Joseph Barjis
Robert Pergl (Eds.)

Enterprise and Organizational Modeling and Simulation

LNBIP 191

10th International Workshop, EOMAS 2014
Held at CAiSE 2014, Thessaloniki, Greece, June 16–17, 2014
Selected Papers

Springer

Lecture Notes in Business Information Processing 191

More information about this series at http://www.springer.com/series/7911

Joseph Barjis · Robert Pergl (Eds.)

Enterprise and Organizational Modeling and Simulation

10th International Workshop, EOMAS 2014
Held at CAiSE 2014, Thessaloniki, Greece,
June 16–17, 2014
Selected Papers

Springer

Editors
Joseph Barjis
Delft University of Technology
Delft
The Netherlands

Robert Pergl
Czech Technical University in Prague
Prague
Czech Republic

ISSN 1865-1348 ISSN 1865-1356 (electronic)
ISBN 978-3-662-44859-5 ISBN 978-3-662-44860-1 (eBook)
DOI 10.1007/978-3-662-44860-1

Library of Congress Control Number: 2014950826

Springer Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Modern enterprises are growing in complexity in all dimensions from IT landscape to intricate business processes and workflows. Not only the internal factors, but also the external ones play a significant role in constant changes that an enterprise has to make. Efficiency of these enterprises is crucial in delivering service to the society and contributing to economic prosperity. For designing and redesigning an efficient and well-integrated enterprise, while qualitative methods play an important role, quantitative methods reinforce to make profound design decisions.

With this purpose, EOMAS was launched to become a forum among researchers and practitioners to share their research and practical findings. In this forum we encourage dissemination of research results under a more generic umbrella called enterprise engineering.

As any system, an enterprise is an object of continuous improvements, redesign, and reimplementation. The departure point for any design or redesign activity pertinent to an enterprise is first to understand the enterprise business processes. Therefore, in the overall enterprise engineering activities, business process modeling plays a central role. However, an extended enterprise and organizational study involves both analysis and design activities, in which not only modeling but also simulation plays a prominent role. Therefore this growing importance of modeling and simulation in the context of enterprises is attracting serious attention from researchers. Today, modeling and simulation are the tools and methods that are effective, efficient, economic, and widely used in enterprise engineering, organizational study, and business process management.

Complementary insights of modeling and simulation in enterprise engineering constitute a whole cycle of study of enterprises. In order to monitor and study business processes and interaction of actors in a realistic and interactive environment, simulation has proven to be a powerful tool and method, especially if simulation is supported with rich animation and gaming elements. In order to explore these topics, address the underlying challenges, find and improve solutions, and demonstrate applications of modeling and simulation in the domain of enterprise, its organization and underlying business processes, peer-refereed papers were accepted for presentation at EOMAS 2014, which was held on June 16–17, 2014, in Thessaloniki, Greece. This year, we had a total of 22 paper submissions, of which 12 were accepted for publication.

June 2014

Joseph Barjis
Robert Pergl

Organization

The EOMAS workshop is annually organized as an international forum for researchers and practitioners in the field of Enterprise & Organization Modeling and Simulation. Organization of this workshop, planning, and review of the contributions were accomplished by an international team of researchers.

Workshop Organizers

General Chair

Joseph Barjis	Delft University of Technology, The Netherlands

Program Chair

Robert Pergl	Czech Technical University in Prague, Czech Republic

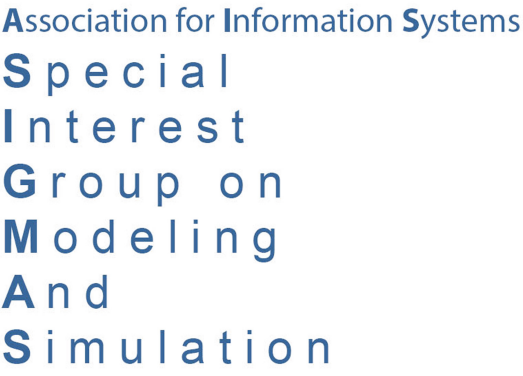
Program Committee

Antonia Albani	University of St. Gallen, Switzerland
Anteneh Ayanso	Brock University, Canada
Joseph Barjis	Delft University of Technology, The Netherlands
Ygal Bendavid	University of Quebec in Montreal (UQAM), Canada
Kawtar Benghazi	University of Granada, Spain
Mahmoud Boufaida	Mentouri University of Constantine, Algeria
Tatiana Bouzdine-Chameeva	BEM, Bordeaux Management School, France
Manuel I. Capel-Tuñón	University of Granada, Spain
Davood Chitchian	XomicX Company, The Netherlands
José Luis Garrido Bullejos	University of Granada, Spain
Rafael Gonzalez	Javeriana University, Colombia
P. Radha Krishna	Infosys Technologies Ltd., India
Peggy Daniels Lee	Indiana University-Purdue University Indianapolis, USA
Prabhat Mahanti	University of New Brunswick, Canada
Vojtech Merunka	Czech University of Life Sciences Prague, Czech Republic
Alta van der Merwe	University of South Africa, South Africa
Martin Molhanec	Czech Technical University in Prague, Czech Republic

Navonil Mustafee	University of Exeter Business School, UK
Manuel Noguera	University of Granada, Spain
Muralimohan Narasipuram	University of Porto, Portugal
Robert Pergl	Czech Technical University in Prague, Czech Republic
Ghaith Rabadi	Old Dominion University, USA
Victor Romanov	Russian Plekhanov University of Economics, Russia
Irina Rychkova	University Paris 1 Pantheon, France
Mamadou Seck	Delft University of Technology, The Netherlands
Natalia Sidorova	Eindhoven University, The Netherlands
Michel Soares	Federal University of Uberlandia, Brazil
David Sundaram	The University of Auckland, New Zealand
José Tribolet	Technical University of Lisbon, Portugal

Sponsoring Institutions

- SIGMAS (**S**pecial **I**nterest **G**roup on **M**odeling And **S**imulation of the Association for Information Systems)
- CAiSE 2014 (International **C**onference on **A**dvanced **I**nformation **S**ystems Engineering)
- Czech Technical University in Prague (Faculty of Information Technology, Department of Software Engineering)



CZECH
TECHNICAL
UNIVERSITY
IN PRAGUE

FIT

Contents

Enterprise Conceptual Modelling and Simulation

Extraction and Reconstruction of Enterprise Models

Mario Sánchez(✉), Julio Cesar Reyes, and Jorge Villalobos

Systems and Computing Engineering Department, School of Engineering, Universidad de Los Andes, Bogotá, Colombia
{mar-san1,jc.reyes159,jvillalo}@uniandes.edu.co

Abstract. Enterprise Models for analysis, and especially for automated analysis, should have five characteristics: they have to be accurate representations of the reality; they have to be well structured; they have to be complete with respect to their intended usage; they have to be kept up-to-date; and the cost of their construction and maintenance has to be as low as possible. In this paper we present an approach for the semi-automatic construction of enterprise models which gathers and weaves information from multiple sources such as information systems, databases, files (system's logs, source code, configuration), and previously existing models. This approach is based on modeling and metamodeling techniques, and has been implemented in a tool called EM-AutoBuilder.

Keywords: Enterprise modeling · MDE · Automatic documentation · Model analysis

1 Introduction

Enterprise Modeling (EM), the discipline and practice of building and analyzing models representing one or many concerns of an enterprise, is progressively becoming mature and widespread. The value that can be gained from doing EM is directly proportional both to the quality of the models, and the quality of the available tools and methods to perform the analyses: if models are small, have low level of detail, have low fidelity, or are unstructured (e.g., text documents), it is difficult to perform insightful analyses; on the other hand, if analyses are simple and naive, there is no point in building advanced and detailed models. A clear example of this are simulation-based analyses, which are very advanced but require high-quality models with information spanning several domains. In this paper we focus on the first concern (model quality) and make a proposal to address this problem and thus increment the value that can be gained from EM, and especially from model analysis.

The biggest issue affecting the quality of enterprise models is the elevated costs of construction and maintenance. Since building these models is typically a human-intensive task, compromises are made which go against quality. For example, the scope of the model may be limited, or its depth, or its completeness. Furthermore, the lack of widespread modeling tools usually results in the

J. Barjis and R. Pergl (Eds.): EOMAS 2014, LNBIP 191, pp. 3–20, 2014.
DOI: 10.1007/978-3-662-44860-1_1

usage of inadequate technologies (e.g., text processors, spreadsheets, unstructured diagrams) that produce models that are impossible to process. Finally, enterprises are ever changing, and thus enterprise models must be permanently maintained. However, the current situation makes it very expensive to make these permanent upgrades.

The hypothesis that we attempt to validate with this work is that a lot of the information that should be included in an enterprise model can be obtained and structured in a *largely automated* way. For example, it should be possible to gather information about the architecture of deployed information systems, combine it with information about the enterprise coming from structured documentation, and finally enrich it with real statistics about its behavior coming from systems' log registries. The result of this, would be a comprehensive enterprise model which can be easily kept updated. Automating steps in the process of collecting and structuring the models, should also remove potential sources of errors and inconsistencies. Ultimately, this all should lead to increasing the quality of enterprise models and the value that can be obtained from them.

To validate this hypothesis, we designed an approach and architecture for building enterprise models using information obtained from different sources. This approach was implemented in a tool called EM-AutoBuilder, which has already been tested in an internal case study, and is now starting to be applied in real scenarios.

The structure of the paper is as follows. Section 2 discusses in more detail what Enterprise Models are, the possible sources of information to build them, and the current state of art in automatic construction of enterprise models. Section 3 introduces a scenario to illustrate the solution. Then, the proposed approach and its implementation are presented and illustrated in Sect. 4. Section 5 concludes the paper and discusses the outlook for the presented work.

2 Automatic Construction of Enterprise Models

An Enterprise Model is a representation of elements of an enterprise that typically belong to different domains (e.g., business processes, business and regulatory environment, organizational structure, or information technology). The cost of building enterprise models varies depending on two factors: the required level of detail, and the scope that the model should cover (i.e., how much of the enterprise and how many domains should be represented in the model). Before making a commitment to build a model, these two variables should be analyzed and balanced against the cost of construction, and against the benefits that the model can eventually bring. To further complicate the matter, these benefits are not intrinsic to the model, but depend on how it is used. For instance, it can be used for (i) documentation or communication purposes; (ii) as a way to increase understanding of the enterprise; (iii) as the starting point for analyzing the current situation of the company; (iv) or to evaluate transformation projects.

Given the aforementioned potential uses of enterprise models, there are a number of desirable qualities that said models should posses. The first and most

important one is *accuracy*: a model that does not reflect the reality cannot answer truthfully any kind of answer, and thus is useless. On top of that, making decisions based on incorrect information can be worse than not making the decisions at all. A second, related quality is that a model has to be *up to date* (old information is just a particular kind of incorrect information, and just as risky as inaccurate information). Next, an enterprise model should to be *structured*: while models are frequently represented using unstructured means such as documents, diagrams, and spreadsheets, their real value can only be achieved if they are built around well defined structures and using representations that favor automatic processing. The fourth quality is *completeness*: a model should be complete with respect to its intended usage, and it should not lack information that it is expected to have. Finally, the *cost of building and maintaining* an enterprise model should be as low as possible; otherwise, it will probably be incomplete or will quickly cease to be up to date.

To build an enterprise model, it is critical to discover the potential sources of information. The typical sources include personnel of the company that deliver the information by means of interviews; manuals and documentations about processes, procedures, organizational structures, and responsibilities; and technical documentation about applications and technological elements. Other powerful sources of information are the Information Systems (IS) themselves, which are usually capable of providing structural and behavioral information. This can be achieved by direct observation of the IS technological components (interfaces, configuration files, source code, etc.), or by studying the relevant documentation and architectural documents. Furthermore, observing the storage systems and logging records of those IS provides valuable information to build models that are also behavioral, instead of purely structural.

Automatically building enterprise models is ideal, considering the discussed qualities. However, most sources of information are not suitable for this: only very well structured documentation (e.g., based in a Quality Management System Software), and the information systems themselves, can be automatically studied in order to build accurate, up to date, structured, and complete models, at relatively low costs.

In the past, some projects have made attempts at automatically building models with different levels of detail and focusing on particular domains. The work of Buschle et al. [1] reconstructs models of enterprise architecture based on network scanning and retrieving, in a graph-based structure, the main components of the application infrastructure of information systems. In this work they collected information using network analysis tools and vulnerability examination tools, and the results were application deployment models. Similarly, the work of Binz et al. [2] combined a manual and a semi-automatic approach for model construction. The automatic part was limited to network assets discovery (e.g., operative systems, DBMS, Application Servers), and the result was a graph representing the topology of enterprise systems.

Other works have tackled the problem from a source code perspective. Instead of obtaining the information from the systems already deployed, they have

collected the information from their source code. These works include MoDisco [3], the work of Schmerl et al. [4], and the work of Song et al. [5]. MoDisco is an extensible framework that is targeted to support software modernization. Moreover, as part of this, MoDisco is capable of analyzing artifacts such as source code, database structures, and configuration files, in order to create models representing existing systems. The work that we present in this paper borrows some architectural ideas from MoDisco, especially with respect to the extensibility capabilities.

On the other hand, the work of Schmerl et al. combines the analysis of source code with the analysis of low-level system events to obtain a representation that relates events that happen in specific use cases of the system, in a specific runtime scenario. Finally, the work of Song et al. [5] analyzes application API and calls, and uses this information to reconstruct the structure of the systems.

