been discovered using historical data.
7. *Promote.* Based on an analysis of the differences between a de facto model and a de jure model, it is possible to promote parts of the de facto model to a new de jure model. By promoting proven "best practices" to the de jure model, existing processes can be improved.
8. *Explore.* The combination of event data and models can be used to explore business processes at runtime. Ongoing process instances can be visualized and compared with similar process instances that were handled earlier.
9. *Predict.* By combining information about running process instances with models (discovered or hand-made), it is possible to make predictions about the future, e.g., the remaining execution time.
10. *Recommend.* The information used for predicting the future can also be used to recommend suitable actions (e.g., to minimize costs or time).

The *Discovery* phase in Fig. 9.2 corresponds to the *discover* activity in Fig. 9.3. The discovered models are de facto models as they aim at capturing what is really happening in the post mortem logs. After analysis by domain experts, these de facto

models become de jure models, i.e., models describing the acceptable behavior. The *Monitoring* phase in Fig. 9.2 corresponds to the *detect* activity in Fig. 9.3. The de jure models are used to analyze the pre mortem logs.

9.3 Constraint-Based Process Models

In this section, we introduce constraint-based process models, and compare them to procedural models. We also introduce the *Declare* language that we use in this chapter for describing constraint-based models.

9.3.1 Motivation

Processes can be described using two different types of models: procedural models and constraint-based models. A procedural model explicitly describes all the acceptable sequences of activities in a process. In contrast, a constraint-based model consists of a list of restrictions to be satisfied by the process.

Procedural models are more supportive for guiding a process during execution. Referring to the process spectrum in Fig. 9.4, procedural languages are more suitable to represent structured processes. In contrast, procedural models become large and complex [4, 6] when used to represent unstructured processes, e.g., when decisions are not made centrally. Constraint-based models can remain compact for unstructured processes with a low degree of predictability.

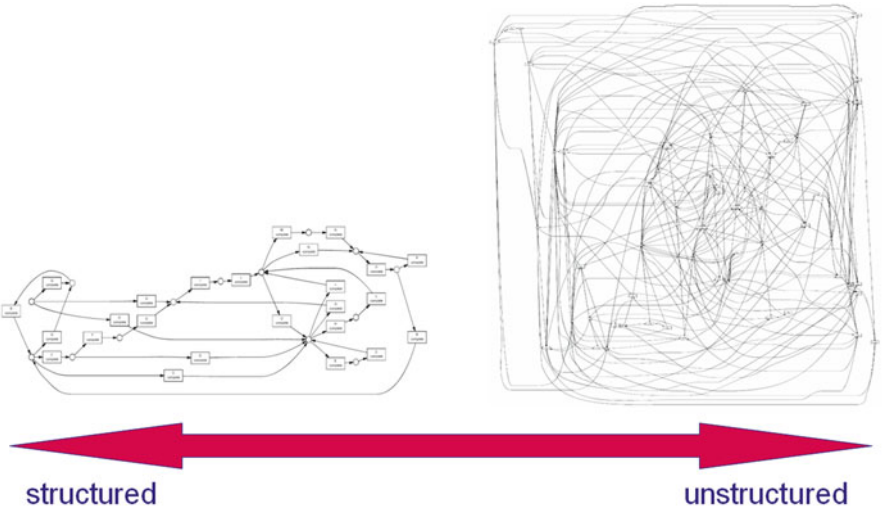


Fig. 9.4 Process spectrum: two procedural models discovered using process mining

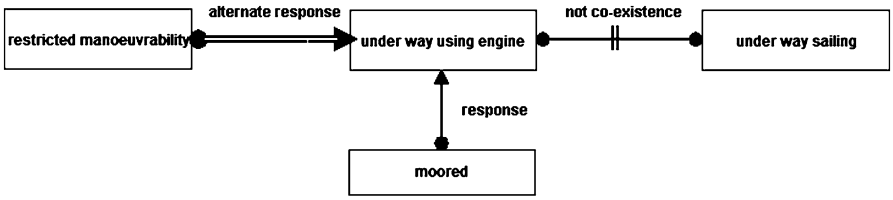


Fig. 9.5 Example *Declare* reference model for vessel type dredger

Table 9.1 Summary of basic temporal LTL operators and their semantics

Operator	Semantics
$\bigcirc\varphi$	Formula φ is satisfied in the next state
$\Box\varphi$	Formula φ is always satisfied in the future
$\Diamond\varphi$	Formula φ is eventually satisfied in the future
$\varphi \sqcup \psi$	Formula φ is satisfied until formula ψ is satisfied, and formula ψ is eventually satisfied in the future

9.3.2 Declare Language

The *Declare* language is a constraint-based language that combines a formal semantics for analysis purposes with a graphical representation for users. Figure 9.5 shows an example of a *Declare* model in the context of maritime safety and security. We use this example to explain the main concepts and refer the reader to [12, 13, 17] for more information about the *Declare* language.

The model involves four *activities* (depicted as rectangles, e.g., *under way using engine*) and three *constraints* (depicted as connections between activities). In this case, each activity relates to a change in the navigational state of a vessel. Each constraint is based on a template (e.g., *not co-existence*) that determines the graphical appearance and the semantics.

Constraints highlight mandatory and forbidden behaviors. The *not co-existence* constraint indicates that a vessel can, during its lifetime, perform events *under way using engine* or events *under way sailing* but not a combination. The *response* constraint indicates that, after any event *moored*, each vessel must eventually perform an event *under way using engine*. Finally, the *alternate response* constraint indicates that, whenever a vessel performs any event *restricted manoeuvrability*, it must eventually perform an event *under way using engine* without other occurrences of event *restricted manoeuvrability* in between.

The *Declare* language also has a formal semantics that enables verification and automated reasoning. The semantics of each individual constraint can be formalized as an LTL (Linear Temporal Logic) [14] formula using the temporal operators from Table 9.1 and some basic logical operators ("not" $\neg$, "and" $\wedge$, "or" $\vee$, "implies" $\Rightarrow$). The three constraints in Fig. 9.5 can be formalized as follows:

- Not co-existence constraint: $\neg(\Diamond E \wedge \Diamond S)$
- Response constraint: $\Box(M \Rightarrow \Diamond E)$
- Alternate response constraint: $\Box(R \Rightarrow \bigcirc(\neg R \sqcup E))$

where *M*, *S*, *E* and *R* respectively denote *moored*, *under way sailing*, *under way using engine* and *restricted manoeuvrability*.

9.4 Discovering Declare Models

We have developed an approach [10] to support the automated discovery of *Declare* models; see the *Discovery* phase in Fig. 9.2, and the *discover* activity in Fig. 9.3. It has been implemented as the *Declare Miner*[1] plug-in for the process mining tool *ProM*.[2] In Sect. 9.4.1 we introduce the *Declare Miner*, and in Sect. 9.4.2 and in Sect. 9.4.3 we illustrate its use on AIS data.

9.4.1 Declare Miner

The *Declare Miner* automatically discovers a *Declare* model of a process based on an event log. To guide the *Declare Miner* to any particular kind of properties, an additional input is a set of *Declare* templates. The basic *Declare Miner* then produces all constraints based on these templates that are satisfied on the event log.

By applying the *Declare Miner* to the AIS case study, we have identified three issues: (a) historical logs may be truncated, (b) some constraints may be satisfied, but trivially, (c) historical logs may contain noise. To be useful in practice, we have solved these issues in the *Declare Miner* as summarized in Table 9.2. In what follows, we briefly introduce the identified issues and their solutions.

9.4.1.1 Historical Logs May Be Truncated

Traditional business processes typically have a clearly defined begin and end point. In contrast, in the context of AIS, the processes are non-terminating, and hence the logs only contain fragments of process executions. This characteristic

Table 9.2 Advanced features of the *Declare Miner*

Synopsis	*Declare Miner* parameter
Historical logs may be truncated	Weak LTL semantics
Some constraints may be satisfied, but trivially	Percentage of interesting witnesses (PoIW)
Historical logs may contain noise	Percentage of instances (PoI)

[1] http://www.win.tue.nl/declare/declare-miner/

[2] http://www.processmining.org/

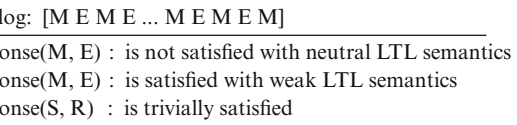


Fig. 9.6 Example log of a single passenger ship

affects the discovered *Declare* models because some constraints can be temporarily violated on the available part of a process instance, but satisfied on its continuation.

For instance, consider the event log in Fig. 9.6, where *M* and *E* denote events *moored* and *under way using engine* respectively. The *response* constraint "whenever event *moored* occurs, eventually event *under way using engine* occurs" would not be discovered using the basic *Declare Miner*. The reason is that event *moored* is the last event, and hence this constraint is not satisfied with the normal (neutral) LTL semantics. However, this constraint could be satisfied in a continuation of this log.

To address this issue, we apply the *truncated semantics* from [3]. At the end of each (prefix of an) event log, four evaluations of a constraint are possible:

- *Satisfied*: it is currently satisfied and cannot become violated in the future;
- *Possibly satisfied*: it is currently satisfied, but can become violated in the future;
- *Possibly violated*: it is currently violated, but can become satisfied in the future;
- *Violated*: it is currently violated and cannot become satisfied in the future.

In Fig. 9.6, the response constraint explained before is possibly violated at the end of the log.

Based on these four evaluations, several LTL semantics [3] have been proposed. Using the *neutral* semantics, a constraint is discovered if it is satisfied or possibly satisfied at the end of the event log. Using the *weak* semantics, a constraint is discovered if the evaluation at the end of the log is different from violated. Hence, the discussed response constraint in Fig. 9.6 would be discovered using the weak semantics. The *Declare Miner* offers the option to choose the LTL semantics (neutral or weak). For truncated logs, the weak semantics is recommended.

9.4.1.2 Some Constraints May Be Satisfied, But Trivially

Some constraints are satisfied on the log, but still do not seem to capture the behavior in the log. For instance, consider again the event log in Fig. 9.6, where *S* and *R* denote events *under way sailing* and *restricted manoeuvrability* respectively. The *response* constraint "whenever event *under way sailing* occurs, eventually event *restricted manoeuvrability* occurs" is satisfied. This constraint is even trivially (or vacuously [2,7,15]) satisfied, as these two activities do not occur in the log. As such it does not capture the behavior in this event log, and hence the user might not be interested in it.

To address this issue, the *Declare Miner* offers the option to set the parameter *Percentage of Interesting Witnesses (PoIW)* that specifies that a *Declare* constraint can be discovered only if there is a certain percentage of process instances in the log

Table 9.3 Experimental settings for passenger ships

Experiment	Template	Semantics	PoIW (%)	PoI (%)
1	Chain response	Neutral	0	100
2	Chain response	Weak	0	100
3	Chain response	Weak	5	100

where the constraint is non-trivially satisfied (these instances are called in literature "interesting witnesses" [2, 7, 15]). For example, if PoIW = 20%, a constraint will be discovered only if it is non-trivial in at least 20% of the process instances in the log.

9.4.1.3 Historical Logs May Contain Noise

The historical logs may contain some noise or even some anomalies. "Noise" is a general, technical term that, in the context of maritime safety and security, can be related to AIS messages that may be lost or corrupted during transmission, e.g., due to weather conditions or bad reception. In this case, there may be constraints that are satisfied in most process instances, but that are violated in only a few instances (due to the presence of noise), and hence they are not discovered.

To address this issue, the *Declare Miner* offers the option to set the parameter *Percentage of Instances (PoI)* that specifies that a *Declare* constraint can be discovered even if it is not satisfied for all process instances in the log. For instance, if PoI = 80%, a constraint will be discovered if at least 80% of the process instances satisfy the constraint.

9.4.2 Case Study: Passenger Ships

To illustrate the discovery of *Declare* models, in this section we aim at discovering the normal behavior of 60 vessels of type passenger ship.[3] Most of the process instances look like a regular alternation of events *under way using engine* and *moored*; see for example Fig. 9.6. In particular, we want to identify events that always appear next to each other, and focus on *chain response* constraints. An example of *chain response* constraint is "whenever event *under way using engine* occurs, event *moored* occurs next". Table 9.3 shows the settings of the *Declare Miner* for the three experiments that we discuss next.

In the first experiment, we use the basic *Declare Miner* without the advanced features from Table 9.2. That is, we use neutral semantics, we do not require any interesting witnesses, and we require that constraints are satisfied in all process

[3] All the experiments illustrated in this chapter use AIS data registered along the Dutch coast in the first week of June 2007

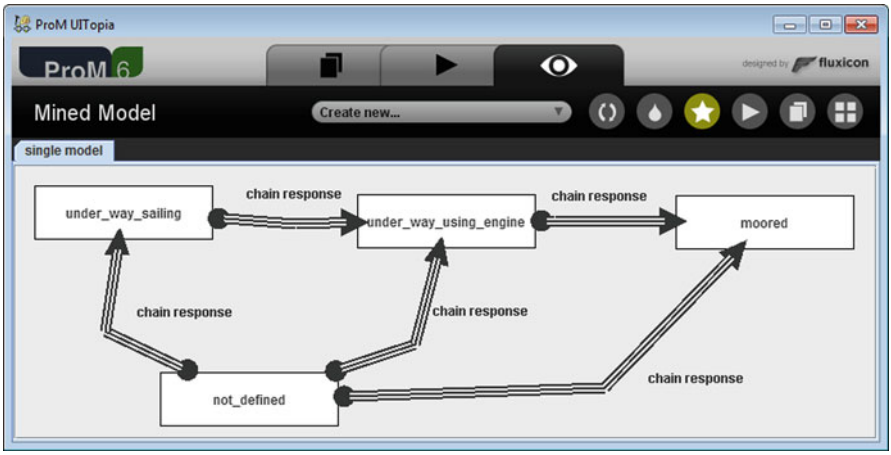


Fig. 9.7 Discovered *Declare* model for passenger ships in ProM (experiment 2 in Table 9.3)

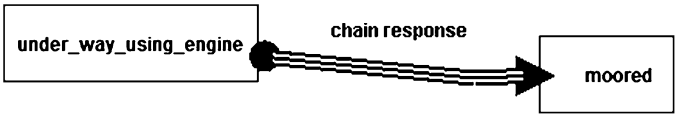


Fig. 9.8 Discovered *Declare* model for passenger ships (experiment 3 in Table 9.3)

instances. As a result, the *Declare Miner* generates an empty model: it is not possible to discover any *chain response* constraint using these settings.

In the second experiment, we take into account that the log contains truncated process instances, and hence use the weak semantics. As a result, the *Declare Miner* generates the *Declare* model shown in Fig. 9.7. This model contains five *chain response* constraints. The *chain response* constraint between events *under way using engine* and *moored* reflects the characteristic alternation of events *under way using engine* and *moored* for this vessel type. However, the discovered model contains several extra constraints (note that one of the possible values for an event according to the AIS standard is *not defined*).

In the third experiment, we require that the discovered constraints are non-trivially satisfied in at least 5% of the process instances, by setting the percentage of interesting witnesses to 5%. As a result, the *Declare Miner* generates the *Declare* model shown in Fig. 9.8. Comparing this one with Fig. 9.7, it is clear that most of the constraints from Fig. 9.7 are trivially satisfied in most of the process instances. The single constraint in Fig. 9.8 is the only one that is non-trivial in (at least) 5% of the process instances.

Table 9.4 Experimental settings for dredgers

Experiment	Template	Semantics	PoIW (%)	PoI (%)
1	Alternate response	Weak	5	100
2	Alternate response	Weak	10	100
3	Alternate response	Weak	20	90

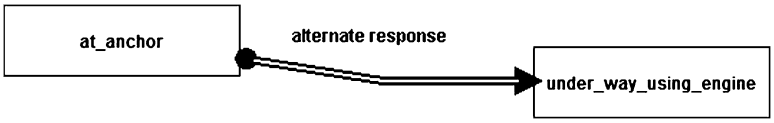


Fig. 9.9 Discovered *Declare* model for dredgers (experiment 1 in Table 9.4)

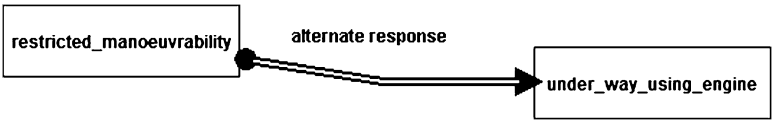


Fig. 9.10 Discovered *Declare* model for dredgers (experiment 3 in Table 9.4)

9.4.3 Case Study: Dredger

In this section, we continue by discovering the normal behavior of 56 vessels of type dredger. In particular, we focus on *alternate response* constraints. An example of an *alternate response* constraint is "whenever event *at anchor* occurs, eventually event *under way using engine* occurs without repetitions of *at anchor* in between". Table 9.4 shows the settings of the *Declare Miner* for the three experiments that we discuss next.

In the first experiment, we continue with the settings of the last experiment from Sect. 9.4.2. As a result, the *Declare Miner* generates the *Declare* model shown in Fig. 9.9, containing one alternate response constraint.

In the second experiment, we try to further increase the required number of interesting witnesses to 10% of the process instances. As a result, the *Declare Miner* generates an empty model. For the constraint discovered in the first experiment, this means that it was trivially satisfied in many process instances.

In the third experiment, we take into account that there may be some noise or anomalies in the log, and decrease the required number of instances to 90%. At the same time, we increase the required number of interesting witnesses even more. As a result, the *Declare Miner* generates the *Declare* model shown in Fig. 9.10. This single constraint is not satisfied in all process instances, but it is non-trivially satisfied in many process instances. Note that this constraint is part of the reference model in Fig. 9.5.

9.5 Monitoring Declare Models

We have developed an approach [9, 11] to support monitoring based on *Declare* models; see the *Monitoring* phase in Fig. 9.2, and the *detect* activity in Fig. 9.3. It has been implemented as the *Mobucon*[4] (*Mo*nitoring *bu*siness *con*straints) framework that has been implemented using the process mining tool *ProM*. In Sect. 9.5.1 we introduce the *Mobucon* framework, and in Sect. 9.5.2 we illustrate its use on AIS data.

9.5.1 Mobucon

The *Mobucon* framework monitors current event logs based on a reference model expressed in *Declare*. After every received event, the *Mobucon* framework reports the evaluation of all constraints. The four possible evaluations are the ones described in Sect. 9.4.1: *satisfied*, *possibly satisfied*, *possibly violated* and *violated*.

Fine-grained diagnostics are provided to the end users about the evaluation of each constraint, and in case of a violation also the reason of the violation. Apart from a single constraint becoming violated, it is also possible that two (or more) constraints become conflicting. A *conflict* indicates that there is no possible future continuation such that all the constraints become satisfied.

Also a health notion is computed for each process instance that indicates the degree of compliance to the constraints. It can be computed using different metrics, which can consider the current evaluations of the constraints as well as any weight that can be assigned to each individual constraint. For example, the *health* at some time t can be computed through the formula

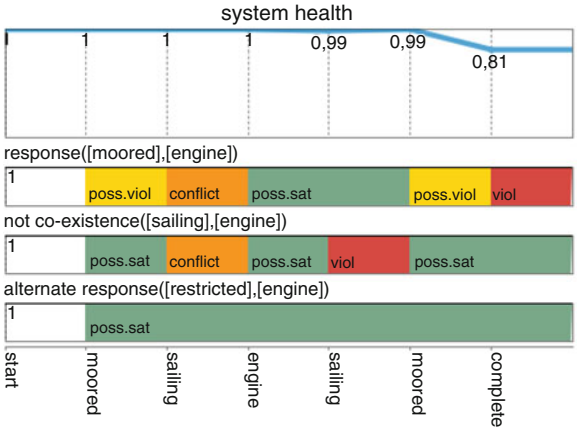
$$health(t) \quad = \quad 1 - \frac{\sum_i \#violations_i(t) \cdot weight_i}{\sum_i \#events(t) \cdot weight_i}$$

where i ranges over the constraints. For each constraint i, $weight_i$ denotes the assigned weight, and $\#violations_i(t)$ denotes the number of violations until time t; $\#events(t)$ denotes the total number of events until time t.

In the maritime safety and security domain, it is crucial to continue monitoring after any violation. To provide such continuous monitoring capabilities, *Mobucon* has several recovery mechanisms that allow a violated constraint to become non-violated again. For instance, after a violation it is possible to move the constraint to the state before the violation or to reset the constraint to the initial state.

Other monitoring techniques such as [1, 8] typically limit themselves to produce as output only a truth value representing whether the current data comply with the monitored constraints. Furthermore, monitoring usually halts as soon as a (permanent) violation is encountered.

[4] http://www.win.tue.nl/declare/mobucon/



Meaning of the colors:

- blue: satisfied;
- green: possibly satisfied;
- yellow: possibly violated;
- red: violated;
- orange: conflict.

Fig. 9.11 Monitoring a single vessel of type dredger in *Mobucon*

9.5.2 Case Study: Dredger

To illustrate the monitoring of *Declare* models, in this section we aim at monitoring the vessels of type dredger with respect to a reference model describing the normal behavior. As a reference model we use the *Declare* model from Fig. 9.5, which contains the three constraints that were explained in Sect. 9.3.2. In particular, the alternate response constraint was discovered in Fig. 9.10 based on the third experiment in Sect. 9.4.3.

Figure 9.11 shows a screenshot of the main window of *Mobucon* for a specific vessel. The events are displayed on the horizontal axis, whereas the constraints from the reference model are displayed on the vertical axis; for the sake of readability, all activity names have been abbreviated. On top, the health metric is displayed.

The first event (mentioned at the bottom after start) is *moored*, and when it occurs, the *response* constraint becomes possibly violated; in the future it requires an event *under way using engine* to become satisfied again. The second event is *under way sailing*, which leads to a conflict between the *not co-existence* and the *response* constraints. The *not co-existence* constraint forbids a future event *under way using engine*, whereas the *response* constraint requires such an event. As a recovery mechanism, the conflicting constraints are afterwards moved to the state before the violation.

The third event is *under way using engine*. The *response* constraint was expecting this one, and now becomes possibly satisfied. The fourth event is *under way sailing*. This event makes the *not co-existence* constraint violated, because both *under way using engine* and *under way sailing* have occurred. Also in this case the violated constraint is afterwards moved to the state before the violation. The fifth event is *moored*, which makes the *response* constraint possibly violated. Note that, when the process instance completes, the possibly violated constraints (in this case the *response* constraint) become violated because it is not possible to satisfy them anymore.

The displayed health trend is based on the metric from Sect. 9.5.1. The health decreases in correspondence with each violation, but conflicts do not affect the system health. In this example, violations of the *response* constraint influence the health more than violations of the *not co-existence* constraint, because of the assigned weights.

9.6 Conclusions

In the context of maritime safety and security, electronic sensors such as radar and AIS receivers provide large amounts of data that can be used to analyze the behavior of vessels. In order to work in an effective and efficient manner, operators need to extract a complete but understandable "picture" of the vessels' behaviors.

We have demonstrated how process mining techniques can contribute to this. In particular, process discovery enables users to automatically extract reference models that describe the normal behavior of vessels based on large amounts of (historical) data. Moreover, process mining can also be applied on running cases for online monitoring of data streams.

To model the behavior of vessels, we have used the *Declare* language. *Declare* is a constraint-based language that can specify complex behavior in terms of a couple of restrictions. *Declare* combines a formal semantics for analysis purposes with a graphical representation for users.

We have developed an offline discovery technique to automatically extract constraint-based reference models from logs. It allows users to guide the discovery to the specific constraint templates they are interested in. Based on experiments with AIS data, several advanced features have been implemented to improve the practical usability. Moreover, in our experiments, we have analyzed in a few seconds data sets containing up to 900 vessels and 250,000 events. This demonstrates the scalability of our approach.

We have also developed an online monitoring technique to support anomaly detection based on constraint-based reference models. In addition to providing diagnostics about any constraint violations, the framework offers various possibilities to continue monitoring after a violation has occurred, which is crucial for applications in maritime safety and security.

Acknowledgements This research has been carried out as a part of the POSEIDON project at Thales under the responsibilities of the Embedded Systems Institute (ESI). This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

The authors wish to thank Marco Montali and Michael Westergaard for their contribution in the development of the approach to monitor *Declare* models described in this chapter.

References

1. Bauer A, Leucker M, Schallhart C (2010) Comparing LTL semantics for runtime verification. J Log Comput 20(3):651–674
2. Beer I, Ben-David S, Eisner C, Rodeh Y (2001) Efficient detection of vacuity in temporal model checking. Form Methods Syst Des 18(2):141–163
3. Eisner C, Fisman D, Havlicek J, Mcisaac A, Lustig Y, van Campenhout D (2003) Reasoning with temporal logic on truncated paths. In: International conference on computer aided verification, vol 2725. Springer, Berlin/New York, pp 27–40
4. Günther CW, van der Aalst WMP (2007) Fuzzy mining – adaptive process simplification based on multi-perspective metrics. In: Business process management conference, vol 4714. Springer, Berlin/New York, pp 328–343
5. International Telecommunications Union (2001) Technical characteristics for a universal shipborne Automatic Identification System using time division multiple access in the VHF maritime mobile band, Recommendation ITU-R M.1371-1
6. Jagadeesh Chandra Bose RP, van der Aalst WMP (2009) Abstractions in process mining: a taxonomy of patterns. In: Business process management conference. Springer, Berlin/New York, pp 159–175
7. Kupferman O, Vardi MY (2003) Vacuity detection in temporal model checking. Int J Softw Tools Technol Transf 4(2):224–233
8. Leucker M, Schallhart C (2009) A brief account of runtime verification. J Log Algebraic Program 78(5):293–303
9. Maggi FM, Montali M, Westergaard M, van der Aalst WMP (2011) Monitoring business constraints with linear temporal logic: an approach based on colored automata. In: Rinderle-Ma R, Toumani F, Wolf K (eds) International conference on business process management (BPM 2011). Lecture notes in computer science, vol 6896. Springer, Berlin/New York, pp 132–147
10. Maggi FM, Mooij AJ, van der Aalst WMP (2011) User-guided discovery of declarative process models. In: IEEE symposium on computational intelligence and data mining (CIDM). IEEE Computer Society, Piscataway, pp 192–199
11. Maggi FM, Westergaard M, Montali M, van der Aalst WMP (2012) Runtime verification of LTL-based declarative process models. In: Khurshid S, Sen K (eds) International conference on runtime verification (RV 2011). Lecture notes in computer science, vol 7186. Springer, Berlin/New York, pp 131–146
12. Pesic M (2008) Constraint-based workflow management systems: shifting Controls to users. Ph.D. thesis, Beta Research School for Operations Management and Logistics, Eindhoven University of Technology
13. Pesic M, Schonenberg H, van der Aalst WMP (2007) Declare: full support for loosely-structured processes. In: IEEE international EDOC conference. IEEE, Los Alamitos, pp 287–300
14. Pnueli A (1977) The temporal logic of programs. In: Annual IEEE symposium on foundations of computer science, Providence, pp 46–57
15. Simmonds J, Davies J, Gurfinkel A, Chechik M (2010) Exploiting resolution proofs to speed up LTL vacuity detection for BMC. Int J Softw Tools Technol Transf 12(5):319–335
16. van der Aalst WMP (2011) Process mining: discovery, conformance and enhancement of business processes. Springer, Berlin Heidelberg/New York
17. van der Aalst WMP, Pesic M, Schonenberg H (2009) Declarative workflows: balancing between flexibility and support. Comput Sci Res Dev 23(2):99–113
18. Weske M (2007) Business process management: concepts, languages, architectures. Springer, Berlin/New York

Chapter 10
The Simple Event Model

Willem Robert van Hage and Davide Ceolin

10.1 Introduction

In this chapter we will give a gentle introduction to event modeling using the Simple Event Model (SEM) [5], a graph model for events and related concepts, like involved actors, places, and time. We will take the perspective of setting up a simple quantitative experiment to show how the Simple Event Model can be applied to the analysis of maritime event data. This allows us to give a structured presentation of event modeling, semantic web languages like the Resource Description Framework (RDF)[1] RDF Vocabulary Description Language (also known as RDF Schema language, or RDFS),[2] the Simple Event Model, and how to use the query language SPARQL[3] to take selections from complex event models.

We focus on a specific type of task in the field of maritime situation awareness, the analysis of past events. This is also called historical analysis, where historical means before the current moment, but not necessarily ages ago. Historical event analysis can be used to answer questions like:

Which type of ships most often violate traffic rules?
Is there a correlation between the type of piracy activity and the place in the world?
How many oil tankers have been hijacked in the past year?

[1]RDF, http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

[2]RDFS, http://www.w3.org/TR/rdf-schema/

[3]SPARQL, http://www.w3.org/TR/rdf-sparql-query/

W.R. van Hage (✉) • D. Ceolin
Web & Media Group, VU University Amsterdam, Amsterdam, The Netherlands
e-mail: w.r.van.hage@vu.nl; d.ceolin@vu.nl

P. van de Laar et al. (eds.), *Situation Awareness with Systems of Systems*,
DOI 10.1007/978-1-4614-6230-9_10,

These questions can be answered using simple statistics on counts of records of observed events. The hard part is to specify exactly which events to count and which to disregard. The specification of this selection can be seen as the core of the experimental set-up.

What this specification looks like depends on the structure of the event records. In simple cases it suffices to use a simple table format to represent events. For instance, if one takes the last question, which is the number of hijacked oil tankers in the past year, it would suffice to simply collect a table (e.g. in a spreadsheet or database) where each row stands for a piracy event instance. In this table the first column could hold the type of pirate activity (e.g. hijacking or robbery), the second column could hold the type of the victimized ship (e.g. oil tanker or cruise ship), the third column could hold the date of the event. Given this representation, the specification of the question is a selection of rows where the value in the first column equals "hijacking", the second column equals "oil tanker", and the third column holds a value between the current date and 1 year before. Answering the question entails counting the number of selected rows. If the table is stored in a relational database in a table called "event" with columns named ("event_type", "ship_type", and "timestamp") this question can be formalized in the SQL query language as:

```
SELECT COUNT(*)
FROM event
WHERE event_type = "hijacking"
  and ship_type  = "oil tanker"
  and timestamp >= DATE_SUB(CURDATE(),
                            INTERVAL 1 YEAR)
```

In real life events are often hard to fit into such a simple and clear table format. Most of the hard decisions already have to happen to make event data into the table. For example, there can be multiple ships involved in a single event; the date can be unknown; it can be known that the ship is some kind of tanker, but whether it is an oil tanker is not specified; or it can be the case that the events are specified at different levels of abstraction, such as a single observation of a ship's position (e.g. by radar or AIS message[4]) or a cross-ocean journey. One solution is to improve the data collection process, but when the data were collected by a third party that can not be influenced – a very common situation when using data from the Web or Semantic Web – this is impossible. The most difficult issue occurs when one wants to reuse event records that were collected for the purpose of answering one question to answer a related but different question which requires additional information, like which company owns the ship, which multinational owns that company, where is this company based, etc. This can require a drastic redesign of the existing table structure.

The Simple Event Model (SEM) was designed to tackle these issues of missing and extraneous bits of information and of differing levels of abstraction. The crucial

[4]AIS, http://nl.wikipedia.org/wiki/Automatic_Identification_System

property of SEM that solves all of the issues mentioned above is that it uses a graph representation of events as opposed to a table representation. This makes it possible to duplicate any property (e.g. to have more than one actor or event type), to omit any property, to interlink any part of the event with external information (e.g. to link ships with their owners that exist in some external model, or to use place descriptions from some existing geographical ontology, i.e., a description of a geographical conceptualization), and to use subclass hierarchies wherever needed (e.g. to specify that all oil tankers are tankers and all tankers are merchant vessels). This flexibility gives the developer the opportunity to postpone many experimental design decisions from the moment of data acquisition to the moment at which the question has to be turned into a formal query.

There are many event models. Some, like CIDOC-CRM by Doerr et al. [2], are large (about 140 classes, i.e., concepts), others, like the "Event Ontology" by Raimond and Abdallah [3], are small (4 classes).[5] The Simple Event Model is a relatively small model with 16 classes in total of which 4 are by far the most important ones, sem:Event, sem:Actor, sem:Place, and sem:Time. These four core classes outline the basic idea underlying SEM, the motto *"Who did what where and when?"* (*"Why"* and *"How"* are omitted, because the answers to these questions are usually complex stories by themselves that require their own representation languages). The other classes are used to further specify events and their context in various ways. SEM is defined using the Resource Description Framework Schema language (RDFS) and a few bits of Web Ontology Language (OWL).

The kind of cases where SEM is the most useful are those where the level of abstraction is high (e.g. ships entering a harbor, not sensor readings) and the set of event types, actor types or place types changes often, and where information sources are being used that are not under the user's control (e.g. crowd sourced observation with an open vocabulary of types). The interpretation of ship behavior by combining ship tracks with external sources, like harbor logs, GeoNames,[6] and extra ship information from Web sites, as done in the Maritime Safety and Security, is a good example of such a case.

In Sect. 10.4 we will walk through all the constructs of SEM step by step, from the basic notion of event along actors, places, time, and types to the complex notions of temporary and subjective validity of properties. All constructs will be illustrated with examples taken from two running example event models: a real example from the domain of international maritime piracy monitoring, a palm oil tanker being hijacked in Indonesia, and a fictional example from the domain of harbor situation awareness, a tug towing an oil tanker into a harbor. These are described in their entirety, but without attention to the details in Sects 10.2 and 10.3. The details will be explained in Sect. 10.4 when the meaning of the SEM constructs is discussed. We choose to use two examples for multiple reasons. We want at least one example that is based on open data so that the reader can look up and try out the examples in

[5]For more information about these and related ontologies see the extensive discussion in [5].

[6]GeoNames, http://www.geonames.org/

this chapter (how this can be done in practice will be shown in Sect. 10.5). However, the dataset that is most suitable for this purpose, the Linked Open Piracy data set,[7] does not use all SEM constructs, so we use the second fictional example of the tugged oil tanker to cover the rest of the SEM model.

10.2 Example 1: The Hijacking of a Palm Oil Tanker in Indonesia

The first example we use to illustrate the SEM constructs is an example from the Linked Open Piracy[8] data set (LOP) containing piracy reports from the International Chamber of Commerce. The event we focus on is the hijacking of an anonymous (anonymized) palm oil tanker (classified as "product tanker" by the ICC) while underway from Sulawesi to Surabaya in the Indonesian Exclusive Economic Zone (EEZ). The complete RDF graph representing this event in SEM is shown in Fig. 10.1. We do not expect the reader to understand this graph at this moment. Parts of the graph will be discussed one by one in Sect. 10.4. For more information about the origin, representation, and use of the data set from which this example was drawn, see [4].

A legend of the visual elements used in all the diagrams in this chapter is shown in Fig. 10.2. The colored ellipses stand for classes, that is, sets of things. The colors themselves are only for visual distinction of the various parts of the Simple Event Model. The core classes are green, the types, that represent sets of instances, are blue, and the constraints orange. White ellipses or circles stand for instances of the classes, boxes for (concrete) data values. Regular arrows stand for various

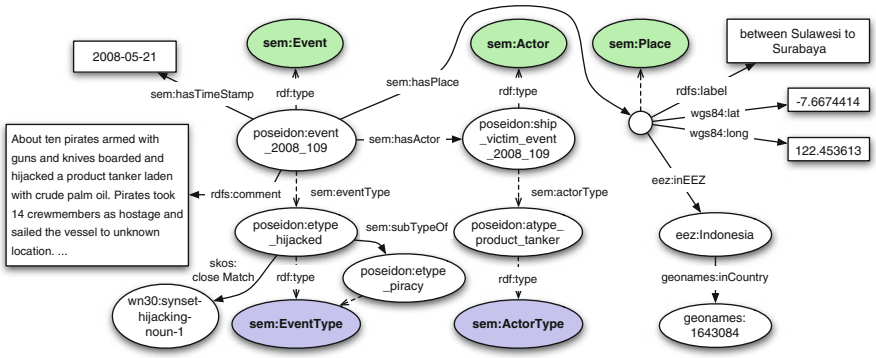


Fig. 10.1 An example event description of a palm oil tanker hijacking in Indonesia

[7]Linked Open Piracy: http://semanticweb.cs.vu.nl/lop

[8]Linked Open Piracy: http://semanticweb.cs.vu.nl/lop

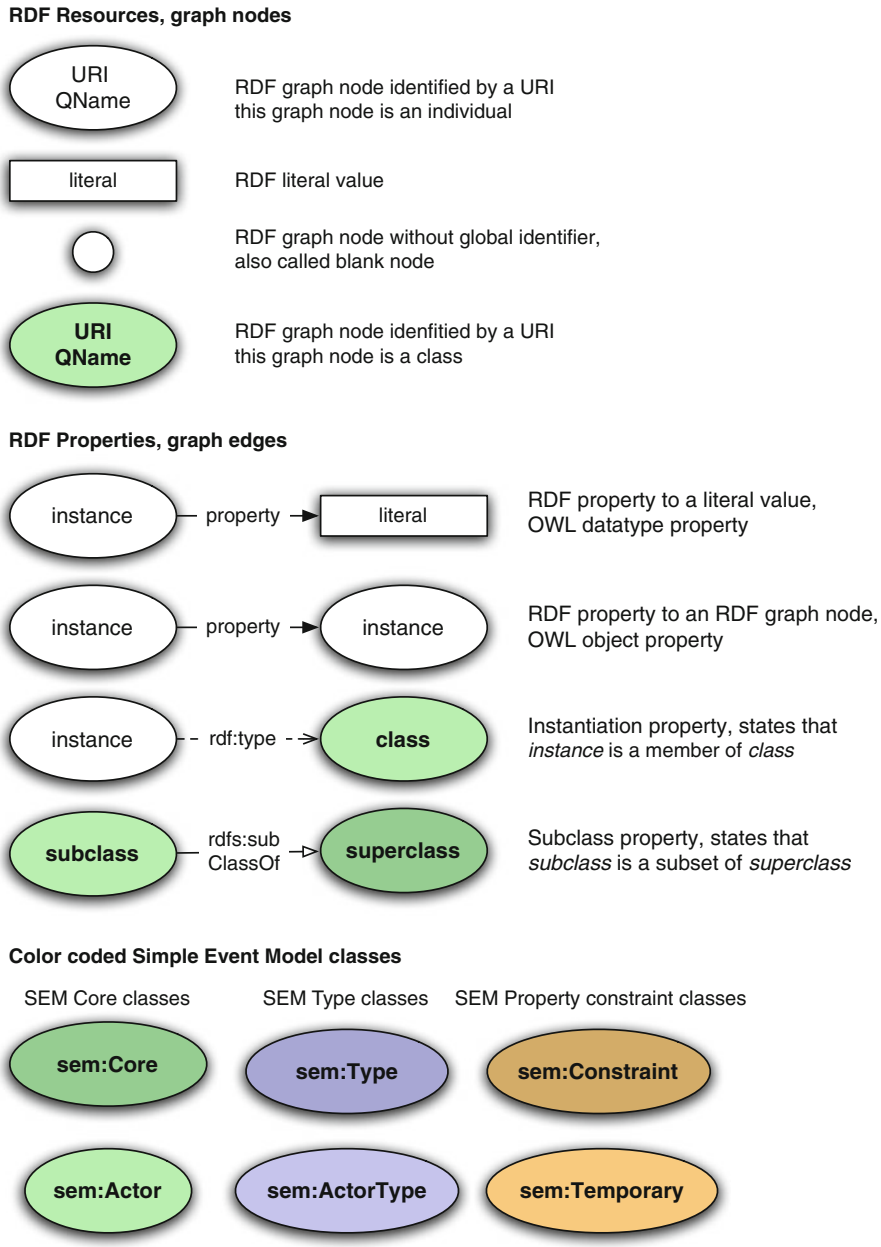


Fig. 10.2 Legend of the visual elements used in this chapter

properties between the instances or classes. Dashed arrows, like in UML,[9] stand for instantiation relations, i.e. *is-a* relations. Open-ended arrows stand for subclass relations.

10.3 Example 2: A Crude Oil Tanker Tugged into a Harbor

The second example we use to illustrate the SEM constructs is a fictional example of a crude oil tanker, the "sts W. Alton Jones (2nd)" coming into a harbor while being tugged by the tug boat "Ääron". The distinctive features of this event description are the following:

- It contains a subevent hierarchy containing an AIS observation of the ship by an AIS station, the arrival of the ship and the entire trip from Egypt to the Netherlands.
- There are two ships involved in a single event, both of which have a distinct role in the event (the tugger and the tow).
- It has a time description using sem:hasTime as opposed to sem:hasTimeStamp (see Sect. 10.4.4 for details about these two kinds of descriptions).
- The ais:flagLabel of the ship has a time constrained validity.

