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, © Springer Science+Business Media New York 2013

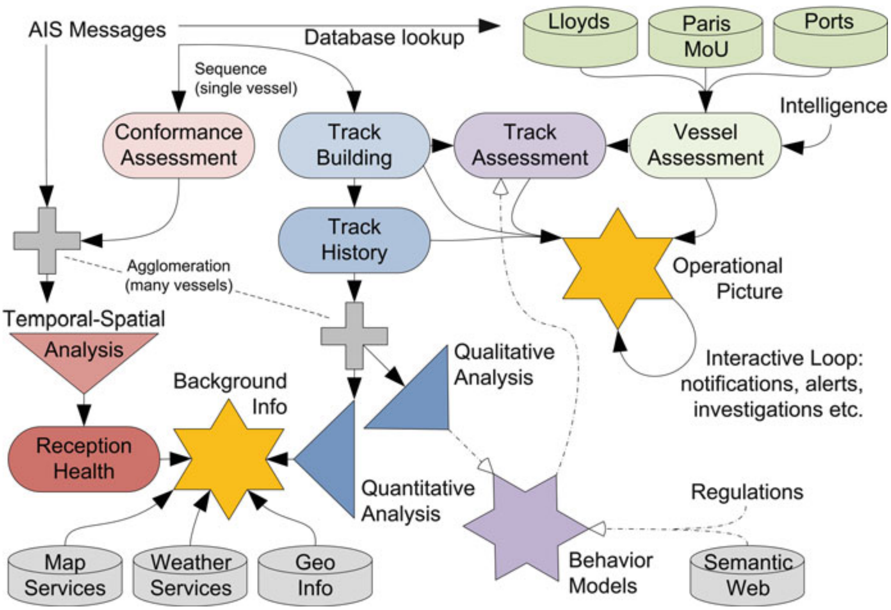

Fig. 3.1 Information processing for maritime situation awareness (Overview)

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 time-scales, 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

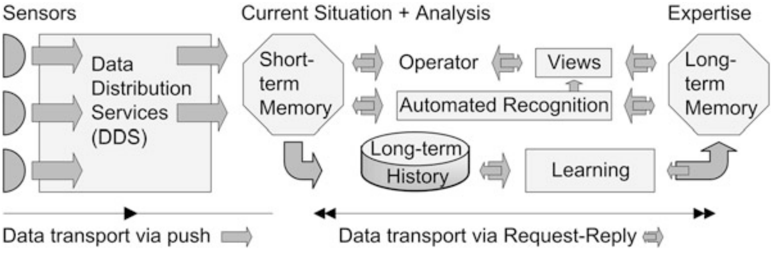

Fig. 3.2 Multi-stage information architecture

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.