Our approach differs from the above mainly because they focus on a single domain, whereas we intend to create multi-dimensional enterprise models. We later show that our enterprise models are produced by combining static information obtained from documents, structural information coming from the observation of deployed information systems and/or their source code, and from the capture of behavioral information that is usually stored in such information system's logfiles.

3 An Illustrative Case Study

To illustrate the work presented in this paper, a case study is now introduced. This case study is a fictitious but realistic company called BPO Los Alpes[1], which offers outsourced services. In particular, we are going to focus on services for creating and managing donation campaigns. These include managing information, publicity, and press releases about the campaigns; gathering information about potential donors; contacting the donors; collecting payments; and tracking the success or failure of the campaigns.

The BPO bases its operation on the following internal information systems, which are illustrated in Fig. 1.

- **Donations System:** this is the system that handles donation campaigns and it is responsible both for managing the business logic as for storing the relevant information. This includes information about donation campaigns, potential and actual donors, donations, payments, and certificates. Information is stored in a relational database (DB DON).
- **CCA:** this is the system used by the call-center of the BPO to contact the potential donors and register information about the outcomes of the calls.

[1] We call the company fictitious because it does not really exist or offer any service. However, it is realistic because it was modeled after real companies that provide similar services, and because its information systems are completely build and operational. We use this, and other similar scenarios, to support research and initial prototypes, and also for educational purposes in the courses we teach.

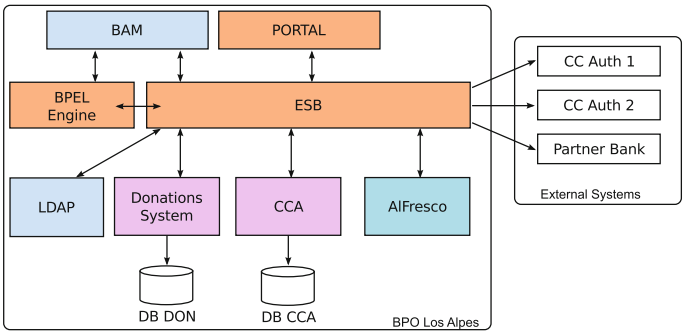


Fig. 1. Architecture of the case study

This system handles the contact-center agents, the lists of pending calls and their assignment, and an annotated registry of calls. Information is stored in a relational database (DB CCA).

- **AlFresco (ECM):** this is the enterprise content management system used to store the certificates for donors (which have legal validity for tax-exemption purposes), and to publish information about donation campaigns. The storage of its information is entirely handled by AlFresco.
- **LDAP:** this is the system that stores information about the users of the different applications and is capable of granting or denying usage privileges. All the applications use this LDAP system for authentication purposes.

In addition to these internal systems, the BPO depends on consuming services from payment transaction systems provided by allied banks and by credit card authorization systems. There are also three elements used for integration and coordination purposes: these are a BPEL engine, an Enterprise Service Bus (ESB), and a Portal. Finally, there is a BAM system to calculate and display indicators based on information obtained from the different applications, the bus and the BPEL engine.

In addition to this, there are several documents and artifacts describing the structure and operation of the BPO. Some of them are unstructured documents created with a common text editor, and thus are very difficult to process; others are diagrams and models created with specialized tools (i.e., ArchiMate and BPMN editors) and thus can be processed with relative ease.

4 Automatic Documentation: EM-AutoBuilder

The goal of this section is to present an approach to build enterprise models with the qualities that were discussed in Sect. 2. This approach has been implemented in a tool called EM-AutoBuilder, has been validated using the scenario presented in Sect. 3, and is currently starting to be used in a real scenario.

The approach was designed to address four critical requirements. The first one required models to be constructed in a way as automatic as possible. Therefore, the approach is targeted to collecting information from sources that can be automatically processed, i.e. information systems and well-structured documents and models. The second requirement was supporting heterogeneity. That is, it should be possible to gather information from various sources that have different structures, support different purposes, and are built around different technologies. These sources may also include documents, further increasing the complexity and heterogeneity. The next requirement is also closely related: the approach should be extensible, in order to be applicable to new sources of information. The fourth and final requirement regards the output of the approach: it should be a single, integrated artifact that can be processed or loaded into other tools that provide analysis capabilities.

Given these requirements, we designed the approach that is illustrated in Fig. 2. The core of this approach is a component that hosts several *configurable extractors* and processes their outputs. Extractors are independent components, and each one is capable of connecting to some specific kind of source to extract information. This information is returned to the core structured as a model. After each extractor has provided one or several models, the core has to process them to build an integrated one. However, the models may conform to different metamodels, and the core cannot know those metamodels before hand. Therefore, each extractor also has the capacity to provide the metamodels that it uses to build the models.

A final point in the strategy is the capacity of the core to weave the models, based on the weaving of the metamodels. It has been shown that completely automating the latter procedure is not feasible [6]. Therefore, we decided to let this step under the responsibility of the user, which has to specify the necessary relations between the metamodels. Finally, models are woven based on the information that the user provided, and a single tuple <metamodel, model> is produced.

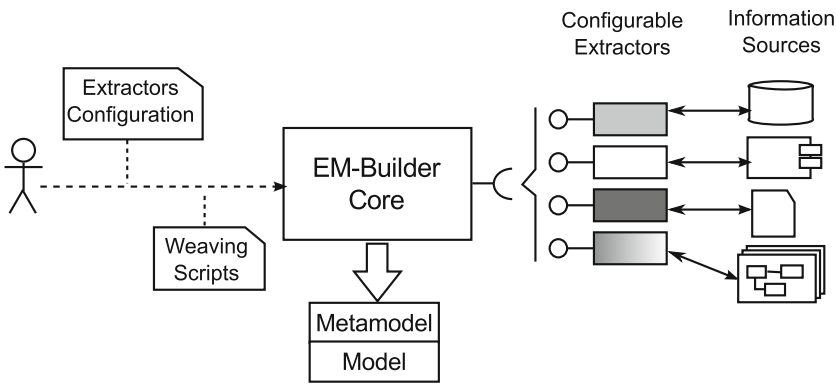


Fig. 2. Overview of EM-AutoBuilder architecture

The approach has been implemented in a tool called EM-AutoBuilder. This tool is based on Java and EMF[2] technologies because this makes it compatible with many tools for model processing, analysis, and visualization. Moreover, it should be possible to build similar tools using other technologies that match other tool environments. The following sections provide more details about each of the elements involved and about the responsibilities of each one.

4.1 Individual Extraction of Information

The first step in the automatic creation of enterprise models is to collect information from all the relevant data sources. However, these sources have a level of diversity that makes it impossible to have an universal component capable of querying them all. Even in the simplest cases, such as extracting information from relational databases, small technical differences in the way of managing the schema structure may prevent the same component for querying any RDMS.

In EM-AutoBuilder, the solution for this diversity problem was to build a framework and define an abstract component, the *extractor*, that is capable of querying systems to obtain and structure information. Concrete extractors are built using the framework, and they only share an API and some libraries to assemble and process models. The API that all concrete extractors implement has the following two main methods:

- `configure`: this method receives a set of Java properties with the information that the extractor needs to find its data source. For example, in the case of an extractor to collect information from a relational database that is accessed through JDBC, the properties include the name of the driver to use, the url to locate the DB, and the username and password to connect.
- `collectInformation`: this method uses the configuration information to obtain information and structure it in an EMF model. The output of this method is a tuple containing a model and a metamodel that are used subsequently by the EM-AutoBuild core.

It is worth noting that the metamodel that each extractor returns is sometimes calculated as part of the process of collecting information, i.e. it cannot be known a priori and it depends on the information obtained from the source. The reason for this has two parts. Firstly, the relation *instanceOf*, between *instances* and their *types* naturally exists within some domains. Secondly, both the types and the instances should appear in the same model, and the structure of the instances should conform to the restrictions imposed by their corresponding types. Unfortunately, this is not something typically supported by modeling frameworks, and thus it would require ad hoc solutions in each case. We experimented with some alternatives to manage these situations, but they required capabilities not supported by EMF, such as deep instantiation and potencies [7]. Ultimately, the chosen strategy was also adopted because it simplifies the weaving process.

[2] EMF - Eclipse Modeling Framework: http://www.eclipse.org/modeling/emf/.

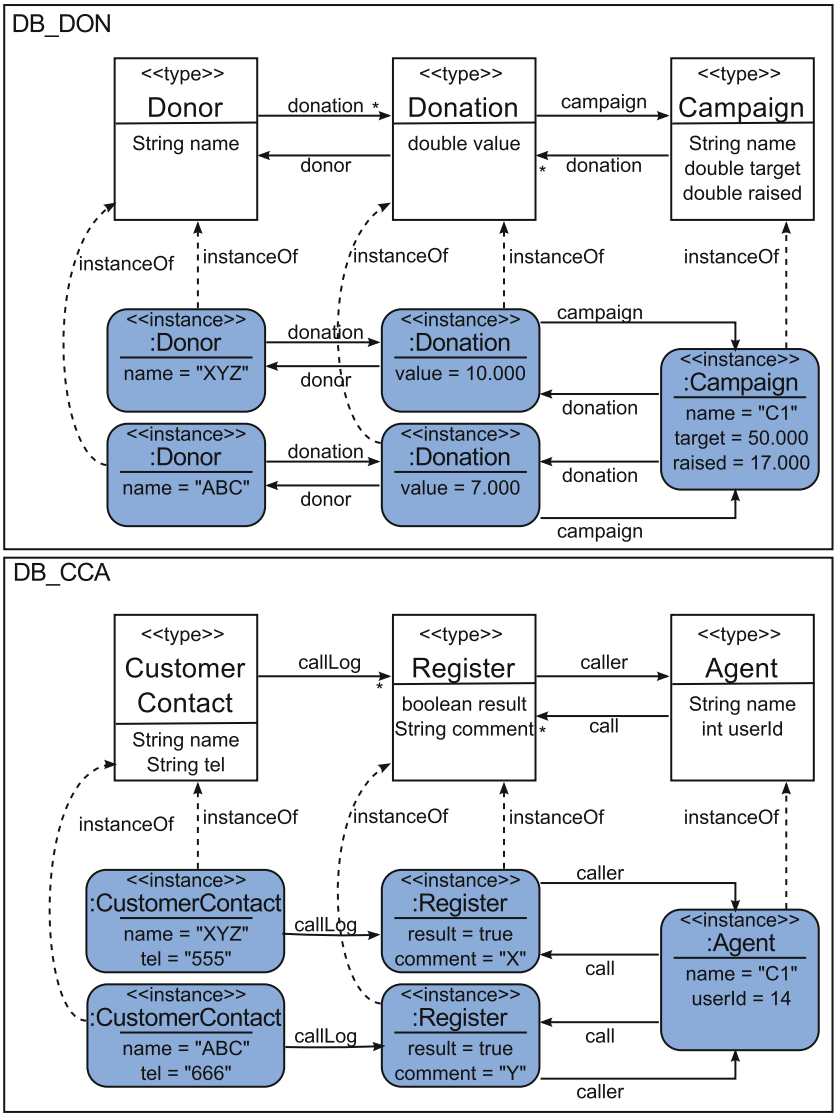


Fig. 3. Models obtained from the CCA and Donations System database

Figures 3 and 4 present a fragment of the results obtained from applying extractors to the BPO case study. In this case, the extractors utilized were capable of collecting information from relational databases, and they were applied to the Donations System database and to the CCA database. The results obtained were one model and one metamodel for each system.

Figure 3 presents the two models: on the top of the figure, there is the model obtained from analyzing the Donations System database; on the bottom of the

figure, there is the model obtained from analyzing the CCA database. Both models include two kinds of elements. Firstly, there are elements that have been represented as classes and are marked with the `<<type>>` stereotype. These elements appear in the model because the databases contained tables with those names, and because it was useful to include those concepts in the models. Secondly, there are elements represented as objects which received the stereotype `<<instance>>`. These elements appear in the model because there were corresponding registers in the tables. The relation `instanceOf` between an `<<instance>>` and a `<<type>>` is of *ontologic* nature [8], and it is a reflection of the relation between a registry in the database and the table that contains it. Furthermore, the elements marked as `<<type>>` also appear on the metamodels, thus making it possible to have *linguistic* instances of them in the models. Since the same extractor was used to collect data from both databases, the structure of both models is very similar.