We will not go into what these features mean exactly here in this section. This will be made clear in Sect. 10.4 as we go through all the constructs step by step. The complete RDF graph describing this event is shown in Fig. 10.3. For more examples of similar event descriptions from the maritime domain, see [6].

10.4 SEM Constructs

The first thing to note about the Simple Event Model is that it consists of three parts: the core, types, and property constraints. The core classes deal with events and their basic properties and constituents (actors, places and times). The types can be used to categorize events and their constituents into classes. The property constraints can be used to qualify event properties in some way. An overview of the classes of the Simple Event Model and how they relate to each other is shown in Fig. 10.4. An overview of all the properties of the Simple Event Model is shown in Fig. 10.5. A detailed description of the design decisions and motivations underlying SEM can be found in [5]. In the following sections we will discuss each of the elements depicted in these figures one by one. We will start with events, then the other core classes and then work our way down the model to the types and constraints.

[9]UML: http://en.wikipedia.org/wikiUnified_Modeling_Language

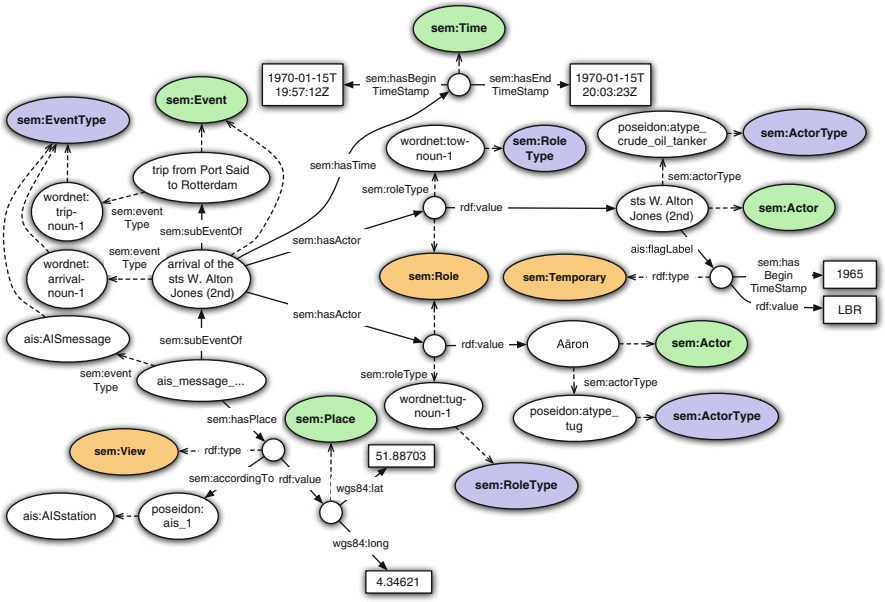


Fig. 10.3 An example event description of a crude oil tanker being tugged into a harbor

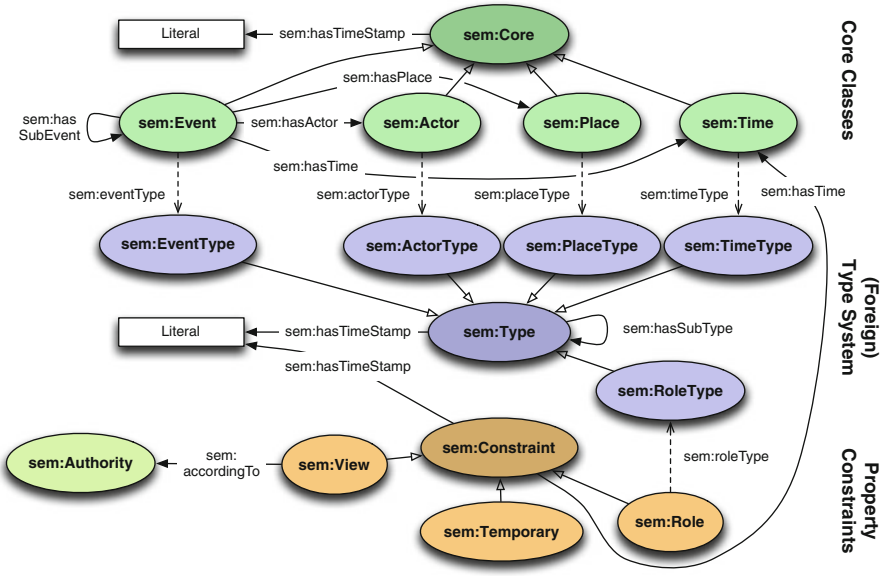


Fig. 10.4 The Simple Event Model. *Solid arrows* represent regular properties, *open-headed arrows* represent subclasses (rdfs:subClassOf), *dashed arrows* represent instantiation (rdf:type)

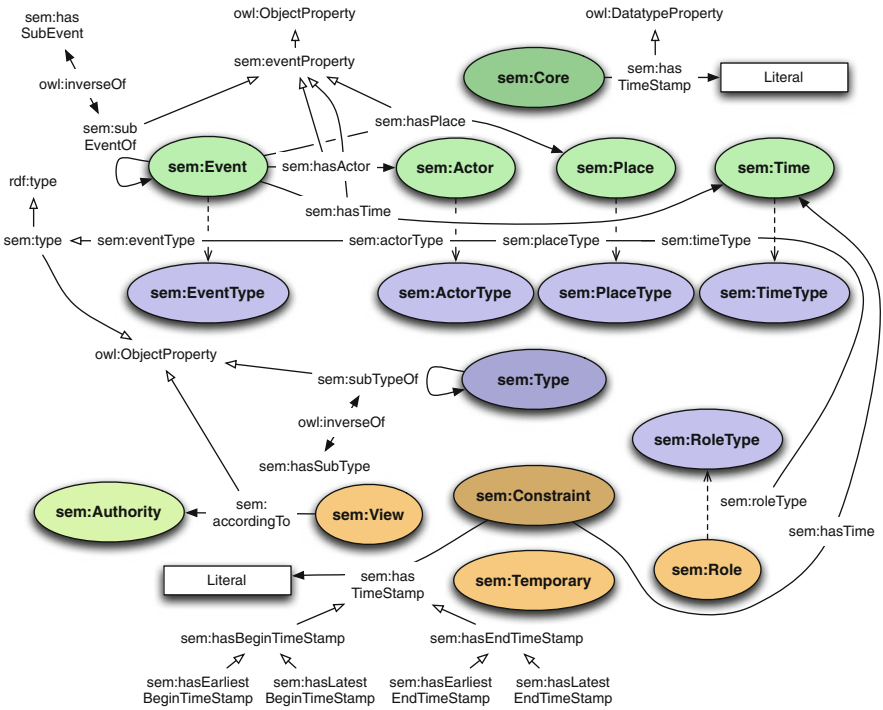


Fig. 10.5 The Simple Event Model property hierarchy. *Solid arrows* represent the range and domain of the properties, *open-headed arrows* represent subproperties (rdfs:subPropertyOf), *dashed arrows* represent instantiation (rdf:type)

In explaining the representation of SEM models in RDF it is impossible to refrain from using the appropriate technical terms. Below is a short list of terms that we will use along with a concise description. For more information about these or other technical RDF terms please consult the RDF Primer[10] and Specification[11] or refer to the book by Allemang and Hendler [1].

URI: Uniform Resource Identifier, a string of characters used to identify anything we can talk about. These can be tangible or intangible things, sets of things, or properties of things, anything denotable. The URI is a global identifier of the thing it denotes. It is good practice that the dereferencing (i.e. looking up using HTTP) of a URI returns a description in RDF of the thing it stands for.
Namespace: An XML namespace, or a URI prefix that is often used to group and shorten URIs from a common authority.

[10]RDF Primer: http://www.w3.org/TR/2004/REC-rdf-primer-20040210/
[11]RDF Spec: http://www.w3.org/TR/rdf-syntax-grammar/

QName: A shortened notation of a URI that consists of a colon-separated pair of a short name indicating a namespace and a name. When the prefix and the name are concatenated they form the entire URI of the denoted thing. For example, if the short name "ex" stands for the namespace "http://example.org/" then "ex:one" stands for the URI "http://example.org/one". The short name of the namespace can be chosen freely and has no meaning. It merely exists to shorten the URI.
Triple: The atomic building block of RDF graphs, three identifiers or two identifiers and a data value. A triple stands for an edge in an RDF graph. The first place of the triple is the identifier of the source node of the edge, the second place is the identifier that indicates the type of the edge, and the third place is either the identifier of the target node of the edge or a data value. The identifiers can be either global identifiers in the form of a URI, or local identifiers of nodes that can not be referenced to from other RDF graphs. These nodes are also called "blank nodes".
Property: An edge in the RDF graph. A triple can be seen as a relation between the source and target URI. In this sense, the triple also states that the source has a certain property, the type of which is denoted by the second place of the triple. This is why we call edges in the RDF graph properties.
Class: A set of RDF nodes. RDF nodes can be assigned to a class by asserting a triple from that node to the URI of the class with the predefined RDF property "rdf:type". For example, if we state the triple "ex:apple rdf:type ex:fruit" this makes "ex:apple" a member of "ex:fruit".
Individual/instance: Members of a class are also called individuals or instances.
RDF: The language we use to describe graphs.
RDFS: The language we use to talk about classes, individuals, properties, subclasses, and subproperties. All RDFS statements can be written down in RDF.
OWL: The language we use to talk about more complex properties of individuals, classes and properties then can be expressed with RDFS. For example, OWL contains a property to state that one property is the inverse of another property. We will only use a few OWL constructs in this chapter: owl:ObjectProperty (properties pointing to a graph node), owl:DatatypeProperty (properties pointing to a data value), and owl:inverseOf (specifies that two properties are the inverse of each other, like sem:subEventOf and sem:hasSubEvent).

10.4.1 Events

The sem:Event class represents the class of all event instances. These can be events of any kind, from an earthquake to the taking of a photograph, or the transmission and reception of an AIS message. Events generally have actors, a place, and a time. Two examples of events are shown in Fig. 10.6. These figures are snippets from two larger event descriptions shown in Figs. 10.1 and 10.3. The irrelevant

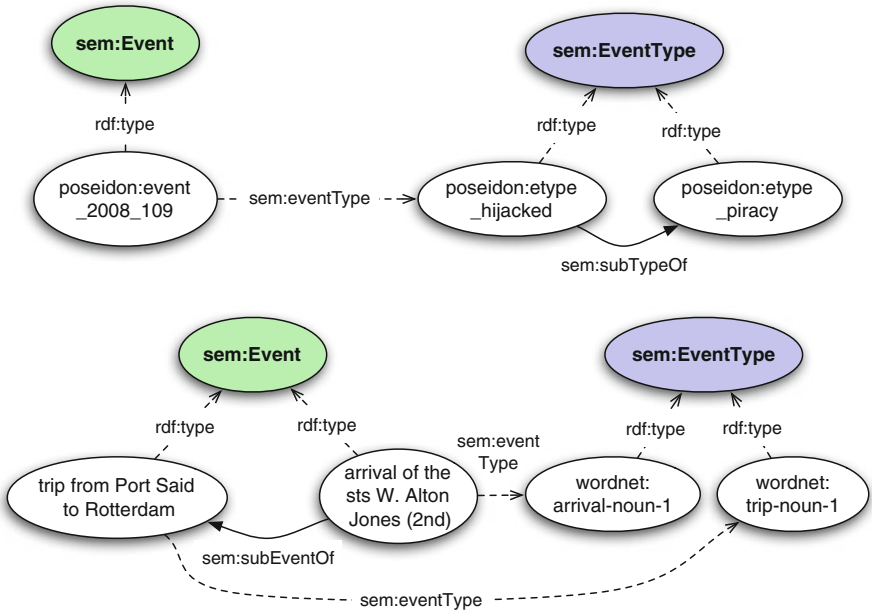


Fig. 10.6 Example events and their types showing the use of sem:subEventOf and sem:subTypeOf, taken from the examples outlined in Figs. 10.1 and 10.3

parts are omitted for the sake of legibility. The upper example shows the event instance poseidon:event_2008_109.[12] The dashed arrow labeled rdf:type denotes the triple ⟨poseidon:event_2008_109, rdf:type, sem:Event⟩ that states it is a member of the class of all sem:Events. The other dashed arrow labeled sem:eventType denotes that this event is a hijacking by assigning a sem:EventType instance to the event, in this case poseidon:etype_hijacked. The sem:eventType property is a subproperty of rdf:type that should only be used to assign types to events (not for categorization of any other kind). This means that hereby we define that the event poseidon:event_2008_109 is a member of the class poseidon:etype_hijacked that also contains all other described hijackings. We can also see in Fig. 10.6 that there is a sem:subTypeOf property between poseidon:etype_hijacked and poseidon:etype_piracy. This means that all the hijacking events are also piracy events and that possibly there are other events that are piracy events, but not hijackings (e.g. high sea robberies). sem:subTypeOf is a subproperty of rdfs:subClassOf. This means sem:subTypeOf defines a subclass hierarchy of event types.

[12]The entire URI of this instance is http://semanticweb.cs.vu.nl/poseidon/ns/instances/event_2008_109, which is shortened to the QName poseidon:event_2008_109. Resolving the entire URI in a browser will return more information about this resource. An RDF crawler or a triple store loading over HTTP will get descriptive RDF about this URI instead.

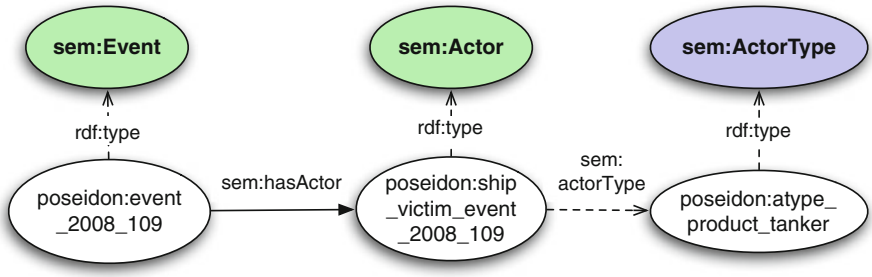


Fig. 10.7 Example event with actor and its actortype

In the lower example two event instances are shown with two different event types. In this case there is also a sem:subEventOf property between the two events. This defines that the "arrival of the sts W. Alton Jones (2nd)" event is a part of the "trip from Port Said to Rotterdam" event. sem:subEventOf is an aggregation relation (in the same sense as in UML) that builds a part-whole hierarchy (as opposed to sem:subTypeOf, which builds a subclass hierarchy) and hence is not a subproperty of rdfs:subClassOf. The sem:subEventOf and sem:subTypeOf properties have inverse properties, respectively sem:hasSubEvent and sem:hasSubType, so that it is possible to define the hierarchies top-down or bottom-up and end up with the same semantics.

10.4.2 Actors

When it is necessary to state more than just *"what happened"*, like *"who"* or *"where"* or *"when"*, just Events and EventTypes are not enough. The most commonly represented part of events is actors. Actors can be anything, from a star in a movie to the person or the camera taking a photograph to machinery used in an experiment. They are represented as instances of the sem:Actor class. The involvement of an actor in an event is represented with the sem:hasActor property. For example, as shown in Fig. 10.7, where the ship that is hijacked, poseidon:ship_victim_ship_event_2008_109 is connected to the hijacking event poseidon:event_2008_109. Actors can also have types, modeled analogously to the way event types are modeled. In this case, the victim ship is a poseidon:atype_product_tanker, which is defined using the sem:actorType property. It is quite possible that an event has many actors and that one actor is involved in many events. It is also possible that an event has no actors or that an actor is defined without any association with an event. (There are event languages where events are modeled as predicates and all knowledge about actors is contained inside the event predicate. In these languages it is not possible to define properties of actors without defining at least one event in which the actor participates.) *All* SEM properties (see Fig. 10.5) can

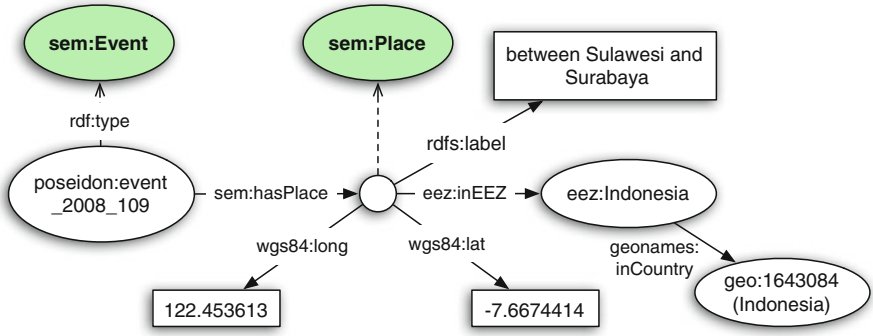


Fig. 10.8 Example event with place showing the omission of a type and the use of a blank node to specify an anonymous place somewhere in the exclusive economic zone of Indonesia

be omitted or duplicated as needed. That is, they are all optional, and it is also possible to have more than one if necessary. Optionality is a useful feature when there is incomplete knowledge. For example, it can be the case that the type is not known, because it was not provided by the source of the information. An example of this is shown in Fig. 10.8, where the sem:Place instance has no sem:PlaceType. Duplicability can be useful when more than one instantiation of a property is valid. For example, there can be more than one actor associated with an event and this can be denoted by stating multiple sem:hasActor properties about the same event.

10.4.3 Places

In SEM the description of Places works exactly the same as the description of Actors. They can be typed with the sem:placeType property and associated to an event with the sem:hasPlace property. SEM itself does not provide any means to define the geometry of a place or to put a Place in some kind of geographical categorization. These need to be borrowed from other ontologies, like the W3C Basic Geo (WGS84) vocabulary,[13] GeoRSS,[14] or Geonames.[15] Figure 10.8 shows an example that demonstrates both of these features (sem:placeType and sem:hasPlace). The wgs84:lat and wgs84:long properties from the Basic Geo vocabulary are used to assign coordinates to a place and the geonames:inCountry property (in combination with the domain specific eez:inEEZ property) is used to put the place in the Geonames place hierarchy under Indonesia.

[13] Basic Geo: http://www.w3.org/2003/01/geo/

[14] GeoRSS: http://www.georss.org/rdf_rss1

[15] Geonames Ontology: http://www.geonames.org/ontology/

On the Semantic Web, anybody can say anything about anything. Authorities owning the domain name of a URI can state triples about this URI, but others can also state things about this URI. It is quite possible to state triples that contradict other triples. Therefore it is important to keep track of the origin of triples. Most RDF systems record this origin in the fourth position of a triple.

It is possible to leave out parts of the description of SEM individuals. This is particularly useful when describing places. One can distinguish two kinds of underspecification with places: Places with no "name", i.e. no global identifier in the form of a URI, and places with no geometrical denotation. An example of the latter is the place of your car keys. They must be somewhere, but the exact location can be unknown. This can be represented by leaving out the geometry predicates of the place denoting the location of the keys. An example of the former is the place of a ship somewhere in the middle of the sea. Such a place may or may not be specified by coordinates, but as opposed to the entire sea it does not have a name of its own, and as opposed to the location of your car keys, it may not be an important subject of discussion. We call both such places anonymous places and they can be represented by blank nodes as shown in Fig. 10.8. Blank nodes are local identifiers of a node. That means if someone else wants to refer to it from some other piece of RDF that is impossible. The advantage is that it is not necessary to think up (e.g. generate) a URI for a nameless place. If it is relevant to relate the anonymous place of the ship to the place of the sea one can make use of geographical part-whole relations from external ontologies, for example, geonames:parentFeature or geonames:inCountry.

10.4.4 Time

Time has a unique place in SEM. Whereas places are only used to localize events, time can be associated to any SEM class instance. For example, actors can be assigned a time. This can be used to denote the "life span" of an entity. Time descriptions can refer to instants, intervals, or they can be symbols that abstractly denote some time (e.g. "ex:t1").

There are two ways to specify time in SEM. One works exactly analogous to places, using the sem:hasTime property, sem:Time class and sem:TimeType class. The other uses the subproperties of the sem:hasTimeStamp datatype property. These two alternatives are shown in Fig. 10.9. The timestamp notation exists to reduce the representational complexity of assigning time to everything. Timestamps directly refer to a literal denoting the time, for example, a string in XML Schema ISO 8601 format. This means the time as such does not have a node in the RDF graph that can be linked to, neither a blank node nor a URI. Therefore, nothing more can be said about that time instance. If something has to be said about a time instance, for example, how it was acquired, whether it was before or after some other time (which can be the only thing known about a moment, like some unknown time of death in a detective novel), then timestamps do not suffice and sem:hasTime will need to be used.

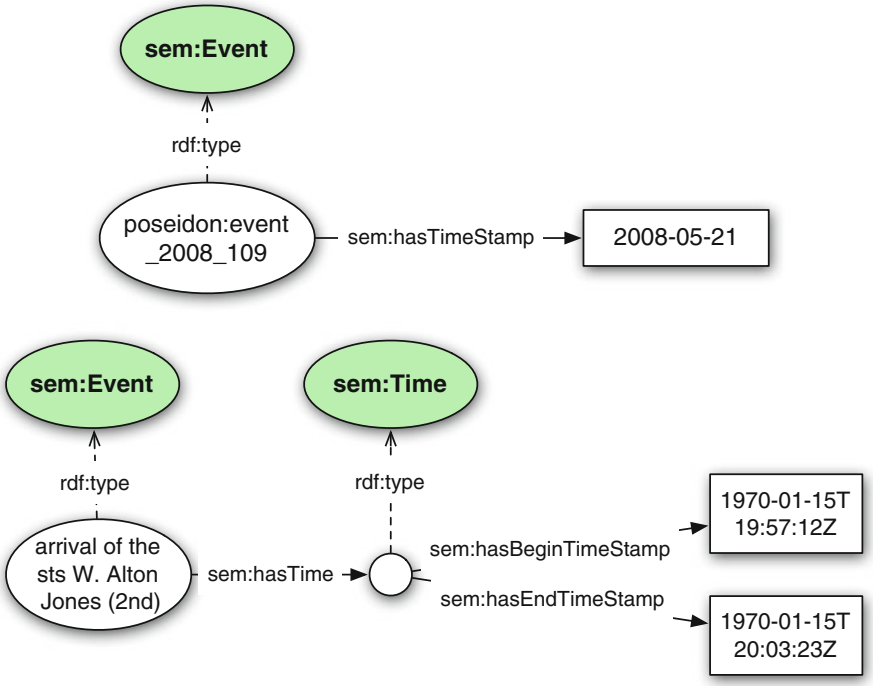


Fig. 10.9 Example event with time indications showing the two alternative ways to indicate time and the use of two-value timestamps

In some cases it can be necessary to state more about a time than just a single timestamp. Apart from single timestamps, denoted by sem:hasTimeStamp, SEM has two additional levels of timestamps: two-value time for denoting intervals, and four-value time for denoting intervals with an uncertain beginning and/or end. An example of two-value time is a tea break with a start time, denoted with sem:hasBeginTimeStamp and a time at which it ends, denoted with sem:hasEndTimeStamp. Things that start and do not end or have no known end point can be denoted by omitting the sem:hasEndTimeStamp property. To say that the end of the tea break happens sometime between four and five, there is the four-value time properties sem:hasEarliestEndTimeStamp and sem:hasLatestEndTimeStamp. An uncertain beginning works analogously. Figure 10.9 shows an example of one-value time and two-value time without uncertainty. Another option is to use the OWL Time ontology to describe a sem:Time instance denoting the time instance. OWL Time has a large array of classes and properties to describe time.

10.4.5 Role

Constraints are the most complicated tool in the SEM toolbox. They are meant to say things about the properties between SEM instances (e.g. sem:hasActor or sem:placeType), for example, to qualify the way in which an actor is involved in an event by constraining the sem:hasActor property. SEM has three ways to constrain the meaning of one of its core event properties: Roles, Temporaries (see Sect. 10.4.7), and Views (see Sect. 10.4.6). The first of the sem:Constraints we will discuss is the role.

As opposed to Types, which are used to assign permanently valid categories to entities, Roles can be used to denote that *in the context of a specific event* the individual referred to belongs in a certain category, i.e. plays a specific role. For example, the actor of the "mail sending" event referred to by the constrained sem:hasActor property is the receiver of the mail, not the sender. One can argue that a simpler way to represent this role is to make a subproperty has Receiver of the sem:hasActor property. This is indeed good practice. The advantage of treating the role as an object with properties as opposed to a property is that one can reuse terms that are not properties from external vocabularies to denote the role. For example, "receiver" might exist in WordNet, but as an object, not as a property. Since it is an object we can not reuse it by making it a subproperty of sem:hasActor. Saying the actor is of sem:actorType receiver would mean he is the receiver in every event. That is probably not what we want to say. We only want to say he is the receiver in this single event. This is when it becomes useful to make a role with the WordNet URI for receiver as sem:roleType. In Fig. 10.10 we show an example of a role constraining the sem:hasActor property to the role of wordnet:tow-noun-1 (i.e. being tugged). The rdf:value property connects the blank node representing the constrained property to the (original) value of that property, in this case the ship that is being towed. In some other event this ship might not be towed, but sailing on its own accord, while the sem:ActorType assigned to the ship is valid in all events.

10.4.6 View

Views can be used to denote which source claims that a certain property holds. This can be useful when different, possibly conflicting values need to be stored at the same time. The source stating a property is called an Authority. This can refer to any source of facts, such as a person, a file, or a sensor. A set of SEM statements should be seen as a truthful representation of some possibly incorrect description of an event. The actual real-world truth is irrelevant to SEM models. They exist solely to convey a description of an event from one party to another. Figure 10.11 shows a View constraint applied to the sem:hasPlace property to denote that the AIS station poseidon:ais_1 was the source of the anonymous place with the attached coordinates. There could be other AIS stations that passed through this message.

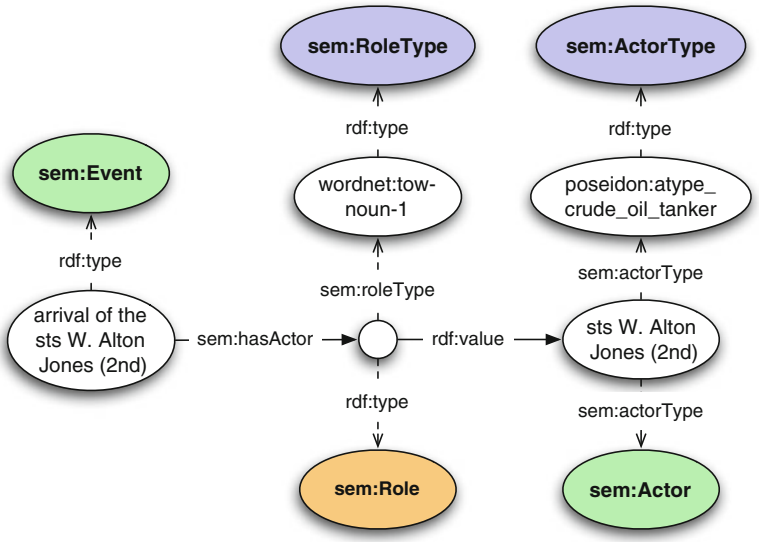


Fig. 10.10 Example event and an actor with a specific role. In this case the crude oil tanker is the tow, not the tug. This role is local to this event. In another event it might be the one doing the tugging

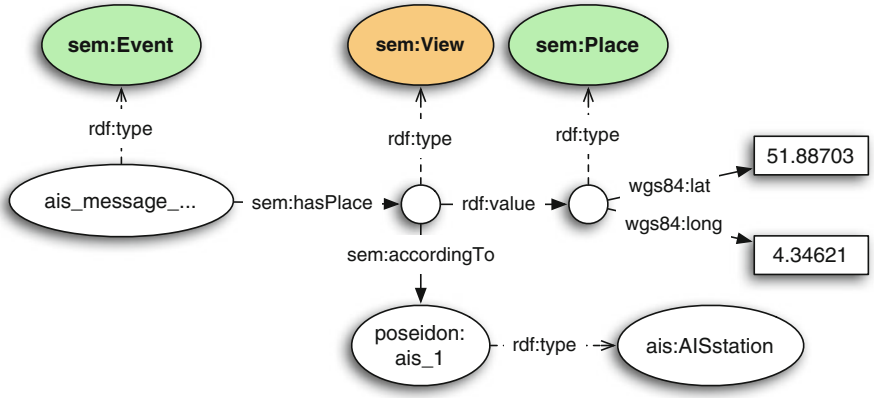


Fig. 10.11 Example of a View denoting the source of the place of an event. In this case, the place was acquired from an AIS station

These sources could be modeled with additional sem:accordingTo properties on the blank node or with additional separate sem:Views. Views can be applied to all subproperties of sem:eventProperty and sem:type.

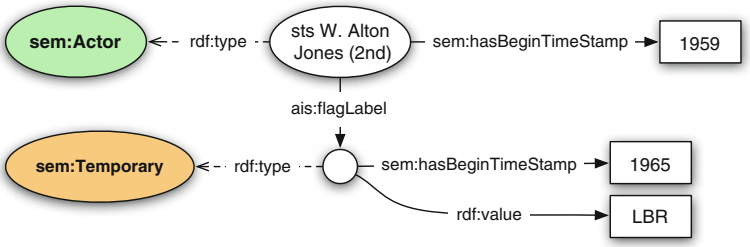


Fig. 10.12 Example of a Temporary constraint applied to a property of an actor. This example shows that the actor predates one of its properties. 6 years after the ship was built it changed ownership which caused a change of flag

10.4.7 Temporary

Temporaries can be used to denote that a property holds during a certain time. In the description of Time we have already seen that any instance of a SEM class can be timestamped. Temporaries can be used to accomplish the same thing for SEM properties. Stating that the place "Den Haag" has a certain begin and end timestamp is the same as to state that the place exists between these points in time. Saying that the type of the place is "seat of government" between two points can be done by applying a sem:Temporary constraint to the sem:actorType property pointing at "seat of government". Figure 10.12 shows how to constrain the validity of the ais:flagLabel property of the actor to the period starting at 1965. This can be any date, in this case it is 6 years later than the begin date of the actor itself, which can be interpreted as the build date of the ship. SEM does not put any limitations on which values that can be assigned to temporaries. They are allowed to be inconsistent. For example, the flag could be valid from before the build date of the ship, even though this is impossible according to international regulations. From the perspective of SEM this is perfectly fine. Solving such inconsistencies is up to the user of the models and can be done by retrieving the conflicting values and resolving them in some way, be it manually or automatically.

10.5 Consumption

The previous sections showed how to represent different kinds of events by means of the Simple Event Model. The information about such events is encoded by means of RDF triples and then stored in triple stores that, from the point of view of data storage and management, perform the same function as traditional relational database management systems (RDBMS). However, since RDF data is represented by means of graphs, RDF repositories extend the potential offered by these data storages by increasing their flexibility and the number of possible

applicable operations. The increase in flexibility is primarily due to the fact that tables, the basic data representation model in RDBMS, are much more static and rigid than graphs. Each record in a database has to be representable by means of a table row with given constraints and, although also with graph representation it is possible to apply constraints to the data, this latter paradigm is less rigid. Adding new relations and columns to a database is quite hard in practice, while in the graph paradigm this corresponds to adding new triples while all existing triples can be left untouched. The graph paradigm allows also to easily make use of functions that are more complicated to implement in traditional RDBMS like, for instance, inheritance and subsumption reasoning (e.g. concluding that type *a* is a subclass of type *b* by chaining sem:subTypeOf properties between them and that *a*'s individuals therefore also individuals of *b*). In principle, the same expressivity that an RDF-based repository exposes is reachable also by traditional RDBMS (indeed, many triple stores are implemented on top of relational databases), especially when all constraints are known a priori. The latter is not always the case. Sometimes constraints become known after the schema has been declared.

In this section we will see how it is possible to "consume" data previously stored in RDF storages, that is, how to retrieve them. The retrieval process is enforced through a query, expressed in a language, SPARQL[16] that allows to model queries for RDF storages exactly as SQL allows to retrieve information from databases. Piracy attacks that have been previously described are stored in a server and are retrievable from the Linked Open Piracy[17] (LOP) repository, that exposes the RDF transposition of all the reports exposed by the International Chamber of Commerce (ICC-CCS), see [4]. The LOP website exposes a SPARQL endpoint that allows to query the repository. We will see now how to use SPARQL to make use of this flexibility to retrieve data from the repository.

Suppose that we want to retrieve all hijacking events involving merchant vessels. We could think to look for all events having an actor of type merchant vessel as follows:

```
SELECT DISTINCT *
WHERE {
    ?actor sem:actorType poseidon:atype_merchant_vessel .
    ?event sem:hasActor ?actor .
    ?event sem:eventType poseidon:etype_hijacked .
}
```

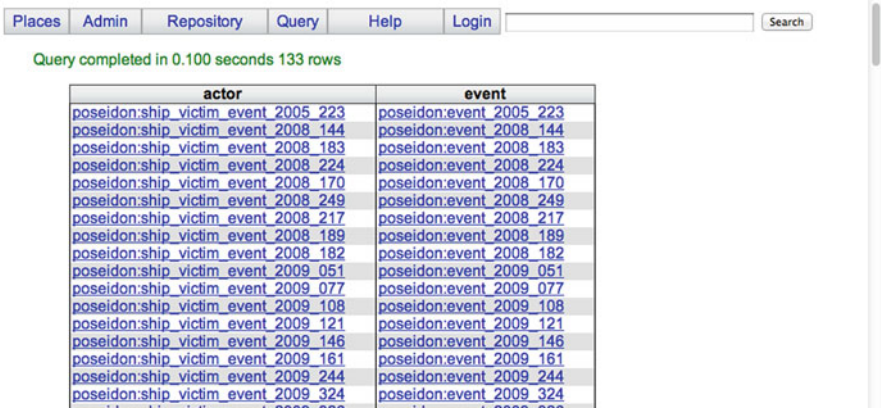
Listing 101 SPARQL query retrieving all hijacking events that involve an actor of the type merchant vessel

In the following screenshot of the result list as displayed by the ClioPatria[18] triple store we can see that this matches 133 event-actor pairs (Fig. 10.13) .

[16]SPARQL: http://www.w3.org/TR/rdf-sparql-query/

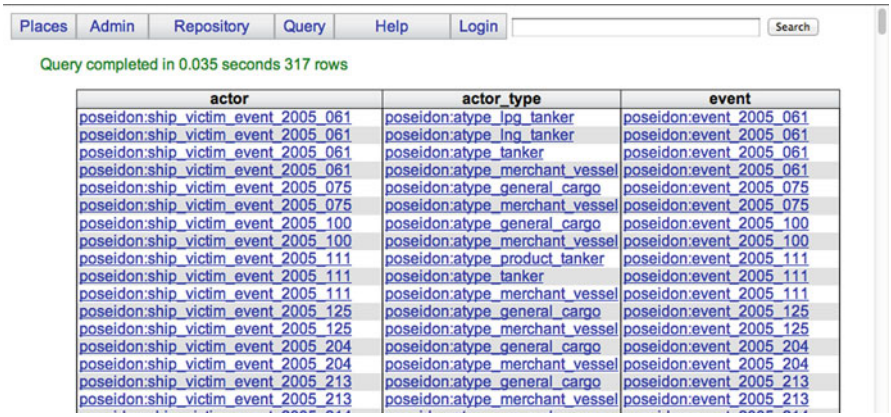
[17]Linked Open Piracy: http://semanticweb.cs.vu.nl/lop/

[18]ClioPatria triple store: http://cliopatria.swi-prolog.org/



actor	event
poseidon:ship_victim_event_2005_223	poseidon:event_2005_223
poseidon:ship_victim_event_2008_144	poseidon:event_2008_144
poseidon:ship_victim_event_2008_183	poseidon:event_2008_183
poseidon:ship_victim_event_2008_224	poseidon:event_2008_224
poseidon:ship_victim_event_2008_170	poseidon:event_2008_170
poseidon:ship_victim_event_2008_249	poseidon:event_2008_249
poseidon:ship_victim_event_2008_217	poseidon:event_2008_217
poseidon:ship_victim_event_2008_189	poseidon:event_2008_189
poseidon:ship_victim_event_2008_182	poseidon:event_2008_182
poseidon:ship_victim_event_2009_051	poseidon:event_2009_051
poseidon:ship_victim_event_2009_077	poseidon:event_2009_077
poseidon:ship_victim_event_2009_108	poseidon:event_2009_108
poseidon:ship_victim_event_2009_121	poseidon:event_2009_121
poseidon:ship_victim_event_2009_146	poseidon:event_2009_146
poseidon:ship_victim_event_2009_161	poseidon:event_2009_161
poseidon:ship_victim_event_2009_244	poseidon:event_2009_244
poseidon:ship_victim_event_2009_324	poseidon:event_2009_324

Fig. 10.13 Output of the SPARQL query of Listing 10.1 posed at http://semanticweb.cs.vu.nl/lop/user/query



actor	actor_type	event
poseidon:ship_victim_event_2005_061	poseidon:atype_lpg_tanker	poseidon:event_2005_061
poseidon:ship_victim_event_2005_061	poseidon:atype_lng_tanker	poseidon:event_2005_061
poseidon:ship_victim_event_2005_061	poseidon:atype_tanker	poseidon:event_2005_061
poseidon:ship_victim_event_2005_061	poseidon:atype_merchant_vessel	poseidon:event_2005_061
poseidon:ship_victim_event_2005_075	poseidon:atype_general_cargo	poseidon:event_2005_075
poseidon:ship_victim_event_2005_075	poseidon:atype_merchant_vessel	poseidon:event_2005_075
poseidon:ship_victim_event_2005_100	poseidon:atype_general_cargo	poseidon:event_2005_100
poseidon:ship_victim_event_2005_100	poseidon:atype_merchant_vessel	poseidon:event_2005_100
poseidon:ship_victim_event_2005_111	poseidon:atype_product_tanker	poseidon:event_2005_111
poseidon:ship_victim_event_2005_111	poseidon:atype_tanker	poseidon:event_2005_111
poseidon:ship_victim_event_2005_111	poseidon:atype_merchant_vessel	poseidon:event_2005_111
poseidon:ship_victim_event_2005_125	poseidon:atype_general_cargo	poseidon:event_2005_125
poseidon:ship_victim_event_2005_125	poseidon:atype_merchant_vessel	poseidon:event_2005_125
poseidon:ship_victim_event_2005_204	poseidon:atype_general_cargo	poseidon:event_2005_204
poseidon:ship_victim_event_2005_204	poseidon:atype_merchant_vessel	poseidon:event_2005_204
poseidon:ship_victim_event_2005_213	poseidon:atype_general_cargo	poseidon:event_2005_213
poseidon:ship_victim_event_2005_213	poseidon:atype_merchant_vessel	poseidon:event_2005_213

Fig. 10.14 Output of the SPARQL query of Listing 10.2 posed at http://semanticweb.cs.vu.nl/lop/user/query

So, does that mean that there are 133 actors that have been assigned the sem:actorType poseidon:atype_merchant_vessel? No. Actually, there are no direct references to this type. All of the ships that matched have some other type assigned to them, for example poseidon:atype_lpg_tanker or poseidon:atype_general_cargo as shown in Fig. 10.14, but these have been defined to be subtypes of poseidon:atype_merchant_vessel using sem:subTypeOf. The following query displays the same 133 matching events, but also displays the actual types of the ships, making use of the fact that sem:subTypeOf is a subproperty of rdfs:subClassOf.

```
SELECT DISTINCT *
WHERE {
    ?actor sem:actorType ?actor_type .
    ?actor_type rdfs:subClassOf poseidon:atype_merchant_vessel .
    ?event sem:hasActor ?actor .
    ?event sem:eventType poseidon:etype_hijacked .
}
```

Listing 102 SPARQL query retrieving hijacked merchant vessels and all their types

The results of this query are shown in Fig. 10.14.

There are more than 133 rows in this result table, 317 to be exact. This is due to the fact that if there is more than one sem:ActorType in the type hierarchy that is a rdfs:subClassOf poseidon:atype_merchant_vessel then we will get more than one row for that event. The first four rows of the result table in Fig. 10.14 all pertain to the same event and same actor. In this case the actor is an LNG tanker, so this single event will match the query for merchant vessels for four reasons: It is an event involving (1) an LNG tanker, (2) an LPG tanker, (3) a tanker, and (4) a merchant vessel, all of which are subclasses of poseidon:atype_merchant_vessel (in the fourth case this is because every class is a subclass of itself). The only way in which there is exactly one row in the results list per event is when that event has exactly one sem:Actor and that actor has exactly one sem:actorType and that type is exactly the type requested in the query. None of these conditions have to be the case. We have just shown that even though an actor has a single type then this type can still match one of its supertypes. If an actor actually has multiple assigned types there will be some results with the same event and actor, and if there are multiple actors per event there will be results with the same event but with different actors and perhaps different actor types.