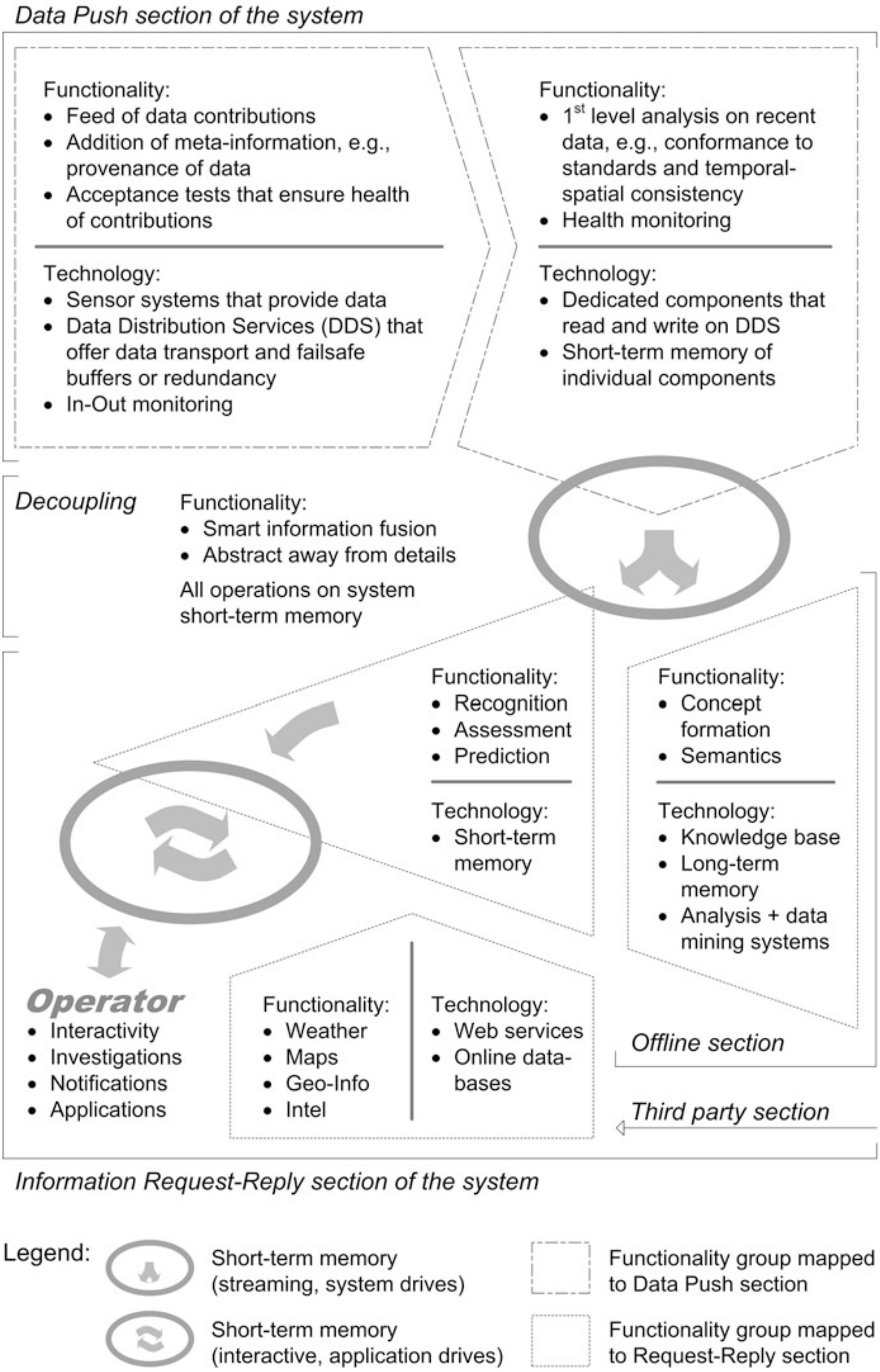
Data Push section of the system
Functionality:
• Feed of data contributions
• Addition of meta-information, e.g., provenance of data
• Acceptance tests that ensure health of contributions
Technology:
• Sensor systems that provide data
• Data Distribution Services (DDS) that offer data transport and failsafe buffers or redundancy
• In-Out monitoring
Functionality:
• 1st level analysis on recent data, e.g., conformance to standards and temporal-spatial consistency
• Health monitoring
Technology:
• Dedicated components that read and write on DDS
• Short-term memory of individual components
Decoupling
Functionality:
• Smart information fusion
• Abstract away from details
All operations on system short-term memory
Functionality:
• Recognition
• Assessment
• Prediction
Technology:
• Short-term memory
Functionality:
• Concept formation
• Semantics
Technology:
• Knowledge base
• Long-term memory
• Analysis + data mining systems
Operator
• Interactivity
• Investigations
• Notifications
• Applications
Functionality:
• Weather
• Maps
• Geo-Info
• Intel
Technology:
• Web services
• Online data-bases
Offline section
Third party section
Information Request-Reply section of the system
Legend:
Short-term memory (streaming, system drives)
Short-term memory (interactive, application drives)
Functionality group mapped to Data Push section
Functionality group mapped to Request-Reply section

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