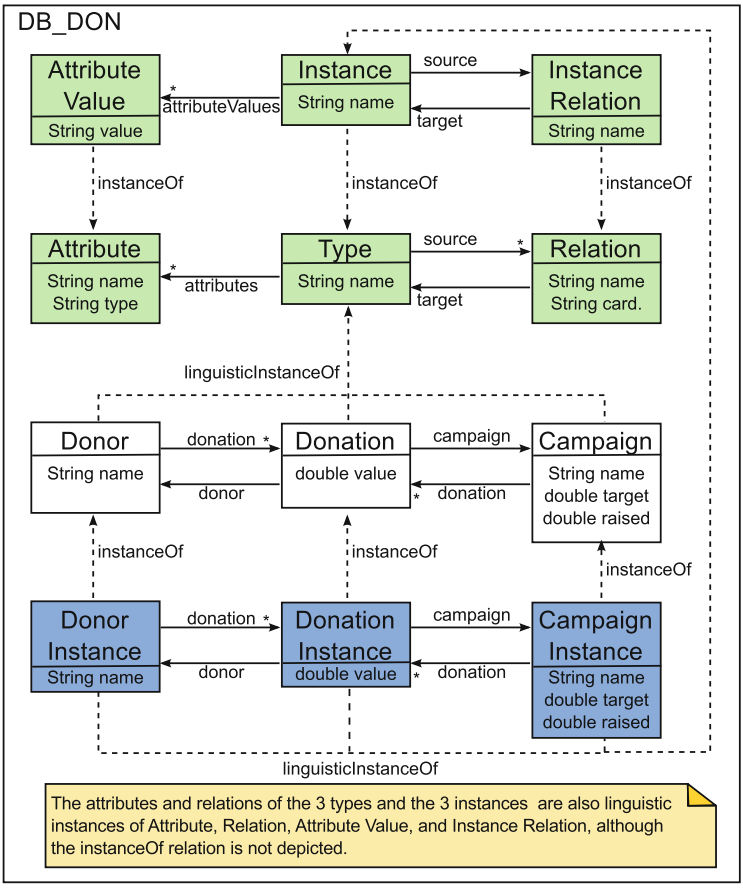


Fig. 4. Metamodel obtained from the Donations System database

Figure 4 depicts the metamodel obtained by the extractor when it analyzed the Donations System database. The model on the top of Fig. 3 conforms to this metamodel, which can be divided in three parts. The top part is composed by elements that are generic in nature and are usually found in generic meta-metamodels (e.g., MOF, Ecore): `Type`, `Attribute`, `Relation`, `Instance`, `AttributeValue`, and `InstanceRelation`. These elements appear in every metamodel created by an extractor, independently of the information source analyzed. In the models, the elements with stereotype `<<type>>` are instances of `Type`. The second part of the metamodel is composed by the elements `Donor`, `Donation`, and `Campaign`. These are all *linguistic instances* of `Type`, and they match the `<<types>>` that were previously described in the model. Finally, the third part of the metamodel is formed by `DonorInstance`, `DonationInstance`, and `CampaingInstance`. In the models previously described, the elements with stereotype `<<instance>>` are instances of the elements in this third part of the metamodel.

There are several reasons that explain the apparent complexity of this metamodel. First of all, relations of ontological and linguistic instantiation are put together instead of creating a separated meta-meta-model. This strategy was selected mainly because current, mainstream modeling tools only support two modeling levels (model and metamodel) and thus making it impossible to introduce a meta-meta-model. Furthermore, metamodel weaving (or type-level weaving), which will be discussed in the next section, is easier to perform when some elements are constant (in this case, the meta-meta-elements). Based only on this, one could wonder whether the rest of the metamodel is really necessary, since `<<types>>` are found in the models. The answer to this is twofold. On the one hand, the case of database analysis is special, because tables determine conceptual *types* which are important also in the model level; analyzing other types of information sources, such as BPEL specifications, does not produce any `<<types>>`, just `<<instances>>`. On the other hand, for many of the actions that we want to perform on the models, it is important to have a *domain metamodel* instead of a *generic* or *linguistic* one. On top of making analysis and queries much more powerful, it simplifies the specification of operations. This is analogous to defining a specific data model for a relational database instead of using a generic data model. Finally, an unexpected element in the metamodels are the attributes and relations found in the `Instances`. This is really duplicated information that can also be found in the types. The explanation for this lays again in the limitations of current modeling tools: they cannot support a mechanism similar to *potencies* [7] to force `<<type>>` to have certain attributes or relations. However, putting the attributes and relations in the `Instances` accomplishes the same goal.

The extractor used to collect this information works as follows. Firstly, it queries the database catalog to know the structure of the schema, including the tables, columns and keys. This information is stored in a temporal model, with a fixed structure. Afterwards, the extractor queries each table to obtain information about the registers, and stores that information in the temporal model.

Finally, the obtained model is transformed to obtain the definitive metamodel, which depends on the original structure of the database schema, and the definitive model.

In addition to the relational database extractor, the BPO case study required other extractors for a number of sources such as the BPEL engine, the BPEL process description files, the ESB Engine (for services, routes and transformations), the LDAP service, BPMN models (built with Bizagi Process Modeler[3]), and ArchiMate models (built with the Archi editor[4]), among others.

There is a further kind of extractors that stands apart to the ones that have been described: extractors that collect information about behavior. What makes these extractors special is that, instead of focusing on structural information, they focus on information about the behavior of elements in the architecture. For example, they can observe the execution logs of a BPEL engine and collect information about the frequency of execution of each process, the frequency of each path within the process, and the frequency of errors. This capacity adds a new dimension to enterprise modeling that is seldom available, and makes it possible to use the resulting models for purposes such as dynamic analysis and simulation.

4.2 Cross-Domain Types and Instance Weaving

Since extractors are completely independent, it is only logical that their outputs are completely independent as well. However, a great part of the value of an enterprise model lies in the identification and *reification* of relations between domains. Unless these relations are materialized, it is impossible to perform analysis involving multiple domains.

In order to obtain a single enterprise model, our approach uses two kinds of model-weaving techniques:

1. **Weaving of types:** this is the kind of weaving performed between the metamodels and also between elements marked with the `<<type>>` stereotype in the models.
2. **Weaving of instances:** this is the kind of weaving performed between elements marked with the `<<instance>>` stereotype.

Combined, these two types of weaving are capable of producing a unified tuple containing an enterprise model and its metamodel. The rest of this section presents in more detail these two kinds of weaving. In EM-AutoBuilder they are implemented on top of a model-weaving machine that is controlled using two different textual languages. These languages are enough to demonstrate the expressiveness and capacity of the machine, but we are in the process of developing a more usable interactive interface.

[3] Bizagi Process Modeler homepage: http://www.bizagi.com/index.php/en/products/bizagi-process-modeler.

[4] Archi homepage: http://archi.cetis.ac.uk/.

Type-Level Weaving. The first type of weaving that we are going to describe is the one that is done between elements marked as `<<type>>`. This process is performed at two levels: on the one hand, it is performed at the metamodel level, to obtain a single metamodel for the enterprise model; on the other hand, it is performed at the model level, to describe the same relations within the enterprise model. The process is performed at both levels following a single specification (i.e., it is not necessary to write one weaving script for each level), which has to be written by a modeler after inspecting the types produced by the extractors. Ideally, this task should be automated, but past experiences show that metamodel-weaving always requires human intervention [6]. Furthermore, this is a task that has to be done only when a new extractor is introduced.

The type-level weaving specification is written using a language based on Fusion [9]. Fusion is a simple language and a platform for weaving metamodels, which we developed for a previous project. Compared to other approaches, such as ATL, Fusion is extremely simple. However, it is powerful enough for its intended purpose, and it is more than enough for specifying type-level weavings. In fact, we only need to use two of the operators included in Fusion: `mergeEntities` and `newLink`. The script in listing 1.1, which is based on the example presented in Sect. 4.1, illustrates the language. This script is processed by the weaving engine that is part of the EM-AutoBuilder Core, and which is derived from the Fusion Weaver.

```
1 mergeEntities Donor {
2   entity1 = DB_DON.Donor
3   entity2 = DB_CCA.CustomerContact
4 }
5
6 newLink responsible {
7   source = DB_DON.Donation
8   target = DB_CCA.Agent
9   containment = false
10  minCard = 0;
11  maxCard = 1;
12 }
```

Listing 1.1. Example of type-level weaving

The example has two parts which depend on the metamodels called DB_DON and DB_CCA. The former refers to the metamodel produced by the extractor that explored the database of the Donations System. The latter refers to the metamodel produced by the extractor that explored the database of the CCA system. In the first part of the example, it is specified that the types `Donor` and `CustomerContact` must be merged into a single type, in the resulting metamodel, also called `Donor`. This kind of operation is used when the same concept appears in two different domains and must be merged. This results in the creation of a new type with a union of the attributes and relations of the original types. In this case, the new type is called `Donor`, has attributes `name` and `tel`, and has relations to `Donation` and `Register`.

The second part of the listing specifies a new relation called `responsible` between the type `Donation` and the type `Agent`. In this case, the relation is added because the modeler wants to make explicit the relation between a certain donation and the agent in the call center that got it. Finally, the types in the source metamodels that are not mentioned in the Fusion script, are simply copied into the combined metamodel.

While the above example is rather simple, it illustrates all the capacities required to create a unified metamodel. In the BPO case study we specified many other relations between elements from different domains. Some of these relations are represented in Table 1. In this table, the types in the columns and rows have been grouped by the domain that they represent, and an `X` has been included when the types have been related in the unified metamodel. For example, a new relation has been added between `BPMN/Process` and `BPEL/Process` to represents the fact that some BPEL processes are automated versions of BPMN processes. Similarly, the `X`s between `BPEL/Partner Link` and `Application/Service` and `External Application/Service` serve to relate BPEL processes and the applications that they rely on, by means of the processes' partner links and the services that they are bound to.

Instance-Level Weaving. The elements described in the previous section are only useful to weave elements marked with the `<<type>>` stereotype. Now we show how elements marked with the `<<instance>>` stereotype are woven in order to produce the expected unified model. In this point, it is important to note that the potential number of `instances` is much larger than the potential number of `types`. Therefore, it is important to have a largely automated process to produce the final woven model. Furthermore, it should be possible to easily run the instance-weaving scripts whenever the model has to be updated.

The language to describe the weaving process between `instances` is a reflection of the language used at the type level, and thus it has two weaving instructions. The first instruction, `mergeInstances`, is used to specify which instances have to be combined, following the merging of their types. The script in listing 1.2 specifies that `Donors` and `CustomerContact` should be combined into a `Donor` whenever the attribute `name` of the former matches the attribute `name` of the latter.

```
1 mergeInstances Donor where
2   (entity1.name == entity2.name)
```

Listing 1.2. Example of usage of instruction mergeInstances

The second instruction in the instance weaving language, `relateInstances`, is used to create relations between instances, based on the new relations created in the metamodel. The script in listing 1.3 continues the example and illustrates this instruction.

Table 1. Selected cross-domain relations in the BPO case study

	e1	e2	e3	e4	e5	e6	e7	e8	e9	e10	e11	e12	e13	e14	e15	e16	e17	e18
DB_DOM																		
e1: Donor				x						x	x							
e2: Donation						x				x	x							
e3: Campaign										x	x							
DB_CCA																		
e4: CustomerContact	x									x	x							
e5: Register										x	x							
e6: Agent		x								x	x		x	x				
BPEL																		
e7: Partner Link															x	x		
e8: Process											x				x	x	x	
e9: Invoke															x	x		
e10: Execution Log	x	x	x	x	x	x					x	x			x	x	x	
BPMN																		
e11: Process	x	x	x	x	x	x				x							x	
e12: Role										x				x				x
LDAP																		
e13: User						x				x								
e14: Org. Unit						x						x						
Internal Application																		
e15: Service							x	x	x	x								
External Application																		
e16: Service							x	x	x	x								
ArchiMate																		
e17: Business Service								x		x	x							
e18: Business Role												x						

The semantics of this script is as follows. Firstly, the model weaving engine has to lookup all the instances of the type `Donation`, which was the `source` type when the new relation was added. Then, for each `Donation` the engine has to find `Agents` using the expression in the second line of the script. This expression is written with the Cumbia Navigation Language [10], which is based mainly on the Object-Graph Navigation Language (OGNL)[5], the JSP Expression Language

```
1 relateInstances Donation.responsible ->
2   #this.donor.callLog.caller
```

Listing 1.3. Example of usage of instruction relateInstances

[5] OGNL Language Guide: http://www.opensymphony.com/ognl/html/LanguageGuide/index.html.

[11], and the MVEL[6]. The expression in the example specifies a path from each `Donation` to 0 or more `Agents`, following the relations `donor`, `callLog`, and `caller`.

Instance weaving instructions are executed in their order of appearance. Thus, when listing 1.3 is run, the `Donors` have already been merged with the `CustomerContact`, thus making `donor.callLog` a valid relation. Figure 5 depicts the resulting merged model.