This section described a sample of the potential that RDF and SPARQL offer. SPARQL is syntactically not very different from SQL, but it permits to exploit all the functionalities allowed by RDF including, for instance, subsumption reasoning like in the example introduced. Type reasoning can be implemented in an RDBMS with a type hierarchy table, but to make use of this table the queries will have to be altered. In an RDFS/OWL triple store with SPARQL the queries can remain the same, even when the type hierarchy is changed. The proper new results will simply be returned as soon as new knowledge is put in the triple store.

10.6 Conclusion

In this chapter we introduced the Simple Event Model and gave a step by step description of how to use all of its components. We illustrated this introduction with two examples from the maritime domain. Finally, we showed how SEM event models can be queried using the SPARQL query language and how RDF(S) type hierarchy reasoning is used behind the scenes of the triple store.

We have shown that SEM provides a model for representing events at very different levels of abstraction, from single sensor readings to long lasting events such as journeys of a ship. When performing queries on datasets with event descriptions at varying levels of abstraction, it is important to be able to return events that have been described at a different level of abstraction than the query. SEM and RDFS provide us with the tools needed to accomplish this. Each part of the event description, the event itself, the place, the time, and the actor, can have a separate type that can be part of a type hierarchy. These hierarchies can come from existing ontologies or can be added as needed. RDFS reasoning underneath SPARQL queries on SEM event models can provide type tolerant retrieval of events.

An important part of RDF, and thus also of SEM, is that it does not constrain what kind of statements can be added to the RDF store. Whereas database schemas prescribe what kind of columns may exist in a database, an RDF store can always be extended with new properties, even about old instances. This means apart from being tolerant to types, SEM models are also tolerant to future extensions. When new information becomes available, for example, the organization of which actors are part, this information can be added to the RDF store later without changing the existing RDF. This will immediately enable the answering of richer queries about the context of events.

Acknowledgements This research has been carried out as a part of the POSEIDON project at Thales under the responsibilities of the Embedded Systems Institute (ESI). This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

We would like to thank Véronique Malaisé, Laura Hollink and Roxane Segers for their work on the design of SEM and Marieke van Erp for her work on the representation of the Piracy event data set from which one of the examples was taken. Thanks also go to Guus Schreiber for steering all of the work touched upon in this chapter.

References

1. Allemang D, Hendler J (2008) Semantic web for the working ontologist: effective modeling in RDFS and OWL. Morgan Kaufmann, San Francisco
2. Crofts N, Doerr M, Gill T, Stead S, Stiff M (eds) (2009) Definition of the cidoc conceptual reference model. Online, November 2009. http://www.cidoc-crm.org/
3. Raimond Y, Abdallah S (2007) The event ontology. Online. http://purl.org/NET/c4dm/event.owl
4. van Hage WR, Malaisé, V, van Erp M (2011) Linked open piracy. In: van Erp M, van Hage WR, Hollink L, Jameson A, Troncy R (eds) Interanational workshop on detection, representation, and exploitation of events in the semantic web (DeRiVE 2011), pp 88–97. http://ceur-ws.org/Vol-779
5. van Hage WR, Malaisé V, Segers R, Hollink L (2011) Design and use of the Simple Event Model (SEM). J Web Semant 9(2):128–136
6. van Hage WR, Malaisé V, de Vries GKD, Schreiber G, van Someren MW (2012) Abstracting and reasoning over ship trajectories and web data with the Simple Event Model (SEM). Multimed Tools Appl 57(1):175–197

Part III
Systems of Systems

Chapter 11
Specification and Generation of Adapters for System Integration

Arjan J. Mooij and Marc Voorhoeve[†]

11.1 Introduction

Large systems-of-systems are developed by integrating several smaller systems that have been developed independently. In this context, the integration needs to be achieved without modifying these independent systems, even though their development may not have considered this particular integration. So the systems can only interact with each other through their existing external interfaces.

Even if the systems to be integrated fit conceptually, the external interfaces may not fit technically. Examples of technical incompatibilities include the encoding of the data that is communicated, and the communication protocols that are used to communicate the data. An approach to resolve such incompatibilities is to develop a custom adapter, which is sometimes called mediator or glue logic. An adapter is a (small) additional system that is compatible with each of the original systems.

System integration and adapter development are known to be time-consuming activities. There is a range of possible causes, including the limited knowledge about the systems to be integrated, the large number of interface technologies involved, the high degrees of parallelism, and the usual efforts of developing a new system. The present chapter explores techniques for developing complex adapters in a faster way, in particular by generating (parts of) them.

Approach We consider two aspects of systems: the communication behavior and the communicated data. The communication behavior of a system describes

[†]7 October 2011

A.J. Mooij (✉)
Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands

Current affiliation: Embedded Systems Institute, Eindhoven, The Netherlands
e-mail: arjan.mooij@esi.nl

P. van de Laar et al. (eds.), *Situation Awareness with Systems of Systems*,
DOI 10.1007/978-1-4614-6230-9_11, © Springer Science+Business Media New York 2013

the orders in which the system can send and receive messages over its external interfaces. The communicated data describes the contents of the messages.

Discussions on adapter generation often emphasize the potential output, viz., an automatically generated adapter. However, it is unreasonable to expect that an adapter can be generated without any description of the application domain or the systems to be integrated. Hence studies on adapter generation should also consider adapter specification. On the one hand, the specification style should be convenient for modeling the integration problem; on the other hand, the specification style should provide enough detail for generating an acceptable, executable adapter.

To this end, we have studied several example adapters. For each example we have investigated how we would like to specify the adapter, and we have investigated which existing techniques could be used for generating the adapter. This has resulted in two kinds of adapter specifications with two techniques for adapter generation.

The first approach focuses on incompatibilities in the communication behavior, and it is based on techniques for (supervisory) controller synthesis [15, 16] from the field of control theory. The second approach focuses on incompatibilities in the communicated data, and it is based on techniques for incremental view maintenance [4, 9] from the field of database theory.

Regarding adapter generation, our emphasis is on generating the behavior of adapters, but the data aspect is not ignored. We illustrate and evaluate the techniques using an example from the domain of maritime safety and security.

Overview In Sect. 11.2 we first introduce the running example. We continue in Sect. 11.3 with a discussion on sources of incompatibilities. In Sects. 11.4 and 11.5, we discuss the two techniques for resolving such incompatibilities. Finally, in Sect. 11.6, we draw some conclusions and sketch further work.

11.2 Running Example of System Integration

To illustrate and evaluate the techniques for developing adapters, we use a (small) running example from the domain of maritime safety and security. This example considers the integration of two industrial systems, viz., a sensor and a display, that were developed independently of each other. The integration goal is to show the most recent vessel information from the sensor on the display; see Fig. 11.1. For any discussions on visualization techniques we refer to Chap. 5.

11.2.1 Sensor: AIS Receiver

The sensor that we consider is a maritime AIS receiver (Automatic Identification System, [5]). Every vessel has an on-board AIS transponder that broadcasts some information about the vessel using several message types and reporting frequencies.

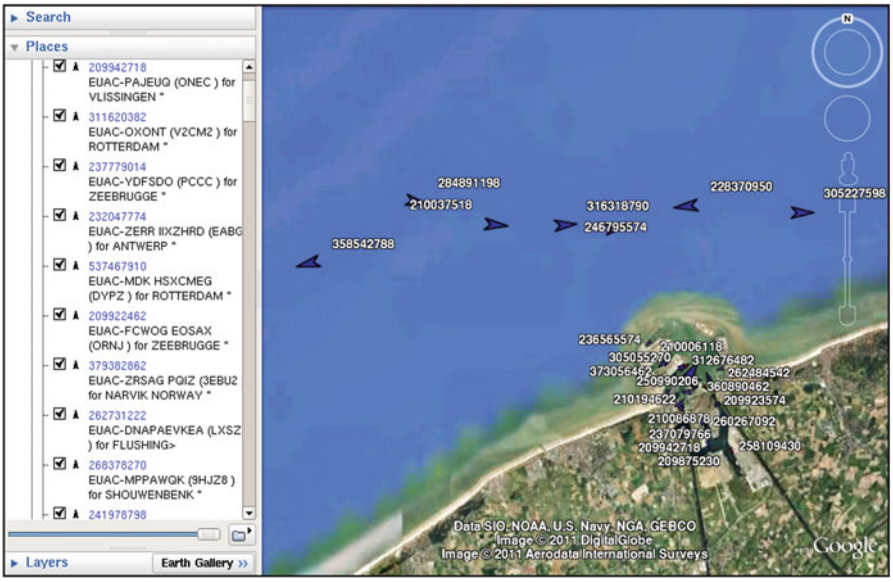


Fig. 11.1 Integration of AIS receiver and Google Earth (displayed: Zeebrugge area) (Note: for privacy reasons, the MMSI's, names and call signs have been anonymized)

The AIS receiver that we consider (locally) collects these broadcast AIS messages, and produces a single TCP/IP stream of messages that are encoded using a standard from the NMEA (National Marine Electronic Association).

We focus on the two main types of messages with the following attributes:

- *point* message: MMSI, longitude, latitude, heading, speed, ... ;
- *info* message: MMSI, name, call sign, destination, ...

Each message type contains an MMSI number that uniquely identifies the reporting vessel, and provides the most recent information about the vessel. Message type *point* corresponds to AIS message types 1 and 3, and represents dynamic data such as position (longitude and latitude), compass heading, and speed. Message type *info* corresponds to AIS message type 5, and represents relatively static data such as name, call sign, and destination.

11.2.2 Display: Google Earth

The display that we consider is Google Earth,[1] which shows a 3D representation of the globe together with some overlays. We intend to create an overlay that displays

[1] http://earth.google.com/

for each vessel an icon, such that the position and rotation of the icon correspond to the vessel, and such that the size of the icon relates to the speed of the vessel.

Each overlay is described using KML (Keyhole Markup Language), and contains a set of placemarks that correspond to geographic positions, polygons or paths. In order to dynamically update an overlay, Google Earth provides network links that periodically reload a (possibly updated) KML file using an HTTP connection.

We consider placemarks with the following attributes:

- *KML* placemark: id, name, longitude, latitude, heading, scale, description, ...

Each placemark has a position (longitude and latitude) and a name that is displayed near the placemark. The heading and scale affect the rotation and size of the icon. In addition, each placemark has a description with additional information, and, for technical purposes, a unique id (which is not visible in Google Earth).

11.3 Interface Incompatibilities

The integration goal is to display in Google Earth per vessel the most recent information from the AIS receiver. These two systems fit conceptually as the AIS receiver provides (data related to) geographic positions of vessels, and Google Earth can display (data related to) geographic positions of placemarks. However, as these systems were not designed to be used together, their technical interfaces do not fit.

It turns out that there are a lot of incompatibilities; see Table 11.1. To start with, they use different interface technologies: the TCP/IP protocol (server side) versus the HTTP Get protocol (client side). These protocols do not fit technically, but also the underlying interface behavior is incompatible: a pushing behavior versus a request/reply behavior. When looking at the transmitted data, it is immediately clear that they use different representations: the NMEA encoding and the KML encoding. This difference also affects the semantic entities that are used: an AIS message of a single vessel versus an overview picture of all vessels in an area.

In turn, the four incompatibilities from Table 11.1 can be classified using two dimensions; see Table 11.2. The one dimension distinguishes incompatibilities related to behavior from those related to data. The other dimension distinguishes abstract designs from concrete implementations.

Table 11.1 Overview of interface incompatibilities

	AIS receiver	Google Earth
Interface technology	TCP/IP (server side)	HTTP Get (client side)
Interface behavior	Pushing	Request/reply
Semantic entities	AIS message of a vessel	Overview picture of an area
Data representation	NMEA	KML

Table 11.2 Classification of interface incompatibilities from Table 11.1

	Behavior	Data
Abstract design	Interface behavior	Semantic entities
Concrete implementation	Interface technology	Data representation

We focus on the two incompatibilities related to abstract design as these affect the core behavior of the adapter. In Sect. 11.4 we describe adapter techniques for interface behavior, and in Sect. 11.5 for semantic entities.

11.4 Adapters Based on Controller Synthesis

In this section we focus on incompatibilities with respect to interface behavior. We present an adapter specification, from which an adapter can be generated using algorithms for controller synthesis [15, 16] from the field of control theory. We first describe the chain from specification via generation to implementation, and then briefly evaluate it.

11.4.1 Specification

To address interface behavior, the starting points are the behavioral interface models of the systems to be integrated. Such models describe the orders in which messages can be sent and received by the systems. Interface behavior is typically modeled in terms of formalisms such as state transition systems, process algebra, or (open) Petri nets [6, 7].

For our running example, some models are shown in Fig. 11.2a and Fig. 11.2c; the model in Fig. 11.2b will be discussed in Sect. 11.4.2. These models are open Petri nets, but they can be interpreted as state transition systems as follows. The external interface is depicted as a dashed border; for each type of message there is a circle on the border. Each circle that is not on a border can be interpreted as a state; the initial state is marked with a black dot. Each square can be interpreted as a transition which has one input state (incoming arrow from a state) and one output state (outgoing arrow to a state), and possibly sends (outgoing arrow to an interface) and receives (incoming arrow from an interface) several messages. The communication channels are not guaranteed to preserve the message order.

The AIS receiver (see Fig. 11.2a) can send a message AisPoint (AIS point message) or a message AisInfo (AIS info message), and afterwards it waits to receive a message AisAck (acknowledgment message). Similarly, Google Earth (see Fig. 11.2c) can send a message KmlReq (request for KML data), and afterwards it waits to receive a message KmlResp (response with the requested KML data).

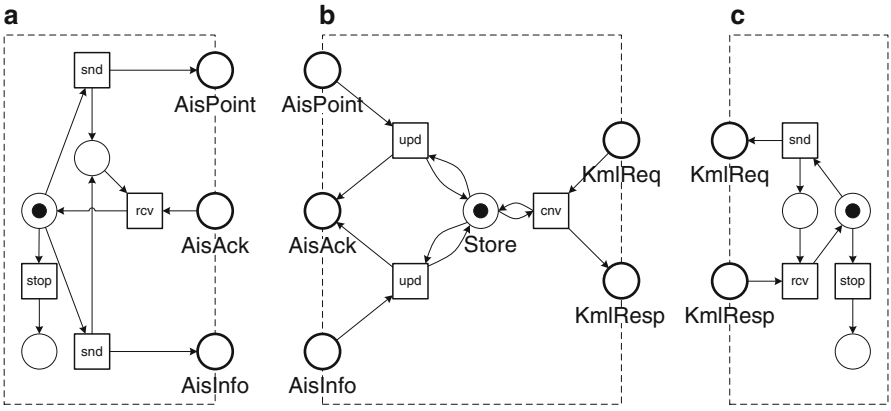


Fig. 11.2 Controller-based adapters: behavioral models of the systems. (**a**) AIS receiver. (**b**) Generated adapter. (**c**) Google Earth

After any number of iterations of this behavior, each of these two systems can (independently) decide to stop and reach its final state.

Message AisAck is introduced because the AIS receiver provides an ordered stream of messages, whereas the open Petri net formalism considers communication channels that are not guaranteed to preserve the message order. The messages AisPoint, AisInfo and KmlResp contain data, whereas messages AisAck and KmlReq are empty signal messages.

Mismatches in the interface behavior result in the violation of a desired behavioral property, like absence of deadlocks (in every non-final state). In our running example, this can happen when messages AisPoint (or AisInfo) and KmlReq have been sent, and the systems are waiting for AisAck and KmlResp respectively. To resolve such a deadlock, an adapter should provide (at least) one of the messages that the systems are waiting for.

However, absence of deadlocks is not enough as a specification. For example, we do not want an adapter that can simply destroy messages with important data, such as AisPoint and AisInfo. So, although we focus on generating the behavior of the adapter, apparently the semantics of the data has an impact on possible behavior. To model these domain-specific data dependencies we extend the specification with transformation rules that indicate what data can be created, deleted, converted, etc.

For our running example, an example set of transformation rules can be found in Fig. 11.3. Each transformation rule consists of the messages that are consumed, followed by a $\mapsto$ sign, and the messages that are produced.

This specification uses an auxiliary message called Store, that represents an internal copy of the most recent information about each vessel; initially there is a default (empty) Store, and a Store still exists in the final state. The first transformation rule specifies that the Store can be updated using an AisPoint message, thereby generating an AisAck message; similarly in the second transformation rule for an AisInfo message. By only generating an AisAck message as part of these rules,

Initially:	Store	
Finally:	Store	
AisPoint, Store	↦	Store, AisAck
AisInfo, Store	↦	Store, AisAck
Store	↦	Store, KmlResp
KmlReq	↦	

Fig. 11.3 Controller-based adapters: domain-specific transformation rules

it is guaranteed that the AIS messages update the Store in the right order. The third transformation rule specifies that a KmlResp message can be created based on the Store; the fourth transformation rule specifies that a KmlReq message can be deleted.

The last two transformation rules could be combined, but this is not necessary in the specification as KmlReq is just a signal message. In Sect. 11.4.2 we will see that these two rules are automatically combined in the generated adapter.

The set of transformation rules indicates the elementary operations that the adapter can perform, but it does not indicate whether or when the adapter should apply them. The transformation rules from Fig. 11.3 also do not explain how the Store is updated, or how to derive a KmlResp from the Store; it is not even visible which rules change the Store. To obtain an executable adapter, we need to associate to each rule a detailed data conversion, but these have been omitted in Fig. 11.3.

Consequently the adapter specification [3] consist of three parts:

- *Behavioral interface models*: models that describe the interface behavior of the systems to be integrated;
- *Transformation rules*: domain-specific rules that describe the elementary operations that adapters can perform;
- *Behavioral property*: property (like absence of deadlocks) that is required in the integrated system.

In case no behavioral interface models are available for the systems to be integrated, there may still be execution logs with typical interface behavior. Such logs may be usable to discover behavioral interface models using process mining techniques; see also Chap. 9.

In some cases an adapter needs to be developed such that one system can be replaced by another system together with the adapter. Thus not all systems to be integrated are known. The hardest possible environment of the system can be used as an approximation, and it can be computed as the maximal controller [11, 12] (sometimes called canonical test) of the original system.

11.4.2 Generation

Given a behavioral model of a system and a behavioral property, a controller (if one exists) is a system such that the composition of the original system and the controller

satisfies the behavioral property. In the absence of transformation rules, adapter generation would be equivalent to applying controller synthesis to the composition of all systems to be integrated.

However, the transformation rules add some restrictions to adapter generation. These can be addressed by generating an adapter that consists of two parts [3] (see also the core adapter in Fig. 11.4a):

- *Engine*: The engine guarantees that the final adapter can only apply the provided data transformations; it is connected to the systems to be integrated and to the controller. The engine is generated directly from the transformation rules.
- *Controller*: The controller guarantees that the required behavioral property holds in the integrated system; it is only connected to the engine. The controller is generated by applying controller synthesis [15, 16] for the required behavioral property to the composition of the engine and the systems to be integrated.

Thus this approach applies a strict separation between data and behavior. In general, the adapter specification admits several, non-equivalent adapters to be generated. For example, controller synthesis algorithms can produce several controllers, which all produce an adapter that satisfies the adapter specification. In [3] we also show two ways to model an engine, but these result in equivalent adapters.

For our running example, a behavioral model of a generated adapter can be found in Fig. 11.2b; in this model it is not possible to distinguish the engine and the controller any more. Once an AisPoint message is received, the Store is updated and an AisAck message is generated; similarly for an AisInfo message. Once a KmlReq message is received, the data from the Store is copied and converted into a KmlResp message. In this adapter the Store separates the interactions with the AIS receiver and the interactions with Google Earth.

Note that the generated adapter combines the last two transformation rules from Fig. 11.3: it only creates a next KmlResp message after receiving a next KmlReq message. The generated controller combines them to establish absence of deadlocks, as in the final state, there should not be any messages left in the channels. By receiving a next KmlReq message, it is guaranteed that Google Earth has not yet stopped, and hence can consume a next KmlResp message.

11.4.3 Implementation

This adapter generation approach is implemented in the tool Marlene.[2] It is based on open Petri net models, and abstracts from the details of the data conversions.

Within the POSEIDON project we have used Marlene as the basis of a prototype that produces executable adapters, and hence addresses all incompatibilities. For the data-related incompatibilities, we have attached manually-specified data conversions to the transformation rules (and to the engine). For the interface technology,

[2] http://www.service-technology.org/tools/marlene/

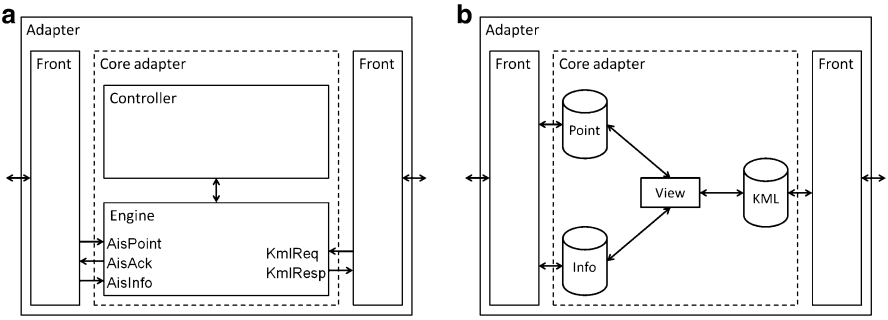


Fig. 11.4 Adapter architectures used in Sects. 11.4 and 11.5. (**a**) Controller-based adapter. (**b**) View-based adapter

we have introduced front-ends that convert all external interface technologies to send or receive operations on message channels. The architecture of such an adapter is depicted in Fig. 11.4a.

11.4.4 Evaluation

Recent research on adapter generation has led to several approaches [1, 2, 8] that are related to controller synthesis. Typically a custom language is introduced for specifying a class of data transformations, and then an adapter generation algorithm is proposed that is somehow related to controller synthesis. By translating the data transformations into an engine, we have shown in [3] that adapters can be generated using any existing controller synthesis algorithm without modification.

Controller synthesis is very powerful, and it can automatically resolve very subtle behavioral mismatches; see the examples in [3, 14]. The resulting adapters allow highly-concurrent interactions without deadlocks. Algorithms for controller synthesis are usually based on state-space exploration. The time needed for generating an adapter varies from specification to specification; in the examples from [3] the adapter is generated within a second.

Our running example has been selected for being understandable and for being usable to illustrate the two approaches to adapter generation. In this example the behavioral models are quite simple, and the transformation rules are already very close to the model of the generated adapter. This kind of examples has motivated our search for other adapter generation techniques, as described in Sect. 11.5. In particular we aim to specify such an adapter in a more convenient way.

11.5 Adapters Based on Incremental View Maintenance

In this section we focus on incompatibilities with respect to semantic entities. We present an adapter specification, from which an adapter can be generated using algorithms for incremental maintenance of materialized views [4, 9] from the field of database theory. We first describe the chain from specification via generation to implementation, and then briefly evaluate it.

11.5.1 Specification

To address semantic entities, the starting point is the communicated data. In our running example, the AIS receiver provides two types of AIS messages with partial updates of a single vessel, whereas Google Earth requests a single overview picture of all vessels in an area. Ignoring the data representations, these communications can nicely be modeled in terms of three (relational) database tables:

- Source table *point*: most recent position message per vessel;
- Source table *info*: most recent info message per vessel;
- Target table *KML*: most recent placemark data per vessel.

The attributes of these three tables correspond to the data attributes mentioned in Sect. 11.2; the primary keys are the attributes MMSI, MMSI, and id, respectively. Then the two systems to be integrated can be modeled as follows:

- AIS receiver: regularly perform an incremental update of tables *point* and *info*;
- Google Earth: periodically query table *KML*.

The AIS receiver performs updates on the two source tables in the order in which it receives the AIS messages.

What remains to be specified is a relation between the target table and the source tables. In the context of databases, this means that we need to specify the target table as a view on the source tables. Although the source tables change continuously, we specify the view statically. For the adapter this means that whenever the source tables do not change for a sufficiently long period of time, then, after a while, the target table should have the specified contents.

For our running example, we specify the target table *KML* as a view on the source tables *point* and *info*; see Fig. 11.5. The join operator $\bowtie_{MMSI}$ indicates that we first combine rows from the tables *point* and *info* with the same MMSI value. Afterwards, we compute for each combined row the attribute values of the corresponding row in the *KML* table. For example, as AIS provides longitude and latitude in 10^{-4} minutes, the longitude and latitude from table *point* need to be divided by $60 * 10^4$ before being stored in table *KML*.

In database theory, many types of joins have been defined. The type of join can be used to indicate when an MMSI value should appear in table *KML*, as summarized in

KML: $point \bowtie_{MMSI} info$	
id	MMSI
name	MMSI
longitude	*point*.longitude/$(60*10^4)$
latitude	*point*.latitude/$(60*10^4)$
heading	*point*.heading
scale	0.5+log(*point*.speed+1)/5
description	*info*.name + " (" +*info*.callsign + ") for " + *info*.destination
...	...

Fig. 11.5 View-based adapters: view specification

Table 11.3 Interpretation of join types for Fig. 11.5

	Table *point*	Table *info*
Inner join	Mandatory row	Mandatory row
Left-outer join	Mandatory row	Optional row
Right-outer join	Optional row	Mandatory row
Outer join	Optional row	Optional row

Table 11.3. The first choice for our running example would be the inner join, which adds a vessel to table *KML* once both types of AIS messages have been received from the vessel. However, as the *info* messages are broadcast infrequently, it may be more useful to choose the left-outer join, which adds a vessel to table *KML* once its position (from a *point* message) is known. Additional information from an *info* message is added when available, and default values are used otherwise.

Consequently the adapter specification [10] consist of two parts:

- *Database interface models*: database schemes that describe the interfaces of the systems to be integrated;
- *Database view relations*: database queries that specify the semantic relation between source tables and target tables.

11.5.2 Generation

In database management systems (DBMS), there are two ways to deal with views:

- *Query abbreviation*: views are seen as an abbreviation of a query on the source tables. This gives no overhead when updating the source tables, but every query on the view needs to be evaluated in terms of the underlying source tables.
- *Materialized query*: views are seen as a materialized query that is stored as a table. This means that queries on the view can efficiently be evaluated in terms of the stored table, but every incremental update in the source tables needs to be propagated to the stored table for the view. This updating is called incremental maintenance of materialized views [4, 9].

```
synchronized public void publishLeft(LI lmsg) {
    Object joinValue = fieldMapLI.getFieldValue(function.getLeftJoinKey(), lmsg);
    Collection<RI> rmsgs = rightInput.subQueryByField(function.getRightJoinKey(), joinValue);
    if (rmsgs.isEmpty()) {
        O result = function.leftInputOnly2output(lmsg);
        output.publish(result);
    } else {
        for (RI rmsg: rmsgs) {
            O result = function.inputs2output(lmsg, rmsg);
            output.publish(result);
        }
    }
}
```

Fig. 11.6 Java code fragment: generic left-outer join responding to an update in the left source

Similarly, the adapter specification leaves some degrees of freedom for the adapter generation, e.g., whether to compute the KML data on request from the source tables, or to continuously update an internal buffer with KML data. In the latter case, we can use solutions for incremental view maintenance; these are often described in terms of relational algebra expressions, from which the behavior of an adapter can be extracted as described in [10]. For example, once an entry in source table *point* is updated, the adapter should query source table *info* for a corresponding row, then perform the specified data conversion, and finally update target table *KML*. In some cases, the query on the source table can also be replaced by a query on the target table, thus avoiding the need to store some of the sources. This leads to a classical performance trade-off between computation time and storage requirements, which should be resolved based on the expected usage characteristics.

11.5.3 Implementation

Within the POSEIDON project, we have developed a Java framework with classes for the various kinds of tables (materialized and non-materialized) and relational operators (like various kinds of joins). Each class has associated methods for establishing the view relations, including code for incremental view maintenance; see for example Fig. 11.6. To generate an adapter, the specification is copied in terms of these classes, in particular by instantiating the data conversions; see for example Fig. 11.7. As the implementation code is already associated to the classes, the generation of the adapter costs no time, but there are parameters to tune the implementation.

For the behavior-related and data representation incompatibilities, we have introduced front-ends that convert all external interface technologies to database operations (such as publish/subscribe and query) on relational database tables. Thus the systems to be integrated can (indirectly) query tables, publish changes in them, and subscribe on them. The architecture of such an adapter is depicted in Fig. 11.4b.

```
public void applyLeft(aisPosReport msg, GoogleEarthDataPointExt partialOutput) {
    partialOutput.id = msg.userid.toString();
    partialOutput.name = msg.userid.toString();
    partialOutput.longitude = (float) (msg.longitude.longValue() / (60 * 10000.));
    partialOutput.latitude = (float) (msg.latitude.longValue() / (60 * 10000.));
    partialOutput.heading = msg.True < 360 ? msg.True : 0;
    partialOutput.scale = (float) (.5 + Math.log(msg.sog + 1)/5);
}

public void applyRight(aismsg_5 msg5, GoogleEarthDataPointExt partialOutput) {
    partialOutput.id = msg5.userid.toString();
    partialOutput.name = msg5.userid.toString();
    partialOutput.description = msg5.name + " (" + msg5.callsign + ")  for " + msg5.dest;
}
```

Fig. 11.7 Java code fragment: specific data conversion from Fig. 11.5

11.5.4 Evaluation

Our study of incremental view maintenance was triggered by several example adapters from the domain of maritime safety and security; see [10]. Incremental view maintenance has been studied extensively in the context of database management systems (DBMS), viz., for efficiently updating materialized views in a transaction that changes the source tables. However, in the adapter context there is typically no database management system. In [10] we have shown that it is still beneficial to consider a lot of interface technologies as database operations, and then apply techniques from the field of databases.

In our prototype implementation, the incremental view maintenance code is attached to the relational operators from the specification; so, the adapter generation itself costs no time. Moreover, the behavior of the generated adapter is intuitively quite predictable, in particular in comparison to the controller synthesis approach. We have informally compared some of the generated adapters with manually-developed adapters, and noticed that their runtime behavior is very similar. Although when running the generated adapters there is a little overhead caused by additional method invocations and generic code, this has never been any problem in the context of the POSEIDON demonstrator as described in Chap. 4.

The POSEIDON demonstrator contains many generated adapters for various interface technologies, such as TCP/IP, HTTP, RMI, DDS [13], Fractal[3]/Atlas[4] (see also Chap. 14), JDBC, etc. There are several adapters that are part of the normal data flow through the system, including some with multiple inputs and multiple outputs. In addition, there are a lot of "probe" adapters that visualize the data in the various processing stages, using Google Earth and process mining tool ProM.[5] Our running example is a probe that visualizes raw sensor data using Google Earth as in

[3] http://fractal.ow2.org/

[4] http://swerl.tudelft.nl/bin/view/Main/Atlas

[5] http://www.processmining.org/prom/

Fig. 11.1. Such probe adapters proved to be pivotal in analyzing the functioning of the POSEIDON demonstrator, especially during integration; see Chap. 4.

Static specifications in terms of data are generally considered to be simpler to make and maintain than dynamic specifications in terms of behavior. In particular, small changes in the type of join can have a large impact on the behavior of the adapter. Even without applying adapter generation, we expect that applying such an adapter specification is already beneficial for developing this kind of adapters.

11.6 Conclusions and Further Work

Adapters are used for bridging interface incompatibilities during system integration. The reported work was driven by a search for convenient ways to specify adapters such that afterwards an adapter can be generated. We use model-based specifications that describe the systems to be integrated, and the requirements that need to be established by the adapter.

The goal of adapter generation is to translate such a specification model of *what* is required into an adapter model of *how* it can be implemented. The main results are two techniques for adapter generation, depending on the primary incompatibility that needs to be resolved. In particular, we have shown how to use several techniques from other fields for generating adapters, viz., (supervisory) controller synthesis from the field of control theory and incremental view maintenance from the field of database theory.

Within the POSEIDON project we have demonstrated these two techniques using prototype implementations that produce executable adapters. These prototypes use architectures with reusable front-ends for the specific details of the interface technologies. Such adapter architectures separate the core functionality from any specific code for the interface technologies.

It is further work to investigate whether it is useful to integrate these two adapter generation approaches into one integration framework. So far we have not seen a good example where this would be really beneficial. A possible way to integrate them is to first use a view-based approach to create a more sophisticated engine, and then use this engine as part of the controller-based approach.

Acknowledgements This research has been carried out as a part of the POSEIDON project at Thales under the responsibilities of the Embedded Systems Institute (ESI). This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

The authors like to thank Maurice Glandrup for providing relevant examples of adapters in the domain of maritime safety and security. The authors also like to thank Christian Günther for integrating the discovery of behavioral interface models using process mining techniques in our prototype based on controller synthesis.

References

1. Bracciali A, Brogi A, Canal C (2005) A formal approach to component adaptation. J Syst Softw. Elsevier 74(1):45–54
2. Brogi A, Canal C, Pimentel E, Vallecillo A (2004) Formalizing web service choreographies. In Proceedings of the 1st international workshop on web services and formal methods (WS-FM 2004). Volume 105 of ENTCS. Elsevier, pp 73–94
3. Gierds C, Mooij AJ, Wolf K (2012) Reducing adapter synthesis to controller synthesis. IEEE Trans Serv Comput 5(1):72–85
4. Griffin T, Libkin L (1995) Incremental maintenance of views with duplicates. In Proceedings of the 1995 ACM SIGMOD international conference on management of data (SIGMOD'95). ACM, New York, pp 328–339
5. International Telecommunications Union (2001) Technical characteristics for a universal shipborne Automatic Identification System using time division multiple access in the VHF maritime mobile band. Recommendation ITU-R M.1371-1
6. Kindler E (1997) A compositional partial order semantics for Petri net components. In Proceedings of the 18th international conference on application and theory of Petri Nets (ICATPN 1997). Volume 1248 of lecture notes in computer science. Springer, Berlin/Heidelberg, pp 235–252
7. Massuthe P, Reisig W, Schmidt K (2005) An operating guideline approach to the SOA. Ann Math Comput Teleinform 1(3):35–43
8. Melliti T, Poizat P, Ben Mokhtar S (2008) Distributed behavioural adaptation for the automatic composition of semantic services. In Proceedings of the 11th international conference on fundamental approaches to software engineering (FASE 2008). Volume 4961 of lecture notes in computer science. Springer, Berlin/Heidelberg, pp 146–162
9. Mohania M, Konomi S, Kambayashi Y (1997) Incremental maintenance of materialized views. In Proceedings of the 8th international conference on database and expert systems applications (DEXA 1997). Volume 1308 of lecture notes in computer science. Springer, Berlin/Heidelberg, pp 551–560
10. Mooij AJ (2013) System integration by developing adapters using a database abstraction. Inf Softw Technol. 55(2):357–364. http://dx.doi.org/10.1016/j.infsof.2012.08.015
11. Mooij AJ, Voorhoeve M (2009) Proof techniques for adapter generation. In Bruni R, Wolf K (eds), Web services and formal methods (WS-FM 2008). Volume 5387 of lecture notes in computer science. Springer, Berlin/Heidelberg, pp 207–223
12. Mooij AJ, Parnjai J, Stahl C, Voorhoeve M (2011) Constructing replaceable services using operating guidelines and maximal controllers. In Bravetti M, Bultan T (eds) Web services and formal methods (WS-FM 2010). Volume 6551 of lecture notes in computer science. Springer, Berlin/Heidelberg, pp 116–130
13. Object Management Group (2007) Data distribution service for real-time systems. Version 1.2. http://www.omg.org/spec/DDS/1.2/PDF/
14. van der Aalst WMP, Mooij AJ, Stahl C, Wolf K (2009) Service interaction: patterns, formalization, and analysis. In Bernardo M, Padovani L, Zavattaro G (eds) Advanced lectures of the 9th international school on formal methods for web services. Volume 5569 of lecture notes in computer science. Springer, Berlin/Heidelberg, pp 42–88
15. Wolf K (2009) Does my service have partners? Trans Petri Nets Other Models Concurr 2: 152–171
16. Wonham WM (2007) Supervisory control of discrete-event systems. Technical Report ECE 1636F/1637S 2007-08, University of Toronto, Department of Electrical and Computer Engineering, Systems Control Group

Chapter 12
The POLIPO Security Framework

Daniel Trivellato, Sandro Etalle, Erik Luit, and Nicola Zannone

12.1 Introduction

Systems of systems are coalitions of autonomous systems that collaborate to achieve a common goal. The systems in a system of systems often belong to different security domains, which are governed by different authorities employing heterogeneous protocols, vocabularies, data models and organizational structures. Furthermore, systems of systems are often dynamic, with systems joining and leaving the coalition at runtime. An example of system of systems is a fleet of ships from different NATO countries collaborating in a patrolling mission.

Despite offering a high degree of operational flexibility, the systems of systems paradigm has a strong impact on systems interoperability and on the security requirements of the coalition members (hereafter called *parties*). In fact, during the operation of a system of systems parties are required to exchange information (e.g., their current location) with the other members of the coalition. This information, however, might be sensitive and should be accessed exclusively by authorized parties, which may vary depending on the context (e.g., the location of the requester, the criticality of a situation) and the content of the information. Therefore, along with the development of systems of systems comes the demand for a flexible security framework that faces the related security challenges.

D. Trivellato (✉) • E. Luit • N. Zannone
Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands
e-mail: d.trivellato@tue.nl; e.luit@tue.nl; n.zannone@tue.nl

S. Etalle
Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands

Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, Enschede, The Netherlands
e-mail: s.etalle@tue.nl; sandro.etalle@utwente.nl

P. van de Laar et al. (eds.), *Situation Awareness with Systems of Systems*,
DOI 10.1007/978-1-4614-6230-9_12,

In particular, to deal with the dynamic nature of systems of systems, a security framework should incorporate security-relevant contextual information into access control decisions [2]. Contextual information may consist of "basic" environmental conditions (e.g., the location of the requester, the time of access), or more complex conditions derived from the basic ones (e.g., an emergency situation due to the collision between two vessels). Context-aware access control models [2, 6] can be employed to serve this purpose.

In addition, contrarily to centralized systems where users and resources belong to a single, trusted domain, in a system of systems parties often do not know each other beforehand. It is therefore not possible to rely on identity-based approaches to regulate the access to local resources. Trust management [3] has been proposed as a solution to this problem. Trust management is an approach to access control in distributed systems where access decisions are based on the attributes of a requester, which are certified by means of digital credentials. Credentials are certificates issued by a party attesting that a subject has a certain attribute, and they are digitally signed to ensure their authenticity and integrity.

The problem of most of the existing trust management frameworks (e.g., [1, 12, 14]) is that they assume a complete agreement among the parties in a system of systems on the vocabulary used to denote subjects' attributes and to describe the concepts and relationships that characterize a given domain. In dynamic coalitions of heterogeneous systems, however, parties will more likely "speak" different languages and employ different organizational models; nevertheless, they must be able to collaborate to achieve the coalition's goal. As a first step towards enabling mutual understanding and thus interoperability among parties in a system of systems, semantic approaches have been adopted for policy specification [11, 26]. In particular, ontologies have been used to assign a precise structure and semantics to information, and to define domain knowledge. Accordingly, parties can refer to ontologies to provide semantics to the terms used to specify their policies and to denote the concepts in the application domain.

The use of ontologies alone, however, is not enough to achieve interoperability. In fact, parties might refer to different ontologies to denote the same (or similar) concepts in the domain. For instance, each NATO country may use different terms (and a different hierarchy) to denote the ranks of the officers on its ships. Semantic alignment techniques [7, 9] need thus to be employed to map concepts from different ontologies, i.e., to align vocabularies and organizational models. A major drawback of the existing semantic alignment techniques is that they require complete knowledge of the ontologies to be aligned. In many systems of systems, however, this requirement is not admissible since parties do not know each other beforehand or might want to keep part of their knowledge base confidential. Therefore, a solution that is effective also when working with partial knowledge needs to be devised.

In this chapter we present POLIPO, a security framework that protects the *confidentiality* and *integrity* of information while enabling *autonomy* and *interoperability* among the parties in a system of systems. POLIPO combines context-aware access control with trust management to protect information

from unauthorized access (confidentiality) and improper modification (integrity). Autonomy and interoperability are enabled by the use of ontology-based services. More precisely, parties may refer to different ontologies in the specification of their policies and to describe domain knowledge and context information. This allows each party to employ the organizational model and terminology that they consider most appropriate within their system. The semantic alignment technique presented in [22] is then employed to align their vocabularies, allowing for mutual understanding.

We present an application of POLIPO to a scenario in the maritime safety and security domain, where a prototype implementation of the framework is employed by the parties in the system of systems to protect the local resources. The framework's architecture, inspired by XACML [17], consists of a set of core security components (e.g., the access control and trust management components) complemented with the ontology-based services. All components and services have been implemented following the service oriented architecture paradigm [18] to facilitate their integration and deployment into existing systems of systems. The modularity of the framework allows for the integration of additional services to support the evaluation of policies and provide additional functionalities, such as a reputation system for the identification of trustworthy information sources (see Chap. 13).

The remainder of this chapter is organized as follows. Section 12.2 presents a use case scenario for a system of systems in the maritime safety and security domain, and elicits a set of basic requirements that a security framework for systems of systems should satisfy. Section 12.3 introduces the ingredients of the POLIPO framework and presents the framework's architecture. A prototype implementation of the framework is then presented in Sect. 12.4. Finally, Sect. 12.5 concludes the chapter.

12.2 Requirements Elicitation

In this section we first introduce a scenario for systems of systems in the maritime safety and security domain. Then, we identify the main requirements that a security framework for systems of systems should satisfy.

12.2.1 Case Study: Maritime Surveillance

We present a scenario which focuses on the dynamic evolution of systems of systems in response to emergency situations. In particular, we consider a system of systems headed by the European Union (EU) which has the goal of detecting and preventing terrorist attacks against European harbors. The system of systems consists of vessels of the EU Naval Force (EU NAVFOR) which patrol the coast of

Somalia, gathering and exchanging information about the vessels transiting in that area. This information is collected, processed and analyzed by an operation control center in Northwood, UK, which coordinates the activities of the EU NAVFOR vessels. In addition to the EU NAVFOR vessels, search and rescue vessels of EU countries may temporarily join the coalition and get access to the information collected by the EU NAVFOR in case of emergency (e.g., to give assistance to vessels getting into troubles). We point out that all the names and events introduced in this scenario are purely fictitious.

The scenario consists of the following steps:

1. A cargo ship called Blue Star is transiting through the Gulf of Aden towards its final destination, Copenhagen. Blue Star is under investigation by the Danish navy because it is suspected of being involved in terrorist activities. The Danish navy has infiltrated agents who are investigating the evolution of these activities.
2. In the proximity of the Dutch coast the cargo ship Blue Star gets into trouble due to a storm and starts drifting. A vessel of the Dutch coastguard (NL-Lifeboat) is nearby and prepares to intervene to give assistance to Blue Star's crew. In order to prepare the intervention, NL-Lifeboat needs information about the cargo transported by Blue Star. By checking the port from which Blue Star departed, NL-Lifeboat infers that the cargo ship has transited off the Somali coast. Therefore, NL-Lifeboat sends a request for additional information about Blue Star also to the EU NAVFOR operation control center in Northwood.
3. Currently, the situation of Blue Star is not considered by the operation control center critical enough to disclose to NL-Lifeboat the intelligence collected by the Danish navy. Therefore, the operation control center does not provide details about the cargo transported by Blue Star to NL-Lifeboat. Furthermore, NL-Lifeboat is requested not to intervene.
4. After a while, however, the situation becomes more critical and the operation control center loses connection with Blue Star. Consequently, the request for intervention from NL-Lifeboat is accepted: NL-Lifeboat is temporarily allowed by the operation control center to access the details about Blue Star's cargo. Through this information NL-Lifeboat's operators find out that Blue Star's cargo contains Anthrax that was possibly meant to be distributed to terrorist cells in Europe. The rescuers must use protective clothes and other ships must be kept at a safe distance of at least 500 m.

Table 12.1 presents the security policies governing the scenario, i.e., the rules defining the authorized accesses to the information controlled by each party in the system of systems. To each policy rule we assign a unique identifier (column *Policy ID*) that we use to refer to the rule in the rest of the paper. Note that the vocabulary used to specify policy rules by the different parties is not always consistent. For example, policy rule NL_1 states the Dutch navy certifies all the "search and rescue vessels" of the Dutch coastguard; NL-Lifeboat, however, is certified as a "search and rescue lifeboat" by the Dutch coastguard (policy CG_1). In this case, an alignment of the vocabularies of the Dutch navy and coastguard is required in order to allow for a certification of NL-Lifeboat by the Dutch navy.

Table 12.1 Security policies of the parties in the scenario

Party	Policy ID	Policy rule
Operation control center	OCC_1	Operators on vessels of the EU NAVFOR may access all the available information about the ships transiting in the operation area
	OCC_2	Operators on search and rescue vessels certified by the navy of an EU country may access all the available information about a ship in case that the ship needs assistance
Dutch navy	NL_1	All search and rescue vessels certified by the Dutch coastguard are certified as search and rescue vessels by the Dutch navy
Dutch coastguard	CG_1	NL-Lifeboat is a search and rescue lifeboat of the Dutch coastguard

The interactions between the actors in the scenario are shown in Fig. 12.1. Each interaction is labeled with the step of the scenario in which it is described. In case that a step involves multiple interactions, we order them by adding a letter to the label. Since security policies are often deemed to be confidential (see Sect. 12.2.2 for a more comprehensive discussion), in our scenario we assume that no party has access to the security policies of the other parties. Therefore, each time a party requires a certificate issued by another party, an explicit request needs to be made.

Every time a party receives a request, it evaluates the request against its security policy and returns a response accordingly. Notice, for instance, that the same request for details about Blue Star's cargo sent by NL-Lifeboat to the operation control center in Northwood at two different moments in time (messages 2 and 4a in Fig. 12.1) leads to two different responses (messages 3 and 4f respectively). In fact, since in step 3 of the scenario the operation control center does not consider Blue Star to be needing assistance, policy OCC_2 cannot be applied. On the contrary, in step 4 the situation of Blue Star is considered critical and the details about its cargo are provided to NL-Lifeboat, upon verifying that NL-Lifeboat is a search and rescue vessel certified by the Dutch navy.

12.2.2 Security Requirements

As a first step towards the elicitation of our security objectives, we derive the characteristics of systems of systems which are relevant for the design of a security framework. The distinguishing features of systems of systems are as follows:

- *Dynamicity*: systems of systems are constantly evolving. Systems may leave a system of systems at any time while new systems may join the coalition, depending on the context or the progress towards the goal. In the scenario introduced in the previous section, for example, NL-Lifeboat joins the EU NAVFOR system of systems during the emergency of the cargo ship Blue Star,

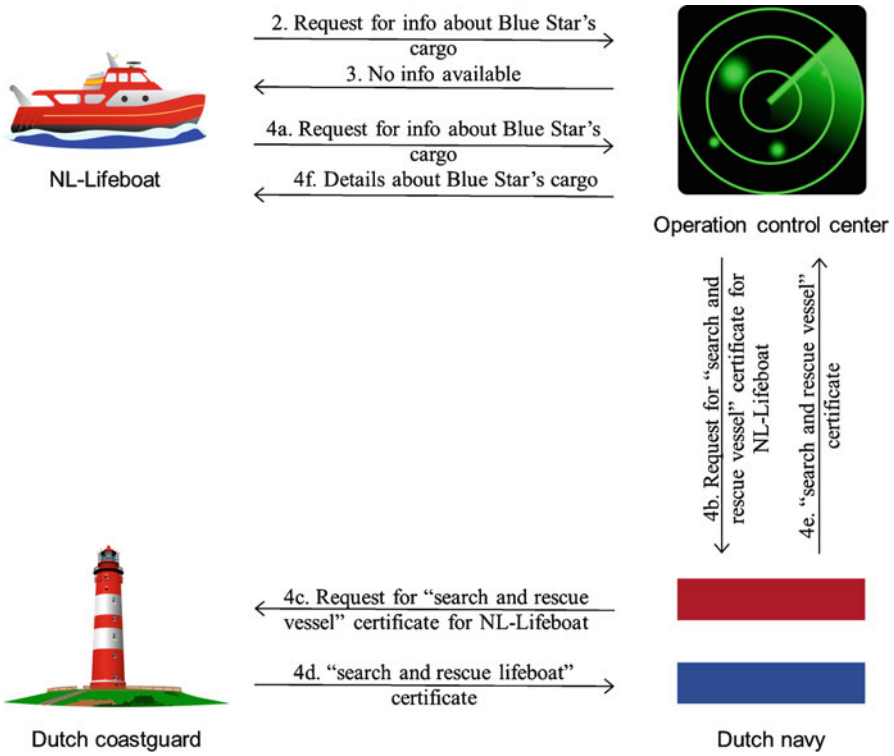


Fig. 12.1 Interactions among the parties in the scenario

and leaves the coalition when the emergency is over. Similarly, the information that systems need to exchange may be context-dependent. For instance, in case of emergency parties may be authorized to access sensitive information that they would normally not be allowed to access, such as NL-Lifeboat in our scenario.

- *Distribution*: contrarily to centralized systems where users and resources belong to a single, trusted domain, systems of systems are characterized by the absence of a central point of control. Each system in a system of systems is an independent, complex system which belongs to a (possibly) different security domain and is governed by a different authority (e.g., the operation control center and the Dutch coastguard). Furthermore, systems of systems are open systems in which parties may not know each other before joining the coalition. For example, the operation control center does not know whether NL-Lifeboat is actually a search and rescue vessel of an EU country, and therefore requests its certification from a trusted party, i.e., the Dutch navy.
- *Heterogeneity*: as a consequence of the autonomy of the systems in a system of systems, each system may adopt different data and organizational structures, and a different vocabulary to define the concepts and relationships in an application domain. In policy CG_1 (Table 12.1), for instance, the Dutch coastguard refers to

NL-Lifeboat as a "search and rescue lifeboat", whereas the Dutch navy issues only certificates with attribute "search and rescue vessel" (policy NL_1).

These features impose serious challenges on the design of a security framework. We identify the following set of core security requirements that a security framework for systems of systems should satisfy:

1. *Protection of information confidentiality and integrity*: sensitive data exchanged among the parties in the system of systems must be protected from unauthorized access and improper modification. For example, if terrorists would be able to access the information gathered by the EU NAVFOR, they would know that the Danish navy is investigating the activities of the cargo ship Blue Star. Security policies, however, may also contain sensitive information which needs to be protected. In particular, the disclosure of a security policy may reveal the relationship among some parties in a domain, such as business relationships or alliances, whose disclosure could be exploited by adversary parties, e.g., terrorists [19]. In addition, the disclosure of a policy may leak information that can be used to exploit vulnerable points of a system [20]: by knowing the security policies protecting the intelligence gathered by the EU NAVFOR, for instance, terrorists would know who are the parties that might be aware of their activities, and under which circumstances. Furthermore, by accessing a policy an adversary would know what credentials he needs to forge to illegitimately gain access to a resource [8]. Therefore, the disclosure of both data and policies shall be protected.

   The distributed and dynamic nature of systems of systems introduces two additional challenges to the confidentiality and integrity requirements. More precisely, policies that regulate the access to information need to:

   - Take into account that parties may not know each other beforehand. Therefore, authorizations cannot (always) be defined based on the identity of the requester. Rather, they should be based on his attributes (e.g., nationality, rank).
   - Be flexible and adaptable to different circumstances. In this respect, the evaluation of access requests should take the current context into consideration (e.g., the criticality of a situation).

2. *Autonomy of the parties involved*: the dynamicity of systems of systems implies that collaborations are often short-lived and the systems involved may change over time. In addition, the parties in a system of systems are independent systems that have individual objectives next to the ones of the coalition, and may be involved in more than one system of systems at a time. In this setting, we cannot expect the parties in a system of systems to employ common data models, organizational structures, and vocabularies for the specification of their security policies. Rather, parties should be able to employ different models and vocabularies.
3. *Interoperability among parties*: despite the heterogeneity in their organizational structures and vocabularies, parties must be able to understand each other for

the success of the coalition. In our scenario, for example, it is evident that NL-Lifeboat should be interpreted by the Dutch navy as a "search and rescue vessel", even though it is defined as "search and rescue lifeboat" by the Dutch coastguard.
4. *Ease of integration into existing systems*: the services and functionalities offered by the systems in a system of systems have strong implications on the design of the security components. On the one hand, the functional components of a system should be designed and implemented as independently as possible from its security components. On the other hand, a security framework for systems of systems should be easy to integrate into existing systems and should easily interface with the system's functionalities to protect the system's confidential information. In addition, the framework should be flexible and allow for an easy integration of additional security components that may become relevant during the lifetime of a system of systems (e.g., a reputation system for service selection).

Clearly, these requirements do not cover all the security aspects that are relevant for systems of systems. For instance, we omit the requirements for the protection of the systems and services in a system of systems from network attacks such as denial of service, eavesdropping, identity spoofing, etc. In the context of the POSEIDON project, however, our focus is mainly on the design of a solution that satisfies the requirements that are characteristic for systems of systems. The combination of our security framework with existing techniques that address other security requirements is out of the scope of this chapter.

12.3 The POLIPO Framework

In this section we present the POLIPO security framework. The contribution of POLIPO lies both in its ingredients, which implement new models and techniques especially designed to meet the security requirements of systems of systems, and in the way in which these ingredients have been combined into a unified framework. In the next two subsections we briefly introduce the framework ingredients (Sect. 12.3.1) and show how they have been integrated in the POLIPO architecture (Sect. 12.3.2).

12.3.1 Framework Ingredients

In order to satisfy the requirements introduced in Sect. 12.2.2, POLIPO combines models and techniques from the fields of computer security, knowledge representation, and software engineering. In particular, it relies on:

1. Context-aware access control models and trust management to protect the confidentiality and integrity of information;

2. Ontology-based services to enable autonomy and interoperability among the parties in a system of systems;
3. A service oriented architecture to allow for an easy integration and deployment of the framework into existing systems.

In the next paragraphs we will provide more details about each of these techniques.

12.3.1.1 Context-Aware Access Control and Trust Management

Context-aware access control is used to tackle the dynamicity of systems of systems: by incorporating context information (e.g., the location of the requester, the criticality of the situation) in access decisions, parties can specify flexible policies which adapt to different situations. Trust management, on the other hand, deals with the distributed nature of systems of systems. In trust management, access decisions are based on the attributes of a requester (e.g., vessels of the EU NAVFOR), which are certified by means of digital credentials issued by an authority, i.e., any party in the system of systems. The contribution of this approach is twofold: (a) contrarily to identity-based approaches, grounding an access decision on the certified attributes of a requester allows parties to exchange information with (previously) unknown entities; (b) each party can choose which authority to trust for certifying which attributes, and accept only credentials issued by that authority. For example, the operation control center trusts only the navies of EU countries for certifying search and rescue vessels.

To combine context-aware access control with trust management we have defined an ontology-based policy specification language [21]. The language allows for the specification of rules to constrain the access to the local resources of a party (called *authorization rules*) and to define the conditions under which a credential is released (*credential rules*). Context information and domain knowledge (e.g., the type of a ship) are incorporated into authorization and credential rules by referring to a knowledge base, which is represented by a set of ontologies. Examples of authorization rules are rules OCC_1 and OCC_2 in Table 12.1. In rule OCC_2, the need for assistance of a ship is determined by the operation control center by analyzing context information (e.g., whether the ship is drifting) available in the knowledge base. Policy rules NL_1 and CG_1 in Table 12.1 are examples of credential rules. More precisely, in rule NL_1 the Dutch navy states that it is willing to release a "search and rescue vessel" credential only to the vessels in possess of an equivalent credential issued by the Dutch coastguard. In rule CG_1, the Dutch coastguard certifies that vessel NL-Lifeboat is a "search and rescue lifeboat".

Other existing work [11, 26] proposes the use of ontologies for the specification of security policies. The expressive power of these languages, however, is limited and does not allow for the specification of several types of security constraints, as for instance separation of duty. Since such constraints are common to many application domains for systems of systems (e.g., maritime safety and security, business-to-business), these languages do not provide a valid solution. To overcome

this limitation, ontology languages have been extended with rules which enable the specification of more complex constraints [10]. In this respect, our contribution lies in the way in which our rules interface with ontologies: rather than allowing for a free interaction between them, which may lead to the introduction of inconsistencies or cause the ontology reasoning to become undecidable (as in [10]), we only allow information to flow *from* ontologies *to* policy rules. In other words, information from the knowledge base can be used to make informed access or credential release decisions, but policy rules cannot be used to derive new knowledge to be integrated into the knowledge base, as this may make the knowledge base inconsistent due to the introduction of contradicting information.

Furthermore, another contribution is represented by the design and development of a novel algorithm for credential retrieval, called GEM [23]. Contrarily to many of the existing algorithms (e.g., [5, 13]), GEM evaluates requests for credentials in a completely distributed way without disclosing the credential rules of parties, thereby preserving their confidentiality. For example, in the scenario in Sect. 12.2.1, when requesting the certificate of "search and rescue vessel" for NL-Lifeboat to the Dutch navy, the operation control center cannot infer anything about the certification procedure (i.e., the policy) of the Dutch navy. Similarly, the Dutch navy and the Dutch coastguard are not able to learn each other's policies. As discussed in Sect. 12.2.2, protecting the confidentiality of security policies is very important, since their disclosure may leak valuable information [8, 19, 20]. Finally, another advantage of GEM is its ability to identify mutually dependent policy rules, preventing loops in the credential retrieval process.

12.3.1.2 Ontology-Based Services

Parties in a system of systems refer to ontologies to assign semantics to the terms used in their policies. More precisely, ontology concepts (or instances) are used to denote the attributes certified by a party's credentials (e.g., "search and rescue vessel"). In addition, ontologies are used to define the data and organizational structures of each party. This, combined with the use of the completely automated semantic alignment technique in [22], which is based on the notion of similarity between ontology concepts, allows parties to use the vocabulary and structures they consider most appropriate within their system (thus accommodating parties' heterogeneity), while preserving mutual understanding with the rest of the coalition.

A major disadvantage of the existing semantic alignment techniques (e.g., [7, 9]) is that they require complete knowledge of the ontologies to be aligned in order to effectively map concepts from different ontologies. When collaborations involve parties that do not know each other beforehand or want to keep part of their knowledge base confidential, however, this condition cannot be guaranteed. To overcome this problem, we adopt the alignment technique introduced in [22] that maps concepts from different ontologies by considering concepts' similarity "estimates" (since they are computed based on partial knowledge) issued by different parties in the system of systems. These estimates are then combined into a single similarity

value by weighing them based on the "reliability" of their issuer. In this way, parties can enable interoperability with parties using different vocabularies and increase the flexibility of their policies by accepting credentials about possibly unknown attributes, provided that they are similar to a known attribute for at least a certain degree. For example, in the scenario in Sect. 12.2.1, the Dutch navy can use the estimates computed by the EU, the operation control center, the Dutch coastguard and itself to assess the similarity between the concepts "search and rescue vessel" and "search and rescue lifeboat". According to the resulting combined similarity estimate, the Dutch navy can then decide whether the two terms are similar enough to certify NL-Lifeboat as a "search and rescue vessel" (policy NL_1 in Table 12.1). The technique in [22] abstracts from the way in which similarity estimates are computed. Several existing algorithms (e.g., [15, 16]) can be employed for this purpose, and each party in the system of systems can adopt a different algorithm without affecting its interoperability with the other parties.

12.3.1.3 Service Oriented Architecture

Service oriented architecture is a paradigm for organizing and utilizing software solutions that promotes reuse and interoperability [18]. A system based on service oriented architecture implements functionality as a suite of interoperable services that can be used within multiple systems from different domains. The characteristic features of service oriented architecture are:

- Service reusability: the logic and functionality of a system is divided into services with the intention of promoting reuse.
- Service autonomy: each service is independent and maintains control over the logic it encapsulates.
- Service loose coupling: the relationship and dependencies among services are minimized.
- Service composability: services can be easily combined and integrated into a larger, complex system.

Service oriented architecture is commonly implemented using web services. Accordingly, our security framework is implemented as a web service which can be easily plugged into existing systems and acts as a proxy server intercepting all the outgoing and incoming messages of a system. Furthermore, each component of the security framework introduced in the next subsection is implemented as a service that interfaces with the functional components and the data stored within a system. This modular approach also facilitates the extension of the framework with additional security services that may become relevant during the lifetime of a system of systems (e.g., a reputation system for service selection or a key performance indicator service) [4].

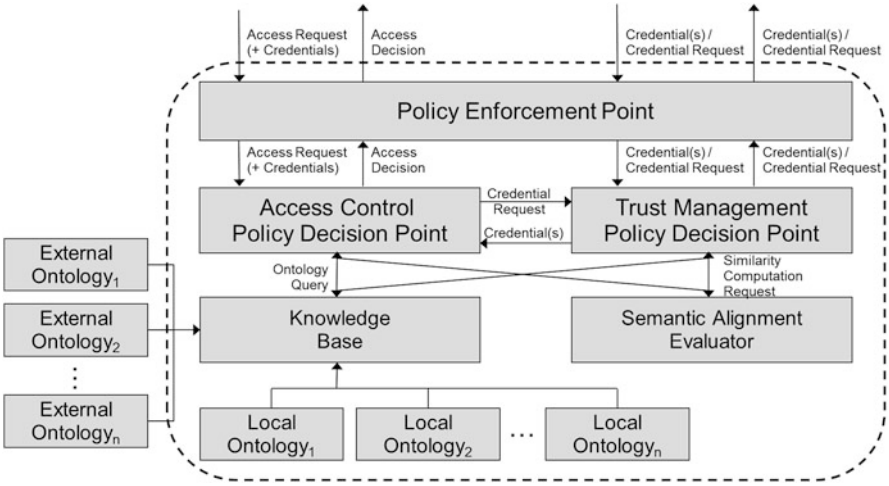


Fig. 12.2 POLIPO framework architecture

12.3.2 Framework Architecture

Each party in a system of systems can employ an instance of the POLIPO framework to protect the local resources. The POLIPO framework intercepts every access request to a local resource and evaluates it against the local security policy. If the request is authorized by the policy, the resource is returned to the requester; otherwise, the access is denied.

An overview of the framework's architecture is shown in Fig. 12.2, where the dashed line separates the local components from the external ones. POLIPO consists of a set of core components (i.e., *policy enforcement point*, *access control* and *trust management policy decision point*), inspired by the XACML architecture [17], and a number of complementary specialized services used to assist the policy evaluation process (i.e., *knowledge base* and *semantic alignment evaluator*). The combination of the core components with the specialized services ensures confidentiality and integrity of information, while preserving autonomy and interoperability among the parties in a system of systems. In the next paragraphs we discuss these five components and services in detail.

Policy Enforcement Point. The policy enforcement point is the interface of a party with the external world, and has three main tasks: (1) intercepting incoming requests for local resources, (2) contacting the appropriate policy decision point to evaluate those requests, and (3) enforcing the decision of the policy decision point. We consider two types of requests: *access requests* and *credential requests*. Access requests are requests for data controlled by the local party (e.g., message 2 in Fig. 12.1), while credential requests are requests for credentials issued by the local party (e.g., message 4b in Fig. 12.1).

Upon receiving a request, the policy enforcement point forwards it to the appropriate policy decision point. In particular, access requests are processed by the access control policy decision point, while credential requests by the trust management policy decision point. Finally, the policy enforcement point enforces the decision of the policy decision point. If the decision is positive, i.e., if access to the requested data is authorized (in case of an access request) or the requested credential is locally available (in case of a credential request), the policy enforcement point returns the requested resource; otherwise, a "deny" response is sent to the requester.

Access Control Policy Decision Point. The access control policy decision point is responsible for the evaluation of access requests. When it receives an access request, the access control policy decision point checks whether the applicable authorization rules depend on some credentials: if this is the case, they are requested to the trust management policy decision point, which takes over the responsibility of retrieving them. Then, depending on whether all the necessary credentials have been successfully retrieved and the other conditions in the authorization rules (e.g., context conditions) are satisfied, the access control policy decision point determines whether the access request should be authorized or denied. Context information and domain knowledge are retrieved by invoking the knowledge base service, while the alignment between ontology concepts is performed by the semantic alignment evaluator.

Trust Management Policy Decision Point. The trust management policy decision point is responsible for the evaluation of credential requests. The credential retrieval algorithm within the trust management policy decision point defines the procedure to compute the answers to a credential request; in our framework we employ GEM [23] as credential retrieval algorithm. The evaluation of a credential request may depend on credentials which are not locally available and, consequently, need to be retrieved from some other party (e.g., a credential issued by the Dutch coastguard in policy NL_1 in Table 12.1). In this case, the trust management policy decision point sends the request for the missing credential to the policy enforcement point, which forwards it to the appropriate party, and feeds the response back to the trust management policy decision point. As for the access control policy decision point, ontology queries in credential rules are resolved by the knowledge base service, while ontology mappings are requested to the semantic alignment evaluator.

Knowledge Base. The knowledge base service is used by a party to retrieve the context and domain information that is relevant for an access or credential release decision. For example, in rule OCC_2, the need for assistance of a ship is determined by the operation control center based on an "outlier factor" (see Chap. 8) associated to the ship, which is available in the knowledge base. The knowledge base consists of a set of local ontologies that define the concepts and relationship employed in the party's policy, the structure of the information it controls (i.e., metadata annotating the local resources), and all the domain and context information (e.g., the current location of a vessel) collected and derived within the system. A party can enlarge

its knowledge base by importing external ontologies defined and published by other parties or institutions. Within the POSEIDON project, POLIPO relies mainly on the simple event model ontology (Chap. 10).

Semantic Alignment Evaluator. This service computes the similarity between ontology concepts, relationships, and instances. The result of the computation is a similarity value which can be used in a constraint in authorization and credential rules. In the evaluation of rule NL_1 in Table 12.1, for instance, the similarity value between "search and rescue vessel" and "search and rescue lifeboat" is used by the Dutch navy to determine whether vessel NL-Lifeboat should be certified as "search and rescue vessel". In Fig. 12.2, the semantic alignment evaluator is depicted as a service directly controlled by the local party; however, parties in a coalition may rely on (possibly shared) external semantic alignment services.

12.4 Prototype Implementation

We have deployed a prototype implementation of POLIPO [24, 25] into a system of systems developed within the POSEIDON project. The system of systems consists of five parties: the operation control center, the Danish navy, the Dutch navy, the Dutch coastguard, and the Dutch coastguard's lifeboat NL-Lifeboat. Parties in the system of systems use Google Earth to visualize and analyze the maritime traffic in their operational area. Communication among parties (i.e., access and credential requests) is via HTTP. In this setting, the policy enforcement point acts as a web proxy that intercepts all the incoming HTTP requests, loads the requested data, and returns an HTTP response based on the policy of the party controlling the data.

In the prototype, the policy enforcement point has been divided into two modules: an interface module and a services module that consists of a set of responders, one for each service offered by the local party. The interface module waits for incoming requests and passes them to the appropriate responder which takes care of processing the request and generating a response. The policy enforcement point of each system in the POSEIDON system of systems has two responders: one to process access requests from the visualization service (Google Earth), and one to process credential requests. The division of the policy enforcement point into two modules enhances the flexibility of the POLIPO framework, as it allows new services offered within the system of systems to be secured by simply adding the relative responders to the policy enforcement point component.

We now show an application of POLIPO to the scenario introduced in Sect. 12.2.1. Figures 12.3 and 12.4 show the output of the visualization service and the details about Blue Star's cargo returned to the operator of NL-Lifeboat respectively before and during the emergency situation declared by the operation control center, based on the policies in Table 12.1. As mentioned in Sect. 12.2.1, all the names and details about ships introduced in this chapter are purely fictitious.

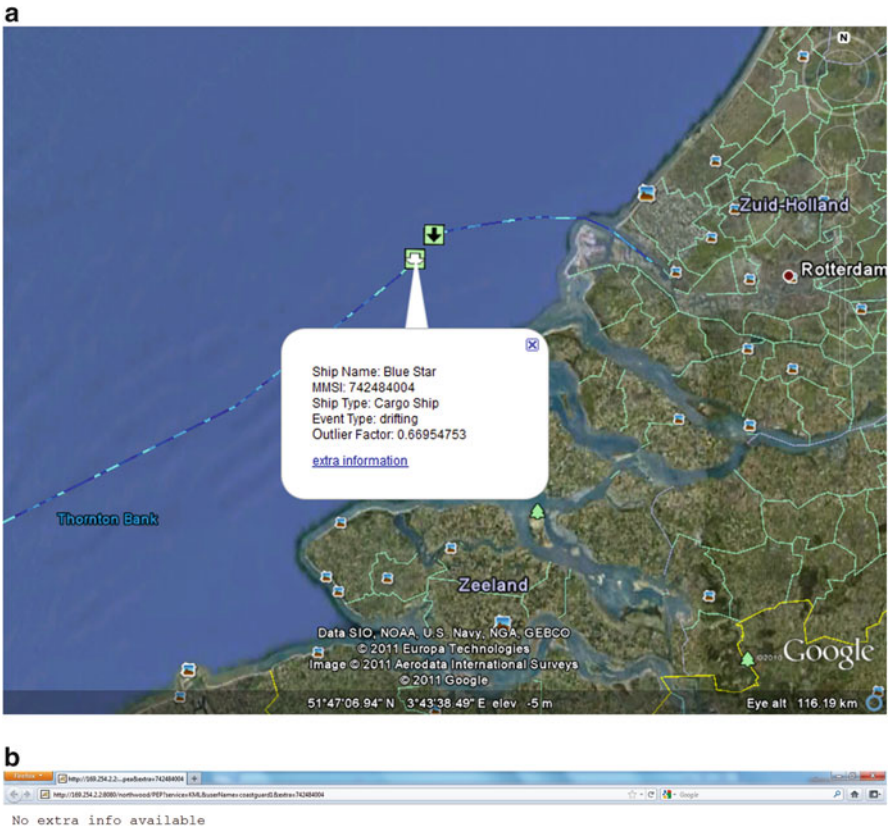


Fig. 12.3 Data view and extra information displayed to the operator of NL-Lifeboat before the emergency. (**a**) Data view for the operator of NL-Lifeboat. (**b**) Information about Blue Star's cargo displayed to the operator of NL-Lifeboat

In the visualization, the color of a segment reflects the outlier factor computed for that segment (see Chap. 8). The color scale goes from blue to red as the outlier factor increases. The current position of each ship is represented by an icon, whose type corresponds to the type of the vessel it represents: more precisely, icons depicting an arrow pointing downwards represent vessels which are (or become) part of the EU NAVFOR; all the other vessel types are represented by an icon depicting a white boat. By clicking on an icon or a segment the operator can see the name, maritime mobile service identity (MMSI), and type of a ship, as well as the event type (e.g., the ship is slowing down, has stopped, etc.) and the outlier factor associated to the event. For instance, in Figs. 12.3a and 12.4a, information about the ship Blue Star is displayed. The information box shows that Blue Star is a cargo ship which is currently drifting, and the outlier factor is approximately 0.67 and 0.84 respectively.

When the operator of NL-Lifeboat requests details about Blue Star's cargo to the operation control center, the security framework of the operation control

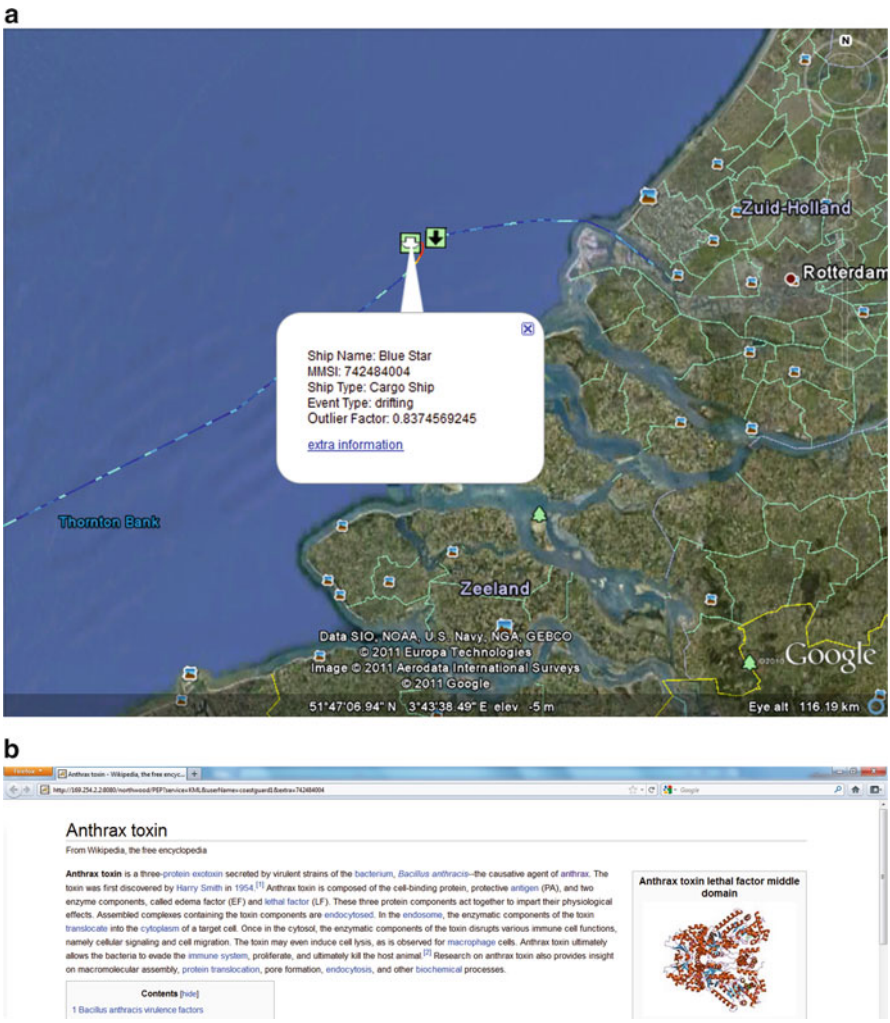


Fig. 12.4 Data view and extra information displayed to the operator of NL-Lifeboat during the emergency. (**a**) Data view for the operator of NL-Lifeboat. (**b**) Information about Blue Star's cargo displayed to the operator of NL-Lifeboat

center verifies whether there are rules in the local policy which might grant access to the requested information, and if so, it attempts to collect the credentials of NL-Lifeboat required to authorize the access. For example, the first time that NL-Lifeboat requests the cargo information, the operation control center does not return any details because the situation of Blue Star is not considered critical (Fig. 12.3b, corresponding to message 3 in Fig. 12.1), and therefore rule OCC_2 in Table 12.1 cannot be applied. On the contrary, when the conditions become more critical (as can be evinced by the high outlier factor associated to Blue Star and the event type

– drifting – in Fig. 12.4a), the intelligence gathered by the EU NAVFOR about Blue Star's cargo is provided to the operator of NL-Lifeboat (message 4f in Fig. 12.1). In this case, since the cargo contains Anthrax, the information returned to NL-Lifeboat includes details about the Anthrax toxin and measures to prevent the infection (Fig. 12.4b). The authorization of NL-Lifeboat to access the intelligence collected by the EU NAVFOR follows from the verification that NL-Lifeboat is a "search and rescue vessel" certified by the Dutch navy. In turn, the certification of NL-Lifeboat as a "search and rescue vessel" from the Dutch navy results from the alignment of the vocabularies of the Dutch navy and the Dutch coastguard (see Table 12.1 for the vocabulary employed in the two parties' policies).

12.5 Discussion and Conclusion

The security challenges in systems of systems are different from those affecting centralized systems. In a dynamic, inter-organizational coalition of systems, parties might not know each other beforehand, might employ different data and organizational models and "speak" different languages; nevertheless, they must be able to collaborate for the success of the coalition. We have identified four main requirements that a security framework for systems of systems must satisfy: (1) protection of the *confidentiality* and *integrity* of data and security policies; (2) *autonomy* of parties in the choice of data and organizational model and vocabulary used to specify policies and describe the local resources; and (3) *interoperability* among parties. In addition, (4) the security framework must be *easy to integrate* into existing systems.

Several security frameworks for systems of systems have been proposed in the literature. These frameworks can be divided into two categories: semantic frameworks [11, 26] and trust management frameworks [5, 12, 14]. Semantic frameworks rely on ontologies for the specification of security policies and the definition of domain knowledge and context information. This enables interoperability among parties at the cost of limiting the expressive power of the policy language, which does not allow the specification of several types of security constraints (e.g., mutually exclusive roles). On the other hand, trust management frameworks rely on an attribute-based approach to access control where access decisions are based on digital certificates, called credentials. Trust management frameworks employ expressive policy specification languages to ensure data confidentiality and integrity; however, they either assume all parties in a system of systems to use the same vocabulary [12, 14], or do not provide a mechanism to align different vocabularies [5]. Thus, none of the existing frameworks satisfies all the security requirements of systems of systems.

In this chapter we have introduced POLIPO, a security framework for systems of systems satisfying all the aforementioned requirements. Confidentiality and integrity of information are protected by complementing context-aware access control with trust management; a distributed policy evaluation algorithm guarantees

that policies' confidentiality is not violated when a credential request is evaluated. Security policies are specified by means of an ontology-based specification language [21]. The use of ontologies allows parties to provide semantics to the terms employed in their policies and to describe domain and context information. This, combined with a semantic alignment technique [22], gives parties the autonomy to employ different organizational models and vocabularies while preserving interoperability among the parties in the system of systems.

The applicability of POLIPO has been demonstrated by a prototype implementation for a scenario in the maritime safety and security domain, where communication among the parties in the system of systems is via HTTP. In this setting, POLIPO acts as a web proxy which intercepts all the incoming access and credential requests directed to a party, and returns a response based on the security policy of the party. This facilitates the deployment of the framework into existing systems. In addition, all the framework components have been implemented following the service oriented architecture paradigm to allow for an easy integration of additional components to support the evaluation of policies and provide additional functionalities.

Even though POLIPO has been mainly tested in systems of systems in the maritime safety and security domain, its characteristics make it suitable for many other domains. For example, we have deployed POLIPO also in systems of systems in the e-health and the employability domain [4]. Furthermore, its integration with ontology-based services allows for an easy deployment of POLIPO into systems of systems on the semantic web. More generally, POLIPO represents a valid security solution for all the domains characterized by the need for collaborations among parties from different security domains, who possibly do not know each other beforehand and employ different vocabularies and different data and organizational structures.