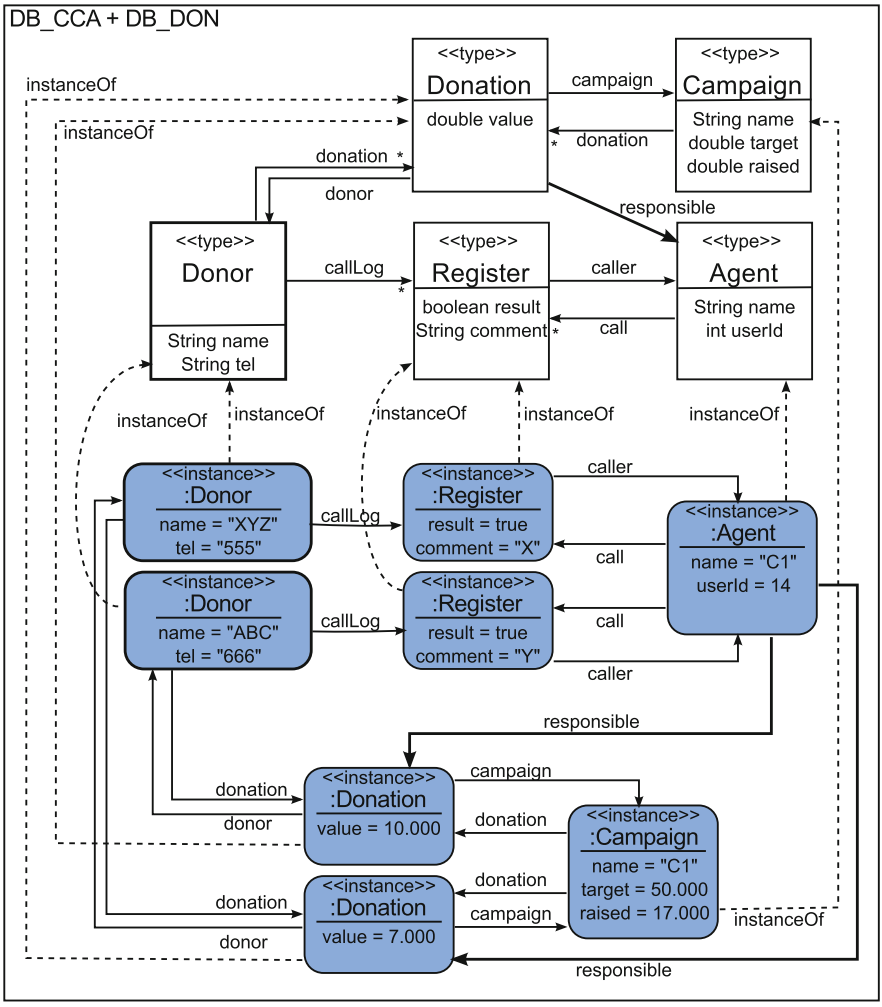


Fig. 5. Model obtained from merging the CCA and Donation models

[6] Language Guide for MVEL 2.0: http://mvel.codehaus.org/Language+Guide+for+2.0.

When all the instance-weaving instructions have been executed, the metamodel and the enterprise model are finally ready to be used.

4.3 Compatibility Concerns and Using the Models

The result of the weaving process is a model and the metamodel it conforms to. By themselves, these artifacts are not really valuable. However, with adequate tools a lot of value can be gained from them.

In consequence, a key requirement for EM-AutoBuilder was to produce models and metamodels that were compatible with other tools. In the end, the tool was given the capacity to export the artifacts to XMI and ecore. Thus, the enterprise models obtained are easily compatible with other tools based on EMF, and even with tools that are capable of using XMI models. Furthermore, the models can be converted from XMI into other representations whenever it is necessary.

Figure 6 presents a screenshot of a fragment of the model obtained during the case study. In the figure, the model was loaded into a tool for supporting visual analysis of models [12]. Within this tool, it is possible to configure how certain characteristics of the model and its elements should be mapped to visual attributes. For example, colors can be mapped to domains, sizes can be mapped to the number of relations, and forces (which the tool uses for calculating layouts) can be mapped to topological proximity. Other tools that can use these enterprise models include generators of documentation, quantitative and qualitative analysis tools (e.g., see [13]), and reporting tools.

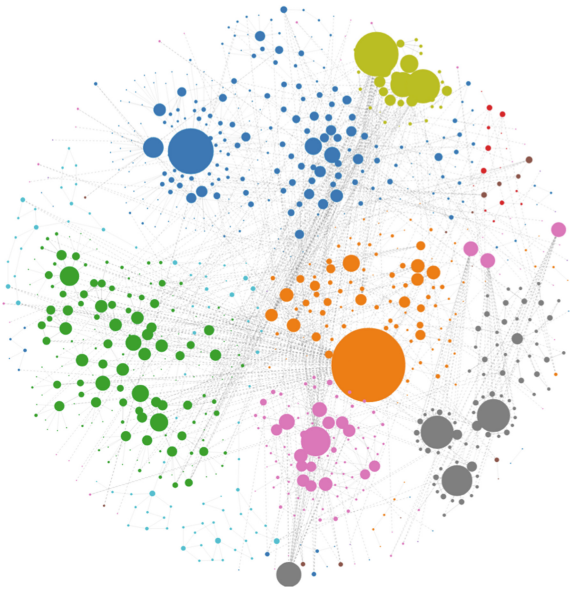


Fig. 6. Visualization of a fragment of the BPO model

5 Conclusion

In this paper we have studied the problematic of building enterprise models that are accurate, complete, structured, and up-to-date. For this purpose, we presented an approach to build and maintain enterprise models in a largely automated way, and we presented the tool that implemented the approach: EM-AutoBuilder. EM-AutoBuilder provides fixed and variable elements. The variable part is represented by a framework and an API for the implementation of extractors that connect to specific information sources. Among the fixed elements, there is a core that hosts the extractors, weaves the models and metamodels, and produces a single model and a single metamodel. This core is not completely automatic: it requires some input from a modeler in order to know how to put together the different pieces that should form the complete model.

In Sect. 2 we stated five desirable qualities for enterprise models. The presented approach, and the EM-AutoBuilder, are helpful for obtaining all of these qualities: the resulting models are *accurate*, because their base information comes from trustworthy sources; they are *well structured* because they conform to known metamodels; they are *complete* with respect to the information available as long as the right extractors are used; finally, the *cost* of building them may not be low if new extractors have to be developed from scratch, but the maintenance cost to keep them up to date is low compared to traditional EM methods.

The presented approach and its implementation have been tested so far with a case study, but they are starting to be used in project with a big company. Within this project, we are also building a reusable library of extractors. These will include extractors for source code, and for performing security vulnerability analysis, which will produce enterprise models *annotated* with security and risk information.

References

1. Buschle, M., Holm, H., Sommestad, T., Ekstedt, M., Shahzad, K.: A tool for automatic enterprise architecture modeling. In: CAiSE Forum. CEUR Workshop Proceedings, vol. 734, pp. 25–32. CEUR-WS.org. (2011)
2. Binz, T., Leymann, F., Nowak, A., Schumm, D.: Improving the manageability of enterprise topologies through segmentation, graph transformation, and analysis strategies. In: Proceedings of the 2012 IEEE 16th International Enterprise Distributed Object Computing Conference (EDOC '12), pp. 61–70. IEEE Computer Society, Washington, DC (2012)
3. Bruneliere, H., Cabot, J., Jouault, F., Madiot, F.: MoDisco: a generic and extensible framework for model driven reverse engineering. In: Proceedings of the IEEE/ACM International Conference on Automated Software Engineering (ASE '10), pp. 173–174. ACM, New York (2010)
4. Schmerl, B., Garlan, D., Yan, H.: Dynamically discovering architectures with DiscoTect. In: Proceedings of the 10th European Software Engineering Conference held Jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering (ESEC/FSE-13), pp. 103–106. ACM, New York (2005)

5. Song, H., Huang, G., Xiong, Y., Chauvel, F., Sun, Y., Mei, H.: Inferring meta-models for runtime system data from the clients of management APIs. In: Petriu, D.C., Rouquette, N., Haugen, Ø. (eds.) MODELS 2010, Part II. LNCS, vol. 6395, pp. 168–182. Springer, Heidelberg (2010)
6. Fabro, M., Bézivin, J., Jouault, F., Breton, E., Gueltas, G.: AMW: a generic model weaver. In: Proceedings of the 1ere Journée sur l'Ingénierie Dirigée par les Modèles (IDM05) (2005)
7. Atkinson, C., Kuhne, T.: Rearchitecting the UML infrastructure. ACM Trans. Model. Comput. Simul. **12**(4), 290–321 (2002)
8. Kuhne, T.: Matters of (meta-) modeling. Softw. Syst. Model. **5**(4), 369–385 (2006)
9. Florez, H., Sánchez, M., Villalobos, J.: EnAr-Fusion. A Tool for Metamodel Composition. Technical Report, Universidad de los Andes, ISIS-01-2012 (2012) http://backus1.uniandes.edu.co/~enar/dokuwiki/doku.php?id=fusion
10. Sánchez, M.: Executable Models for Extensible Workflow Engines. Ediciones Uniandes, Bogotá (2012)
11. Armstrong, E., Ball, J., Bodoff, S., Carson, D.B., Evans, I., Green, D., Haase, K., Jendrock, E.: JSP expression language. In: The J2EE 1.4 Tutorial, ch. 12: JavaServer Pages Technology, pp. 499–506. Sun Microsystems Inc., Santa Clara (2005)
12. Naranjo, D., Sánchez, M., Villalobos, J.: Connecting the dots: examining visualization techniques for enterprise architecture model analysis. In: Grabis, J., Kirikova, M., Zdravkovic, J., Stirna, J. (eds.) Short Paper Proceedings of the 6th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modeling (PoEM 2013), vol. 1023, pp. 29–38. CEUR-WS (2013)
13. Ramos, A., Gomez, P., Sánchez, M., Villalobos, J.: Automated enterprise-level analysis of ArchiMate models. In: Bider, I., Gaaloul, K., Krogstie, J., Nurcan, S., Proper, H.A., Schmidt, R., Soffer, P. (eds.) BPMDS 2014 and EMMSAD 2014. LNBIP, vol. 175, pp. 439–453. Springer, Heidelberg (2014)

Towards Multi-perspective Process Model Similarity Matching

Michael Heinrich Baumann, Michaela Baumann,
Stefan Schönig(✉), and Stefan Jablonski

University of Bayreuth, Bayreuth, Germany
{michael1.baumann,michaela1.baumann,
stefan.schoenig,stefan.jablonski}@uni-bayreuth.de

Abstract. Organizations increasingly determine process models to support documentation and redesign of workflows. In various situations correspondences between activities of different process models have to be found. The challenge is to find a similarity measure to identify similar activities in different process models. Current matching techniques predominantly consider lexical matching based on a comparison of activity labels and 1-to-1-matchings. However, label based matching probably fails, e.g., when modellers use different vocabulary or model activities at different levels of granularity. That is why we extend existing methods to compute candidate sets for N-to-M-matchings based on power-sets of nodes. Therefore, we impose higher demands on process models as we do not only consider labels, but also involved actors, data objects and the order of appearing. This information is used to identify similarities in process models that use different vocabulary and are modelled at different levels of granularity.

Keywords: Business process model · Process similarity · Model matching

1 Introduction

Organizations increasingly determine business process models for supporting the documentation and redesign of actual workflows as well as information system implementation [9]. In order to cover all the different peculiarities of a process typically several expert modellers from diverse business domains are involved in modelling activities [5].

In various situations correspondences between elements of different process models have to be found, e.g., when analysts of different departments modelled the same process or when merging similar processes of recently merged companies [1]. Furthermore, it is conceivable to detect correspondences in conjunction with process improvement, pattern identification or increase of efficiency. The challenge is to find a similarity measure to identify similar activities in different process models [1]. Therefore, current process model matching techniques

J. Barjis and R. Pergl (Eds.): EOMAS 2014, LNBIP 191, pp. 21–37, 2014.
DOI: 10.1007/978-3-662-44860-1_2

predominantly consider lexical matching scores based on a comparison of activity labels that appear in process models [12]. Furthermore, these methods only consider 1-to-1-matchings, i.e., only single nodes are compared per model [1].

Think about analysts of different departments who model the same process. Some analysts use a more technical vocabulary than others and some analysts tend to get more granular when modelling the process. As a consequence, it is likely that activity labels of resulting models considerably deviate from each other or that activities are divided in different chunks. In such situations, label based matching methods probably fail, i.e., lead to a low recall [6].

The intention of the work at hand is to find an adequate similarity measure to identify similar activities in process models that use rather different vocabulary and are modelled at different levels of granularity. Therefore, we extend existing process model matching methods to compute candidate sets for N-to-M-matchings based on power-sets of nodes. That is why we compute similarity measures to identify similar sets of activities in different process models. Of course, this implicates a considerably higher complexity. Furthermore, it is useless or even impossible to use only label matching to analyse sets of activities. Therefore, we impose higher demands on process models as we do not only want to match activities based on their labels, but also by analysing involved actors, data objects and their order of appearing in the model. Our approach is based upon the different perspectives of the perspective-oriented process modelling approach [7]. Consider the simple example of Fig. 1. Here, we identified a similarity between the activities A and B of the first model and the activity C in the second model since the combination of data objects produced by A and B relates to the set of data objects produced by C.

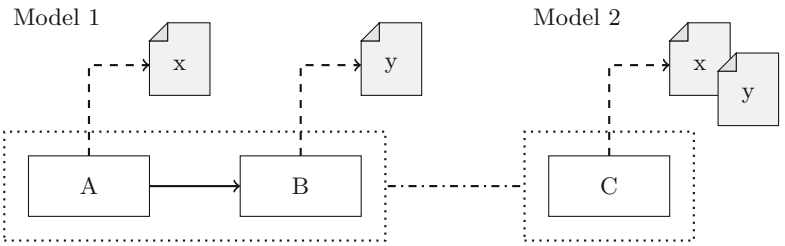


Fig. 1. Example process model similarity matching based on data objects

This information can help to reduce computation time and to identify similar activities in process models that use different vocabulary and are modelled at different levels of granularity.

2 Background and Related Work

The problem of matching two process models of a general form has already been discussed in several papers, like in [1–3,10–16]. A lot of modeling notations

are available to capture business processes, e.g., Event-driven Process Chains (EPCs), UML Activity Diagrams and the Business Process Modeling Notation (BPMN) [9]. In the work at hand, we seek to abstract as much as possible from specific notations and therefore a process model is given according to following definition.