Acknowledgements This research has been carried out as a part of the POSEIDON project at Thales under the responsibilities of the Embedded Systems Institute (ESI). This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

References

1. Becker MY, Sewell P (2004) Cassandra: distributed access control policies with tunable expressiveness. In: Proceedings of the 5th IEEE international workshop on policies for distributed systems and networks, POLICY'04, Washington, DC, USA. IEEE Computer Society, Los Alamitos, pp 159–168
2. Bhatti R, Bertino E, Ghafoor A (2005) A trust-based context-aware access control model for web-services. Distrib Parallel Database 18(1):83–105
3. Blaze M, Feigenbaum J, Lacy J (1996) Decentralized trust management. In: Proceedings of the 1996 IEEE symposium on security and privacy, SP'96. IEEE Computer Society, Los Alamitos, pp 164–173
4. Böhm K, Etalle S, den Hartog J, Hütter C, Trabelsi S, Trivellato D, Zannone N (2010) Flexible architecture for privacy-aware trust management. J Theor Appl Electron Commer Res 5(2):77–96

5. Czenko M, Etalle S (2007) Core TuLiP logic programming for trust management. In: Proceedings of the 23rd international conference on logic programming, ICLP'07, Porto, Portugal. Lecture notes in computer science, vol. 4670. Springer, Berlin, pp 380–394
6. Dersingh A, Liscano R, Jost A (2008) Context-aware access control using semantic policies. Ubiquitous Comput Commun J (UBICC) 3:19–32. Special issue on autonomic Computing Systems and Applications
7. Doan A, Madhavan J, Dhamankar R, Domingos P, Halevy A (2003) Learning to match ontologies on the semantic web. VLDB J 12(4):303–319
8. Frikken K, Atallah M, Li J (2006) Attribute-based access control with hidden policies and hidden credentials. IEEE Trans Comput 55:1259–1270
9. Heeps S, Sventek J, Dulay N, Filho AS, Lupu E, Sloman M, Strowes S (2007) Dynamic ontology mapping for interacting autonomous systems. In: Proceedings of the 2nd international workshop on self-organizing systems, IWSOS'07, The Lake District, UK. Lecture notes in computer science, vol 4725. Springer, Berlin, pp 255–263
10. Horrocks I, Patel-Schneider PF, Bechhofer S, Tsarkov D (2005) OWL rules: a proposal and prototype implementation. J Web Semant 3(1):23–40
11. Kagal L, Paolucci M, Srinivasan N, Denker G, Finin T, Sycara K (2004) Authorization and privacy for semantic web services. IEEE Intell Syst 19(4):50–56
12. Li N, Mitchell JC, Winsborough WH (2002) Design of a role-based trust-management framework. In: Proceedings of the 2002 IEEE symposium on security and privacy, SP'02, Washington, DC, USA. IEEE Computer Society, Los Alamitos, pp 114–130
13. Li N, Winsborough WH, Mitchell JC (2003) Distributed credential chain discovery in trust management. J Comput Secur 11(1):35–86
14. Nejdl W, Olmedilla D, Winslett M (2004) PeerTrust: automated trust negotiation for peers on the semantic web. In: Proceedings of the 2004 VLDB workshop on secure data management, SDM'04. Lecture notes in computer science, vol 3178. Springer, Berlin, pp 118–132
15. Ngan LD, Hang TM, Goh AES (2006) Semantic similarity between concepts from different OWL ontologies. In: Proceedings of the 5th IEEE international conference on industrial informatics, INDIN'06. IEEE Computer Society, Los Alamitos, pp 618–623
16. Nguyen HA, Al-Mubaid H (2006) A Combination-based semantic similarity measure using multiple information sources. In: Proceedings of the 2006 IEEE international conference on information reuse and integration, IRI'06. IEEE Systems, Man, and Cybernetics Society, Piscataway, pp 617–621
17. OASIS (2005) eXtensible Access Control Markup Language (XACML) Version 2.0. Technical report, OASIS standard. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf
18. OASIS (2006) Reference model for service oriented architecture 1.0. OASIS standard. http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf
19. Seamons KE, Winslett M, Yu T (2001) Limiting the disclosure of access control policies during automated trust negotiation. In: Proceedings of the network and distributed system security symposium, NDSS'01, San Diego, CA, USA. The Internet Society, Reston
20. Stine K, Kissel R, Barker WC, Lee A, Fahlsing J (2008) Guide for mapping types of information and information systems to security categories. Special publication SP 800–60 Rev. 1. National Institute of Standards and Technology (NIST), Gaithersburg
21. Trivellato D, Spiessens F, Zannone N, Etalle S (2009) POLIPO: Policies & OntoLogies for Interoperability, Portability, and Autonomy. In: 10th IEEE international symposium on policies for distributed systems and networks (POLICY'09). IEEE Computer Society, Los Alamitos, pp 110–113
22. Trivellato D, Spiessens F, Zannone N, Etalle S (2009) Reputation-based ontology alignment for autonomy and interoperability in distributed access control. In: IEEE international conference on computational science and engineering (CSE'09), vol 3. IEEE Computer Society, Los Alamitos, pp 252–258
23. Trivellato D, Zannone N, Etalle S (2010) GEM: a distributed goal evaluation algorithm for trust management. Computer science report CS 10–15, Eindhoven University of Technology. http://alexandria.tue.nl/repository/books/695281.pdf

24. Trivellato D, Zannone N, Etalle S (2011) A security framework for systems of systems. In: 12th IEEE international conference on policies for distributed systems and networks (POLICY'11). IEEE Computer Society, Los Alamitos, pp 182–183
25. Trivellato D, Zannone N, Etalle S (2011) Poster: protecting information in systems of systems. In: Chen Y, Danezis G, Shmatikov V (eds) 18th ACM conference on computer and communications security (CCS'11). ACM, New York, 2011, pp 865–868
26. Uszok A, Bradshaw JM, Johnson M, Jeffers R, Tate A, Dalton J, Aitken S (2004) KAoS policy management for semantic web services. IEEE Intell Syst 19(4):32–41 (2004)

Chapter 13
Assessing Trust for Determining the Reliability of Information

Davide Ceolin, Willem Robert van Hage, Guus Schreiber, and Wan Fokkink

13.1 Introduction

In the naval domain, particular messages, called "Automated Identification System" (AIS) messages are periodically exchanged between ships and captured by particular receivers that allow ship and land based naval authorities to avoid collisions, to locate and to identify ships. These messages contain important information, about the identity of the ship (identification code, name, flag, ship dimension, etc.), about its location (latitude, longitude, timestamp of the message, that is, temporal identification of the moment when the message is sent) and about kinematic data (for instance, speed and heading).

The importance of these messages is therefore evident from the fact that they allow to keep track of the position of the ships, together with their identity. However, these messages can, in principle, be intentionally or unintentionally manipulated by the senders. For instance, there exist episodes of ships willing to impersonate the identity of others to evade controls.[1]

Trust plays a crucial role when dealing with these messages, because the information that they provide is not always certain, but a naval operator that reads them, would like to know whether he can trust them. We will see in this chapter how to calculate a trust level for messages that are received, and how to live with

[1]For instance, in 2010 the fleet of an Iranian company tried to disguise its identity. See: http://www.nytimes.com/2010/06/08/world/middleeast/08sanctions.html?pagewanted=all

D. Ceolin (✉) • W.R. van Hage • G. Schreiber
Web & Media Group, VU University Amsterdam, Amsterdam, The Netherlands
e-mail: d.ceolin@vu.nl; w.r.van.hage@vu.nl; guus.schreiber@vu.nl

W. Fokkink
Theoretical Computer Science Group, VU University Amsterdam, Amsterdam, The Netherlands
e-mail: w.j.fokkink@vu.nl

P. van de Laar et al. (eds.), *Situation Awareness with Systems of Systems*,
DOI 10.1007/978-1-4614-6230-9_13,

the fact that trust can only be estimated, i.e., it cannot be computed precisely or with absolute certainty. Trust is a strategy chosen to deal with this lack of information. If we would have known for sure that a message is correct, we would have simply used it, without the need to estimate its trust level.

The concept of trust in computer and information science directly reflects the idea of reliance on third parties typical of the view of trust in sociology or psychology [1]. Chapter 12 describes techniques that agents (ships or fleets) can use to trust each other in order to safely exchange sensitive information. Instead, we are interested in trusting messages that these ships send, i.e. in determining the reliability of the information that they contain. We will assess the level of trust in these messages by means of decomposition: messages are composed of different information about the ship. By assessing the trust level of these pieces of information and then aggregating all the assessments related to a given message, we are able to provide an assessment about the whole message. Clearly, the different parts composing the messages are not always independent of each other. This problem will be addressed in Sect. 13.5. Because of the tight relation between messages and sender (the message is sent by the ship that it refers to and it is used to identify it), trusting or distrusting the message can be considered equivalent to trusting the so-called agent, that is, the ship.

This chapter will describe how it is possible to estimate appropriate trust in the information that the message exposes. We will base our evaluations on two factors: the reputation of the sender (and, more generally, the "provenance" of the message, that is, who produced it and how), and the co-occurrence of multiple observations supporting or contrasting the information provided by the message itself. Roughly speaking, this means that we will trust messages when they are sent by well reputed agents and we will trust information that is confirmed by many agents.

In the following sections we will show how to deal with all the information contained in the messages, reason about it and take the decision to trust it, with the goal to reduce the probability to take a wrong decision. In Sect. 13.2 we introduce evidential reasoning, and in Sect. 13.3, we describe first and second order probabilities. Evidential reasoning and first and second order probabilities are the statistical foundations of our approach. In Sects. 13.4 and 13.5 it is explained how provenance information (that is, information about how a given piece of information has been produced) plays a key role in trust assessments. Some results and achievements are discussed in Sect. 13.6. Finally, Sect. 13.7 describes future work.

13.2 Evidential Reasoning

Often, we are not able to directly determine whether a message is correct by simply reading it. Suppose, for instance, that we receive a message from $Ship_{123}$ stating that the name of the ship is "Beauty". Shall we trust it? It may be lying or not, and we do not know it yet. What we can do in this situation is to look for other messages sent by $Ship_{123}$ in order to see whether it coherently says that its name is "Beauty" and look for other sources (e.g., official registries) confirming or disagreeing with this fact.

Of course, a trust assessment based only on evidence coming from one single source of information (the ship itself) is possibly biased and partial. Moreover, in case the ship wants to hide its real identity, it is likely to send consistently messages containing wrong information, and any trust level based only on them will be misleading. However, this is only a starting point of our reasoning. We will shortly see how to "weigh" these evaluations taking into account the reliability of the source that produces them and how to incorporate information coming from other sources than the ship itself.

We can refer to the evidence provided by the messages as "direct" evidence, because it consists of observations that directly focus on our subject of interest, that is, the name of the ship. On the other hand, if, for instance, we know that the reputation of the ship is positive, that is, we know that $Ship_{123}$ has always been reliable in the past, then, although this does not say anything directly about the correctness of its name, still will make us incline to believe in what the ship says. Therefore, this kind of evidence can be regarded as "indirect", since it helps us taking a decision although it is not directly related to what we are evaluating. Indirect evidence includes the reputation of the ship but, for instance, might also include evidence about the possible theft of the ship identity. Clearly, this latter event would change our opinion about the information provided by the ship.

Evidential reasoning allows to deal with these two kinds of evidence. We will see how a probabilistic logic can allow to represent the previously introduced facts ($Ship_{123}$ is named "Beauty") as logical statements and how to deal with the evidence that permits to determine whether these statements are true or false. In this particular case, we are not interested in determining the name of $Ship_{123}$, but we are assessing the trust level for "Beauty" being it. Therefore, instead of having a possibly infinite range of values, our range is composed of only two values, true and false, which are the truth values that the assertion we are assessing can assume.

13.2.1 Subjective Logic

Subjective Logic [8] is a kind of probabilistic logic, that is, a formalism that combines the traditional logic with probability theory to express uncertainty about the truth level of statements. The "Subjective" name is due to the fact that the assertions treated by this logic are evaluated subjectively, according to the evidence that one can gather on a statement. For instance, if we have at our disposal 100 messages sent by $Ship_{123}$ stating that its name is "Beauty", this will give us a subjective opinion different from that of another person relying on only 10 messages. We have a limited view on the world and we base our judgements on all the evidence that we have been able to gather until a given moment.

Opinions are the basic element of Subjective Logic, because they are the means to link logical statements to probabilities and to contextualize them. For instance, if our source (or "subject") is $message_1$ and we are determining whether the name of the ship is "Beauty" (this statement constitutes our "object"), then we can represent

it as an opinion (represented by the symbol $\omega_{object}^{subject}$) in one of these two equivalent means:

$$\omega^{message_1}_{\textit{the name of Ship}_{123}\textit{ is ``Beauty''}}\left(\frac{1}{3},0,\frac{2}{3}\right)=\omega^{message_1}_{\textit{the name of Ship}_{123}\textit{ is ``Beauty''}}\left(\frac{1}{3},0,\frac{2}{3},\frac{1}{2}\right)$$

Any valid opinion has four components ranging between zero and one (belief, disbelief, uncertainty and a priori). If the a priori value is omitted, it is implied that it assumes a default value. In this case, as we will see later in detail, the default value is $\frac{1}{2}$. The sum of the first three components has to be equal to one. These components implicitly represent probabilities:

Belief: the value that represents how much we believe that this statement can be true. It is computed as follows:

$$b^{message_1}_{\textit{the name of Ship}_{123}\textit{ is ``Beauty''}}=\frac{\#positive\_evidence}{\#total\_evidence+\#range\_values}=\frac{1}{3}=0.333$$

In this case, we have only one evidence because we are considering only one message, therefore $\#positive\_evidence = \#total\_evidence = 1$. Note also that $\#range\_values$ is the number of values that the statement can assume. In this case $\#range\_values = 2$, since the statement can be either true or false.

Disbelief: the value that summarizes all the observations that falsify the statement. It is computed as follows:

$$d^{message_1}_{\textit{the name of Ship}_{123}\textit{ is ``Beauty''}}=\frac{\#negative\_evidence}{\#total\_evidence+\#range\_values}=\frac{0}{3}=0$$

Since the opinion is based on the only message that we are considering at the moment, there is no observation contrasting with what the message reports.

Uncertainty: the quantification of the uncertainty about the correctness (or truthfulness) of a given statement.

$$u^{message_1}_{\textit{the name of Ship}_{123}\textit{ is ``Beauty''}}=\frac{\#range\_values}{\#total\_evidence+\#range\_values}=\frac{2}{3}=0.667$$

A priori: it represents the prior bias that "subject" has against one of the possible outcomes before collecting any observation. The range of this value is [0..1]. An a priori value equal to zero or one means higher bias towards disbelief and belief, respectively. A priori value equal to 0.5 means no bias: in this case, for convention the value can be omitted from the opinion representation. Vice-versa, an opinion not reporting that value, implicitly indicates that its value is $\frac{1}{\#range\_values}$ so, in this case, 0.5.

$$a^{message_1}_{\textit{the name of Ship}_{123}\textit{ is ``Beauty''}}=\frac{1}{\#range\_values}=\frac{1}{2}=0.5$$

These formulas are useful in the maritime domain, where different "subjects" may have different points of view with regard to the same fact. By "subjects", in this domain, we may mean, for instance, a message from a ship or from a radar station. Different levels of granularity are allowed. These formulas allow to correctly represent a particular point of view and the operators that we shall introduce later in this section, allow to "merge" and "weigh" opinions, providing opinions at different levels of granularity, based on actual observations.

It is important to note how the uncertainty value is both inversely correlated to the amount of evidence (the more evidence we have, the less uncertain our evaluation will be) and correlated to the number of possible different values (the bigger the number of possible values that our object of interest can have, the more uncertain our evaluation will be).

Of course, it could also be that if a certain ship is willing to make us believe that its name is "Beauty" (although this is not true), it will probably consistently send messages reporting that name. That is why we group our evidence in opinions, which record also the source of the evidence, and then combine these with the reputations of the sources. In other words, these opinions are weighted according to the opinion that we own about the source itself. We will see later in this section how this weighting is possible. Note also that the opinion representation focuses on a particular "object" (the name of $Ship_{123}$ is "Beauty") and that it resembles the point of view (based on the observations currently at disposal) that a certain "subject" ($message_1$) has.

13.2.2 Subjective Logic Operators for Combining Opinions

The opinions represent facts, both with respect to the point of view that they incorporate (they are based on the evidence observed by the "subject" of the opinion) and to their focus, since they represent belief, disbelief, uncertainty and a priori value about a specific "object", that is a specific fact. In several situations, like in the maritime domain, single opinions are not enough to answer questions that involve many atomic facts related to each other in disparate manners. The solution to this issue is to combine the opinions about these atomic facts by means of specific operators. The results of these operations will be "complex" opinions that reflect the relations incurring between the facts.

Suppose that many subjects expose their (possibly disagreeing) opinions about the same object. In order to be able to take a decision about the truth value of the common object of these opinions, we could merge all these opinions into one, unique opinion that summarizes them all. The subject of this final opinion is represented as a concatenation of the original subjects and the evidence on which the opinions are based are merged. This way, we mediate possible disagreement between the input opinions, and, at the same time, we incorporate all the contributions. This merge is done by means of a so-called "fusion operator" (represented by the symbol $\oplus$). The resulting opinion is equivalent to an opinion based on the

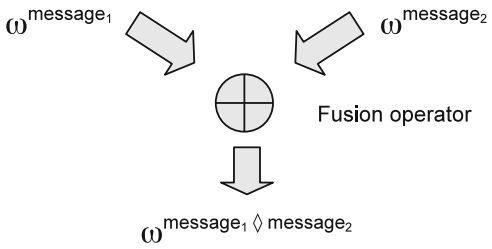


Fig. 13.1 Small network of opinions

(possibly weighted) sum of all the evidence on which the opinions are based. The fusion operator works as follows:

$$\omega_x^{message_1} \oplus \omega_x^{message_2} =$$

$$\omega_x^{message_1 \diamond message_2} = \begin{cases} b_x^{message_1 \diamond message_2} = \frac{u_x^{message_2} \times b_x^{message_1} + b_x^{message_2} \times u_x^{message_1}}{u_x^{message_1} + u_x^{message_2} - u_x^{message_1} \times u_x^{message_2}} \\ d_x^{message_1 \diamond message_2} = \frac{u_x^{message_2} \times d_x^{message_1} + d_x^{message_2} \times u_x^{message_1}}{u_x^{message_1} + u_x^{message_2} - u_x^{message_1} \times u_x^{message_2}} \\ u_x^{message_1 \diamond message_2} = \frac{u_x^{message_2} \times u_x^{message_1}}{u_x^{message_1} + u_x^{message_2} - u_x^{message_1} \times u_x^{message_2}} \\ a_x^{message_1 \diamond message_2} = \frac{2 \times a_x^{message_1} \times a_x^{message_2}}{a_x^{message_1} + a_x^{message_2}} \end{cases}$$

In this formula, the superscript names indicate the sources considered when building the opinion. The subscript x is a generic statement (like "$Ship_{123}$ is named Beauty") and b, d, u and a are respectively the belief, disbelief, uncertainty and a priori value described before. Basically, what this operator allows to do is to combine opinions about the same object, but coming from different sources and based on possibly different amount of evidence. The resulting opinion will have decreased uncertainty (because it will incorporate evidence coming from both inputs) and increased belief and/or disbelief, for the same reason. Belief and disbelief of the resulting opinion are computed by determining the average of the input belief and disbelief, weighted on the inverse of the uncertainty, that is, implicitly weighted on the evidence supporting them, since the uncertainty is inversely proportional to the amount of evidence considered. Two opinions combined into an output opinion can be seen as a simple, almost trivial, network and represent the first attempt to organize the information that we have at disposal, as can be seen from Fig. 13.1.

Note that the merged opinion is exactly equivalent to an opinion based on the amount of evidence used to build the input opinions. This fact allows to compute opinions in an incremental and possibly distributed way. If we get opinions from different sources, we do not need to look for the original evidence that led to these opinions in order to obtain a global opinion that takes all this evidence into consideration. It is possible to simply combine the opinions that we see and the

resulting combined opinion will be equivalent to an opinion actually based on that evidence. The same holds in case evidence is not all available at the same time, but rather, is collected progressively. More information about these operators is available in [8].

Suppose, further, that one of the messages is retrieved through a receiver that we know is not always reliable. This means that at least the uncertainty of the opinion computed on the basis of such a message should be increased (because we do not know if the receiver was working properly or not, when it recorded the message). This is obtained by another operator, called "discount" operator (represented by the symbol $\otimes$) that weighs the opinion on the message itself according to the opinion on the receiver, that is, on the reputation of the receiver. This will allow us to "smooth" strong opinions coming from subjects (that is, sources) of which the reputation is not surely positive, while allowing us to incorporate opinions about facts of which we do not have direct observations, but that are "told us" by third parties (in this case, the receiver). Here is an example. If our opinion about $message_1$ was

$$\omega^{we}_{message_1}(0.4, 0.4, 0.2)$$

and the opinion given by $message_1$ is the one we have seen before,

$$\omega^{message_1}_{the\ name\ of\ Ship_{123}\ is\ \text{“}Beauty\text{”}}(0.333, 0, 0.667)$$

we can weigh this opinion on the basis of $message_1$'s reputation by applying the discount operator:

$$\omega^{we}_{message_1} \otimes \omega^{message_1}_{the\ name\ of\ Ship_{123}\ is\ \text{“}Beauty\text{”}} = \omega^{we:message_1}_{the\ name\ of\ Ship_{123}\ is\ \text{“}Beauty\text{”}} =$$

$$\begin{cases} b^{we:message_1}_{the\ name\ of\ Ship_{123}\ is\ \text{“}Beauty\text{”}} = b^{we}_{message_1} b^{message_1}_{the\ name\ of\ Ship_{123}\ is\ \text{“}Beauty\text{”}} \\ d^{we:message_1}_{the\ name\ of\ Ship_{123}\ is\ \text{“}Beauty\text{”}} = b^{we}_{message_1} d^{message_1}_{the\ name\ of\ Ship_{123}\ is\ \text{“}Beauty\text{”}} \\ u^{we:message_1}_{the\ name\ of\ Ship_{123}\ is\ \text{“}Beauty\text{”}} = d^{we}_{message_1} + u^{we}_{message_1} + b^{we}_{message_1} u* \\ a^{we:message_1}_{the\ name\ of\ Ship_{123}\ is\ \text{“}Beauty\text{”}} = a^{message_1}_{the\ name\ of\ Ship_{123}\ is\ \text{“}Beauty\text{”}} \end{cases}$$

where $u* = u^{message_1}_{the\ name\ of\ Ship_{123}\ is\ \text{“}Beauty\text{”}}$, and the result will be:

$$\omega^{we}_{message_1}(0.4, 0.4, 0.2) \otimes \omega^{message_1}_{the\ name\ of\ Ship_{123}\ is\ \text{“}Beauty\text{”}}(0.333, 0, 0.667) =$$

$$\omega^{we:message_1}_{the\ name\ of\ Ship\_123\ is\ \text{“}Beauty\text{”}}(0.133, 0, 0.867)$$

In case the source (which, in this case, is the message), is known to be potentially malicious, then we can use another type of discount operator that, instead of rising the uncertainty, rises the disbelief. Other operators are available, in order to allow

different logical operations to be applied to the statements of our interest and to have the corresponding beliefs, disbeliefs and uncertainties properly updated. The choice of the correct operator to be applied on the opinions at our disposal depends on the relations among objects and subjects and is usually related to domain knowledge. For instance, a given $object_1$, in order to be true may need two other objects, namely $object_2$ and $object_3$, to be true. If we own an opinion about $object_2$ and one about $object_3$, we can use the AND operator to infer an opinion about $object_1$. The Subjective Logic version of it (represented by the symbol $\cdot$) works as follows:

$$\omega_x^{subject} \cdot \omega_y^{subject} = \omega_{x \wedge y}^{subject} = \begin{cases} b_{x \wedge y}^{subject} = b_x b_y + \frac{(1-a_x)a_y b_x u_y + a_x(1-a_y)u_x b_y}{1-a_x a_y} \\ d_{x \wedge y}^{subject} = d_x + d_y - d_x d_y \\ u_{x \wedge y}^{subject} = u_x u_y + \frac{(1-a_y)b_x u_y + (1-a_x)u_x b_y}{1-a_x a_y} \\ a_{x \wedge y}^{subject} = a_x a_y \end{cases}$$

where x and y are two different statements and $x \wedge y$ is the opinion that "subject" has about both x and y being true. These "subjective" versions of the boolean operators allow to logically combine all the statements at our disposal according to our needs. Opinions having belief or disbelief equal to one are equivalent to the boolean values of true and false. In boolean logic, the truth value of statements combined by these operators is determined by truth tables that is, tables that determine the results (true or false) for all possible combinations of inputs (true or false). The corresponding Subjective Logic version produces the same output as the truth tables do, when applied to opinions having belief or disbelief equal to one. In addition, it allows to compute the conjunction of opinions neither true nor false (having belief and disbelief less than one).

The wide range of operators available (including subjective versions of the boolean operators, together with the fusion and discounting operators), makes this logic particularly expressive, since it allows to capture many different existing relations between the statements. For an extensive example of application of the logic, see [3]; for more details about the logic, its foundations and its operators, see [5, 8, 9, 13].

13.3 First and Second Order Probabilities

We saw how an opinion represents the evidence that a certain subject has up to a certain moment by means of four values. Evidence might be partial or misleading but since it is the only "view" we have on the world, one should grasp all the information that it exposes without fully relying on it. This is the reason why uncertainty is quantified and why uncertainty is inversely related to the total amount of evidence, because we assume that an opinion is less uncertain if it is supported by

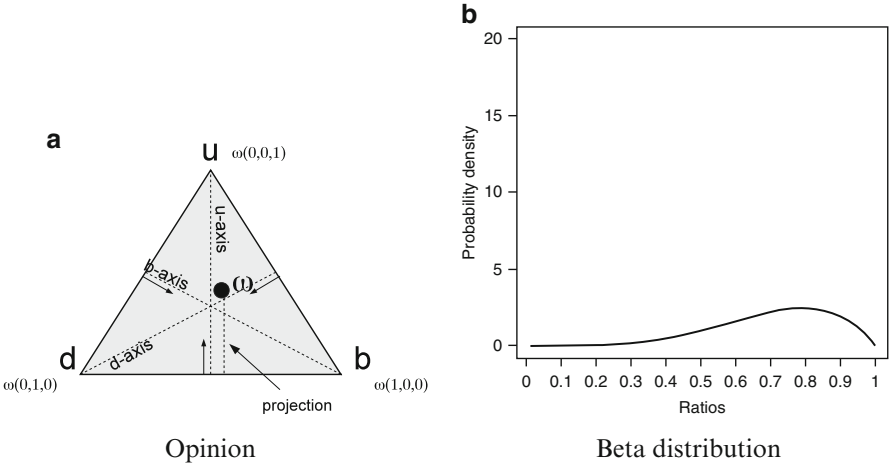


Fig. 13.2 ω is an opinion based on four positive and one negative evidence. Figure (**a**) represents ω in the triangular space. Figure (**b**) represents the corresponding Beta distribution (Beta(4+1, 1 + 1) = Beta(5,2)). The parameters of the Beta are equal to the amount of positive and negative evidence respectively, increased by one

more evidence. However, even in case a lot of observations support a given opinion, the opinion could be wrong, so then the opinion will always have an uncertainty component higher than zero.

An opinion can be graphically represented as in Fig. 13.2. The triangular representation is the simplest and most direct graphical representation of opinions. Therefore, although they are equivalent, it is often preferred to the Beta distribution representation shown in Fig. 13.2b. First, we will explain the triangular representation. Later in this section we will explain the Beta distribution representation of opinions, together with a clarification about the statistical implications of this probabilistic representation. The triangle depicted in Fig. 13.2 is not a triangle in a cartesian space, rather it represents the space of all the possible opinions where each of the sides is a "dimension" of such a space. Each side is an axis of this particular type of plain. Each side represents the space of all the opinions having a dimension equal to zero. For instance, the side that links the belief and disbelief vertices represents the set of all opinions having uncertainty zero, and the same holds for other sides. On the other hand, each vertex represents the opinion having only one dimension equal to one, that is the component that names the vertex. If we connect each vertex with the median of the opposite side, we discover three orthogonal axes, represented by dashed lines in Fig. 13.2, that range between zero (the median point of the side) and one, the vertex. Each component of the opinion determines the position with regards to each of these sides. So, for instance, if we take opinion $\omega(0.4, 0.1, 0.5)$, this opinion will be situated at distance 0.4 from the side disbelief-uncertainty, since its belief value is 0.4. The intersection of all these three distances determines the position of the opinion.

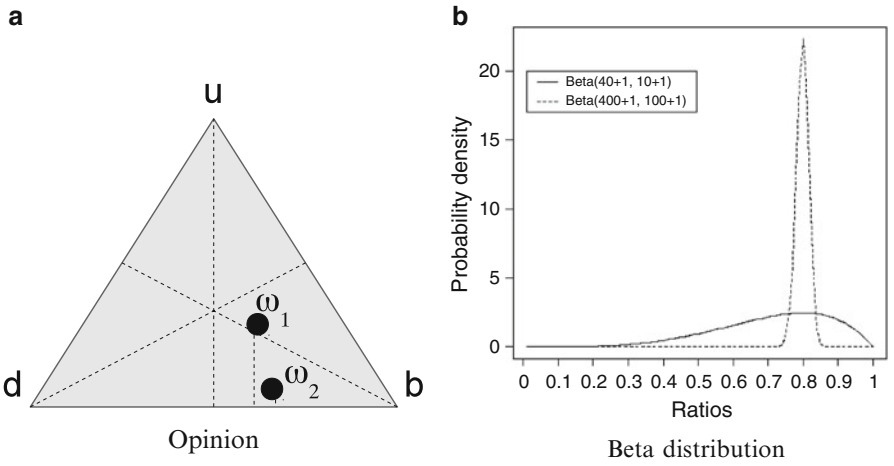


Fig. 13.3 ω_1 is an opinion based on four positive and one negative evidence, while ω_2 is based on 400 positive and 100 negative evidence. The *left* picture locates the opinions in the triangular space. The *right* picture shows the shape of the two distributions describing the opinions. Clearly opinion ω_2 has less variance than opinion ω_1, since it is based on more evidence

The opinion determines the correct point of the lower side of the triangle that represents the right proportion of the population, because the lower side is the set of all "certain" opinions. If we know everything about the whole population, there is no uncertainty in the ratio that we know. The lower side of the triangle ranges between zero (disbelief) and one (belief). What it represents is the proportion between sources of information that agree with the name of $Ship_{123}$ being "Beauty" and sources of information disagreeing with that. This proportion can be interpreted as the probability to observe a source of information agreeing with the statement *the name of* $Ship_{123}$ *is "Beauty"*. So, in turn, this probability is an estimate of the probability of the statement being true. However, having an opinion with uncertainty higher than zero, what is the population proportion that this opinion implies? That is, onto which point of the belief-disbelief side should we "project" our opinion?

The higher the point, the higher the uncertainty, because the height represents the uncertainty. Once the opinion is represented in the triangular space, it is possible to "project" it into the lower side in order to see the estimated proportion. The projection is a way to derive information about the population given the current available opinion. The slope of the projection is determined by the a priori component. If $a = 0.5$, then there is no bias, and the projection is therefore orthogonal with respect to the lower side. Otherwise, the projection will "penalize" (in case $a < 0.5$ the projection will be closer to the disbelief vertex of the triangle) or "reward" ($a > 0.5$) the opinion. Figures 13.3 and 13.4 show how two opinions based on observations with the same ratio (4:1) but with two different observation set sizes (500 vs. 5) are affected by 50 new observations, still keeping the same ratio. Of course, the opinion based on the smaller set of data is the most susceptible, since

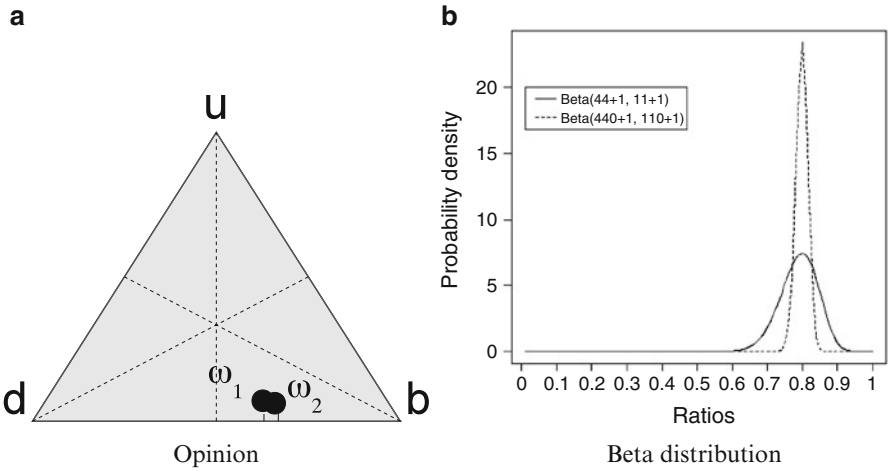


Fig. 13.4 ω_1 and ω_2 after update. Now ω_1 is based on 44 positive and 11 negative evidence. ω_2 is based now on 440 positive and 110 negative evidence. The shape of opinion ω_1 is much more affected by this change than the shape of opinion ω_2

it was more uncertain. This explains also why opinions based on less observations are more susceptible to the influence of a biased prior: if we base our opinion only on few observations, then it is natural to rely more on our prior knowledge.

The evidence that we observe is representable by a Binomial distribution, that is a discrete probability distribution describing two mutually exclusive outcomes, one that can happen with probability p and one with probability $1 - p$. Indeed, we want to determine whether the name of the $Ship_{123}$ is really "Beauty": the possible outcomes are two, true or false, and if the true value has p probability to appear, the false value will appear with probability $1 - p$. The observations that we own are observations about this fact being true or false.

The fact that we observed for instance two messages confirming that the name of $Ship_{123}$ is "Beauty" and one that disagrees, should not imply that we have $1/3$ of probability that the ship is not named "Beauty": there is uncertainty about the p parameter of the Binomial distribution. We should define a probability distribution describing the possible values for p on the basis of the current observations.

The probability distribution that meets our requirements is the Beta distribution. The shape of this distribution is determined by two parameters, a "rewarding" and a "penalizing" one (α and β) equal to the number of positive and negative observations plus one, respectively, because Beta(1,1) is by definition the Beta without any evidence. This distribution is used to describe the possible values that the parameter of the Binomial can assume. This relation is named "conjugacy" and the mathematical proof of this relation is available in [7, 12]. Roughly speaking, the Beta distribution works the same way as the opinions do: the ratio between rewarding and penalizing parameter (determined by positive and negative evidence) determines the position of the "peak" of the distribution (same value as the expected

value of the opinion). The Beta distribution behaves exactly as Subjective Logic's opinions do. The two parameters of the Beta can be expressed in the belief, disbelief, uncertainty and a priori value of the opinions:

$$\alpha = \frac{2 * b}{u} + 2 * a \qquad \beta = \frac{2 * d}{u} + 2 * (1 - a)$$

Finally, the expected value of the Beta distribution can be computed as $E = b + a * u$ and this value is exactly the value that we obtain by "projecting" the opinion onto the lower side of the triangular representation of the opinions, that connects belief and disbelief.

We have described our situation as a stratification of probability layers. The lower layer, the "first order probability", is represented by the Binomial distribution. This is the probability distribution that describes the probability to observe correct (true) information. The Beta distribution, that is, the "second order probability", situates on top of this layer. The Beta determines the choice of the model for the first order probability, because the Beta describes the possible values for the parameter of the Binomial and the parameter chosen for the Binomial determines its shape.

13.4 The Open Provenance Model

Provenance represents another important component of our trust evaluations. It is a complementary component with respect to the probabilistic and logic part, since it focuses on representing "where the data we are dealing with come from" and here, by "come from" we mean who produced them and how. Therefore, whereas the probabilistic logic described before gives us an important tool to consistently deal with the observations that we face, even when they are available in a small amount or when they disagree with each other, the provenance model allows us to capture information about the origin of the data that we are evaluating. Although this information does not provide direct evidence about the content reliability and correctness, it can be helpful in giving implicit evidence about that. For instance, if we know that certain information is provided by a certain "agent" and such an agent is known to have a bad reputation, we will probably distrust that data, in case we do not have any other information. In Sect. 13.2.2 we saw how to deal with sources' reputations by means of the discount operator $\otimes$, and here we will see how to capture a range of other information useful for our assessments. It is possible to represent this information logically by means of the components previously introduced. The next Sect. 13.5 will explain how this has been implemented in the maritime domain.

To represent this additional information, we use the Open Provenance Model [10], which is an open source model developed for representing and recording information about the provenance of artifacts, that is, who produced them and how. Its components are:

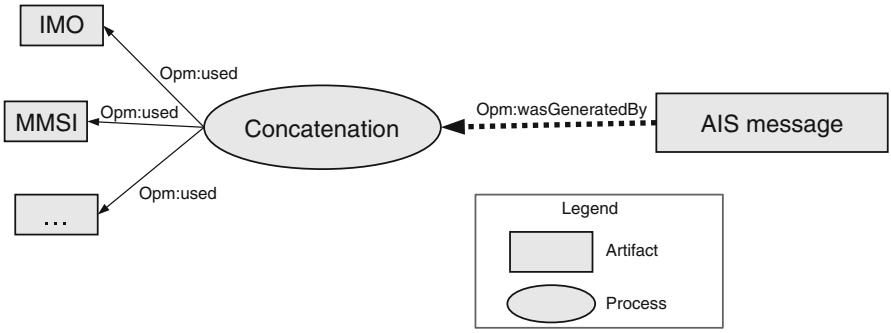


Fig. 13.5 Example of an Open Provenance Model graph. An AIS message is an artifact generated by a concatenation process. This process uses the values of the IMO, MMSI and other fields as input

Artifact Immutable piece of state, which may have a physical embodiment in an physical object, or a digital representation in a computer system.

Process Action or series of actions performed on or caused by artifacts, and resulting in new artifacts.

Agent Contextual entity acting as a catalyst of a process, enabling, facilitating, controlling, and affecting its execution.

Since the model aims at describing all the information about how the artifact of our interest was produced, the Open Provenance Model focuses on capturing the causal relations between agents, processes and other artifacts within such a production phase. These relations are defined as follows:

- Process *used* Artifact
- Artifact *wasGeneratedBy* Process
- Agent *wasControlledBy* Process
- Process1 *wasTriggeredBy* Process2
- Artifact1 *wasDerivedFrom* Artifact2

So, the model allows to represent formally all the information that we have about the provenance of the "objects" of our opinions, which can be considered as artifacts: all the processes and artifacts that were used (input and intermediate data), together with the agents that controlled them. The model also allows to infer new relations. For instance, if Artifact1 *wasDerivedFrom* Artifact2, then implicitly, there is a set of processes in between the two artifacts, that used Artifact2 as input in order to obtain Artifact1. For our purposes, the most important thing is that the model allows to represent the network of dependencies and relations that influences the reliability level of the data that we are analyzing. Figure 13.5 shows an example of an Open Provenance Model instance. We will see in the following section how to build such a network and how to apply probabilistic logic reasoning on top of it.

The Open Provenance Model might be considered as overlapping with the Simple Event Model described in Chap. 10. However, the two differ in their focal

point: whereas the Simple Event Model focuses on the description of events (what happened, when, where, who was involved, etc.), the Open Provenance Model focuses on the causal relations among events, in order to describe the chain of events that led to a given artifact. A mapping between these models is provided in [2].

13.5 Using Provenance for Improving Trust Assessments of AIS Messages

This section will describe how to compute trust levels for the AIS messages by combining Subjective Logic with provenance information. An AIS message contains, amongst others, the following fields:

- IMO: unique identification code from the International Maritime Organization;
- MMSI: the Maritime Mobile Service Identity code is a nine digits code used for communication purposes. Its first three digits are determined on national basis;
- CallSign: four or five digits communication code. Its first two digits are determined on the basis of the nation of the ship;
- Name of the ship;
- Flag of the ship.

We applied Subjective Logic reasoning on the fields reporting static information about the ship (like IMO, MMSI and CallSign), and not on the fields reporting kinematic information (like speed or heading). For each field we computed an opinion based on all the available evidence, that is, AIS messages and information crawled from the Web.[2] Then, we merged all the opinions taking into account provenance information, that is, how information contained in the fields is produced. Since there exists also a dependency relation between certain fields, provenance allows to encode dependencies between them, as we will see later. The trust level of the whole static part of a message is determined by combining the trust level computed for each field apart. These pieces of information are combined by the "AND" operator. The reason why we use this operator, is that, in order to be considered "trustworthy", a message should carry only correct (or "trustworthy") information. If one or more pieces of information are not, then the whole rating should be affected by this.

There are two categories of fields: independent and dependent (or partially dependent) fields. Fields like, for instance, the width or the name of the ship, are not bound to any other information within the message itself. So, to compute the trust value for these fields, we gathered all the evidence available and properly count them to build an opinion.

Other fields, like the MMSI code and the CallSign are dependent on the flag field, which represents the nationality of the ship. The Open Provenance Model can help

[2]In particular, we crawled www.vesseltracker.com and www.shipais.com websites

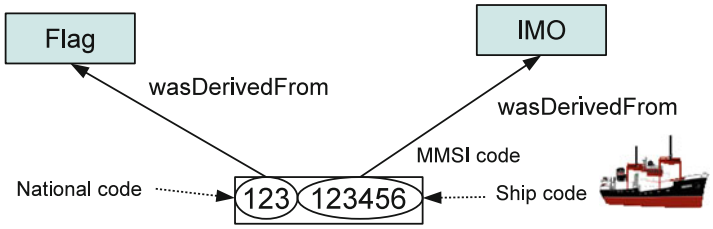


Fig. 13.6 Graph representing the provenance of the MMSI field of an AIS message. The code depends on the Flag field for the national code part and for the IMO code for the ship part

us in recording this information. Assuming that the IMO is a code able to uniquely identify the ship, we can record the following relations:

MMSI national code *wasDerivedFrom* Flag
MMSI ship code *wasDerivedFrom* IMO
MMSI *wasDerivedFrom* MMSI national code
MMSI *wasDerivedFrom* MMSI ship code

The CallSign field is defined exactly in the same way. For each field we have a small graph (for instance, see Fig. 13.6) with the field itself being dependent on two components. We know from the domain that the process that produced the codes is the concatenation process. From trust perspective, it means that the two input elements do not influence each other, because they determine the value of the two elements that, once concatenated lead to the overall code. Therefore, these two elements need to be both true so that the whole message can be true ("AND" operation). So, we computed the opinion for the second part of the MMSI, of the CallSign and of the flag, based on the available evidence. Fig. 13.7 shows the network of information that we have just described.

We did not have the possibility to determine the MMSI local part given the IMO code from a reliable service, but by employing Subjective Logic and exploiting provenance information we could obtain reliable estimates for this 6-digit code (that is, the ship code of the MMSI). With regards to the national part, instead, we have the possibility to map it into the nation that it represents. Therefore, we can merge all the evidence we have about the flag, the MMSI national part and the CallSign national part into a single opinion about the nationality of the ship. The national part of the MMSI code is a 3-digit code. Before doing this, we need to have a map that collects all the national codes for MMSI and CallSign. We retrieved these maps from a Web repository of places-related information and of communication codes[3] (see Fig. 13.8a).

[3]In particular we crawled the International Telecomunication Union website (http://www.itu.int/online/mms/glad/cga_mids.sh?lng=E) and from Citymap HQ website (http://www.citymaphq.com/codes/itu.html)

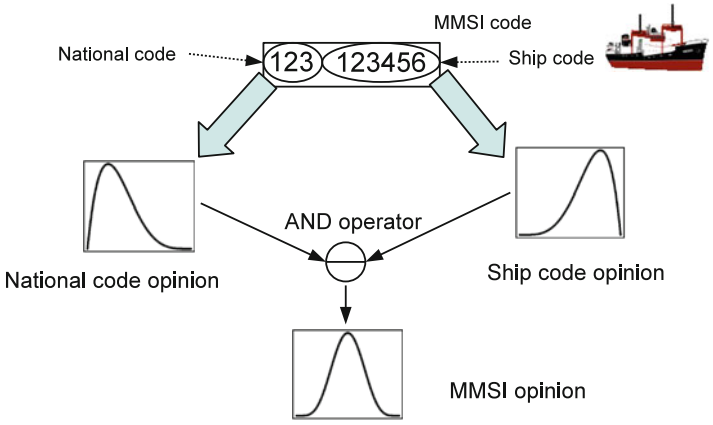


Fig. 13.7 The computation of an opinion regarding the communication code of $Ship_{123}$ is made by merging opinions on the national and ship components of the code. The national part is evaluated by considering the nationality of the ship

Finally, in order to determine the trust values from the AIS messages, we computed:

- The trust values for all the "independent" fields (e.g. the flag). Note that, the AIS messages also report a timestamp, that is, a field indicating the moment when they are sent. When computing the trust value for these elements, we consider all the evidence available up to that point: messages arrived before the one that we are analyzing, and all the Web data considered (websites providing AIS-related information).
- The trust values for all the "dependent" components, like the CallSign and the MMSI code. The trust value of these elements is computed by applying the "AND" operation over the input elements (national and local codes).
- The trust value of the entire message, by computing the AND operation of all components.

We computed the trust values for all the messages in our dataset, that covers 1 week period. In addition to these data, we consulted a few Web sources[4] to increase the amount of data at our disposal. An example of the visualization of the results of these calculations is available in Fig. 13.8b. Table 13.1 reports some summarizing statistics about the trust levels computed. We did not have any information about the reputation of the sources at our disposal, so the belief average concentrates around the middle of the range because of an initial situation of high uncertainty. Moreover, when different disagreeing values were proposed by different sources for a given field, we computed the trust values for all of them.

[4] www.shipsais.com and www.vesseltracker.com

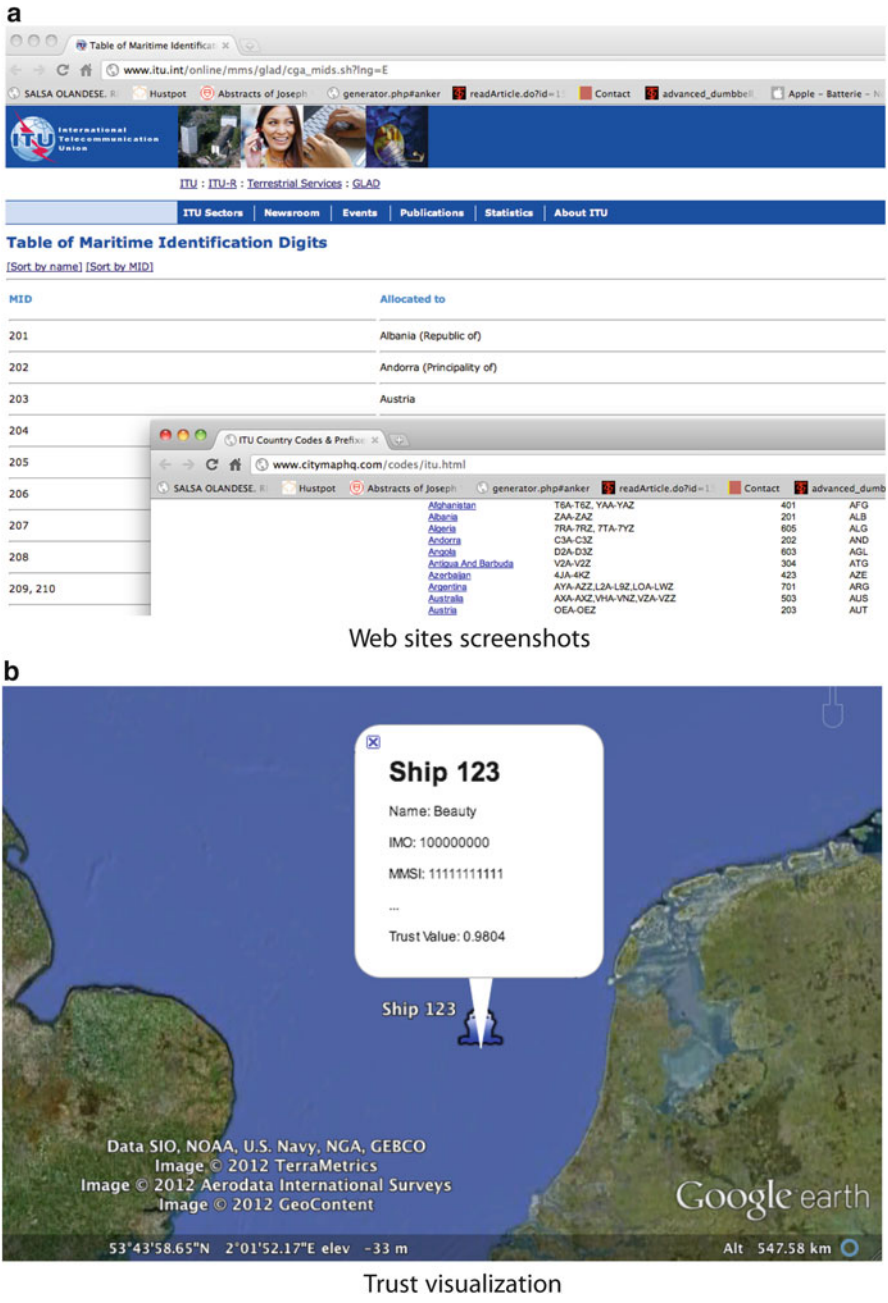


Fig. 13.8 Screenshot of International Telecommunication Union and Citymap HQ websites (**a**) and trust value visualization (**b**). The ship is localized thanks to AIS data

Table 13.1 Statistics about the belief and uncertainty of the trust level computed

	Min value	Max value	Average	Median
Belief	0.0005	0.9985	0.5834	0.5
Uncertainty	0.0015	0.5	0.2578	0.1667

This explains why the range of the beliefs is so wide: correct values were very popular and so had a high trust value, and consequently, wrong or non-comforming values (like messages reporting MMSI code value "0") got a lower trust value. The maximum value for uncertainty is 0.5 because it corresponds to the uncertainty of an opinion based on one observation, that is, on the first message (see Sect. 13.2.2 for additional details about how uncertainty is computed). Opinions are computed incrementally, so we computed an opinion for each message, considering all the messages observed up to that moment. This means that consecutive opinions will manifest decreasing uncertainty and the belief in rare values will decrease, while the belief in common values will increase. For instance, a belief of 0.0005 corresponds to 1 positive evidence over 1998 total evidence ($\frac{1}{(1998+2)} = 0.0005$; 2 is the range of possible evidence (true, false)). Vice-versa, a belief of 0.9985 corresponds to 1997 positive evidence over 1998 total observations ($\frac{1997}{(1998+2)} = 0.9985$). For further details see [2].

13.6 Discussion

This chapter describes current research. As we have seen in Sect. 13.5, some experiments have already been executed and some results are already available. Based on these experiments and results, we can derive some interesting conclusions and learn about interesting facts.

The choice of the model was driven by the clear requirements that the problem had. One important requirement is the impossibility to assume that the evidence at our disposal is the result of a random sampling process. This is due to the fact that the evidence considered by us is not the result of a controlled drawing process. Rather, we used all the observations at our disposal without any information about their reliability or representativity. This is an important consideration and motivates why, for instance, we did not take a "classical bayesian" approach, and, more precisely, why we did not assume that our data were normally distributed: if there was biased manipulation in the messages, their distribution could have taken any shape. This fact led us to the following differences with regards to a "classical bayesian" approach:

- We could not assume that our observations were "identically independently distributed", for the reasons that we have just explained, so we could not make use of estimators based on normality assumptions;
- Sources' reputations had to be explicitly recorded and incorporated in our evaluations. Initially, for instance, in case we had no prior information, the

reputation was neutral (neither positive nor negative), but the data provided by that source were considered as particularly uncertain;
- We did not know what the representativity of the model was that we inferred from the observation, hence we estimated the likelihood of the various possible models, instead of directly estimating the most likely values. We inferred two orders of probability: one about the possible models and one about the outcomes, given the most likely model.
- Finally, the uncertainty component of opinions as such is a typical characteristic of Subjective Logic that concisely quantifies the lack of information. There is no parameter in bayesian models directly corresponding to it.

The use of first and second order probabilities is useful when dealing with multiple levels of uncertainty (uncertainty about the outcomes and uncertainty about the model representing the data), because it prudently computes a probability distribution based on the actual observations. This probability distribution can be used, for instance, by a decision strategy with the aim of deciding whether a message is trustworthy or not. The prudence of the model is due to the limiting assumptions on which it is based (for instance, it does not assume that the observations are randomly obtained).

As we previously stated, this is ongoing work. The results obtained are useful, since they allow to highlight messages containing untrustworthy information. However, by gathering further information, we might improve the reliability of our evaluations and increase the speed of convergence of beliefs and disbeliefs. First, we have to take into account that, given that we did not have prior knowledge about any ship, evaluations started with an initial period of high uncertainty for almost any ship, and this explains why average uncertainty is quite high, and average belief is not really high. Moreover, having at our disposal more domain knowledge and information, would have helped in better understanding the results and motivating them. For instance, suppose that a sequence of messages confirms that the name of $Ship_{123}$ is really "Beauty" and that, after that sequence of messages, we receive a disagreeing message. A database of ownership of ships, could have allowed to show that the change of the value of the owner field (or the name of the ship) is not due to an error, but correctly reflects the change of ownership. The incorporation of provenance information in our model is a solution that goes in this direction. Furthermore, if we would have had at our disposal more domain data like, for instance, the reputation of the sources, we would have reduced the initial situation of high uncertainty, avoiding the evaluation of messages only on frequency basis.

13.7 Future Work

A combination of provenance and probabilistic logic is apparently a promising direction with regards to trust assessments. So far we focused on the representativity of the ratio between agreeing and disagreeing observations, as an indicator of the

trustworthiness of observed values. A straightforward extension of this approach would be computing the representativity of each observed value (taking into account that our observations could not have covered the entire range of possible values). Preliminary work in this direction has already been published [4]. This preliminary work is based on extensions of the probability distributions described in this chapter (for additional information about these extensions see the work of Ferguson and Pitman [6, 11]). A possible development of this work will link these extended probabilistic models with provenance information, similarly to what has been described in this chapter.

Acknowledgements This research has been carried out as a part of the POSEIDON project at Thales under the responsibility of the Embedded Systems Institute (ESI). This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

We would like to thank the editors for their effort spent in reviewing this chapter.

References

1. Castelfranchi C, Falcone R (2000) Trust is much more than subjective probability: mental components and sources of trust. In: Proceedings of the 33rd Hawaii international conference on system sciences, HICSS2000, vol 6, pp 154–165
2. Ceolin D, Groth P, van Hage WR (2010) Calculating the trust of event descriptions using provenance. In: Sahoo SS, Zhao J, Missier P, Gomez-Perez JM (eds) Second international workshop on the role of semantic web in provenance management (SWPM 2010). http://ceur-ws.org/Vol-670/completeSWPM10Proceedings.pdf
3. Ceolin D, van Hage WR, Fokkink W (2010) A trust model to estimate the quality of annotations using the web. In: Web science conference (WebSci10), 26–27 Apr 2010. http://journal.webscience.org/315
4. Ceolin D, van Hage WR, Fokkink W, Schreiber G (2011) Estimating uncertainty of categorical web data. In: Bobillo F et al (eds) 7th international workshop on uncertainty reasoning for the semantic web (URSW 2011), pp 15–26. http://ceur-ws.org/Vol-778/proceedings.pdf
5. Dempster A (1967) Upper and lower probabilities induced by a multivalued mapping. Ann Math Stat 2(38):325–339
6. Ferguson TS (1973) A bayesian analysis of some nonparametric problems. Ann Stat 1(2):209–230
7. Fink D (1995) A compendium of conjugate priors. Technical report, Cornell University
8. Jøsang A (2007) Probabilistic logic under uncertainty. In: Gudmundsson J, Jay CB (eds), Theory of computing 2007. Proceedings of the thirteenth computing: the Australasian theory symposium (CATS2007), Ballarat, Victoria, Australia, proceedings, ser. CRPIT, Jan 30 – Feb 2, 2007, vol 65, pp 101–110. Australian Computer Society, Darlinghurst
9. Jøsang A, McAnally D (2005) Multiplication and comultiplication of beliefs. Int J Approx Reason 38(1):19–51
10. Moreau L, Clifford B, Freire J, Futrelle J, Groth P, Gil Y, Kwasnikowska N, Miles S, Missier P, Myers J (2010) The open provenance model core specification (v1.1). Futur Gener Comput Syst (online). Available: http://dx.doi.org/10.1016/j.future.2010.07.005
11. Pitman J (1995) Exchangeable and partially exchangeable random partitions. Probab Theory Relat Fields 102(2):145–158
12. Raiffa H, Schlaifer R (1968) Applied statistical decision theory. MIT, Cambridge
13. Shafer G (1976) A mathematical theory of evidence. Princeton University Press, Princeton

Chapter 14
Online Fault Localization and Health Monitoring for Software Systems

Éric Piel, Alberto Gonzalez-Sanchez, Hans-Gerhard Gross, and Arjan J.C. van Gemund

14.1 Introduction

It is generally accepted that all but the most trivial software systems will inevitably contain residual defects. Large and complex software systems, such as systems of systems, will face these problems. Nowadays, the high reliability, availability, and flexibility imposed on many systems require support for online reconfiguration and join/leave of external components (a coupled and cohesive part of a system). This further increases the chances of unexpected behavior during execution, as they are hard to take into account in the validation phase. As such problems cannot be avoided, the system should be prepared to handle them as quickly as possible. Typically, after a failure (a deviation from the expected behavior) has been detected the following steps are taken: diagnosis, bug fix design, re-validation, and update. To reduce the time of this process, we focus here on automating the diagnosis step, which very few previous works have tried to automate. This step focuses on finding the location of the fault, i.e., the cause of one or more failures in the system.

So far automated diagnosis techniques, also called fault localization, have been applied solely offline, during the testing phase. In this chapter, we detail approaches to apply fault localization in an online context, i.e., when the system is in operation. One of the obstacles is that typical *active testing* used offline cannot be applied online, because of interference with the normal operations. So continuous validation must come from observations provided by monitors, also referred to as *passive testing*. While there exist other approaches to fault localization [3, 9, 10], SFL is one of the most light-weight fault localization techniques available to be used for the provision of health information and for identifying problematic components in software systems. This technique is described in Sect. 14.2. Section 14.3 presents

É. Piel (✉) • A. Gonzalez-Sanchez • H.-G. Gross • A.J.C. van Gemund
Department of Software Technology, Delft University of Technology, Delft, The Netherlands
e-mail: e.a.b.piel@tudelft.nl; a.gonzalezsanchez@tudelft.nl; h.g.gross@tudelft.nl; a.j.c.vangemund@tudelft.nl

P. van de Laar et al. (eds.), *Situation Awareness with Systems of Systems*,
DOI 10.1007/978-1-4614-6230-9_14,

the modeling of the problem. Our proposal of online fault localization is presented in Sect. 14.4. Section 14.5 summarizes the main approaches we have used to implement fault localization on actual software systems. Section 14.6 evaluates the technique on a case study. Finally, Sect. 14.7 summarizes and concludes the chapter.

Let us note that we will discuss fault localization further in Chap. 15. We will then examine how SFL can be shortened during the testing phase by picking the most relevant test cases.

14.2 Spectrum-Based Fault Localization

The objective of fault localization is to pinpoint the precise locations of faults in a system. Before delving into the usage of the SFL approach for online fault localization, and the provision of health information, let us introduce SFL in its offline version.

The following data are usually used as inputs in SFL approaches:

- A finite set $\mathscr{C} = \{c_1, c_2, \ldots, c_j, \ldots, c_M\}$ of M *components* (e.g., source code statements, functions, classes) which are potentially faulty. We will denote the number of faulty components in the system as M_f.
- A finite set $\mathscr{T} = \{t_1, t_2, \ldots, t_i, \ldots, t_N\}$ of N given tests with binary outcomes $O = (o_1, o_2, \ldots, o_i, \ldots, o_N)$, where $o_i = 1$ if test t_i failed, and $o_i = 0$ otherwise.
- An $N \times M$ *coverage matrix*, $A = [a_{ij}]$, where $a_{ij} = 1$ if test t_i when executed involves (covers) component c_j, and 0 otherwise. Each row a_i of the matrix is called a *spectrum*.

Table 14.1 shows an example of SFL applied on a small program with a component granularity at the statement level. This program aims at counting different types of characters. The component c_3 contains a fault, mishandling uppercase

Table 14.1 Example program, spectrum, and output in fault localization

$\mathscr{C}$	Program: character counter	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	SC
	`function count(char *s) {`									
	`  int let, dig, other, i;`	0	0	0	0	0	0	0	0	0
c_0	`  let = dig = other = i = 0;`	1	1	1	1	1	1	1	1	0.87
c_1	`  while (c = s[i++]) {`	1	1	1	1	1	1	1	1	0.87
c_2	`   if ('A'<=c && 'Z'>=c)`	1	1	1	1	1	1	0	1	0.93
c_3	`    let += 2;`	1	1	1	1	1	1	0	0	1.0
c_4	`   else if ('a'<=c && 'z'>=c)`	1	1	1	1	1	0	0	1	0.83
c_5	`    let += 1;`	1	1	0	0	1	0	0	0	0.71
c_6	`   else if ('0'<=c && '9'>=c)`	1	1	1	1	0	0	0	1	0.73
c_7	`    dig += 1;`	0	1	0	1	0	0	0	0	0.71
c_8	`   else if (isprint(c))`	1	0	1	0	0	0	0	1	0.47
c_9	`    other += 1;}`	1	0	1	0	0	0	0	1	0.47
c_{10}	`  printf("%d %d %d\n",`	1	1	1	1	1	1	1	1	0.87
	`  let, dig, other);}`									
	Test case outcomes	1	1	1	1	1	1	0	0	

characters. Eight tests are executed against this implementation. The columns t_1 to t_8 present the coverage spectrum and the test outcomes when executing each of the tests. The last column shows the similarity coefficients, a value computed by the SFL, which we will describe later.

The output of fault localization is a *diagnosis*, which is a ranking of the components ordered according to their assumed likelihood to contain a fault.

In program debugging, the granularity of a *component* is often very small, typically at the statement level, since SFL benefits from variations in program control flow (i.e., different branches of a `if` are taken). However, in an online context, a larger grain size for components is more appropriate. This still permits to monitor a system and to take the appropriate actions in case of degradation, while it reduces the performance overhead, and represents a more realistic component granularity for large systems. In the later study, we selected a granularity at the level of the source code functions.

14.2.1 Statistical Spectrum-Based Fault Localization

Statistical SFL is a well-known approach originating in software engineering [1, 5, 11]. Fault likelihood l_j (and thus assumed health) is quantified in terms of *similarity coefficients*. Intuitively, the goal is to identify the component whose line of test coverage is most similar to the test outcomes. Similarity coefficients measure the statistical similarity between component c_j's test coverage $(a_{1j}, \ldots, a_{Nj})$ and the observed test outcomes, $(o_1, \ldots, o_N)$. It is computed by four values $n_{pq}(j)$ counting the number of times a_{ij} and o_i form the combinations $(0,0), (0,1), (1,0), (1,1)$, respectively, i.e.,

$$n_{pq}(j) = |\{i : a_{ij} = p \wedge o_i = q\}| \;\; p,q \in \{0,1\} \tag{14.1}$$

For instance, $n_{10}(j)$ and $n_{11}(j)$ are the number of tests in which component c_j is executed, and which passed or failed, respectively. For each component, the four counters sum up to the number of tests N. There are several different known similarity coefficients which are efficient. For example, Tarantula [5], and Ochiai [1] are both very common similarity coefficients. We use the latter one, given by

$$\text{Ochiai:} \;\; SC = \frac{n_{11}(j)}{\sqrt{(n_{11}(j)+n_{01}(j)) \cdot (n_{11}(j)+n_{10}(j))}} \tag{14.2}$$

Ordering the components by their similarity coefficients results in the ranking of the diagnosis algorithm.

In Table 14.1, the similarity coefficient for each component is indicated. As c_3 was the part most used when a test failed and less used when a test passed, its similarity coefficient is the highest. The SFL will therefore rank c_3 as the most likely location of the fault, which is correct.

A by-product of statistical SFL is the *component health*. The health of a given component can be simply approximated by $h = 1 - SC$, where SC is the similarity coefficient. This permits the system, or system of systems, to also be self-adapting to the failures. Components which have access to redundant information can adapt the weight of each input depending on the health of the components that provide it. For example, in the maritime safety and security context, when a radar starts behaving incorrectly, the situation awareness component can reduce automatically the importance of the data from this radar in its computations.

Despite their lower diagnostic accuracy [2], similarity coefficients have a ultra-low computational complexity (compared with probabilistic diagnosis approaches, such as Bayesian reasoning [5]), which is ideal for online diagnosis. Another advantage is the fact that statistical SFL is incremental. Only the counters n_{pq} must be kept per component, so there is no need to compile a (possibly huge) test coverage matrix. Finally, unlike other approaches, statistical SFL is robust with respect to uncertainties in the test outcomes. While all techniques tolerate false negatives (i.e., a test involving a faulty component and not returning a failure), statistical approaches are more robust with respect to false positives (i.e., a test reports a failure although the system actually behaved correctly), which is essential in online monitoring as the oracles are often less sophisticated than in offline testing.

14.2.2 Diagnosis Effort

In order to compare different diagnosis approaches, there is a need to measure how well a diagnosis performed. This measure, the diagnostic performance, should represent how well the diagnosis algorithm can pinpoint the true root cause of an observed problem. In software fault localization, this performance is often expressed in terms of a metric C_d that measures the theoretical *effort* still needed for a diagnostician to find all faulty components after reading the generated diagnosis [2]. C_d is expressed as the position of the last faulty component in the ranking given by the fault localization. C_d measures *wasted effort*, independent of the number of faulty components M_f in the system, to enable an unbiased evaluation of the effect of M_f on C_d. Thus, regardless of M_f, $C_d = 0$ represents an ideal diagnosis technique (all M_f faulty components are ranked at the top, and no effort is wasted for a human to check healthy components), while $C_d = M - M_f$ represents the worst diagnosis technique (checking all $M - M_f$ healthy components before the M_f faulty ones), with M the total number of components. For example, consider a diagnosis algorithm that returned the ranking $\langle c_{12}, c_5, c_6, \ldots \rangle$, while c_6 contains the actual fault. This diagnosis leads the developer to inspecting c_{12} and c_5 first.

As both components are healthy, C_d is increased by 2, and the next component to be inspected is c_6. As it is faulty, no more effort is wasted and $C_d = 2$. To ease comparison between systems, a *relative wasted effort* is often used: $\frac{C_d}{M-M_f}$. A perfect diagnosis gives therefore a relative effort of 0, while the worse possible one gives an effort of 1, and an algorithm picking randomly a component gives on average a relative effort of 0.5.

14.3 Simulation of a Faulty System

For initial illustration and evaluation of online SFL we use synthetic system simulations next to an actual case study. The main advantage of the simulations is that they can be executed quickly (e.g., for our case study system we can simulate 1 h of operation in just a few seconds). They also avoid implementation details which could cause noise in the observations (e.g., test outcomes with false positives), and they allow to vary many properties of a base system, in order to generalize the findings according to many different (synthetic) system configurations.

14.3.1 System Model

The simulations use system models with different topologies all based on the surveillance system used as case study, which is presented in Sect. 14.6. The simulation of a system generates outputs similar to the ones given by the actual SFL algorithm, i.e., a ranking of the components according to their assumed health over the whole period of execution of the simulation. The simulator and example models are available for download.[1]

Figure 14.1 shows an example of a system model, with seven functional components and three monitors (A, B, and C). As we will see in Sect. 14.4, monitors are placed in order to replace test cases in an online context. Component 2 is set to be faulty, with a fault happening 60 % of the time it is used. The model represents a typical data-flow system where component 1 receives the inputs and passes them on to the other components. More information about the simulation setup and a description of the type of model that is used in the simulator can be found in [8].

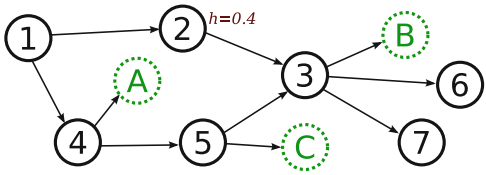


Fig. 14.1 Example topological layer with seven functional components and three monitors

[1] http://swerl.tudelft.nl/bin/view/Main/SOFL

14.3.2 Simulated System Generation

One of the most difficult parts of simulation is to obtain models of systems which are representative of the reality. If a model is generated fully randomly with respect to every possible parameter, there is little chance that it corresponds to a potential real system. That is because only some topologies, order of execution, etc. are reasonable for the software of a system-of-systems. Therefore, as basis for creating many simulations, we used the topology of a known surveillance system. It comprises 63 components for the functionality. For each component, a configuration was generated with that component being the faulty one. For each fault location, 10 different system configurations were generated by randomly placing 15 monitors, and producing a set of 20 execution paths (with random frequencies between 0.2 and 50 Hz). Therefore, each technique can be evaluated on 630 system configurations. Results are presented in the next section.

14.4 Online Fault Localization

Applying SFL online brings up three issues: (1) test cases would disrupt the normal operation of the system (to be discussed in Sect. 14.4.1, (2) the range of a coverage spectrum (to be discussed in Sect. 14.4.2), (3) the adequacy of the diagnosis with the current system behavior (to be discussed in Sect. 14.4.3). In an offline context, tests are run separately, so the start and end of a test and the coverage spectrum are clear, as well as associated inputs and outputs. However, in the case of continuous diagnosis these boundaries disappear, or, at least, become blurred. In this section, we present solutions for adapting SFL to an online context.

14.4.1 Obtaining Test Outcomes Online

In order to bring fault localization online, the usage of test cases must be reevaluated. Test cases are *active*, as they provides their own inputs to the system. If done during operation, such input can interfere with the usual behavior of the system, and can cause a large performance overhead. Therefore, in the online context, *monitoring* is more fitting. Monitoring is well understood, easy to apply, and event-based, due to its passive nature, e.g., triggered by the arrival of new data, or a timer interrupt. A monitor is a specific component in the system that observes and assesses the correctness of the functionality without interfering through test inputs.

A monitor observes data or behavior at specific locations and decides based on built-in oracle logic whether an observation is expected (*pass*) or unexpected (*fail*), for example through checking the range of a variable, consistency between different data, or through comparison with a state model. The monitor outcomes

replace the test outcome. Because SFL requires to know when the system is deemed behaving both correctly and incorrectly, it is of prime importance when writing a monitor that whenever a *fail* could be sent, it sends a *pass* if no failure is detected.

14.4.2 Spectrum Sampling

In many cases, interactions in a live system are not clearly separable by time or space boundaries (such as a complete test transaction in testing). Input stimuli are continuously arriving and the system responds accordingly changing its internal state and/or producing some output. For example, in our case study (cf. Sect. 14.6), input messages arrive at any time, and sometimes simultaneously in separate threads. Previous inputs influence the behavior of a component either explicitly such as in a database, or implicitly by affecting its internal state. When applying SFL offline, the coverage spectrum is recorded since the system was started for a test case. In an online context, after a short period of operation, the coverage matrix will contain only 1's: "everything covered". Although this approach would guarantee a theoretical strong causal relationship between fault execution and failure observation (i.e., if a failure is observed, the spectrum will contain the fault information), a solid 1's spectrum does not provide any diagnostic information for the SFL, because it infers the diagnosis from differences of the various spectra in the coverage matrix A and the outcome O. The curve named *time inf* of Fig. 14.2 shows the result of never resetting the spectrum. The average diagnostic cost is approximately 0.5 all the time. Guessing the fault locations randomly would yield a similar performance.

The coverage of components represented as binary values in the spectrum must be reset regularly, in order to provide a meaningful diagnosis. We propose two different solutions, which are adapted to different development contexts, that is, a transactional approach, and a time frame approach.

14.4.2.1 Transactional

A monitor validates the correctness of a specific component transaction in the system, corresponding to particular interactive functionality. The provision of an outcome through the monitor correlates to the end of this transaction. The *transactional* policy assigns a separate spectrum to every monitor. Every monitor is also associated to a scope, which represents which components might be involved in the monitored interaction.[2] Each time a component is involved, the current spectrum of every monitor whose scope contains that component is updated. When a monitor generates an outcome, its associated spectrum is used as a row for the matrix A and is then completely reset to zero.

[2]Each execution of the interaction can be considered a *transaction*, hence the name of the policy

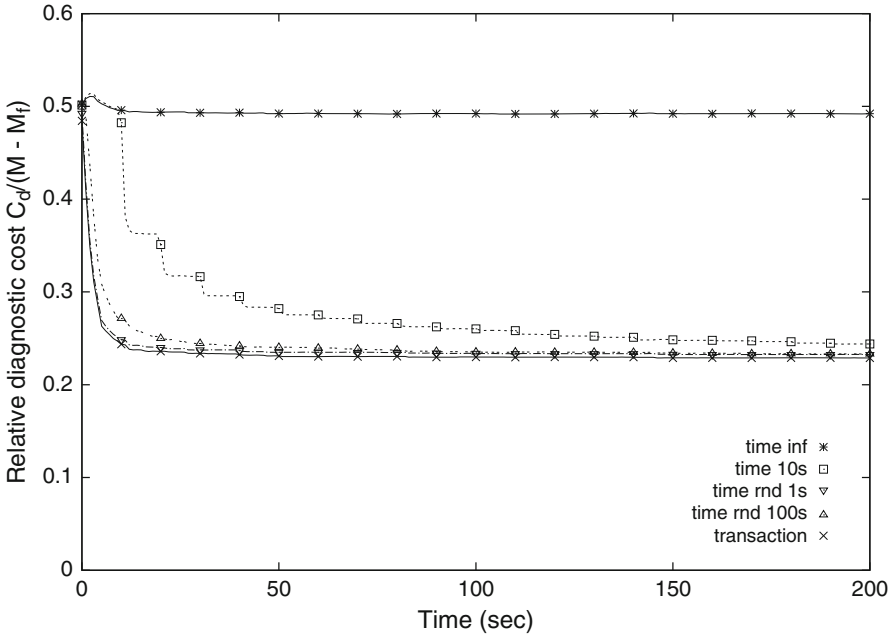


Fig. 14.2 Average diagnostic cost along the time of observation for various observation policies, with simulated systems having one fault

The list of the components in the scope associated to each monitor is provided before the start of the system (and is updated after each modification). It is either manually created by the user (the developer of the monitor, most likely), or it could be determined by code or configuration analysis. Figure 14.2 shows with the curve *transaction* that this solution is the most effective one, with a low average diagnostic cost throughout the execution of the systems. The curve tends towards an asymptote close from 0.2. This asymptote corresponds to the average diagnostic cost that can be achieved by the SFL algorithm with all possible spectra for the specific set of systems in the simulation.

However, if a fault modifies how components interact (i.e., the control flow is modified), the difference between the expected behavior and the implementation could lead to an inaccurate scope. In such a case, this policy would cause a faulty component to be omitted from every spectrum associated with a *fail* outcome. The quality of the diagnosis would be adversely affected. In addition, pre-analysis of the system for every monitor can be time consuming, and needs to be done every time the system is modified. It might be difficult to perform if external components (from different companies) are used. In order to avoid this analysis we investigate a technique requiring less information about the system, i.e., the *time frame* technique.

14.4.2.2 Time Frame

The *time frame* policy uses expiration of time as transaction boundary to establish causality between components covered and monitor outcome. Over a given time period, the component activity is recorded into a global "current spectrum". When the time expires, the bits of the involved components are reset and the recording of a new current spectrum is started. Every monitor outcome during this period, is associated with the current spectrum.

Time frame-based sampling avoids spectra with too many 1's if the time window is properly adjusted to the working speed of the system. To avoid using a period which could hide a specific fault our approach uses a random frame length. After expiration of a time frame, the length of the next frame is determined randomly within reasonable bounds. An exponential distribution is used, in order to have a broad set of period sizes. An average period must be selected according to the system under observation, but it can be relatively roughly estimated to the average processing time of a typical transaction. In Fig. 14.2 it can be seen how a fixed time period leads a limited accuracy of the fault localization, with the curve *time 10 s*. The curves *time rnd 1 s* and *time rnd 100 s*, corresponding respectively to a randomized time frame with an average of 1 s and 100 s, both provide on average a low diagnostic cost.

We recommend that the observation policy should be selected according to the system context: if it is possible to gather precise information on which interaction is observed by a monitor, then the transactional policy should be applied. Otherwise the randomized time frame policy should be implemented, with just enough validation to ensure the average period is adapted to the system.

14.4.3 Spectrum Matrix Size

When using SFL offline, the size of the spectrum matrix and the test outcome vector are finite and, in practice, relatively small, which is not the case online. For example, in our case study, approximately 100,000 monitor outcomes are generated for a single hour of observation. This could eventually lead to excessive storage requirements and processing overheads. This potential size problem is addressed through application of statistical SFL, on which our approach relies. It is incremental, so that accumulating counters can be used.

However, another issue is the timeliness of a spectrum, for example "is a week-old observation relevant for the current state of the system?" A fault may appear long time after the system was started (e.g., memory leakage, unexpected combination of inputs that affect the internal state of the system, an unnoticed third-party component update). Old spectra might mislead the fault localization. The detection of a new failure should always lead to the same diagnosis, independent of how long the system has been running.

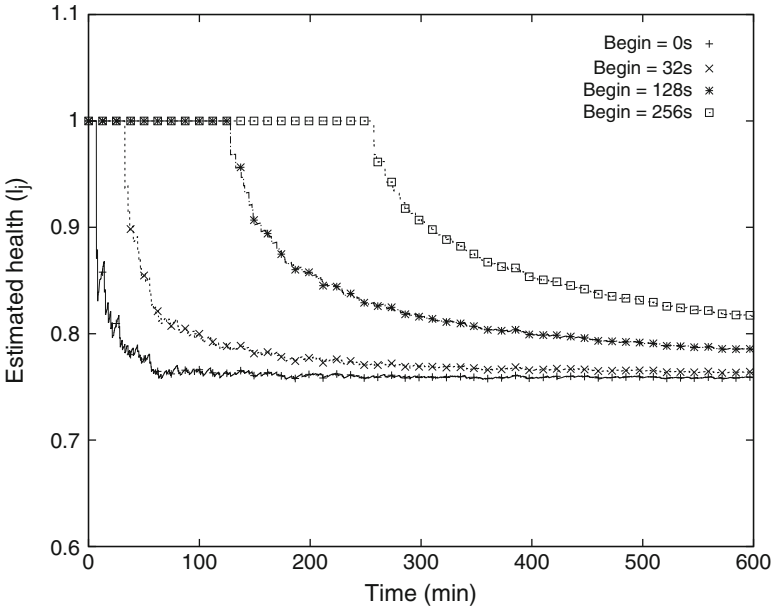


Fig. 14.3 Estimated health with an infinite window

Note however that the problem is not symmetric, when conversely, a fault is fixed, or the failures are not observed anymore. If the fault is fixed, it is easy to reset the matrix at the same time to avoid this "aging effect". If the failures stop appearing without the fault having been fixed, it is better to still report the component as faulty for some sufficiently long time to acknowledge the problem and deal with it.

Figure 14.3 shows the health estimated by the SFL algorithm for a faulty component yielding a failure at different times, when all spectra are kept. The later the failure surfaces, the slower is the convergence of health. From the point of view of the system maintainer, when a given failure happens, the algorithm output should be identical independently from the time system has been running previously.

To overcome this problem, we defined the *sliding window* policy. Spectra that are older than a given age are discarded. In practice, as the SFL counters are accumulated, we approximate the window by decomposing it into a fixed number of small periods. An array of counters allows to keep track of the SFL counters for each period. When the current period is over, the oldest set of counters is discarded and replaced by a new set for the next coming period. The global counters are replaced by an addition of the counters for each available period. In our implementation we used 32 sub-periods, which appeared to be of sufficient precision.

The ideal window size (leading to stable health values) depends on the frequency of the monitors generating observations and the frequency of failures being detected. In our experiments, we observed that short sliding windows yield a relatively high diagnostic cost and unstable output over time, because they are too small to contain

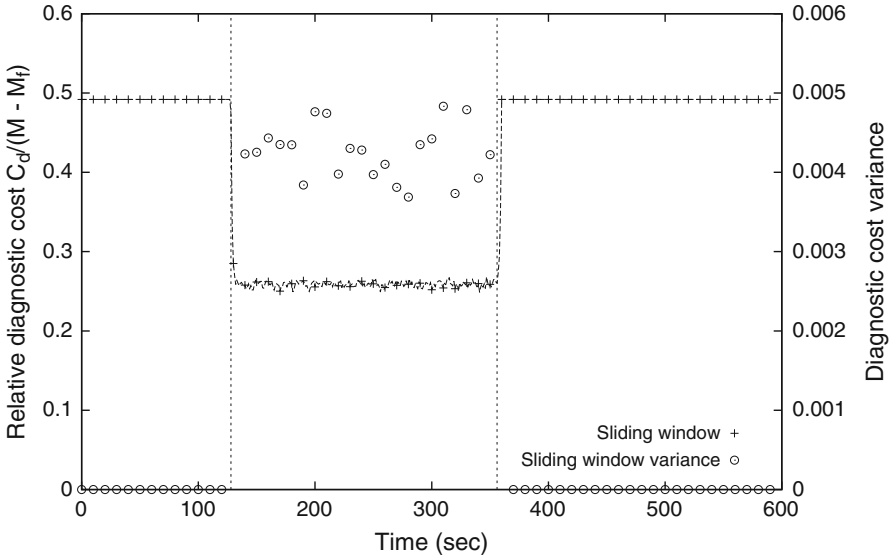


Fig. 14.4 Average diagnostic cost on simulated systems with a sliding window policy of length of 4 s (component fails in the period 128–356 s, *dotted lines*)

enough test outcomes for adequate diagnosis. When the size of the window is extended, it reaches a point where the diagnostic performance does not improve anymore. Increasing further the length solely leads to a bigger latency to react to the failure disappearance. The size of the window after which the diagnosis presents no more noise depends on the frequency at which the failures are detected. We observed that the minimum efficient window size depends on the amount of *fail* outcomes that are captured. The amount of *pass* outcomes is usually far superior, so it is not a bottleneck. We observed that if a window is long enough to contain at least approximately 10 *fail* outcomes, it is sufficient to keep a good quality diagnosis.

Therefore, we recommend selecting a size of the window which is sufficiently long to receive many monitor outcomes. The main restriction on the maximum length is to ensure a fairly fast reaction in terms of health. The window size can be set as the minimum duration for which a single failure occurrence should be seen when looking at the diagnosis.

In order to observe the effect of applying the sliding window policy, we simulate a system where a new failure is seen, lasts for 228 s, and disappears. Figure 14.4 shows the average diagnostic cost when a window size of 4 s is applied. Approximately 4 s after the first failure appears the diagnostic cost reaches its minimum. Similarly, the diagnostic reacts within seconds to the disappearance of the failures. As the failure frequency is high enough that a window contains several *fail* observation outcomes, the diagnostic variance is relatively low. Increasing the window size would stabilize even further the diagnostic over the period that the failure happens.

14.5 Implementation of Online Fault Localization

There are many ways to implement the proposed techniques. We outline here two different implementation approaches that we have carried out successfully. The first approach is centralized, while the second one is metadata-based.

14.5.1 Centralized Approach

A first implementation approach, which we have used in our Atlas framework,[3] relies on a centralized spectrum recording. Its architecture can be broken down into four parts. An example system using such an approach is displayed in Fig. 14.5. For each architectural part, we will refer to this example. The *coverage manager* component takes care of keeping the coverage spectrum of the system. In the example, this component is represented by the box of the same name. The spectrum is reset periodically according to the randomized time frame policy as described previously. By request from the coverage instrumentation part (discussed later), it sets a position in the spectrum to indicate a specific component has been covered. When a monitor sends a new observation, the coverage manager receives this observation, attaches the current spectrum, and forwards it to the SFL component.