Definition 1 (process model). *Let* $\mathcal{L} \subset \{s_1 s_2 \dots s_n \mid s_i \text{ is a character } \forall i \in \{1, 2, \dots, n\}, n \in \mathbb{N}\}$ *be a set of labels. Then, a process graph is a tuple* (N, E, λ)*, where*

- N *is a set of nodes,*
- $E \subseteq N \times N$ *is a set of edges and*
- $\lambda : N \to \mathcal{L}$ *is a function, that maps nodes to labels.*

To determine the similarity between such models, first, the similarity of two nodes, i.e., of their labels has to be specified. This is usually done with a construct called string-edit similarity which is a measure for how strong one string resembles another one. It is defined with help of the so-called string-edit distance, *sed*. The string-edit distance of two strings is the minimal number of atomar string operations, that means insertion, deletion and substitution of one character, that is needed to transform one string into the other. Thus, the string-edit distance is an integer with a value not more than the length of the longer string.

Definition 2 (string-edit similarity). *For two strings* s *and* t*, the string-edit similarity* Sim *is given through*

$$Sim(s,t) = 1 - \frac{sed(s,t)}{\max(|s|, |t|)}.$$

Sim takes values between 0 and 1, where 1 can only be reached, if *sed* equals 0, that means the two compared strings are the same. As mentioned in [1,3], it is also possible to do stemming before computing the string-edit similarity in order to get better values. Stemming is the name for a collection of several techniques, like deleting symbols, fillers, often and repeatedly used words, reducing words to their stem, translating words into one language, sorting words, etc. to get a standardized basis for the strings that have to be matched [17]. If possible, one can even use synonym dictionaries or a thesaurus, as suggested in [3], or ontologies to get optimal results for comparing two strings. Of course, the problem of homonyms cannot be solved this way, and there is no help when it comes to spelling errors or neologisms. In [6] another label-based similarity is proposed, namely the basic bag-of-words similarity which may be combined with label pruning.

The next step in getting an optimal matching of two models $G_1 = (N_1, E_1, \lambda_1)$ and $G_2 = (N_2, E_2, \lambda_2)$ is to consider a partial and injective mapping $M : N_1 \to N_2$, that maps nodes of G_1 to nodes of G_2. This mapping is partial, as not all nodes of G_1 have to be mapped, and injective, as not all nodes of G_2 have to be met and those that are in the image of M may only have a one-elemental

inverse image. If $|N_1| < |N_2|$, not all nodes in N_2 can be met by M. With this mapping M, all nodes and edges of the two graphs G_1 and G_2 are element of one of the following sets.

Definition 3 (Substituted and deleted nodes). *For a mapping M as defined above, the set*

$$subn := \{n \in N_1 \cup N_1 \mid n \text{ is in the image or the inverse image of } M\}$$

is the set of all substituted/mapped nodes. Accordingly,

$$skipn := (N_1 \cup N_2) \setminus subn$$

is the set of all deleted nodes.

Definition 4 (Substituted and deleted edges). *Consider the mapping M. For all edges of G_1 and G_2 we say that an edge $(n_1, m_1) \in E_1$ is deleted from G_1 if there is no $(n_2, m_2) \in E_2$ with $M(n_1) = n_2$ and $M(m_1) = m_2$, and vice versa. Then*

$$skipe := \{(n, m) \mid (n, m) \text{ is deleted}\}$$

is the set of all deleted edges and

$$sube := (E_1 \cup E_2) \setminus skipe$$

is the set of all substituted/mapped edges.

Subsequently, the graph-edit similarity, a value of how good two graphs match, is computed with the shares of deleted nodes and edges compared to their total number and an average distance of the substituted edges where the variables *skipn*, *skipe*, *subn* and $Sim(\cdot, \cdot)$ are defined as in Definitions 2, 3 and 4. These values are given as

- $fskipn = \dfrac{|skipn|}{|N_1| + |N_2|}$ (share of deleted nodes)
- $fskipe = \dfrac{|skipe|}{|E_1| + |E_2|}$ (share of deleted edges)
- $fsubn = \dfrac{2 \cdot \sum_{(n_1,n_2)\in M}(1 - Sim(n_1, n_2))}{|subn|}$
  (average distance of substituted nodes)

All these shares are element of the interval $[0, 1]$ and especially $fsubn$ takes values near 1, when there's not much similarity between the two compared graphs. Combining these three values with some weight factors *wskipn*, *wskipe* and *wsubn* that are element of $[0, 1]$ and sum up to 1 leads to the graph-edit similarity defined as

Definition 5 (graph-edit similarity induced by M). *For two models G_1 and G_2 and a mapping M the graph-edit similarity induced by M, $GSim_M$, is given through*

$$GSim_M(G_1, G_2) = 1 - (wkipn \cdot fskipn + wskipe \cdot fskipe + wsubn \cdot fsubn).$$

To get the best matching, that means the best mapping M between the two models, the graph-edit similarity induced by M has to be maximized with respect to M. The resulting value is called graph-edit similarity.

Definition 6 (graph-edit similarity). *The graph-edit similarity GSim for G_1 and G_2 is obtained as*

$$GSim(G_1, G_2) = \max_M GSim_M(G_1, G_2).$$

For the implementation of this maximization problem, efficient algorithms are used, like Greedy or A*-Algorithms (see e.g. [2]), as the problem of finding the best M is of exponential order. With one of these algorithms it is now possible to efficiently match two process models on the same level of abstraction using a similar vocabulary. Obviously, by comparing process models with a strongly differing number of nodes the method presented so far will not provide satisfying results. Furthermore, a lot of information contained in the models is not considered. In [3] there are mentioned some possibilities to expand this matching technique and not only use the nodes' labels, as they might not lead to the desired results, but also their context with predecessor and successor relations. In [1] only the idea of expanding this method to more than one node in a successive/iterative way is mentioned. Nevertheless, all methods described so far are based on mapping single nodes to single nodes. Reference [8] introduces 1-to-n matchings, however, it does not imply other perspectives of business process models and only focuses on sequence flows during analysis. To eliminate the discussed disadvantages of existing techniques a method based on mapping sets of nodes to sets of nodes will be introduced.

3 Extended Definitions for Graph Matching

The work at hand expands previous ideas of single node matching to a procedure where sets of nodes are matched. Therefore, we impose higher demands on the process models as we do not only want to match nodes based on their description, but also based on involved positions, data objects and on their order of appearing in the model. Therefore, our approach is based upon the different perspectives of the perspective-oriented process modelling approach defined in [7]. We need these additional perspectives to get better matches, as the descriptions differ very much when comparing sets of nodes, and to reduce a combinatoric explosion, that results from the exponential number of matches we have to check. We also make some additional assumptions for these perspectives. Involved positions are arranged in some kind of tree, that represents their hierarchical structure. This tree can be seen as a combination of an organigram and maybe a population, which we use to reduce complexity of the model and to avoid introducing another mapping. Furthermore, all data objects appearing in the process models need to have a unique identifier. Based on these assumptions we can define an extended process model.

Definition 7 (Extended process model). *Let* $\mathcal{B} \subset \{s_1 s_2 \dots s_{n_\mathcal{B}} \mid s_i \text{ is a character } \forall i \in \{1, 2, \dots, n_\mathcal{B}\}, n_\mathcal{B} \in \mathbb{N}_0\}$ *be a set of descriptions,* $\mathcal{A}$ *a hierarchical tree of positions with nodes* a *and levels* e*, and* $\mathfrak{D} = \{D_1, \dots, D_{n_\mathfrak{D}}\}$ *a finite set of data objects. Then a process graph is a tuple* (N, E, λ) *with*

- N *being a set of nodes,*
- $E \subseteq N \times N$ *a set of edges and*
- $\lambda : N \to \mathcal{B} \times \mathcal{A} \times \mathcal{P}(\mathfrak{D})$ *a function, that maps nodes to entities.*

For all process models to be matched, the sets $\mathcal{B}$*,* $\mathcal{A}$ *and* $\mathfrak{D}$ *have to be the same. Note, that* $\mathcal{P}(\cdot)$ *indicates the power set.*

Taking two process models, represented by their graphs, we define the extended graph-edit-similarity under consideration of the following sets.

Definition 8 (Set of deleted and substituted nodes). *Let* $G_i = (N_i, E_i, \lambda_i)$*,* $i = 1, 2$ *be two models and* $P_i \subset \mathcal{P}(N_i) \ni \emptyset$ *a complete and disjoint partition of* N_i *(i.e.* $\bigcup_{p \in P_i} p = N_i$ *&* $\forall p, p' \in P_i : p \cap p' = \emptyset, p \neq p'$*),* $i = 1, 2$*. Further, let* $M : P_1 \to P_2$ *be a bijective function (*$\emptyset \mapsto p_2$ *and* $p_1 \mapsto \emptyset$ *means, that* p_2 *and* p_1 *are deleted, respectively,* $p_1 \in P_1$*,* $p_2 \in P_2$*), where* $\neg(\emptyset \mapsto \emptyset)$*. Then*

$$\begin{aligned} skipn = &\{n_1 \in N_1 \mid n_1 \in p_1 \in P_1 : p_1 \overset{M}{\mapsto} \emptyset\} \\ \cup &\{n_2 \in N_2 \mid n_2 \in p_2 \in P_2 : \emptyset \overset{M}{\mapsto} p_2\} \end{aligned}$$

is the set of deleted nodes and

$$subn = (N_1 \cup N_2) \setminus skipn$$

is the set of substituted nodes.

Definition 9 (Set of deleted and substituted edges). *Let* $E_i^* = \{(a, b) \in E_i \mid \exists p_i \neq p_i' \in P_i : a \in p_i, b \in p_i'\}$ *be a set of edges, that connect nodes from different elements of* P_i*, i.e., start node* a *is in* p_i *and end node* b *is in* p_i' *with* $p_i \neq p_i' \in P_i$*. Thus, with the assumption that* $p_1 \neq p_1', p_2 \neq p_2'$ *we name with*

$$\begin{aligned} sube = \{(a, b) \in E_1^* \mid &a \in p_1 \in P_1, b \in p_1' \in P_1, M(p_1) = p_2, M(p_1') = p_2', \\ &\exists a' \in p_2 \in P_2, b' \in p_2' \in P_2 : (a', b') \in E_2^*\} \\ \cup \{(a', b') \in E_2^* \mid &a' \in p_2 \in P_2, b' \in p_2' \in P_2, M(p_1) = p_2, M(p_1') = p_2', \\ &\exists a \in p_1 \in P_1, b \in p_1' \in P_1 : (a, b) \in E_1^*\} \end{aligned}$$

the set of substituted edges, i.e., the set of edges, that remain connectors of mapped pairs of nodes. Like in the definition above, let

$$skipe = (E_1^* \cup E_2^*) \setminus sube$$

be the set of deleted edges.

As one can see, we do not need to consider those edges, that connect nodes both being inherent in the same element of P_i. The similarity can now be defined analoguously to that in the section before. However, we still need to determine the similarity of two sets of nodes p_1 and p_2, "$Sim(p_1, p_2)$". Therefore, we consider different perspectives of nodes separately from each other and define a similarity value for each perspective. These values can then be combined with respect to some weight factor resulting in a global similarity value. In the next section, we look upon the four perspectives given in the process model and how we compute a similarity value for each perspective.

4 Similarity Between Sets of Nodes

We will start with the nodes' description, i.e., the activity label, as we can apply a modification of the similarity concepts of Sect. 2, i.e., a modification of string-edit similarity. For positions, we examine their hierarchical structure in form of the given trees. Data objects have the function of an exclusion criterion due to their unique identifiers. Finally, we examine the order of sets of nodes where we need the concept of partial orders. In this paper, we focus on sequential process models which should be extended in future.

The Functional Perspective. As mentioned above, for the functional perspective, i.e., the nodes' description, we can apply the well-known concepts of label matching, like string-edit similarity. The only difference is that we have to apply it to a set of nodes, i.e., to a set of strings. For this, we concatenate the descriptions of each node of the two sets with whitespace and in their order of appearance in the model.

Definition 10 (Extended string-edit similarity). *Let P_1 be a partition of graph G_1 and P_2 a partition of G_2. Then, with $p_1 \in P_1$ consisting of nodes $n_1, \ldots, n_k$ with description strings $s_1, \ldots, s_k$ and $p_2 \in P_2$ consisting of nodes $m_1, \ldots, m_l$ with description strings $t_1, \ldots, t_l$, we indicate with $s_1 . \vee \ldots \vee . s_k$ and $t_1 . \vee \ldots \vee . t_l$ the concatenated descriptions of p_1 and p_2. The string-edit similarity of p_1 and p_2 is then defined as*

$$BSim(p_1, p_2) = 1 - \frac{sed(p_1, p_2)}{\max(|p_1|, |p_2|)},$$

where $sed(p_1, p_2) = sed(s_1 . \vee \ldots \vee . s_k, t_1 . \vee \ldots \vee . t_l)$ is the string-edit distance of p_1 and p_2. $|p_i|$ stands for the length of the respective, underlying, concatenated string.

The range of $BSim$ is in [0, 1] with a value of 1 if p_1 and p_2 have the same descriptions and a value close to 0 if they differ very much. Of course it is clear that comparing two sets of nodes with a strongly different number of elements, the result of $BSim$ has no chance to come close to 1. Thus, we strongly recommend using stemming-techniques like mentioned in Sect. 2.