The *SFL* component (which is represented by the *SFL* box in the example) receives every monitor observation and adds it to the matrix according to the sliding window policy. In practice, a whole matrix is not needed, only a set of accumulators, which permits a fast processing. Running at a slower frequency, the ranking of the faulty components is computed. This might require a noticeable amount of processing power, but it can be done independently from the rest of the system, even offloaded to separate hardware.

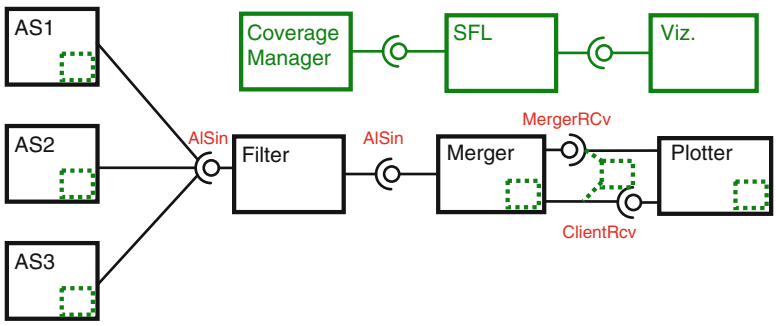


Fig. 14.5 Architecture of the case study system, which is based on the centralized approach

[3] http://swerl.tudelft.nl/bin/view/Main/Atlas

Every functional component of the system is instrumented to report whenever one of its methods is called. In the example, every component part of the core functionality is instrumented. We use Aspect Orientation [6] and Java self-reflection to apply the same code to all the components. This allows to dynamically instrument any component, even when provided by a third party or added a posteriori. However it brings a high overhead to each method call. A static approach, such as found in many code profilers, would likely be more efficient.

Finally, the behavior of the system is validated by a set of monitors, positioned at various places between or around the normal components. Monitors are represented as dash boxes in the example. Every monitor observation, both *fails* and *passes*, is transmitted to the *coverage manager*. A monitor can be replacing what would traditionally be a warning or error check, or can be more complex piece of code which validates the outputs of a component compared to the previously received input (based for instance on a state machine). Watchdogs, which detect the loss of service provided by a component can also be implemented as monitors but care should be taken to report in case of failure not the actual spectrum, but the spectrum that would be expected (so that SFL can point towards the non-responding part of the system).

14.5.2 Metadata-Based Approach

The centralized approach is easy to implement and efficient on systems where all components can access the *coverage manager* with a low latency and where communications have a low overhead. In systems where components are running on physically separate nodes such as systems of systems, or systems which are message-based, it might be more efficient to use a different approach, based on metadata. All data transmitted between components is associated to metadata that contains a coverage spectrum indicating all the components used to generate this data. Every time an output is generated, its metadata must be set, based on the metadata of the inputs. Note that computing the spectrum might be difficult in some cases where many inputs are used. There is still a central component for the coverage, but it is only accessed to request a position in the spectrum when a component initializes. Monitors work similarly to the previous implementation approach except that the spectrum associated to an observation comes from the metadata of the output which is validated. This observation can then be sent directly to the SFL component.

To handle dynamic system architectures, where components can be added and removed online, the coverage spectrum needs to have positions updated when there is a change. We treat this requirement by having the *coverage manager* assign positions to new components. When a component is removed the positions which were assigned to it can be reused, once a certain delay corresponding to the time window length has passed.

14.6 Case Study

All techniques for realizing online fault localization with SFL have been introduced. Synthetic system simulations were used to compare different techniques to each other on a large set of systems. In the following, we evaluate our contributions on a real system. The main goal is to validate the techniques on practical ground, and verify that the simulated systems behave similarly to the actual ones.

The surveillance system that we use as case receives information broadcasts from ships, called *AIS messages* [4] (cf. Chap. 1), and it processes them in order to form a situational picture of a maritime area. The system is made of Atlas components in Java. In total it is comprised of 63 methods (the granularity of the SFL) for the core functionality with an average of 10 lines of Java code each.

The monitoring infrastructure comprises four monitors, each of them guarding different functional and non-functional aspects of the system. Coverage of components is recorded through an ad-hoc Java aspect, as described in Sect. 14.5.

14.6.1 Injected Faults

We simulate two types of faults, loss of data between components (for example due to reset of the component, or unstable connection), and software faults caused by the functionality. Data loss faults are simulated through intermittent connection drops between two components. Software faults are introduced through mutations in the original code (a set of 100 mutants which was created with μJava [7] and manually verified to affect the behavior of the system). For each of the mutation faults, the system was executed for 1 h with the recorded input, producing approximately 100,000 monitor outcomes in total. A posteriori, it is then possible to determine the diagnostic cost at each moment in time. Twelve mutations lead to early system crash (within a minute) and are sorted out (in practice, such a bug would be directly noticed and investigated offline). Fifty five mutations have faults not detected by the monitors, leaving 33 configurations with detectable faults.

14.6.2 Results

The average C_d for *transactional* and *randomized time frame* observation strategies is presented in Fig. 14.6. The *# systems* indicate the number of systems still running at a given time. It decreases whenever a system crashes or stop responding. The SFL algorithm uses a sliding window of 5 min, in order to ensure a good quality of the diagnosis while keeping a relatively fast reaction to any fault correction.

The diagnostic cost C_d, which starts at 0.5, decreases until it reaches some relatively constant value after around a minute. This is similar to the results seen in the simulations (Fig. 14.2). After 5 min of execution (i.e., the length of the sliding window), all C_d graphs increase. This is because some faults lead to failures only at

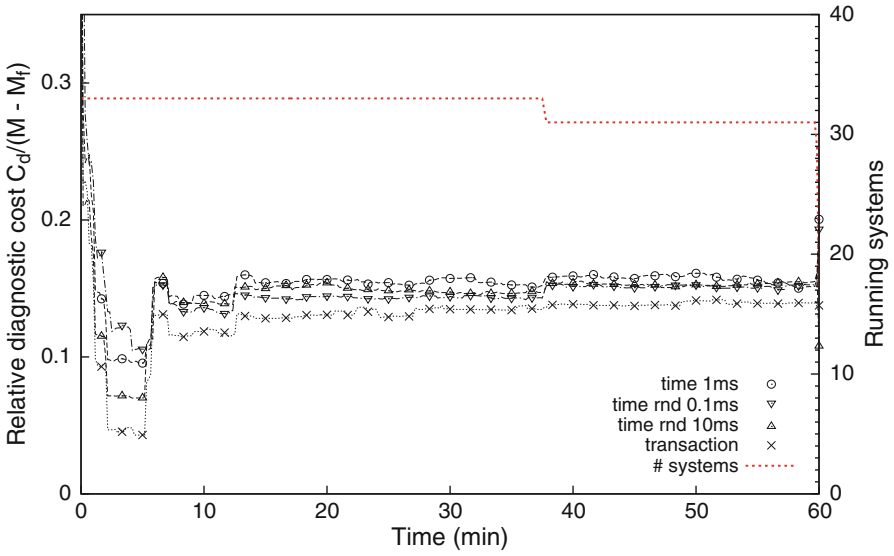


Fig. 14.6 Average diagnostic cost (33 configurations) over the time for three different observation policies

initialization, i.e., they are located in components only used at that time. When these first spectra are removed from the matrix (through the sliding window) the SFL loses information about their location, and assumes a better health, leading typically to a $C_d = 0.5$. Hence, the average C_d increases.

As in the simulation, the *transactional* observation performs best, with an average $C_d = 0.14$. The *time frame* observation yields its best results with 1 ms ($C_d = 0.16$). A shorter or longer period impairs the results, leading to C_d around 0.3 (not shown in the figure to improve readability). This suggests that observation periods of 1 ms are optimal for this system. The *randomized time frame* observation performed equally well as the best fixed time period, for all periods tried between 0.1 and 100 ms.

In our case, transactional observation provides the best results. Nevertheless, this requires that for each monitor the information about which components are observed is known and correct. Otherwise, a randomized time frame allows diagnosis with comparable quality, with only a rough estimation of the processing time needed.

This case study demonstrates the feasibility of online fault localization using the SFL technique in a system inspired by industry. With a diagnostic cost ranging on average below 0.2 just after a minute, it also shows that fault localization is able to point into the right direction for identifying problematic components in software systems. Of course, this works only if residual defects can be detected by the monitors. The fact that the results are relatively similar to the results obtained by simulation suggests that the model employed for the simulation is representative of this real case.

14.7 Conclusions

In complex, large, and evolving systems, such as systems of systems, bugs are unavoidable. Whenever problems appear while a system is in operation, they should be handled and taken care of as quickly as possible. One way to reduce this time is to help the maintainer to locate the error. In this chapter, we have presented an approach for realizing online spectrum-based fault localization to be used in software systems. Techniques have been introduced to obtain a significant spectrum for the SFL algorithm in order to yield good diagnoses. Furthermore, the diagnostic outcome is ensured to be always relevant to the current state of the system by using a time window on the spectrum matrix.

Our work has been validated first by simulation of a large set of randomly generated systems, and through a case study with a system inspired by industry. The diagnostic results on a set of mutated systems corroborate the results of the simulation and confirm that, with our techniques, SFL and monitoring can be applied successfully to online fault localization.

When implementing fault localization for online usage in a system, a lot of care should be taken to keep the performance overhead of the coverage instrumentation as low as possible. Otherwise, this method could slow down considerably the system's execution. Moreover, one should be aware that the quality of the diagnosis is highly dependent on the monitors in the system. Without good detection of the failures, it is impossible to locate the fault. In general, many simple monitors are better than few complex ones. It should also be ensured that the monitors are placed relatively uniformly all over the system's architecture.

Acknowledgements This research has been carried out as a part of the POSEIDON project at Thales under the responsibilities of the Embedded Systems Institute (ESI). This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

References

1. Abreu R, Zoeteweij P, van Gemund AJC (2007) On the accuracy of spectrum-based fault localization. In Proceedings of the testing: academic and industrial conference practice and research techniques – MUTATION, Washington, DC, USA, Aug 2007. IEEE Computer Society, pp 89–98
2. Abreu R, Zoeteweij P, van Gemund AJC (2009) Spectrum-based multiple fault localization. In ASE '09: proceedings of the 2009 IEEE/ACM international conference on automated software engineering, Washington, DC, USA. IEEE Computer Society, pp 88–99
3. Cleve H. Zeller A (2005) Locating causes of program failures. In ICSE '05: proceedings of the 27th international conference on software engineering, St. Louis, MO, USA, May 2005. ACM Press, pp 342–351
4. International Telecommunications Union (2001) Technical characteristics for a universal shipborne Automatic Identification System using time division multiple access in the VHF maritime mobile band. Recommendation ITU-R M.1371-1

5. Jones JA, Harrold MJ, Stasko J (2002) Visualization of test information to assist fault localization. In ICSE '02: proceedings of the 24th international conference on software engineering, Orlando, FL, USA, May 2002. ACM, pp 467–477
6. Kiczales G, Lamping J, Menhdhekar A, Maeda C, Lopes C, Loingtier J-M, Irwin J (1997) Aspect-oriented programming. In: Akşit M, Matsuoka S (eds) Proceedings European conference on object-oriented programming, vol 1241. Springer, Berlin/Heidelberg/New York, pp 220–242
7. Ma Y-S, Offutt J, Kwon YR (2005) Mujava: an automated class mutation system: research articles. Softw Test Verif Reliab 15:97–133
8. Piel E, González-Sánchez A, Gross H-G, van Gemund AJC (2011) Spectrum-based health monitoring for self-adaptive systems. In 5th IEEE international conference on self-adaptive and self-organizing systems (SASO'11). IEEE Computer Society, Washington, DC, USA, pp 99–108
9. Slane D (2009) Fault localization in in vivo software testing. Master's thesis, Bard College, Massachusetts, USA
10. Williams BC, Ingham MD, Chung SH, Elliott PH (2003) Model-based programming of intelligent embedded systems and robotic space explorers. In IEEE special issue on modeling and design of embedded software, IEEE Computer Society, Washington, DC, USA, pp 212–237
11. Zoeteweij P, Abreu R, Golsteijn R, Arjan JC van Gemund (2009) Spectrum-based fault localization in practice. In Mathijsen R (ed) Trader: reliability of high-volume consumer products, Eindhoven, The Netherlands. Embedded Systems Institute, pp 113–124

Chapter 15
Prioritizing Tests for Fault Localization

Alberto Gonzalez-Sanchez, Éric Piel, Rui Abreu, Hans-Gerhard Gross, and Arjan J.C. van Gemund

15.1 Introduction

Devising appropriate quality assurance strategies are amongst the most obvious challenges in building and evolving large-scale Maritime Safety and Security Systems of Systems (MSS SoS), given the large number of different systems contributing to them. Systems of systems evolve dynamically at runtime. Systems can join or leave the system of systems, meaning that offered services may vary in terms of functionality, as well as quality. When a system joins or leaves the system of systems, the other systems may have to be reconfigured to take advantage of new services and improved quality of service, or they may have to be notified that services are degraded. This process should be mostly seamless for the system operators and should be executed within a short time, without any major disruption of the rest of the system of systems.

By their very nature, systems of systems are large-scale systems, formed by a large number of systems and sub-systems. After each runtime evolution, the quality assurance of the integrated system of systems has to be verified again. It is therefore necessary to devise an appropriate verification strategy that not only achieves this goal, but also *minimizes* the cost of checking after each modification. Re-verification must be as little disruptive as possible for the running configuration and the latency between the moment a reconfiguration is requested and the moment it is accepted and deployed must be minimal.

A. Gonzalez-Sanchez (✉) • É. Piel • H.-G. Gross • A.J.C. van Gemund
Department of Software Technology, Delft University of Technology, Delft, The Netherlands
e-mail: a.gonzalezsanchez@tudelft.nl; e.a.b.piel@tudelft.nl; h.g.gross@tudelft.nl; a.j.c.vangemund@tudelft.nl

R. Abreu
Department of Informatics Engineering, University of Porto, Porto, Portugal
e-mail: rui@computer.org

P. van de Laar et al. (eds.), *Situation Awareness with Systems of Systems*,
DOI 10.1007/978-1-4614-6230-9_15,

In the context of MSS, the system of systems will be formed by software systems whose quality assurance will be performed by means of runtime testing and fault localization. Software testing and fault localization (debugging) is a time-consuming but rather important task for improving software reliability. Hence, testing and debugging are an integral part of the quality assurance process of these systems.

In reliability terminology, a *fault* is a defect in the system, for example a programming error, or a contract violation in composing software systems. A fault will produce *errors* when the system containing the fault is used. Errors are inconsistent internal program states (variable values). An error may stay unnoticed for very long, if it is never observed as an output by the users of the system. When an error affects the system in a way that the output (data or behavior) deviates from what the users expect, this event is called a *failure*.

The ultimate goal of testing is to prove the absence of faults for all possible inputs. Since exhaustive testing is hardly ever possible, testing typically aims at *detecting the presence of faults*, by trying to produce a failure (the external manifestation of a fault). The longer the system does not produce any failures, the more certain testers are that the system contains a low number of faults with no or limited impact. To optimize the cost of detecting faults, a reasonable option is to execute only the tests that have a high potential of failure first, followed by tests with lower failure potential, etc. This technique is known as *test prioritization*.

Once failures have been produced, the faults that produced them have to be localized. Optimizing the cost of fault localization is a whole research area on its own, which has recently attracted great interest. The goal of (automated) fault localization is to reduce the effort that the software developers have to invest in manually localizing the fault. Automated fault localization employs algorithms that produce a list of likely fault candidates by using information from the system execution (either test cases or normal system execution) and a model of the system.

Unfortunately, it has recently been shown that tests that aim for very good fault detection can have a negative impact on automated fault localization techniques canceling the savings in fault detection cost by an increase in fault localization cost [4, 6]. In this chapter we explore the causes for this cost increase, and propose a shift in the goal of testing, focusing on techniques that achieve much better fault localization performance, while maintaining a good fault detection performance.

15.2 Software Fault Localization

The objective of fault localization is to pinpoint the precise location of a number of faults in a program (bugs) by observing the program's behavior given a number of tests. Although there is a large number of different fault localization techniques, our work is based on spectrum-based fault localization (SFL).

Table 15.1 Example program and inputs for test prioritization and fault localization

		Tests							
	Program: character counter	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
	`function count(char * s) {`								
	`int let, dig, other, i;`				A				
c_1	`while(c = s[i++]) {`	1	1	1	1	1	1	1	1
c_2	`if ('A'<=c && 'Z'>=c)`	1	1	1	1	1	1	0	1
c_3	**`let += 2; // FAULT`**	1	1	1	1	1	1	0	0
c_4	`elsif ('a'<=c && 'z'>=c)`	1	1	1	1	1	0	0	1
c_5	`let += 1;`	1	1	0	0	1	0	0	0
c_6	`elsif ('0'<=c && '9'>=c)`	1	1	1	1	0	0	0	1
c_7	`dig += 1;`	0	1	0	1	0	0	0	0
c_8	`elsif (isprint(c))`	1	0	1	0	0	0	0	1
c_9	`other += 1;`	1	0	1	0	0	0	0	1
c_{10}	`printf("%d %d %d\n",` `let, dig, other);}`	1	1	1	1	1	1	1	1
	Test case outcomes	F	F	F	F	F	F	P	P

Let us consider the faulty program in Table 15.1. We provide a test suite with eight tests that provide full statement coverage. The inputs that SFL algorithms require are represented in the figure. These are:

- A vector of test outcomes, where the value of each outcome corresponds to 'F' (fail) if the test failed, and 'P' (pass) otherwise.
- A coverage matrix, A, with as many rows as fault sources (e.g., components, interfaces, methods, statements), and as many columns as tests in our test pool. If test t_i involves component c_j, then the corresponding element $a_{ij} = 1$ in A.

Each column in A is called "spectrum", hence the name spectrum-based fault localization for this family of techniques. SFL techniques can be divided into two main groups: *similarity coefficients*, and *Bayesian reasoning*.

Similarity coefficients measure the likelihood that a component is at fault in terms of the statistical similarity between a statement's test coverage and the observed test outcomes, a technique inspired by clustering and machine learning techniques. A great advantage of similarity coefficients is their ultra-low computational complexity compared to Bayesian or model-based diagnosis. Despite their lower diagnostic accuracy [1] similarity coefficients have, therefore, gained much interest.

Bayesian diagnosis uses a very high level model of the system in logic first principles. This model is first solved to obtain a set of explanation candidates. These candidates are then sorted according to a probabilistic approach that uses Bayes rule to update the fault probability of each statement. An advantage of Bayesian diangosis with respect to similarity coefficients is that Bayesian diagnosis has much better accuracy when there are multiple faults in the system.

Both Bayesian diagnosis and similarity coefficients take the same kind of inputs and produce the same kind of output, therefore they are interchangeable depending on the current needs (e.g., speed vs. accuracy).

15.3 Problem Statement

In the previous section, two processes have been identified that are critical to the quality assurance of systems of systems: (1) fault detection, and (2) fault localization. These two processes are represented in Fig. 15.1. Each of these processes has costs attached to it:

1. **Fault detection costs:** these are the costs related directly to detecting the presence of faults. Since in the context of our project we detect faults by testing, testing cost (i.e., the execution time of each test) is the main cost source.
2. **Fault localization costs:** these are the costs associated with the localization of faults in the system. First, testing cost associated to the additional tests that may be executed to gather more information about the error state of the system. Second, the inspection cost (manual effort) incurred when developers have to inspect each of the suspect components until the faults are found.

The overall cost of the system's quality assurance has to be minimized, while maximizing the confidence in the integrated system. A low fault detection cost and a low fault localization cost are antagonistic goals. Good fault detection will cause poor fault localization [4], since both goals depend on test cost but the kind of test that is good for detection is not so good for localization. This is represented in Fig. 15.1 by a snake line between the two.

15.3.1 Why Is Lowering Fault Detection Cost a Bad Idea?

Test prioritization techniques that lower detection cost, are founded on the reasoning that the sooner failures are found, the sooner fault localization (e.g., debugging) can commence. Test prioritization techniques try to execute first those tests with a higher probability of producing a failure, according to an heuristic. The most commonly used heuristics are based on test coverage. The more parts of the system are used in a test, the higher the probability that the test will fail. This idea is implemented in the "additional statement coverage" (ADD-ST) and the "additional fault exposure potential" (ADD-FEP) heuristics [10].

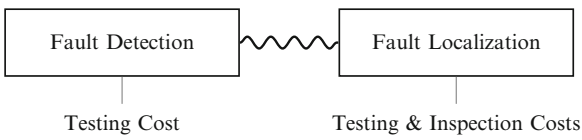


Fig. 15.1 Goals and associated costs

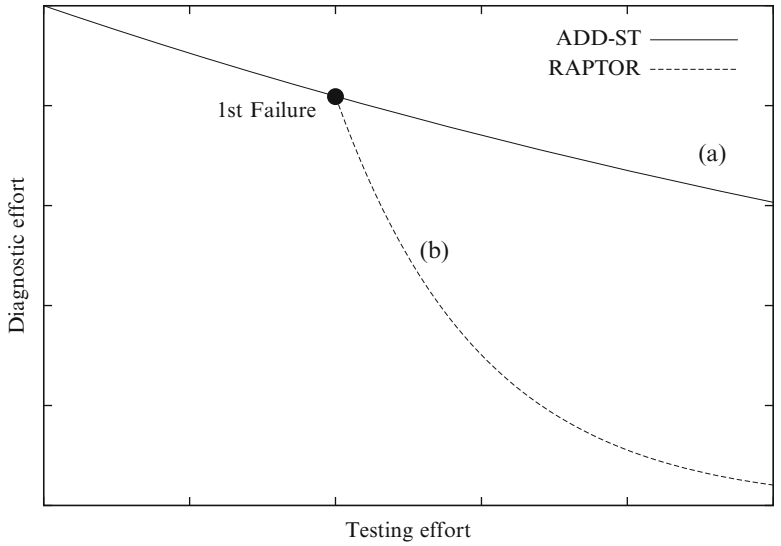


Fig. 15.2 Test prioritization scenarios

Coverage maximization heuristics are a perfectly valid reasoning when our goal is exclusively reducing fault detection cost. However, the kind of test that is required for a good fault localization is different than the one required for good fault detection. SFL algorithms depend on having good *variability* in the observations (i.e., in the spectra). Heuristics such as ADD-ST or ADD-FEP tend to select test cases with very low variability, thus the fault localization information that each executed test provides is very low.

Figure 15.2 illustrates the situation, showing the trade off between fault detection cost and fault localization cost, in two possible scenarios. Scenario (*a*) represents a situation in which a prioritization heuristic for fault detection such as ADD-ST or ADD-FEP is used during testing. If after detecting the fault, we continue using a fault detection test prioritization, refining the diagnosis will require a large number of tests. Scenario (*b*), on the other hand, represents the improvement when using the RAPTOR *diagnostic* test prioritization technique, which will be described in Sect. 15.4. In this case, the refinement of the diagnosis happens much faster.

15.4 Fault Localization Prioritization

In this section we will present *fault localization prioritization*, a test prioritization approach that takes into account the reduction of fault localization cost to determine the next test case to be executed. Our work is inspired by research in *sequential*

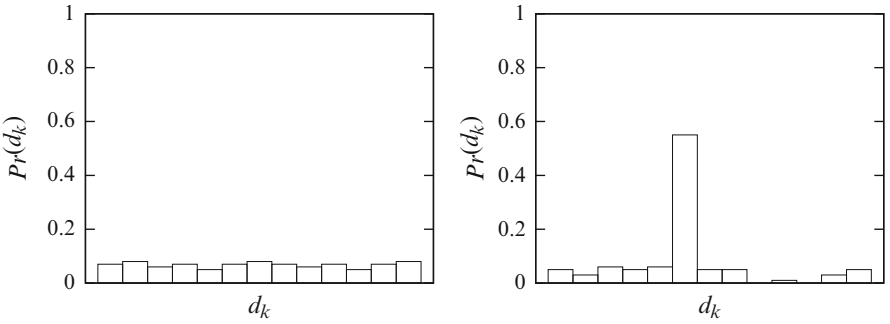


Fig. 15.3 Very ambiguous diagnosis (*left*) and very precise diagnosis (*right*)

diagnosis of hardware systems, where algorithms exist to diagnose systems with permanent [11] and intermittent [9] failures.

In fault localization prioritization, the best tests are those that, at each step, maximize the improvement of the fault localization quality, i.e., the location of the actual fault within the rank of suspects. Since we do not know the faulty component, an approximate indicator, different from localization quality (i.e., an heuristic), has to be used. In general, good diagnoses are those in which one of the candidates has a much larger probability than the rest, i.e, it stands out over all other candidates. Figure 15.3 illustrates what we want to achieve. The horizontal axis represents each of the diagnostic explanations (d_k), and the vertical axis the calculated probability of the explanation being right ($\Pr(d_k)$). The plot on the left shows a poor diagnosis where no clear candidate exists. Our objective is to produce a diagnosis like the one on the right hand side plot.

15.4.1 Solution 1: Information Gain Heuristic

The key for an effective fault localization cost reduction during testing is fault localization *information gain* (IG). The improvement of the quality of the diagnosis can be seen as a reduction of the size entropy of the set of fault candidates [7]. Applying this reasoning, at each test choice in the test sequence, the test yielding the highest average information gain is chosen. Since the outcome of a test is uncertain, either pass or fail, the heuristic must consider the two possible scenarios. Therefore, an important parameter of the IG heuristic is the fault detection capability of tests, i.e., the probability of a test passing or failing.

The information gain heuristic, *IG*, is defined as

$$\mathrm{IG}(D,t_i) = \mathrm{H}(D) - (\Pr(\mathrm{P}) \cdot \mathrm{H}(D_{\mathrm{P}}) + \Pr(\mathrm{F}) \cdot \mathrm{H}(D_{\mathrm{F}})) \tag{15.1}$$

where $\mathrm{H}(D)$ is a measure of the quality of the current diagnosis, usually Shannon's information entropy [7]. The test can fail with probability $\Pr(\mathrm{F})$, and will update the diagnosis to D_{F}. In the case it does not fail, it will update the diagnosis to D_{P}.

Unfortunately, computing information gain is exponentially complex in the number of components of the system when multiple faults can be present. We propose a number of heuristic approximations that reduce the information gain estimation while maintaining good accuracy. We integrate these approximations in a prioritization algorithm called Sequoia (*SEQUencing fOr dIAgnosis*) that employs information gain as heuristic. Detailed information about Information Gain prioritization for systems with only one fault can be obtained in [4]. The Sequoia algorithm for systems where more than one fault may be present simultaneously is described in [3].

15.4.2 Solution 2: Ambiguity Reduction Heuristic

The above information gain heuristic depends heavily on the knowledge of the user about the fault detection ability of tests, which has to be estimated beforehand. Our experiments have shown that the accuracy of IG can be severely affected by errors in the estimation of the fault detection ability. For this reason, we study a low-complexity test prioritization heuristic which does not require any additional input parameters. We integrate this heuristic in a second test prioritization algorithm, dubbed Raptor (*gReedy diAgnostic Prioritization by ambiguiTy grOup Reduction*)

Since the basic working principle of SFL algorithms is analyzing the differences and similarities between the coverage of each of the statements of the system under test (each of the rows in the test coverage matrix described in Sect. 15.2) and the test outcomes, we can use the maximization of these differences to devise a new, simpler fault localization test prioritization heuristic.

Any individual statement belonging to a group of statements with identical rows (signatures) cannot be uniquely identified as faulty. Such a group is termed *ambiguity group* [12]. When no test has been executed, all statements belong to the same ambiguity group. Each test that is executed (i.e., added to the coverage matrix) breaks each ambiguity group into two smaller ambiguity groups, one corresponding to the statements in the ambiguity group that are covered by the test, and one corresponding to the statements that are not covered. Once all the tests are executed, the example system in Table 15.1 has two ambiguity groups: $\{1,10\}$ and $\{8,9\}$.

The *ambiguity reduction* heuristic is defined as the difference in the average size of each group caused by appending one more test to the test matrix, according to

$$\mathrm{AR}(A,t_i) = G(AG(A)) - G(AG(A||t_i)) \tag{15.2}$$

where G represents the ambiguity of metric, the function AG returns the set of ambiguity groups of a test matrix, and the $||$ operator adds test t_i to the coverage matrix A. Using this heuristic has the advantage that it does not require knowing the test outcomes, and does not require estimation of failure probabilities.

15.5 Empirical Evaluation

We compared the performance of Sequoia and Raptor to already existing test prioritization techniques such as ADD-ST and ADD-FEP. We also compared with randomly chosen tests. More information on these methods can be found in [4].

We performed our experiments using the well-known Siemens benchmark set [5], as well as the `flex`, `grep`, `gzip`, `sed`, and `space` programs (obtained from SIR [2]). The Siemens suite is composed of seven programs. Every program has a correct version, and a set of test inputs is also provided, which were created with the intention of providing full statement and branch test coverage. Table 15.2 provides more information about the programs used in the experiments, where *LOC* corresponds to the number of lines of code, and *N* is the number of tests available.

We compare fault localization prioritization effectiveness in terms of fault localization effectiveness [4]. Our experiments showed that Sequoia's effectiveness when combined with similarity coefficients appears to be extremely poor, in contrast with its very good performance when using Bayesian fault localization. This is caused by the fact that similarity coefficients do not exploit all the information available. Table 15.3 contains the results for Bayesian fault localization.

Our results also show how Raptor and Sequoia consistently provide an improvement in the efficiency for all programs with only one fault, and is the best in all cases but one (`tcas`). The results also highlight the fact that the IG heuristic used in Sequoia is largely affected by errors in the input parameter estimations, whereas Raptor is not.

On average, both techniques provide 54% reduction of fault localization cost with respect to randomly selected tests in the single-fault case, and 45% in the multiple-fault case. When compared to the next best technique, they provides 38% reduction in the single fault case on average, and 15% reduction in the multiple fault case. In the best case, cost reductions of up to 80% can be obtained (`schedule2` and `gzip`). Additional analyses and detailed result tables with statistical tests can be found in [3].

Table 15.2 Programs used for evaluation

	Program name	LOC	N	Program type
`pt`	`print_tokens`	539	4,130	Lexical analyzer
`p2`	`print_tokens2`	489	4,115	Lexical analyzer
`re`	`replace`	507	5,542	Pattern recognition
`sc`	`schedule`	397	2,650	Priority scheduler
`s2`	`schedule2`	299	2,710	Priority scheduler
`tc`	`tcas`	174	1,608	Altitude separation
`ti`	`tot_info`	398	1,052	Information measure
`sp`	`space`	9,126	500	ADL parser
`gz`	`gzip`	6,708	211	Compressor
`se`	`sed`	9,014	184	Stream editor
`gr`	`grep`	13,287	809	String matching
`fl`	`flex`	14,194	107	Lexer generator

Table 15.3 Effectiveness in terms of Bayesian diagnosis

	Bayesian diagnosis, 1–3 faults					
	RND	ART	ADDST	FEP	RAPTOR	SEQUOIA
pt	5.71	6.37 11%	25.69 350%	27.38 379%	**3.31 −42%**	5.49 −4%
p2	8.56	6.51 −24%	21.09 146%	27.77 224%	**3.81 −55%**	5.74 −33%
re	10.95	8.84 −19%	10.36 −5%	11.79 8%	**5.08 −54%**	**5.12 −53%**
sc	13.07	9.25 −29%	9.72 −26%	21.83 67%	**4.95 −62%**	5.81 −56%
s2	11.78	9.07 −23%	9.79 −17%	19.81 68%	3.06 −74%	**2.70 −77%**
tc	4.44	**3.94 −11%**	13.16 196%	14.88 235%	7.66 72%	12.53 182%
ti	3.00	2.20 −27%	3.65 22%	10.06 235%	**1.31 −56%**	4.43 48%
sp	12.48	10.04 −20%	7.48 −40%	**6.64 −47%**	7.03 −44%	8.33 −33%
gz	14.07	7.14 −49%	**2.03 −86%**	**1.95 −86%**	**2.64 −81%**	3.69 −74%
se	10.88	6.49 −40%	**3.40 −69%**	4.62 −58%	**3.39 −69%**	3.82 −65%
gr	10.78	8.22 −24%	**3.68 −66%**	3.77 −65%	4.87 −55%	6.52 −40%
fl	9.90	5.04 −49%	3.63 −63%	4.09 −59%	3.41 −66%	**3.26 −67%**

15.6 Lessons Learned and Practical Application

Many factors influence the decision of what test prioritization technique to use in a practical setting, to test for failures or faults from the first test, and to use either Raptor, Sequoia, or an alternative.

Wait for the first failure? The fault detection cost, i.e., the test effort it takes to detect the first failure, can be slightly increased when using fault localization test prioritization. The first question is whether to perform fault localization test prioritization from the first test, or only after the first failing test, since Raptor and Sequoia have a moderately decreased fault detection capability. The most conservative solution, especially if the introduction of a fault is unlikely, is to initially use ADD-ST or ADD-FEP, and switch to fault localization test prioritization if a failure occurs. However, if the probability of a fault is high (some projects can have up to 70% probability [8]), it is better to start directly with fault localization test prioritization since the tests executed until the first failure is produced provide also the most valuable fault localization information. The increased test cost (the first failure occurs slightly later) is greatly outweighed by the much more substantial gains in terms of fault localization cost reduction. Therefore, even if it takes slightly longer to *detect* a fault, the reduction in the cost required to precisely *locate* the fault greatly compensates for this.

Raptor or Sequoia? If using low-cost similarity coefficients, Raptor is the clear choice. However, since Sequoia can potentially provide better results than Raptor combined with Bayesian fault localization, it is necessary to consider the advantages and disadvantages of each technique.

1. **Computational cost:** Even though Raptor and Sequoia have a similar order of complexity, Raptor is based on fast bit-wise operators, whereas Sequoia is based on expensive floating point operations, and requires expensive prior parameter determination. Furthermore, it must be taken into account that Sequoia must perform online prioritization, as the tests are executed, potentially introducing a significant overhead.
2. **Parameter estimation quality:** the heuristic used in Sequoia can be severely affected by poor input parameter estimations. If the estimations are not reliable, e.g., they have a large variance, one should not consider Sequoia.
3. **Ambiguity and non-uniformity:** a test matrix with very few and very large ambiguity groups, or a system with extremely biased fault distribution can cause problems to a static algorithm like Raptor. If the input parameter estimation quality is good enough, one could opt for Sequoia. Otherwise, the best would be to opt for a random approach.

15.7 Conclusion

Runtime testing and fault localization are a developer's main means of performing the quality assurance process of dynamically evolving and high-availability systems, such as Systems of Systems or Service Oriented Architectures.

At the core of this chapter, we have investigated the relationship and trade-off between fault detection and fault localization. Our approach is a departure from the commonplace fault detection-centric paradigm towards a fault localization-centric one, where during testing test cases are not selected by their fault detection potential, but for their potential to improve the current diagnosis. In particular, we have investigated a test selection method based on the information gain heuristic. Although this method provides theoretically the most optimal solution, we showed that in practice the method's performance is not always good due to estimation errors in the input parameters. Motivated by this fact, we investigated a simpler, offline test selection heuristic that achieves comparable performance and does not suffer from these problems. Besides its comparable performance, the offline algorithm does not incur in additional overhead during testing. Furthermore, its computational complexity is much lower than the online algorithm.

Acknowledgements This research has been carried out as a part of the POSEIDON project at Thales under the responsibilities of the Embedded Systems Institute (ESI). This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

References

1. Abreu R, Zoeteweij P, van Gemund AJC (2009) Spectrum-based multiple fault localization. In: ASE '09: proceedings of the 2009 IEEE/ACM international conference on automated software engineering. IEEE Computer Society, Washington, DC, pp 88–99

2. Do H, Elbaum S, Rothermel G (2005) Supporting controlled experimentation with testing techniques: an infrastructure and its potential impact. Empir Softw Eng 10(4):405–435
3. González-Sánchez A, Abreu R, Gross H-G, van Gemund AJC (2011) Spectrum-based sequential diagnosis. In: 25th international conference on artificial intelligence (AAAI'11), pp 189–196, 2011. http://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/download/3565/3847.
4. González-Sánchez A, Piel E, Abreu R, Gross H-G, van Gemund AJC (2011) Prioritizing tests for software fault diagnosis. Software 41(10):1105–1129
5. Hutchins M, Foster H, Goradia T, Ostrand T (1994) Experiments of the effectiveness of dataflow- and controlflow-based test adequacy criteria. In: ICSE '94: proceedings of the 16th international conference on software engineering. IEEE Computer Society Press, Los Alamitos, pp 191–200
6. Jiang B, Zhang Z, Tse TH, Chen TY (2009) How well do test case prioritization techniques support statistical fault localization. In: COMPSAC '09: proceedings of the 2009 33rd annual IEEE international computer software and applications conference. IEEE Computer Society, Washington, DC, pp 99–106
7. Johnson RA (1960) An information theory approach to diagnosis. In: Symposium on reliability and quality control, Washington, DC. DOI: http://dx.doi.org/10.1109/IRE-PGRQC.1960.5007263
8. Kim S, Whitehead EJ, Zhang Y (2008) Classifying software changes: clean or buggy? IEEE Trans Softw Eng 34(2):181–196
9. Raghavan V, Shakeri M, Pattipati K (1999) Test sequencing algorithms with unreliable tests. IEEE Trans Syst Man Cybern Part A 29(4):347–357
10. Rothermel G, Untch RJ, Chu C (2001) Prioritizing test cases for regression testing. IEEE Trans Softw Eng 27(10):929–948
11. Shakeri M, Raghavan V, Pattipati KR, Patterson-Hine A (2000) Sequential testing algorithms for multiple fault diagnosis. IEEE Trans Syst Man Cybern Part A 30(1):1–14
12. Stenbakken GN, Souders TM, Stewart GW (1989) Ambiguity groups and testability. IEEE Trans Instrum Meas 38(5):941–947

Appendix A
POSEIDON Project Partners

This appendix provides an overview of the organisations that have participated in the POSEIDON project.

- Embedded Systems Institute, P.O. Box 513, 5600 MB Eindhoven, The Netherlands.
- Thales Nederland B.V., P.O. Box 42, 7550 GD Hengelo, The Netherlands.
- Noldus Information Technology B.V., P.O. Box 268, 6700 AG Wageningen, The Netherlands.
- Embedded Software Group, Department of Software Technology, Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands.
- Architecture of Information Systems, Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands.
- Security and Embedded Networked Systems, Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands.
- Algorithms and Visualization, Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands.
- Informatics Institute, Faculty of Science, Universiteit van Amsterdam, P.O. Box 94216, 1090 GE Amsterdam, The Netherlands.
- Tilburg center for Cognition and Communication (TiCC), School of Humanities, Tilburg University, P.O. Box 90153, 5000 LE Tilburg, The Netherlands.
- Business Web and Media, Department of Computer Sciences, Faculty of Sciences, VU University Amsterdam, De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands.

P. van de Laar et al. (eds.), *Situation Awareness with Systems of Systems*,
DOI 10.1007/978-1-4614-6230-9, © Springer Science+Business Media New York 2013

Appendix B
POSEIDON Publications

1. W. M. P. van der Aalst, A. J. Mooij, C. Stahl, and K. Wolf. Service Interaction: Patterns, Formalization, and Analysis. In M. Bernardo, L. Padovani, and G. Zavattaro, editors, *Advanced Lectures of the 9th Int. School on Formal Methods for Web Services*, volume 5569 of *Lecture Notes in Computer Science*, pages 42–88. Springer, 2009.
2. C. J. van Aart, B. J. Wielinga, and W. R. van Hage. Mobile Cultural Heritage Guide: Location-Aware Semantic Search. In P. Cimiano and H. S. Pinto, editors, *17th Int. Conf. on Knowledge Engineering and Knowledge Management by the Masses (EKAW'10)*, volume 6317 of *Lecture Notes in Computer Science*, pages 257–271. Springer, 2010.
3. M. L. Bernardi, M. Cimitile, and F. M. Maggi. Model Driven Development of Process-centric Web Applications. In S Hammoudi, M. van Sinderen, and J. Cordeiro, editors, *ICSOFT 2012 – 7th Int. Conf. on Software Paradigm Trends*, pages 340–346. SciTePress, 2012.
4. K. Böhm, S. Etalle, J. den Hartog, C. Hütter, S. Trabelsi, D. Trivellato, and N. Zannone. Flexible Architecture for Privacy-Aware Trust Management. *Journal of Theoretical and Applied Electronic Commerce Research*, 5(2):77–96, August 2010.
5. A. Burattin, F. M. Maggi, W. M. P. van der Aalst, and A. Sperduti. Techniques for A Posteriori Analysis of Declarative Processes. In *16th Int. EDOC Conf. (EDOC 2012)*, 2012.
6. D. Ceolin, P. Groth, and W. R. van Hage. Calculating the Trust of Event Descriptions using Provenance. In S. S. Sahoo, J. Zhao, P. Missier, and J. M. Gomez-Perez, editors, *Second Int. Workshop on the Role of Semantic Web in Provenance Management (SWPM 2010)*, 2010. http://ceur-ws.org/Vol-670/completeSWPM10Proceedings.pdf.
7. D. Ceolin, W. R. van Hage, and W. Fokkink. A Trust Model to Estimate the Quality of Annotations using the Web. In *Web Science Conference (WebSci10)*, April 26–27 2010. http://journal.webscience.org/315.
8. D. Ceolin, W. R. van Hage, W. Fokkink, and G. Schreiber. Estimating Uncertainty of Categorical Web Data. In F. Bobillo *et al.*, editor, *7th Int. Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2011)*, pages 15–26, 2011. http://ceur-ws.org/Vol-778/proceedings.pdf.
9. M. T. M. Emmerich, R. Li, A. Zhang, I Flesch, and P. Lucas. Mixed-Integer Bayesian Optimization Utilizing A-Priori Knowledge on Parameter Dependences. In A. Nijholt, M. Pantic, M. Poel, and G. H. W. Hondorp, editors, *Belgian-Dutch Conference on Artificial Intelligence (BNAIC 2008)*, pages 65–72, Enschede, The Netherlands, 2008. University of Twente. http://eprints.eemcs.utwente.nl/13354/.