The Data/Dataflow Perspective. Our intention is to compute a number that is 0 if the compared sets use completely different data objects and increases to 1 if all used objects appear in both sets. Furthermore, this perspective is meant to fulfill some important function in the context of practicability of our approach to distinctly reduce computation time of the hyper-exponential problem. That means if under a certain mapping M at least one assignment $p_1 \mapsto p_2$ has 0 similarity in the data/dataflow perspective the whole mapping M gets a similarity value of 0 and must not be considered any longer. It is likely that a lot of mappings can be rejected before their concrete similarity values have to be computed.

Before we define a similarity for occurring data objects of sets of nodes, we first have to specify a value for sets of data objects, as in one node more than one data object can be listed.

Definition 11 (Similarity for sets of data objects). *For* $D_1, D_2 \subset \mathfrak{D}$, $D_1 \cup D_2 \neq \emptyset$, *we set*

$$DSim(D_1, D_2) = \frac{|D_1 \cap D_2|}{|D_1 \cup D_2|}.$$

If $D_1 = D_2 = \emptyset$ *we set* $DSim(D_1, D_2) = 1$.

In fact, we do not have to look at single sets of data objects, but at all data objects in $p_i \in P_j$, which can be a set of sets of data objects. To handle this construct, we join all data object sets of the nodes and call this new set

$$D_{p_i} := \{D \mid \exists n \in p_i \in P_j : (\lambda(n))_3 = \mathcal{D} \wedge D \in \mathcal{D}\} = \bigcup\nolimits_{n \in p_i} (\lambda(n))_3.$$

Now, we can define a similarity for the data/dataflow perspective for a set of nodes as follows

Definition 12 (Data/dataflow similarity)

$$DSim(p_1, p_2) = DSim(D_{p_1}, D_{p_2}) = \frac{|D_{p_1} \cap D_{p_2}|}{|D_{p_1} \cup D_{p_2}|},\ p_i \in P_i.$$

The Organizational Perspective. A similarity value has to be found that is 1 if the executing positions in the compared sets are the same and decreases to 0 the more organizational distance lies between the involved positions. In comparing the positions of two nodes of a given hierarchical tree it is possible to find a minimal number of edges, $\tilde{k}$, that have to be passed to get from one position to the other and the number of levels, $\tilde{e}$, that lie between them. Two positions on the same level have $\tilde{e} = 0$. To transform these two values into a similarity value, we set

$$ksim(A, B) := \frac{1}{\tilde{k} + 1} \text{ and } esim(A, B) := \frac{1}{\tilde{e} + 1}.$$

Therefore, by comparing a position with itself, we get a value of 1 for both similarities, that means maximal similarity, and a value tending to 0, the more

edges and levels are between two positions. To combine these two similarity measures, we define

$$HSim(A, B) = \alpha ksim(A, B) + (1 - \alpha) esim(A, B),$$

with $\alpha \in [0, 1]$ being a weight factor, that allows to display some preferences for the position similarity. This value $HSim$ has to be extended to work for sets of nodes, that means a set of positions. This is done the following way:

Definition 13 (Organizational similarity). *Let $\mathcal{M}_{p_i} \subset \mathcal{A}$ be the set of positions occurring in $p_i \in P_j$, i.e.,*

$$\mathcal{M}_{p_i} = \{m \mid \exists n \in p_i \in P_j : (\lambda(n))_2 = m\}.$$

Then, for $p_i \in P_j$, it is

$$HSim(p_1, p_2) = \frac{\sum_{m_1 \in \mathcal{M}_{p_1}, m_2 \in \mathcal{M}_{p_2}} HSim(m_1, m_2)}{|\mathcal{M}_{p_1}| \cdot |\mathcal{M}_{p_2}|}.$$

Hence, we compute the similarity of every pair of positions from the two sets, add this values up and divide through the number of pairings to get an average value for position similarity. Note, that if there is more than one tree representing the hierarchical structure of an organization, a comparison of positions from different trees leads to a similarity value of 0.

The Behavioral Perspective. The behavioral perspective is somehow different to the other perspectives, as it is not a component of λ, but given through the nodes' sequential order in a process model. We examine whether the order of elements from P_1 is maintained, turned around or completely mixed up under the mapping M. To define a similarity with respect to this sequence, we use the partial order on P_i which is a result from the complete, disjoint decomposition of $P_i = \{p_i^1, \ldots, p_i^t\} \subset \mathcal{P}(N_i)$ of the i-th model, $i = 1, 2$. Within this partial order, several states may occur, namely

$$\begin{aligned}
p_i \succ p_i' &\Leftrightarrow \forall n \in p_i, n' \in p_i' : n \succ n', \\
p_i \prec p_i' &\Leftrightarrow \forall n \in p_i, n' \in p_i' : n \prec n', \\
p_i \sim p_i' &\Leftrightarrow p_i = p_i' \vee \exists n, m \in p_i, n', m' \in p_i' : n \succ n', m \prec m', \\
p_i &\nsim \emptyset.
\end{aligned}$$

$\succ$ and $\prec$ on N_i is given through the successive order of nodes in N_i. With this notation, we can now assign values to sets $p, p' \in P_1$ by comparing their order to the order of $M(p), M(p') \in P_2$. For this, we first want to distinguish between comparisons involving the empty set and all other pairs of sets and assign following values:

$$\gamma(p, p') = \begin{cases} 0, & \text{if } p \nsim p' \vee M(p) \nsim M(p') \\ 1, & \text{else.} \end{cases}$$

Next, the individual similarity values are assigned for each possible situation:

$$\nu(p,p') = \begin{cases} 1, & \text{if } p \prec p' \wedge M(p) \prec M(p'), \\ 1, & \text{if } p \succ p' \wedge M(p) \succ M(p'), \\ 1, & \text{if } p \sim p' \wedge M(p) \sim M(p'), \\ 0, & \text{if } p \prec p' \wedge M(p) \succ M(p'), \\ 0, & \text{if } p \succ p' \wedge M(p) \prec M(p'), \\ \frac{1}{2}, & \text{if } p \prec p' \wedge M(p) \sim M(p'), \\ \frac{1}{2}, & \text{if } p \succ p' \wedge M(p) \sim M(p'), \\ \frac{1}{2}, & \text{if } p \sim p' \wedge M(p) \prec M(p'), \\ \frac{1}{2}, & \text{if } p \sim p' \wedge M(p) \succ M(p'), \\ 0, & \text{if } p \nsim p' \vee M(p) \nsim M(p'). \end{cases}$$

It is possible to assign other plausible values to the different cases, for example $\frac{3}{4}$ in the third line. Function γ is necessary to make sure, later on, that we do not divide through 0.

5 The Extended Graph-Edit Similarity

The next step is to combine the defined similarity values for the different perspectives in addition to the values of *skipn* and *skipe*. Therefore, we transform all these values into normalized distances where 0 means full similarity and 1 greatest possible distance. We result in getting the following equations:

$$fskipn = \frac{skipn}{|N_1| + |N_2|}$$

is the share of deleted nodes and

$$fskipe = \frac{skipe}{|E_1^*| + |E_2^*|}$$

is the share of deleted edges, considering only the relevant ones. With

$$fsubb = \frac{\sum_{(p_1,p_2)\in M | p_1 \neq \emptyset \neq p_2} (1 - BSim(p_1,p_2))}{\sum_{(p_1,p_2)\in M | p_1 \neq \emptyset \neq p_2} 1}$$

we get an average normalized distance value for the functional perspective with respect to M. Analoguously, we get a value for the organizational perspective through

$$fsubh = \frac{\sum_{(p_1,p_2)\in M | p_1 \neq \emptyset \neq p_2} (1 - HSim(p_1,p_2))}{\sum_{(p_1,p_2)\in M | p_1 \neq \emptyset \neq p_2} 1}.$$

For the data perspective, we have to do a distinction of cases to enable it to work as an exclusion criterion, as explained in Sect. 4. That is why we get

$$fsubd = \begin{cases} 1, \text{ if } \exists \emptyset \neq p \in P_1 : M(p) \neq \emptyset, DSim(p, M(p)) = 0 \\ \frac{\sum_{(p_1,p_2)\in M | p_1 \neq \emptyset \neq p_2} (1 - DSim(p_1,p_2))}{\sum_{(p_1,p_2)\in M | p_1 \neq \emptyset \neq p_2} 1}, \text{ else} \end{cases}$$

for the data perspective. For the behavioral perspective we also need to distinct between some cases, as there exist some degenerated mappings. Considering such mappings, we get

$$fsubv = \begin{cases} \frac{\sum_{p \neq p' \in P_1} (1 - \nu(p,p'))\gamma(p,p')}{\sum_{p \neq p' \in P_1} \gamma(p,p')}, \exists p \neq p' \in P_1 : \gamma(p,p') \neq 0, \\ 1, \text{ for } fskipn = 1, \\ 0, \text{ else.} \end{cases}$$

Now, these normalized distance measures are simply added with some weight factors and transformed back into a similarity measure with greatest possible similarity = 1 and 0 for no similarity to get the extended graph-edit similarity.

Definition 14 (Extended graph-edit similarity induced by M). *With weight factors $wskipn$, $wskipe$, $wsubb$, $wsubv$, $wsubh \in [0,1]$ and $wsubd \in (0,1]$ that sum up to 1 and can be chosen at one's own discretion we get the graph-edit similarity induced by M through*

$$GSim_M(G_1, G_2) = \begin{cases} 0, \text{ if } fsubd = 1, \\ 1 - (wskipn \cdot fskipn + wskipe \cdot fskipe \\ \quad + wsubb \cdot fsubb + wsubd \cdot fsubd \\ \quad + wsubh \cdot fsubh + wsubv \cdot fsubv), \text{ else.} \end{cases}$$

To get the global graph-edit similarity that means the best fitting mapping M we define.

Definition 15 (Extended graph-edit similarity)

$$GSim(G_1, G_2) = \max_M GSim_M(G_1, G_2).$$

For this task, again algorithms like Greedy or A* are used but adjusted to make use of the exclusion criterion, as the problem we focus here is of hyper-exponential order. Using these algorithms, the problem is still of exponential order, but can be made fairly efficient by using the mentioned exclusion criterion as a lot of possibilities are neglected and not completely computed.

6 Case Study and Evaluation

To give a more detailed insight of how the different similarities are computed and applied we present a case study and find the graph-edit similarity of the two

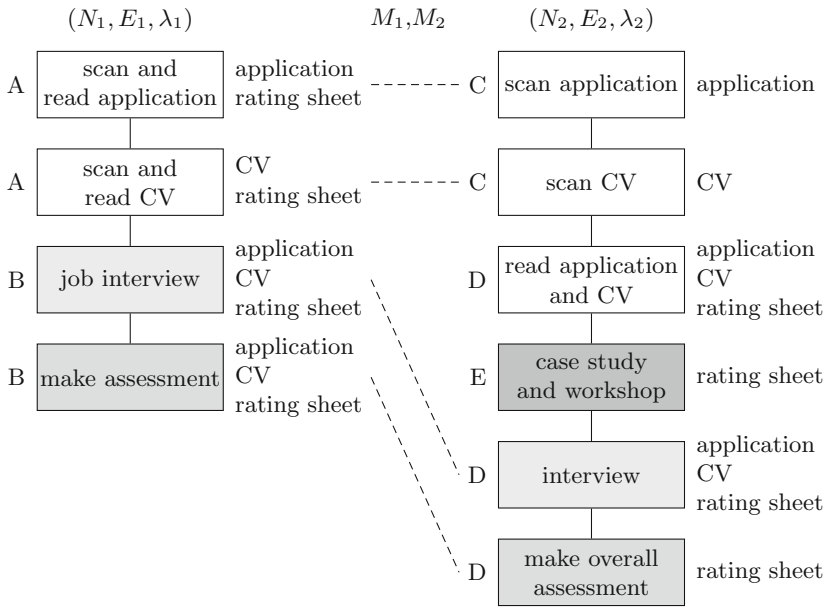


Fig. 2. Two process models for two resembling processes written down differently with 1:1-mapping M_1 and N:M-mapping M_2. The positions at the left-hand side and the data at the right-hand side of the nodes are relevant for mapping M_2.

examplary process models (N_1, E_1, λ_1) and (N_2, E_2, λ_2) under a given mapping M_i. In fact, we will consider two mappings, one 1:1-mapping M_1 and one M:N-mapping M_2. It becomes obvious that our extended approach provides better results than simple node-to-node mappings. For this, our two sequential process models and mapping M_1 are like in Fig. 2, where the nodes mapped by M_1 are indicated with dashed lines. It can be shown, that under application of stemming techniques this mapping M_1 is the best 1:1-mapping for these two models. For computing the graph-edit similarity we need the two values $fskipn = 0.2$ and $fskipe = 0.5$. Applying stemming techniques on the nodes' labels, we get for the average distance of substituted nodes a value of $fsubn \approx 0.54$. With all weights equalling $\frac{1}{3}$ we get for graph-edit similarity with respect to mapping M_1

$$\begin{aligned} GSim(G_1, G_2) = GSim_{M_1}(G_1, G_2) &\approx 1 - \left(\tfrac{1}{3} \cdot 0.2 + \tfrac{1}{3} \cdot 0.5 + \tfrac{1}{3} \cdot 0.54\right) \\ &\approx 0.59. \end{aligned}$$

For our second mapping M_2 we choose the M:N-mapping according to Definition 8 indicated with different colors in Fig. 2. The mappings λ_1 and λ_2 of G_1 and G_2 are given through

- $\lambda_1 : \begin{cases} n_{11} \mapsto (\text{ scan and read application, A, \{ application, rating sheet}\}) \\ n_{12} \mapsto (\text{ scan and read CV, A, \{ CV, rating sheet}\}) \\ n_{13} \mapsto (\text{job interview, B, \{application, CV, rating sheet}\}) \\ n_{14} \mapsto (\text{make assessment, B, \{application, CV, rating sheet}\}) \end{cases}$
- $\lambda_2 : \begin{cases} n_{21} \mapsto (\text{scan application, A, \{application}\}) \\ n_{22} \mapsto (\text{scan CV, A, \{CV, rating sheed}\}) \\ n_{23} \mapsto (\text{read application and CV, B, \{CV}\}) \\ n_{24} \mapsto (\text{case study and workshop, B, \{rating sheet}\}) \\ n_{25} \mapsto (\text{interview, B, \{application, CV, rating sheet}\}) \\ n_{26} \mapsto (\text{make overall assessment, B, \{rating sheet}\}) \end{cases}$

where

- $N_1 = \{n_{11}, n_{12}, n_{13}, n_{14}\}$,
- $N_2 = \{n_{21}, n_{22}, n_{23}, n_{24}, n_{25}, n_{26}\}$,
- $E_1 = \{(n_{11}, n_{12}), (n_{12}, n_{13}), (n_{13}, n_{14})\}$ and
- $E_2 = \{(n_{21}, n_{22}), (n_{22}, n_{23}), (n_{23}, n_{24}), (n_{24}, n_{25}), (n_{25}, n_{26})\}$

and thus the mapping M_2 is the following:

- $M_2 : \begin{cases} \{n_{11}, n_{12}\} =: p_{11} \mapsto \{n_{21}, n_{22}, n_{23}\} =: p_{21} \\ \{n_{13}\} =: p_{12} \mapsto \{n_{25}\} =: p_{22} \\ \{n_{14}\} =: p_{13} \mapsto \{n_{26}\} =: p_{23} \\ \emptyset =: p_{14} \mapsto \{n_{24}\} =: p_{24} \end{cases}$

This leads to edge sets

- $E_1^* = \{(n_{12}, n_{13}), (n_{13}, n_{14})\}$ and
- $E_2^* = \{(n_{23}, n_{24}), (n_{24}, n_{25}), (n_{25}, n_{26})\}$.

With this, the share of deleted nodes has the same underlying set of nodes, whereas the share of deleted edges changes its denominator with respect to this new edge sets and we get

$$fskipn = \frac{1}{4+6} = 0.1$$

and

$$fskipe = \frac{3}{2+3} = 0.6.$$

The next step is to compute $fsubb$. Concatenating the respective descriptions and using the same stemming techniques as before we get for the stemmed descriptions

- sed(application CV read scan, application CV read scan) = 0, $BSim(p_{11}, p_{21}) = 1$
- sed(interview job, interview) = 4, $BSim(p_{11}, p_{21}) = 1 - \frac{4}{13} \approx 0.69$
- sed(assesment make, assessment make overall) = 8, $BSim(p_{11}, p_{21}) = 1 - \frac{8}{23} \approx 0.65$

This leads to a $fsubb$ of value

$$fsubb = \frac{(1-1) + (1 - \frac{9}{13}) + (1 - \frac{15}{23})}{3} \approx 0.22.$$

For $fsubd$ we have to determine $DSim$. It is

- $D_{p_{11}}$ = {application,CV,rating sheet},
- $D_{p_{12}}$ = {application,CV,rating sheet},
- $D_{p_{13}}$ = {application,CV,rating sheet},
- $D_{p_{21}}$ = {application,CV,rating sheet},
- $D_{p_{22}}$ = {application,CV,rating sheet} and
- $D_{p_{23}}$ = {rating sheet}.

So, we get

- $DSim(p_{11}, p_{21}) = 1$, $DSim(p_{12}, p_{22}) = 1$, $DSim(p_{13}, p_{23}) = \frac{1}{3}$

and with that, it is

$$fsubd = \frac{(1-1) + (1-1) + (1 - \frac{1}{3})}{3} = \frac{2}{9} \approx 0.22,$$

as the exclusion criterion does not occur with this mapping M_2.
For computation of $fsubh$ we need the organizational structure for the positions A, B, C, D and E, that is given via the tree in Fig. 3. For the weights, we find it appropriate to give more weight to level similarity, so we choose $\alpha = \frac{1}{4}$. With this, we get

- $HSim(p_{11}, p_{21}) = \frac{HSim(A,C)+HSim(A,D)}{2} = \frac{0.25 \cdot \frac{1}{5} + 0.75 \cdot 1 + 0.25 \cdot \frac{1}{4} + 0.75 \cdot \frac{1}{2}}{2} \approx 0.62$
- $HSim(p_{12}, p_{22}) = HSim(p_{13}, p_{23}) = \frac{HSim(B,D)}{1} = 0.25 \cdot \frac{1}{3} + 0.75 \cdot 1 \approx 0.83$

This leads to a value for $fsubh$ of

$$fsubh \approx \frac{(1 - 0.62) + (1 - 0.83) + (1 - 0.83)}{3} = 0.24.$$

For the last part of the formula for graph-edit similarity we need to compute $fsubv$. For this, we have to consider the order of the sets p_{ij} under our mapping M_2.

- $\gamma(p_{11}, p_{12}) = 1$, $\nu(p_{11}, p_{12}) = 1$,
- $\gamma(p_{11}, p_{13}) = 1$, $\nu(p_{11}, p_{13}) = 1$,

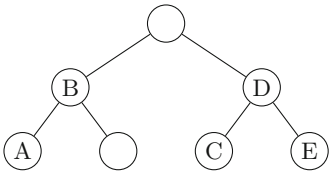


Fig. 3. Organizational structure of the five positions in our example

- $\gamma(p_{11}, p_{14}) = 0$, $\nu(p_{11}, p_{14}) = 0$,
- $\gamma(p_{12}, p_{13}) = 1$, $\nu(p_{12}, p_{13}) = 1$,
- $\gamma(p_{12}, p_{14}) = 0$, $\nu(p_{12}, p_{14}) = 0$,
- $\gamma(p_{13}, p_{14}) = 0$, $\nu(p_{13}, p_{14}) = 0$.

Therefore, it is

$$fsubv = \frac{(1-1)\cdot 1 + (1-1)\cdot 1 + (1-1)\cdot 1}{3} = 0,$$

which means perfect behavioral similarity.
With weights equalling $\frac{1}{6}$, especially $wsubd \neq 0$, we get for the graph-edit similarity with respect to M_2

$$\begin{aligned} GSim_{M_2}(G_1, G_2) &\approx 1 - \left(\tfrac{1}{6} \cdot (0.1 + 0.6 + 0.22 + 0.22 + 0.24 + 0)\right) \\ &= 0.77 \end{aligned}$$

We can conclude that $GSim(G_1, G_2) \geq 0.77$ as there may exist a mapping M_3 that leads to a better matching of the two graphs than M_2.

7 Conclusion and Future Work

The contribution of the work at hand is to find an adequate similarity measure to identify similar activities in process models that use rather different vocabulary and are modelled at different levels of granularity. We extended existing process model matching methods to compute candidate sets for N-to-M-matchings based on power-sets of nodes. In order to cope with the increasing complexity we imposed higher demands on process models. Therefore, we did not only consider activity labels but also comprised involved actors, data objects and the order of activities. Using this additional information we reduced computation time and identified similar activities in process models that use different vocabulary and are modelled at different levels of granularity. Table 1 provides a short comparison of traditional 1-to-1-matching techniques and the N-to-M-matching of the work at hand.

For future work it is conceivable to extend the set-of-nodes-matching to general process models containing gateways and the possibility, that not only one specific position, but roles are allowed for the organizational perspective. For the

Table 1. Comparison of 1:1- and N:M-matching techniques

	1:1	M:N
Utilized dimensions	$fskipn, fskipe, fsubn$	$fskipn, fskipe, fsubb,$ $fsubd^*, fsubh, fsubv$
Process model	(N, E, λ) with $\lambda : N \to \mathcal{L}$	(N, E, λ) with $\lambda : N \to \mathcal{B} \times \mathcal{A} \times \mathcal{P}(\mathfrak{D})$ with $\mathcal{A}$ being a tree
Findings	1:1-mappings (+ extensions)	1:1-, 1:N-, M:N-mappings
Runtime/complexity	Low	High (improved by special assumptions, etc.)
Robustness	Possibly against inaccurate labels	against inaccurate descriptions, different granularities

weights in the formula of the graph-edit similarity and the weights of computing the organizational similarity, we proposed to choose their values according to everybody's own preferences. It is conceivable that if training graphs with given similarities are available, one may find the best suiting values for the weights with help of statistical methods, like maximum likelihood estimation.

Acknowledgement. The presented work is developed and used in the project "Kompetenzzentrum für praktisches Prozess- und Qualitätsmanagement", which is funded by "Europäischer Fonds für regionale Entwicklung (EFRE)".

The work of Michael Heinrich Baumann is supported by Hanns-Seidel-Stiftung e.V.

References

1. Dijkman, R., Dumas, M., García-Bañuelos, L., Käärik, R.: Aligning Business Process Models (2009)
2. Dijkman, R., Dumas, M., García-Bañuelos, L.: Graph matching algorithms for business process model similarity search. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 48–63. Springer, Heidelberg (2009)
3. Dijkman, R., van Dongen, B., Käärik, R., Mendling, J.: Similarity of business process models: metrics and evaluation. Inf. Syst. **36**(2), 498–516 (2011)
4. Minor, M., Tartakovski, A., Bergmann, R.: Representation and structure-based similarity assessment for agile workflows. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 224–238. Springer, Heidelberg (2007)
5. Dijkman, R.: A Classification of Differences between Similar Business Processes (2007)
6. Klinkmüller, C., Weber, I., Mendling, J., Leopold, H., Ludwig, A.: Increasing recall of process model matching by improved activity label matching. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 211–218. Springer, Heidelberg (2013)

7. Jablonski, S., Bussler, C.: Workflow Management: Modeling Concepts, Architecture and Implementation. International Thomson Computer Press, London (1996). ISBN: 1850322228
8. Weidlich, M., Dijkman, R., Mendling, J.: The ICoP framework: identification of correspondences between process models. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 483–498. Springer, Heidelberg (2010)
9. Weske, M.: Business Process Management: Concepts, Languages, Architecture. Springer, New York (2007)
10. Branco, M.C., Troya, J., Czarnecki, K., Küster, J.M., Völzer, H.: Matching Business Process Workflows across Abstraction Levels, Models (2012)
11. Kunze, M., Weidlich, M., Weske, M.: Behavioral similarity – a proper metric. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 166–181. Springer, Heidelberg (2011)
12. Leopold, H., Smirnov, S., Mendling, J.: On the refactoring of activity labels in business process models. Inf. Syst. **37**(5), 443–459 (2012)
13. Dumas, M., García-Bañuelos, L., Dijkman, R.M.: Similarity search of business process models. Bull. Tech. Comm. Data Eng. **32**(2), 23–28 (2009)
14. Ehrig, M., Koschmider, A., Oberweis, A.: Measuring similarity between semantic business process models. In: Proceedings of the 4th Asia-Pacific Conference on Conceptual Modelling, Ballarat, Victoria, Australia, pp. 71–80 (2007)
15. van der Aalst, W.M.P., de Medeiros, A.K.A., Weijters, A.J.M.M.: Process equivalence: comparing two process models based on observed behavior. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 129–144. Springer, Heidelberg (2006)
16. van Dongen, B.F., Dijkman, R., Mendling, J.: Measuring similarity between business process models. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 450–464. Springer, Heidelberg (2008)
17. Lovins, J.B.: Development of a stemming algorithm. Mech. Transl. Comput. Linguist. **11**, 22–31 (1968)

Verifying Cross-Organizational Workflows Over Multi-Agent Based Environments

Mirtha Lina Fernández Venero(✉)

Center for Mathematics, Computation and Cognition,
Federal University of ABC, São Paulo, Brazil
mirtha.lina@ufabc.edu.br

Abstract. The agent oriented approach is increasingly used to provide flexible mechanisms for collaboration between cross-organizational processes. Most of research on this direction has been focused on agents that interact directly with partners hosted at specific locations. However, the design of a reliable model that integrates a multi-agent environment with a cross-organizational workflow becomes rather complex when mobility is considered. Few methods have been proposed advocating for the use of indirect protocols as the link between the environment and the workflows. Such protocols may couple heterogeneous agents and enforce their migration to specific locations in order to accomplish a collaboration. This paper presents a model based on nested Petri nets for achieving this integration. Besides, it explains how some properties of the overall system can be verified using the SPIN model checker.

Keywords: Cross-organizational workflows · Multi-agent systems · nested Petri nets · Verification · Model checking

1 Introduction

The rapid growth (in number and complexity) of business processes crossing the boundaries of their organizations has led to the development of the cross-organizational protocols and workflows that should be carefully designed and analyzed before their deployment. These processes are captured in a flexible and distributed way by the agent oriented approach, since agents are capable of migrating, under their own control, from one organization to another in order to perform a task. Usually, collaboration is a key factor to achieve the agent goal and it is carried out by interacting autonomously and directly with other agents. Nevertheless, there are situations in which the collaboration should be guided or enforced by external protocols in order to satisfy specific requirements at the inter-organizational level. Such protocols may couple several heterogeneous

This work was partly developed during a post-doctoral stay of the author at the Department of Computer Science, University of São Paulo. It was partially funded by the São Paulo Research Foundation (FAPESP) under the grant 2010/52505-0.