P. van de Laar et al. (eds.), *Situation Awareness with Systems of Systems*,
DOI 10.1007/978-1-4614-6230-9, © Springer Science+Business Media New York 2013

10. M. van Erp, W. R. van Hage, L. Hollink, A. Jameson, and R. Troncy, editors. *Int. Workshop on Detection, Representation, and Exploitation of Events in the Semantic Web (DeRiVE 2011)*, Bonn, Germany, 2011. http://ceur-ws.org/Vol-779.
11. M. van Erp, V. Malaisé, W. R. van Hage, V. Osinga, and J.M Coleto. Parse and Tag Somali Pirates (Abstract). In *Computational Linguistics in the Netherlands (CLIN 2011)*, page 53, 2011. http://lt3.hogent.be/clin21/book_clin21_final.pdf.
12. I. Flesch and P. J. F. Lucas. Combining Abduction with Conflict-Based Diagnosis. In M. Ghallab, C. D. Spyropoulos, N. Fakotakis, and N. Avouris, editors, *European Conf. of Artificial Intelligence (ECAI 2008)*, pages 807–808. IOS Press, 2008.
13. I. Flesch and P. J. F. Lucas. Comparing GDE and Conflict-Based Diagnosis. In B. Peischl, N. Snooke, G. Steinbauer, and C. Witteveen, editors, *European Conf. of Artificial Intelligence – Workshop on Model-Based Systems (MBS 2008)*, pages 1–6, 2008. http://www.ist.tugraz.at/mbs08/w11.pdf.
14. I. Flesch and P. J. F. Lucas. The Probabilistic Interpretation of Model-Based Diagnosis. In M. Jaeger and T. D. Nielsen, editors, *Workshop on Probabilistic Graphical Models (PGM 2008)*, pages 113–120, 2008. http://pgm08.cs.aau.dk/Misc/pgm08_proceedings.pdf.
15. I. Flesch and E. O. Postma. Simplifying Learning in Non-repetitive Dynamic Bayesian Networks. In C. Sossai and G. Chemello, editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, volume 5590 of *Lecture Notes in Computer Science*, pages 216–227. Springer, 2009.
16. C. Gierds, A. J. Mooij, and K. Wolf. Specifying and Generating Behavioral Service Adapters based on Transformation Rules. Preprint CS-02-08, Institut für Informatik, Universität Rostock, Rostock, Germany, 2008.
17. C. Gierds, A. J. Mooij, and K. Wolf. Reducing Adapter Synthesis to Controller Synthesis. *IEEE Trans. on Services Computing*, 5(1):72–85, 2012.
18. M. Glandrup. Towards Improving Situation Awareness for Operators in the Maritime Domain. In J. Janssens, E. Postma, and J. Hellemons, editors, *Int. Workshop on Maritime Anomaly Detection (MAD 2011)*, page 17, 2011. http://mad.uvt.nl/mad/mad2011-proceedings.pdf.
19. A. González-Sánchez. *Cost Optimizations in Runtime Testing and Diagnosis*. PhD thesis, Delft University of Technology, 2011. http://repository.tudelft.nl/assets/uuid:58b2acc2-e97e-4c2c-b40f-c05c49b5d1fd/thesis.pdf.
20. A. González-Sánchez, R. Abreu, H.-G. Gross, and A. J. C. Gemund. Prioritizing Tests for Fault Localization through Ambiguity Group Reduction. In *26th Int. Conf. on Automated Software Engineering (ASE'11)*, pages 83–92. IEEE, 2011.
21. A. González-Sánchez, R. Abreu, H.-G. Gross, and A. J. C. van Gemund. Spectrum-Based Sequential Diagnosis. In S. Poll, J. de Kleer, I. Roychoudhury, and M. Daigle, editors, *21st Int. Workshop on the Principles of Diagnosis (DX'10)*, pages 55–62. PHM Society, 2010.
22. A. González-Sánchez, R. Abreu, H.-G. Gross, and A. J. C. van Gemund. An Empirical Study on the Usage of Testability Information to Fault Localization in Software. In *26th Int. Symp. On Applied Computing (SAC'11)*, pages 1398–1403. ACM Press, 2011.
23. A. González-Sánchez, R. Abreu, H.-G. Gross, and A. J. C. van Gemund. Spectrum-Based Sequential Diagnosis. In *25th Int. Conf. on Artificial Intelligence (AAAI'11)*, pages 189–196, 2011. http://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/download/3565/3847.
24. A. González-Sánchez, H.-G. Gross, and A. J. C. van Gemund. Modeling the Diagnostic Efficiency of Regression Test Suites. In *Software Testing, Verification and Validation Workshops – Workshop on Testing & Debugging (TeBug'11)*, pages 634–643. IEEE Computer Society, 2011.
25. A. González-Sánchez, E. Piel, R. Abreu, H.-G. Gross, and A. J. C. van Gemund. Prioritizing Tests for Software Fault Diagnosis. *Software: Practice and Experience*, 41(10):1105–1129, 2011.
26. A. González-Sánchez, E. Piel, R. Abreu, H.-G. Gross, and A.J.C. van Gemund. Prioritizing Tests for Software Fault Localization. In *10th Int. Conf. on Quality Software (QSIC'10)*, pages 42–51. IEEE Computer Society, 2010.

27. A. González-Sánchez, E. Piel, and H.-G. Gross. Architecture Support for Runtime Integration and Verification of Component-Based Systems of Systems. In M. Caporuscio, A. Di Marco, L. Mariani, H. Muccini, A. Ploni, and O. Sheory, editors, *23rd IEEE/ACM Int. Conf. on Automated Software Engineering Workshops – Workshop on Automated Engineering of Autonomous and Run-Time Evolving Systems (ARAMIS 2008)*, pages 41–48. IEEE Computer Society, 2008.
28. A. González-Sánchez, E. Piel, and H.-G. Gross. A Model for the Measurement of Runtime Testability of Component-Based Systems. In *IEEE Int. Conf. on Software Testing, Verification, and Validation Workshops*, pages 19–28. IEEE Computer Society, 2009.
29. A. González-Sánchez, E. Piel, and H.-G. Gross. RiTMO: A Method for Runtime Testability Measurement and Optimisation. In *9th Int. Conf. on Quality Software (QSIC'09)*, pages 377–382. IEEE Computer Society, 2009.
30. A. González-Sánchez, E. Piel, H.-G. Gross, and A. J. C. van Gemund. Minimising the Preparation Cost of Runtime Testing Based on Testability Metrics. In *34th Annual IEEE Computer Software and Applications Conference (COMPSAC)*, pages 419–424, 2010.
31. A. González-Sánchez, E. Piel, H.-G. Gross, and A. J. C. van Gemund. Runtime Testability in Dynamic High-Availability Component-Based Systems. In *2nd Int. Conf. on Advances in System Testing and Validation Lifecycle (VALID'10)*, pages 37–42. IEEE Computer Society, 2010.
32. A. González-Sánchez, E. Piel, H.-G. Gross, and A. J. C. van Gemund. A Diagnostic Point of View for the Optimization of Preparation Costs in Runtime Testing. In *1st Workshop on Testing & Debugging (TeBug'11)*, pages 654–660. IEEE Computer Society, 2011.
33. A. González-Sánchez, E. Piel, H.-G. Gross, and A. J. C. van Gemund. A Runtime Testability Metric for Dynamic High-Availability Component-Based Systems. *Journal On Advances in Systems and Measurements*, 4(1&2):122–134, 2011.
34. A. González-Sánchez, E. Piel, H.-G. Gross, and M. Glandrup. Testing Challenges of Maritime Safety and Security Systems-of-Systems. In *Testing: Academic and Industry Conference – Practice And Research Techniques (TAIC PART'08)*, pages 35–39. IEEE Computer Society, 2008.
35. H.-G. Gross, M. Lormans, and J. Tretmans, editors. *Software Integration and Evolution @ Runtime – SINTER '09; An 2009 ESEC/FSE Workshop*. ACM, 2009. http://doi.acm.org/10.1145/1596495.
36. W. R. van Hage, V. Malaisé, and M. van Erp. Linked Open Piracy. In M. van Erp, W. R. van Hage, L. Hollink, A. Jameson, and R. Troncy, editors, *Int. Workshop on Detection, Representation, and Exploitation of Events in the Semantic Web (DeRiVE 2011)*, pages 88–97, 2011. http://ceur-ws.org/Vol-779.
37. W. R. van Hage, V. Malaisé, M. van Erp, and G. Schreiber. Linked Open Piracy. K-CAP 2011 Poster Session, 2011. http://www.few.vu.nl/~wrvhage/pdf/kcapPoster.pdf.
38. W. R. van Hage, V. Malaisé, R. Segers, and L. Hollink. Design and Use of the Simple Event Model (SEM). *Journal of Web Semantics*, 9(2):128–136, 2011.
39. W. R. van Hage, V. Malaisé, G. de Vries, G. Schreiber, and M. van Someren. Combining Ship Trajectories and Semantics with the Simple Event Model (SEM). In *1st ACM Int. Workshop on Events in Multimedia (EiMM'09)*, pages 73–80. ACM, 2009. http://doi.acm.org/10.1145/1631024.1631039.
40. W. R. van Hage, V. Malaisé, G. K. D. de Vries, G. Schreiber, and M. W. van Someren. Abstracting and Reasoning over Ship Trajectories and Web Data with the Simple Event Model (SEM). *Multimedia Tools and Applications*, 57(1):175–197, 2012.
41. W. R. van Hage, G. de Vries, V. Malaisé, G. Schreiber, and M. van Someren. Spatial and Semantic Reasoning to Recognize Ship Behavior (Demo). In *8th Int. Semantic Web Conference (ISWC 2009)*, 2009. http://www.few.vu.nl/~wrvhage/papers/iswc2009demo.pdf.
42. W. R. van Hage, J. Wielemaker, and G. Schreiber. The Space Package: Tight Integration Between Space and Semantics. In *TerraCognita 2009 Workshop*, 2009. http://www.few.vu.nl/~wrvhage/papers/terra09_vanhage.pdf.

43. W. R. van Hage, J. Wielemaker, and G. Schreiber. The Space Package: Tight Integration between Space and Semantics. *Transactions in GIS*, 14(2):131–146, 2010.
44. W.R. van Hage, N. Stash, Y. Wang, and L.M. Aroyo. Adaptation Step-by-Step: Challenges for Real-time Spatial Personalization. In S. Berkovsky *et al.*, editor, *Workshop on Pervasive User Modeling and Personalization (PUMP'10)*, pages 40–47, 2010.
45. W.R. van Hage, N. Stash, Y. Wang, and L.M. Aroyo. Finding Your Way through the Rijksmuseum with an Adaptive Mobile Museum Guide. In L. Aroyo, G. Antoniou, E. Hyvönen, A. ten Teije, H. Stuckenschmidt, L. Cabral, and T. Tudorache, editors, *The Semantic Web: Research and Applications*, volume 6088 of *Lecture Notes in Computer Science*, pages 46–59. Springer, 2010.
46. K. van Hee, A. Mooij, N. Sidorova, and J. M. van der Werf. Soundness-Preserving Refinements of Service Compositions. In Mario Bravetti and Tevfik Bultan, editors, *7th Int. Workshop on Web Services and Formal Methods (WS-FM 2010)*, volume 6551 of *Lecture Notes in Computer Science*, pages 131–145. Springer, 2011.
47. Intermediair (2009) Wetenschapsbeeld. VNU Media: Amsterdam, The Netherlands 40:34. http://www.intermediair.nl
48. J. Janssens, E. Postma, and J. Hellemons, editors. *Int. Workshop on Maritime Anomaly Detection (MAD 2011)*, Tilburg Center for Cognition and Communication, Tilburg University, The Netherlands, 2011. http://mad.uvt.nl/mad/mad2011-proceedings.pdf.
49. J. H. M. Janssens, I. Flesch, and E. O. Postma. Outlier Detection with One-Class Classifiers from ML and KDD. In *8th Int. Conf. on Machine Learning and Applications*, pages 147–155. IEEE Computer Society, 2009.
50. J. H. M. Janssens, H. Hiemstra, and E. O. Postma. Creating Artificial Vessel Trajectories with Presto. In *22nd Benelux Conf. on Artificial Intelligence (BNAIC 2010)*, 2010.
51. J. H. M. Janssens, F. Huszar, E. O. Postma, and H. J. van den Herik. Stochastic Outlier Selection. Technical report TiCC TR 2012-001, Tilburg University, Tilburg Center for Cognition and Communication, Tilburg, The Netherlands, 2012.
52. J. H. M. Janssens and E. O. Postma. One-Class Classification with LOF and LOCI: An Empirical Comparison. In M. van Erp, H. Stehouwer, and M. van Zaanen, editors, *18th Annual Belgian-Dutch Conf. on Machine Learning (Benelearn 09)*, pages 56–64, 2009. http://benelearn09.uvt.nl/Proceedings_Benelearn_09.pdf.
53. T. Kanstrén, E. Piel, A. González-Sánchez, and H.-G. Gross. Observation-Based Modeling for Testing and Verifying Highly Dependable Systems – A Practitioner's Approach. In A Wagner, editor, *Workshop on Design of Dependable Critical Systems at Safecomp (DDCS'09)*, 2009. http://www.ub.uni-heidelberg.de/archiv/10095.
54. T. Kanstrén, E. Piel, and H.-G. Gross. Observation-Based Modeling for Model-Based Testing. Technical Report Series TUD-SERG-2009-012, Delft University of Technology, Software Engineering Research Group, Delft, The Netherlands, 2009.
55. T. Kanstrén, E. Piel, and H.-G. Gross. Trace-Based Code Generation for Model-Based Testing. Technical Report Series TUD-SERG-2009-017, Delft University of Technology, Software Engineering Research Group, Delft, The Netherlands, 2009.
56. Keulen, J-P (2010) Vaart in kaart. Kijk, 2013 Sanoma Media Netherlands B.V.: Hoofddorp, The Netherlands 24(2):64–65 http://www.kijkmagazine.nl/
57. R. Lagerweij. Learning a Model of Ship Movements. Thesis for Bachelor of Science, University of Amsterdam, Faculty of Science, Amsterdam, The Netherlands, 2009.
58. F. M. Maggi, J. C. Bose, and W. M. P. van der Aalst. Efficient Discovery of Understandable Declarative Process Models from Event Logs. In J. Ralyté, X. Franch, S. Brinkkemper, and S. Wrycza, editors, *Advanced Information Systems Engineering (CAiSE 2012)*, volume 7328 of *Lecture Notes in Computer Science*, pages 270–285. Springer, 2012.
59. F. M. Maggi, M. Montali, and W. M. P. van der Aalst. An Operational Decision Support Framework for Monitoring Business Constraints. In J. Lara and A. Zisman, editors, *Fundamental Approaches to Software Engineering*, volume 7212 of *Lecture Notes in Computer Science*, pages 146–162. Springer, 2012.

60. F. M. Maggi, M. Montali, M. Westergaard, and W. M. P. van der Aalst. Monitoring Business Constraints with Linear Temporal Logic: An Approach Based on Colored Automata. In R. Rinderle-Ma, F. Toumani, and K. Wolf, editors, *Int. Conf. on Business Process Management (BPM 2011)*, volume 6896 of *Lecture Notes in Computer Science*, pages 132–147. Springer, 2011.
61. F. M. Maggi, A. J. Mooij, and W. M. P. van der Aalst. User-Guided Discovery of Declarative Process Models. In *IEEE Symp. on Computational Intelligence and Data Mining (CIDM)*, pages 192–199. IEEE Computer Society, 2011.
62. F. M. Maggi, M. Westergaard, M. Montali, and W. M. P. van der Aalst. Runtime Verification of LTL-Based Declarative Process Models. In S. Khurshid and K. Sen, editors, *Int. Conf. on Runtime Verification (RV 2011)*, volume 7186 of *Lecture Notes in Computer Science*, pages 131–146. Springer, 2012.
63. I. Magilsen. Poseidon Ontsluiert Geheimen op Zee. *Matrix*, 18(1):46–47, 2011. http://w3.tue.nl/fileadmin/csc/matrix/Matrix_2011-1.pdf.
64. M. Montali, F. M. Maggi, F. Chesani, P. Mello, and Aalst W. M.P. van der. Monitoring Business Constraints with the Event Calculus. Technical Report DEIS-LIA-002-11, Università degli Studi di Bologna DEIS, Bologna, Italy, 2012. http://www-lia.deis.unibo.it/Research/TechReport/LIA-002-11.pdf.
65. A.J. Mooij. System integration by developing adapters using a database abstraction. Inf Softw Technol. 55(2):357–364, (2013). http://dx.doi.org/10.1016/j.infsof.2012.08.015.
66. A. J. Mooij, J. Parnjai, C. Stahl, and M. Voorhoeve. Constructing Replaceable Services using Operating Guidelines and Maximal Controllers. In M. Bravetti and T. Bultan, editors, *Web Services and Formal Methods (WS-FM 2010)*, volume 6551 of *Lecture Notes in Computer Science*, pages 116–130. Springer, 2011.
67. A. J. Mooij, C. Stahl, and M. Voorhoeve. Relating Fair Testing and Accordance for Service Replaceability. *Journal of Logic and Algebraic Programming*, 79(3–5):233–244, 2010.
68. A. J. Mooij and M. Voorhoeve. Proof Techniques for Adapter Generation. In R. Bruni and K. Wolf, editors, *Web Services and Formal Methods (WS-FM 2008)*, volume 5387 of *Lecture Notes in Computer Science*, pages 207–223. Springer, 2009.
69. A. J. Mooij and M. Voorhoeve. Trading Off Concurrency to Generate Behavioral Adapters. In *9th Int. Conf. on Application of Concurrency to System Design (ACSD 2009)*, pages 109–118. IEEE, 2009.
70. NewScientist. Visions of Data, September 2009. http://www.newscientist.com/gallery/dn17746-visions-of-data.
71. E. Piel and A. González-Sánchez. Data-flow Integration Testing Adapted to Runtime Evolution in Component-Based Systems. In H.-G. Gross, M. Lormans, and J. Tretmans, editors, *ESEC/FSE Workshop on Software Integration and Evolution @ Runtime (SINTER'09)*, pages 3–10. ACM, 2009.
72. E. Piel, A. González-Sánchez, and H.-G. Gross. Automating Integration Testing of Large-Scale Publish/Subscribe Systems. In A. M. Hinze and A. Buchmann, editors, *Principles and Applications of Distributed Event-Based Systems*, pages 140–163. IGI Global, 2010.
73. E. Piel, A. González-Sánchez, and H.-G. Gross. Built-in Data-Flow Integration Testing in Large-scale Component-Based Systems. In A. Petrenko, A. da Silva Simão, and J. C. Maldonado, editors, *Testing Software and Systems (ICTSS 2010)*, volume 6435 of *Lecture Notes in Computer Science*, pages 79–94. Springer, 2010.
74. E. Piel, A. González-Sánchez, H.-G. Gross, and A. J. C. van Gemund. Spectrum-Based Health Monitoring for Self-Adaptive Systems. In *5th IEEE Int. Conf. on Self-Adaptive and Self-Organizing Systems (SASO'11)*, pages 99–108. IEEE Computer Society, 2011.
75. E. Piel, A. González-Sánchez, H.-G. Gross, A. J. C. van Gemund, and R. Abreu. Online Spectrum-Based Fault Localization for Health Monitoring and Fault Recovery of Self-Adaptive Systems. In F. Bodendorf and W. Powley, editors, *8th Int. Conf. on Autonomic and Autonomous Systems*, pages 64–73. IARIA, 2012.

76. R. Scheepens, N. Willems, Wetering H. van de, and J. J. van Wijk. Density Based, Visual Anomaly Detection. In J. Janssens, E. Postma, and J. Hellemons, editors, *Int. Workshop on Maritime Anomaly Detection (MAD 2011)*, pages 11–12, 2011. http://mad.uvt.nl/mad/mad2011-proceedings.pdf.
77. R. Scheepens, N. Willems, H. van de Wetering, G. Andrienko, N. Andrienko, and J. J. van Wijk. Composite Density Maps for Multivariate Trajectories. *IEEE Trans. on Visualization and Computer Graphics*, 17(12):2518–2527, 2011.
78. R. Scheepens, N. Willems, H. van de Wetering, and J. J. van Wijk. Interactive Visualization of Multivariate Trajectory Data with Density Maps. In *IEEE Pacific Visualization Symposium (PacificVis 2011)*, pages 147–154, 2011.
79. R. Scheepens, N. Willems, H. van de Wetering, and J. J. van Wijk. Interactive Density Maps for Moving Objects. *IEEE Computer Graphics and Applications*, 32(1):56–66, 2012.
80. R. J. Scheepens. GPU-Based Track Visualization of Multivariate Moving Object Data. Master's thesis, Eindhoven University of Technology, Dept. of Computer Science and Engineering, Visualization Group, Eindhoven, The Netherlands, 2010. http://www.win.tue.nl/visnet/wiki/doku.php?id=finished:densityplot.
81. D. M. M. Schunselaar, F. M. Maggi, and N. Sidorova. Patterns for a Log-Based Strengthening of Declarative Compliance Models. In J. Derrick, S. Gnesi, D. Latella, and H. Treharne, editors, *Integrated Formal Methods (IFM 2012)*, volume 7321 of *Lecture Notes in Computer Science*, pages 327–342. Springer, 2012.
82. D. M. M. Schunselaar, F. M. Maggi, N. Sidorova, and W.M.P van der Aalst. Configurable Declare: Designing Customisable Flexible Models. In *20th Int. Conf. on Cooperative Information Systems (CoopIS 2012)*, Lecture Notes in Computer Science, 2012.
83. M. van Someren et al., editor. *Workshop on Modelling Moving Objects in the Netherlands*, University of Amsterdam, The Netherlands, 2010. http://staff.science.uva.nl/~maarten/MMO-webpage.htm.
84. F. Spiessens, J. den Hartog, and S. Etalle. Know what you Trust: Analyzing and Designing Trust Policies with Scoll. In P. Degano, J. Guttman, and F. Martinelli, editors, *Formal Aspects in Security and Trust (FAST 2008)*, volume 5491 of *Lecture Notes in Computer Science*, pages 129–142. Springer, 2009.
85. Thales (2009) Poseidon Verhoogt Veiligheid op Zee. In: Login to Thales-Magazine of Thales Nederland B.V., July/Aug 2009, pp 8–9.
86. X. Theunissen. Poseidon Signaleert Afwijkend Gedrag Schepen. *De Technologiekrant*, 19:11, 2009.
87. X. Theunissen. Zeeverkeersinformatie. *De Ingenieur*, 14/15:HTS 6–7, 2009.
88. D. Trivellato. *Protecting Information in Systems of Systems*. PhD thesis, Eindhoven University of Technology, 2012.
89. D. Trivellato, F. Spiessens, N. Zannone, and S. Etalle. POLIPO: Policies & OntoLogies for Interoperability, Portability, and Autonomy. In *10th IEEE Int. Symp. on Policies for Distributed Systems and Networks (POLICY'09)*, pages 110–113. IEEE Computer Society, 2009.
90. D. Trivellato, F. Spiessens, N. Zannone, and S. Etalle. Reputation-Based Ontology Alignment for Autonomy and Interoperability in Distributed Access Control. In *IEEE Int. Conf. on Computational Science and Engineering (CSE'09)*, volume 3, pages 252–258. IEEE Computer Society, 2009.
91. D. Trivellato, N. Zannone, and S. Etalle. GEM: A Distributed Goal Evaluation Algorithm for Trust Management. Computer Science Report CS 10-15, Eindhoven University of Technology, 2010. http://alexandria.tue.nl/repository/books/695281.pdf.
92. D. Trivellato, N. Zannone, and S. Etalle. A Security Framework for Systems of Systems. In *12th IEEE Int. Conf. on Policies for distributed systems and networks (POLICY'11)*, pages 182–183. IEEE Computer Society, 2011.
93. D. Trivellato, N. Zannone, and S. Etalle. Poster: Protecting Information in Systems of Systems. In Y. Chen, G. Danezis, and V. Shmatikov, editors, *18th ACM Conf. on Computer and Communications Security (CCS'11)*, pages 865–868. ACM, 2011.

94. M. van de Ven. Veilig en Alert Heersen over de Zee. *Cursor*, 52(32):12, June 3 2010. http://web.tue.nl/cursor/internet/jaargang52/_pdf/cursor32.pdf.
95. G. de Vries, V. Malaisé, M. van Someren, P. Adriaans, and G. Schreiber. Semi-Automatic Ontology Extension in the Maritime Domain. In A. Nijholt, M. Pantic, M. Poel, and G. H. W. Hondorp, editors, *Belgian-Dutch Conference on Artificial Intelligence (BNAIC 2008)*, pages 265–272, Enschede, The Netherlands, 2008. University of Twente. http://eprints.eemcs.utwente.nl/13354/.
96. G. de Vries and M. van Someren. Unsupervised Ship Trajectory Modeling and Prediction using Compression and Clustering. In M. van Erp, H. Stehouwer, and M. van Zaanen, editors, *The 18th Annual Belgian-Dutch Conf. on Machine Learning (Benelearn 09)*, pages 7–12, 2009. http://benelearn09.uvt.nl/Proceedings_Benelearn_09.pdf.
97. G. de Vries and M. van Someren. Clustering Vessel Trajectories with Alignment Kernels under Trajectory Compression. In J. L. Balcázar, F. Bonchi, A. Gionis, and M. Sebag, editors, *Eur. Conf. on Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2010)*, volume 6321 of *Lecture Notes in Computer Science*, pages 296–311. Springer, 2010.
98. G. K. D. Vries, Hage W. R. van, and M. van Someren. Comparing Vessel Trajectories Using Geographical Domain Knowledge and Alignments. In Wei Fan, Wynne Hsu, Geoffrey I. Webb, Bing Liu, Chengqi Zhang, Dimitrios Gunopulos, and Xindong Wu, editors, *The 10th IEEE Int. Conf. on Data Mining Workshops (ICDMW 2010)*, pages 209–216. IEEE Computer Society, 2010.
99. G. K. D. Vries, Hage W. R. van, and M. van Someren. Comparing Vessel Trajectories using Geographical Domain Knowledge and Alignments. In J. Janssens, E. Postma, and J. Hellemons, editors, *Int. Workshop on Maritime Anomaly Detection (MAD 2011)*, pages 15–16, 2011. http://mad.uvt.nl/mad/mad2011-proceedings.pdf.
100. G. K. D. Vries, Hage W. R. van, and M. van Someren. Comparing Vessel Trajectories using Geographical Domain Knowledge and Alignments. In P. van der Putten, C. Veenman, J. Vanschoren, M. Israel, and H. Blockeel, editors, *The 20th Annual Belgian-Dutch Conf. on Machine Learning (Benelearn 2011)*, pages 125–126, 2011. http://www.liacs.nl/~putten/benelearn2011/Benelearn2011_Proceedings.pdf.
101. G. K. D. de Vries. *Kernel Methods for Vessel Trajectories*. PhD thesis, University of Amsterdam, 2012. http://dare.uva.nl/document/357350.
102. G. K. D. de Vries and M. van Someren. Machine Learning for Vessel Trajectories using Compression, Alignments and Domain Knowledge. *Expert Systems with Applications*, 39(18):13426–13439, 2012.
103. M. Westergaard and F. M. Maggi. Modelling and Verification of a Protocol for Operational Support using Coloured Petri Nets. In L. M. Kristensen and L. Petrucci, editors, *Applications and Theory of Petri Nets (Petri Nets 2011)*, volume 6709 of *Lecture Notes in Computer Science*, pages 169–188. Springer, 2011.
104. N. Willems. Shipping Routes North Sea. *Geo-info*, 8(6):9, 2011. http://www.geo-info.nl/download/?id=17685718.
105. N. Willems. *Visualization of Vessel Traffic*. PhD thesis, Eindhoven University of Technology, 2011. http://alexandria.tue.nl/extra2/719764.pdf.
106. N. Willems, W. R. van Hage, G. de Vries, J. H. M. Janssens, and V. Malaisé. An Integrated Approach for Visual Analysis of a Multisource Moving Objects Knowledge Base. In *13th AGILE Int. Conf. on Geographic Information Science – Workshop on Geospatial Visual Analytics: Focus on Time (GeoVa(t) 2010)*, 2010. http://www.few.vu.nl/~wrvhage/papers/geoVAt10.pdf.
107. N. Willems, W. R. van Hage, G. de Vries, J. H. M. Janssens, and V. Malaisé. An Integrated Approach for Visual Analysis of a Multisource Moving Objects Knowledge Base. *Int. Journal of Geographical Information Science*, 24(10):1543–1558, 2010.
108. N. Willems, H. van de Wetering, and J. J. Wijk. Evaluation of the Visibility of Vessel Movement Features in Trajectory Visualizations. *Computer Graphics Forum*, 30(3):801–810, 2011. Proceedings of EuroVis 2011.

109. N. Willems, H. van de Wetering, and J. J. van Wijk. Interactive Poster: Visualization of Vessel Trajectories for Maritime Safety and Security Systems. In *IEEE Information Visualization Conference (InfoVis 2008)*, 2008.
110. N. Willems, H. van de Wetering, and J. J. van Wijk. Visualization of Vessel Trajectories for Maritime Safety and Security Systems. In *SIREN 2008*, 2008. Poster; http://www.ictonderzoek.net/3/assets/File/posters/2008_70/2008_70.pdf.
111. N. Willems, H. van de Wetering, and J. J. van Wijk. Visualization of Vessel Movements. *Computer Graphics Forum*, 28(3):959–966, 2009. Proceedings of EuroVis 2009.

Index

A
Access control, 7, 48–49, 56, 58, 60, 190, 191, 200, 201, 205
Adapter, 15, 60, 61, 173, 174, 177–186
  adapter generation, 174, 180, 181, 184–186
AIS. *See* Automatic Identification System (AIS)
AIS receiver, 43, 56, 59–61, 67, 133, 147, 174–178, 182
Alignment
  ontology alignment, 6
  semantic alignment, 190, 191, 198, 200–202, 206
Anomaly detection, 10, 11, 14, 18, 63, 66, 82–83, 113, 116, 117, 120, 134
  density-based anomaly detection, 14, 83, 119–130
  rule-based anomaly detection, 14, 18
  runtime anomaly detection, 18
Architecture, 12, 13, 39–52, 55–58, 65, 68, 77, 78, 181, 184, 186, 191, 196, 197, 199–202, 206, 240, 241, 244, 256
  architectural reasoning, 39, 44–49
Automatic Identification System (AIS), 11–15, 24, 26–28, 30, 31, 33–35, 40, 43, 49, 56, 59, 60, 66, 67, 79, 86, 90, 94, 95, 101, 105–111, 116, 117, 133, 134, 140, 142, 143, 145, 147, 150, 154, 157, 163, 164, 174–177, 179, 182, 183, 209, 221–226, 242

B
Behavior
  behavior analysis, 13
  behavior recognition, 13
  behavior rule, 14
  emergent behavior, 46
  historical behavior, 14
  interface behavior, 15, 176–179
Beta probability distribution, 219, 220

C
Carrying Industrial Partner (CIP), 17, 19, 55, 68
Clustering, 14, 18, 57, 107, 110–117, 249
Coast guard, 10, 12, 24, 26, 28, 30, 31, 33–35, 93, 100
Collaborative system, 7
Confidentiality, 6, 23, 190, 191, 195, 196, 198, 200, 205, 206
Connector, 8
Context-aware access control, 15, 190, 196–198
Controller, 178–181, 186
  controller synthesis, 15, 174, 177–181, 185, 186

D
Database, 6, 15, 21, 40, 49, 58, 59, 106, 116, 150, 165, 166, 169, 174, 182–186, 227, 235
Data Distribution Service (DDS), 43, 56
Data source, 6, 49, 74
Declare miner, 57, 140–144
Demonstrator, 4, 13, 19, 50, 55–68, 185, 186
Density, 62, 66, 74–86, 120–125, 127, 130
  density map, 13, 61, 74, 75, 77–86

P. van de Laar et al. (eds.), *Situation Awareness with Systems of Systems*,
DOI 10.1007/978-1-4614-6230-9, © Springer Science+Business Media New York 2013

Diagnosis, 8, 18, 229, 231–237, 239, 242–244, 249–253, 255, 256
self-diagnosis,
runtime diagnosis, 15, 46, 47
Distance function, 14, 108, 109, 113, 117
Distributed access control, 18

E
EthoVision XT, 89–96, 99–102
Event, 4, 21, 50, 103, 105, 119, 133, 149, 192, 211, 234, 248
Evolving system, 244

F
Fault localization
runtime fault localization, 16, 62
Spectrum-based Fault Localization (SFL), 16, 230–233, 244, 248
Federation of systems, 7
Functional, 12, 42, 50, 196, 199, 233, 241, 242

G
Geographic information layer, 58, 61
Glue logic, 173
Google Earth, 15, 25, 58, 61, 64, 175–178, 180, 182, 185, 202

H
Health
information health, 42, 47–48
system health, 46–47, 147
Human operator, 5, 40, 52, 119

I
IMO. *See* International Maritime Organization (IMO)
Incident, 22, 23, 25, 26, 31–36
Incremental view maintenance, 15, 174, 182–186
Industry-as-laboratory, 17, 18, 55, 65, 68
Information
information aggregation, 4–6
information complexity, 22
information flow, 40–50
information fusion, 6, 41, 66
information overload, 21, 22, 27
information visualization, 6, 19
Integration
runtime integration, 44
system integration, 15, 173–186
Integrity, 6, 190, 191, 195, 196, 200, 205
Interface, 66, 83, 173, 176–177, 179–181, 183–186, 196, 198, 200, 202
International Maritime Organization (IMO), 11, 34, 60, 221–223
International Telecommunications Union (ITU), 11, 225
Interoperability
semantic interoperability, 6, 190, 205
syntactic interoperability, 6
Interpolation, 95, 101

K
Kernel density estimation, 75
Knowledge
domain knowledge, 29, 50, 57, 67, 79, 86, 112–115, 190, 191, 197, 201, 205, 227
knowledge base, 18, 57, 61–63, 112–117, 190, 197, 198, 200–202
knowledge representation, 196

L
Linear temporal logic, 139
Linked data, 152

M
Machine learning, 52, 130, 249
Maritime Mobile Service Identity (MMSI), 11, 34, 60, 62, 63, 175, 182, 203, 221–224, 226
Maritime Safety and Security (MSS), 3–19, 21–24, 26, 28, 29, 31, 36, 37, 39, 55, 59, 67, 89, 93, 100, 103, 105, 106, 112, 113, 116, 117, 119, 124, 133, 136, 139, 142, 145, 147, 151, 174, 185, 191, 197, 206, 232, 247, 248
Mediator, 173
Meta-information, 8, 42, 45, 49
Metis, 19
MMSI. *See* Maritime Mobile Service Identity (MMSI)
Model, 14, 15, 77, 109, 134, 137–140, 143–147, 149, 151, 154, 166, 169, 177–181, 186, 191, 205, 220, 221, 226, 227, 233–234, 243, 248, 249
Monitoring, 134
health monitoring, 15, 16, 62, 229
runtime monitoring, 8, 15, 18
MSS. *See* Maritime safety and security (MSS)

N
Non-functional, 12, 44, 242
North Sea, 4, 22, 26, 55, 60, 85, 100, 111

O
Online. *See* Runtime
Ontology, 14, 15, 18, 113–115, 117, 151, 160, 162, 191, 197–199, 201, 202, 206
OODA loop, 28–30, 33
Open provenance model, 220–222
Operational picture, 18, 50
Opinion, 211–220, 222–224, 226
Outlier, 13, 14, 18, 92, 117, 120–124, 126–130, 201, 203, 204
  outlier detection, 57, 120–124, 126–130

P
Petri net, 177, 178, 180
Piecewise linear segmentation, 14, 107, 108, 116
POLIPO, 190, 191, 196–202, 205, 206
POSEIDON, 3, 4, 13, 16–19, 26, 51, 55–68, 158, 159, 163, 166–168, 180, 184–186, 196, 202
Prioritization
  diagnostic prioritization, 253
  test prioritization, 248–251, 253, 255
Privacy, 6, 9, 60, 175
Probe, 185, 186
Process discovery, 147
Process mining, 14, 18, 61, 133–147, 179, 185
Protocol, 6, 11, 44, 45, 48, 52, 106, 173, 176, 189
Provenance, 41, 42, 45, 49, 210, 220–228

R
RDF. *See* Resource Description Framework (RDF)
RDFS. *See* Resource Description Framework Schema language (RDFS)
Reasoning, 5, 6, 14, 19, 29–31, 36, 46, 49, 52, 57, 58, 139, 166, 168, 169, 198, 210–216, 221, 222, 232, 249–252
  semantic reasoning, 6, 18
Recognition, 7, 13, 14, 41, 44, 52, 94, 105, 109, 117, 254
Reconfiguration, 7–9, 229, 247
  runtime reconfiguration, 8
Reproducibility, 59, 65–66, 91
Resource Description Framework (RDF), 57, 149, 152, 154, 156–158, 161, 165, 166, 168, 169
Resource Description Framework Schema language (RDFS), 149, 151, 157, 168, 169
Runtime, 7, 14, 43, 59, 137, 185, 189, 247

S
Sea lane, 25, 26, 30, 31, 33, 34, 36, 77, 79, 85, 86, 91, 112, 119
Security
  ontology-based security policy, 15
  POLIPO security framework, 15, 189–206
  security framework, 189–191, 195, 196, 199, 203, 205
Segmentation, 18
Segmenter, 57
SEM. *See* Simple Event Model (SEM)
Semantic web, 6, 40, 149, 150, 161, 206
Sensor network, 27–28
Similarity
  similarity calculation, 18
Simple Event Model (SEM), 14, 18, 62, 113, 149–169, 202, 221, 222
Simulation environment, 66–67
SPARQL, 149, 166–169
Spatiotemporal indexing, 57
Specification, 15, 36, 150, 156, 173–186, 190, 191, 195, 197, 198, 205, 206
Stochastic outlier selection, 14, 18, 124–126
Subjective logic, 211–216, 220, 222, 223, 227

T
Testing, 7, 9, 15, 94, 190, 229, 230, 232, 235, 248, 250–252, 256
  runtime testing, 8, 9, 248, 256
Track
  sensor track, 57
  system track, 57

Tracking
animal tracking, 13, 90, 91, 93–94, 99–102
object tracking, 13, 19, 74
vessel tracking, 11–13, 90, 93–94, 99–102
Traffic separation scheme, 25–26
Training set, 6
Trajectory
trajectory alignment, 109, 115
trajectory classification, 107
trajectory clustering, 14, 18, 107, 110, 112
trajectory compression, 117
Transition system, 177
Transport mechanism, 43–44
Trust
trust calculation, 18
trust management, 15, 190, 191, 196–198, 201, 205

U
Uncertainty, 6, 8, 19, 39, 41, 42, 45–46, 48, 86, 162, 211–220, 224, 226, 227
Usability, 147, 199

V
Verification, 52, 139, 205, 247
runtime verification, 8
View
database view, 183
functional view, 13, 49
system architect's view, 40
Visual analysis, 58, 86
Visualization, 6, 12, 13, 19, 30, 56, 73–87, 96, 97, 100, 105–107, 116, 174, 202, 203, 224, 225
multi-objective visualization, 18