J. Barjis and R. Pergl (Eds.): EOMAS 2014, LNBIP 191, pp. 38–58, 2014.
DOI: 10.1007/978-3-662-44860-1_3

agents from different organizations to perform a complex collaboration. Ensuring the well-functioning of the global system in the presence of these protocols is a challenging issue, due to the heterogeneity and uncontrolled mobility of the agents. The first step to obtain a reliable system is to build an abstract model to test the behavior of the agents and processes and analyze undesirable or unpredicted events. The model can be further refined leading to the construction of reliable executable prototypes.

Most of the work concerning the verification of protocols in multi-agent systems (MASs) has been focused on the direct communication of agents that are situated in a concerted environment [3,4]. The same applies when a business process is implemented as an agent-based system [23,25]. However, this tightly coupled approach is not effective for cross-organizational processes that use mobile agents as resources for achieving goals in collaboration. Note that the external agents may have no knowledge about the internal structure of an organization or the local agents with whom they may collaborate. Mobility and collaboration can be integrated in a more flexible way when the agents are used to support the management of the business process [20]. But models based on this approach usually do not consider the external environment as a main component. Besides, there are few techniques that can be applied to verify the system behavior.

This paper explains how to construct a model that is suitable for simulating and verifying a cross-organizational workflow over an environment of mobile agents. The model provides a clear separation between the multi-agent based (MAB) environment and the local workflows and uses indirect protocols for linking both abstractions. This way the agents can autonomously move in the environment as well as inside the organizations. The model is specified as a nested Petri net (NPN) [17] where agents, protocols and local workflows are net tokens. These nets and related variants have been previously used in MAB modeling and cross-organizational workflows [4,8,9,15,21]. However, there are few systematic methodologies for combining both frameworks using general-purpose indirect protocols. Furthermore, there are few methods that can be used to analyze the sources of conflicts on the use agents. In this paper, we outline how the model can be translated into a PROMELA program and how to analyze the system behavior using SPIN model checker.

The remainder of the paper is organized as follows. Section 2 introduces a running example and discusses the related work. The model is presented in Sect. 3, describing in greater detail the NPN representation for indirect protocols. To this end, a simple but flexible language is defined that provides general patterns for mobility, concurrency and collaboration. Section 4 explains how SPIN can be used for verifying behavioral properties of the model. Some concluding remarks and future work are presented in Sect. 5.

2 A Running Example and Related Work

In this section we introduce an example that illustrates how the model can be constructed. The example is an abstraction of an interaction model belonging to

the domain of emergency response management [18]. Agent-based models over this domain require the reliable coordination of several mobile agents that are shared by a number of organizations. During an emergency, the agents may be redirected by external protocols and forced to collaborate with unknown agents. In this context it is a crucial issue to prevent uncompleted interactions due to e.g. agents deadlock.

Our running example uses two basic roles for the agents: emergency chiefs and subordinates. The chiefs distribute a number of assignments between a set of subordinates (R1) and move them to initial locations (R2). An assignment consists on specific instructions about the tasks that the subordinates can execute (R3). Each subordinate may traverse the environment autonomously in order to use resources or solve a task in collaboration (R4). The collaborative tasks may be performed by several agents at the same location (R5). The environment may enforce other forms of collaboration by coupling agents positioned at different locations (R6). A subordinate can move back to its original location after completing all the tasks (R7). The distribution of the assignments is executed as the first activity in the workflow of some organization (R8). Once it is done, the workflows of other organizations may use the agents at the environment, to carry out additional tasks (R9). Note that, in this scenario it is important to ensure not only that all local workflows terminate but also that all agents complete their assignments and return to the original position.

Lot of research has been devoted to study the mobility issues from the perspective of multi-agent systems (MASs). However, most of the work concerning simulation and verification has focused on agents that (1) migrate autonomously in a concerted local environment and (2) interact *directly* with specific agents. This is usually achieved by first providing an abstract model (by means of activity/state/sequence diagrams) that it is then translated into a verifiable formalism such as Petri nets or the language of model checkers, e.g. [3,4,8,15]. In many cases, the same applies when the agent-oriented approach is used for cross-organizational models. For instance, in [23,25] business partners are modeled as agents that engage in direct protocols to accomplish a cross-organizational business process. To this end, it is used the Blindingly Simple Protocol Language that describes a protocol in terms of the messages exchanged by the agents. However, message exchange is not enough for dealing with a complex coordination involving heterogeneous agents that must perform a collaborative task. Reo connectors [1] may help to solve the later problem using channels that can be composed by joining the ends. Nevertheless, the link between the agents actions and the nodes of the connectors should be fixed from design time. Hence, all agents that may be involved in a collaboration should be linked in advance.

There are few results that can be applied to verify general-purpose *indirect* protocols arising from the above scenario. Such protocols enforce the movement of an agent to a new location or organization that maybe the agent cannot reach in a single step by its own or is not allowed to perform autonomously. They may also group agents situated at different (and possibly disconnected) locations in order to form a collaboration unit (see Fig. 1, adapted from [31]). A key issue

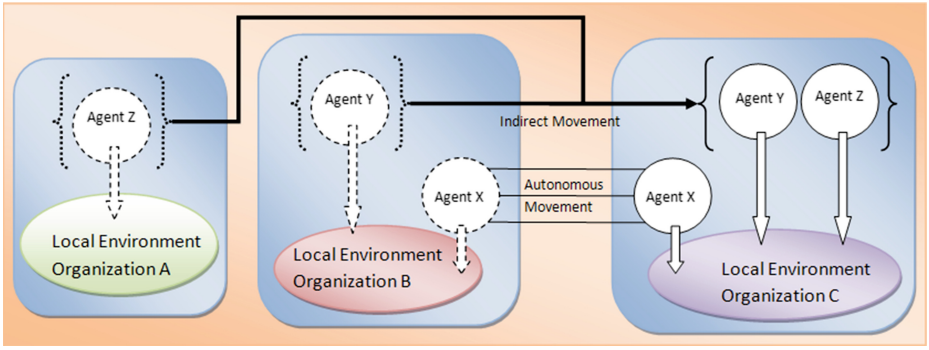


Fig. 1. Autonomous and indirect movements in a cross-organizational environment.

for the latter is to provide means to carry out the collaboration with no a priori knowledge about internal structure or behavior of the participant agents. Languages and frameworks based on process calculi (such as the Join calculus, the Ambient calculus and several π-calculus variants) are expressive enough to specify indirect protocols (as e.g. in [14]), even as first class entities [19]. Unfortunately, they lack suitable tools for simulation and verification. Several results involving mobile agents, protocols and cross-organizational workflows are based on NPNs [4,6,9,27]. However, they deal with indirect movements of a single agent and do not take into account collaboration. Several tools can be used to simulate a wide range of PNs but few of them support the nets-within-nets approach (e.g. RENEW [15], MASGAS [8] and CPN Tools [7]). Currently, no tool provides facilities for simulation and verification of NPNs with arbitrary nesting, synchronization and recursion. The SPIN model checker has been used to verify restricted subclasses of NPNs. For example, the work in [4] focusses on two-level nested nets without horizontal synchronization or net tokens removal. A recent translation [26] allows to analyze multi-level and recursive NPNs with a restricted synchronization and without transportation steps.

Two main approaches have been proposed for integrating the agent-based technologies with workflow and business process management: the *agent-driven* (or agent-based) approach and the *agent-supported* (or agent-enhanced) approach [20,30]. The former consists on a set of agents that manage the business process. The entire process is split into groups of tasks which are then encoded into software agents. Other agents are responsible for controlling the whole workflow logic. This provides a high degree of flexibility, dynamism, and adaptability. In theory, the techniques for verifying a MAS can be applied to a MAB business process. However, the size of the resulting MAS may be too large to be verified in practice. Since agents are more complex because of implementation details, finding the source of an error or conflict (e.g. due to mobility issues) becomes more difficult. In the agent-supported approach, the tasks of a business process may be executed by agents (possibly in collaboration with other agents). This approach is suitable for cloud business process management since

the implementation of the process is not easily modified and the agents may execute sub-processes for dealing with external changes. Nevertheless, mobility and collaboration are usually taken into account inside the local workflows but they are ignored at the global environment. This is due to the fact that in this context, models do not consider the environment as a primary abstraction. The approach can be modeled and simulated using NPNs but, as we explained above, few tools can be applied for verification.

3 A NPN-Based Model Combining MAB Environments, Indirect Protocols and Cross-Organizational Workflows

In this section we present a model for an agent-supported cross-organizational workflow with two important features: (1) it considers the agent environment as a main component of the model and (2) it uses indirect protocols as the link between the workflows and the environment in order to achieve collaboration. The model is based on the construction of a nested Petri net. A Petri net is a tuple (P, T, A, W) where P and T are non-empty, finite and disjoint sets of places and transitions resp; $A \subseteq (P \times T) \cup (T \times P)$ is a set of arcs and W is a function defined from A to multisets of uncolored tokens. The *marking* of the net is a function from P to a multiset of tokens. The transitions represent events (called *firings*) which may change the marking of the net according to W. The uncolored tokens in a PN have no structure or information; therefore they are usually represented as black dots. In colored Petri nets (CPNs), each place has a type; thus, it may host tokens with different data values (colors) [13]. A nested Petri net is a CPN in which tokens can also be Petri nets and thus they fire their own internal transitions [17]. It is usually defined as a tuple $(SN, EN_1, \ldots, EN_n)$ where SN, called *system net*, represents the top level and main component of the NPN. Each EN_i (called *element net*) is considered as a constant and also as a type whose set of values consists of marked net tokens. Therefore, the places in a NPN may have basic or element net types. A marking of a NPN is a marking of its system net.

The firing of a transition in the system net or a net token is performed as usual in CPNs: basic and net tokens are created, removed or transported. In addition, the firing may be synchronized with the firing of other net tokens at the same place (*horizontal synchronization step*) or at different places of the parent net (*vertical synchronization step*). To this end, transitions may be labeled for horizontal or vertical synchronization. The firing of an unlabeled transition is known as *autonomous step*. Transitions labeled for horizontal synchronization fire simultaneously in net tokens situated at the same place. The number of tokens that are involved in a horizontal step depends on the arity of the label[1]. A transition t (in SN or some net token) labeled for vertical synchronization is enabled if there are enough enabled transitions with the complementary label of t in the net tokens situated at the input places of t. If so, t may fire simultaneously with the enabled complementary transitions in its input children nets.

[1] The arity of a horizontal step may be defined for places instead of labels as in [17].

In our model the element nets are derived from the behavior of the agents, the definition of the indirect protocols and the extended workflow of each organization. The system net comprises two subnets: the environment and the net that triggers the workflows. The model at the agent level (behavior and direct interactions) has been largely studied in terms of PNs (see e.g. [5,15,16,24]). Here we only remark that the direct communication between agents can be effectively modeled as horizontal steps while the interaction with the environment is achieved using vertical steps. This has the advantage that several transitions can be synchronized using a single label. It contrasts with models based on reference nets [5,15] where several synchronous channels should be used to synchronize more than two transitions. In the next we outline how the remaining components of our model can be constructed.

3.1 Modeling the Environment

When the environment represents a graph, its model directly leads to a 2-level net where the places represent locations and the links between them are encoded using transitions. The agents are modeled as 1-level element nets with labels for synchronization. The autonomous movement of an agent net in the environment can be allowed or restricted using typed variables on the arc inscriptions. Other restrictions can be enforced by means of additional places and labels for vertical synchronization. A location represents a general abstract concept (e.g. a role, a physical position, a grid) and may have its own local environment with objects and agents situated at internal places [8]. Therefore, the global environment may also be modeled as a multi-level net. However, the migration of agents between adjacent levels is hard to specify in the NPN framework. In such a case, it is better to unfold the locations to obtain a 2-level representation.

A key feature to allow indirect movements is to share places at the environment, especially those that may host agents nets. This way the net tokens corresponding to protocols or workflows may have access to the agents. To avoid conflicts in a synchronizing step, we require that labeled transitions in the element nets have no shared place as input. This condition ensures that the behavior of the NPN is not affected. In addition, we assume that all shared places store tokens of basic types or 1-level nets. Hence, the agents nets have only access to shared places of basic type. We use further restrictions on the arc inscriptions to avoid the replication of agents or the comparison of their internal structure or state. The formal definition of these NPNs can be found in [28].

Figure 2 in the left shows a net that can be used to model the environment of the scenario described in Sect. 2. The circles represent basic places while the ellipses represent net-typed places. The places for the roles (*Chiefs* and *Subordinates*) and resources (*Assignments*) are shared as well as some places representing physical positions ($L1, L2, L3, L4, Lf$). Other positions are represented as non-shared places ($L5, L6$). In this paper the shared places are colored in blue and the element nets are enclosed in rectangles. Nets enclosed in rounded rectangles belong to the system net and substitution transitions (standing for subnets) are drawn using bold boxes.

Figure 2 in the right depicts simple element nets that may be used for the roles. The net for a *Chief* agent has a transition for synchronizing the subordinates (R1) while the net for a *Subordinate* agent uses a source transition for receiving the instructions. In a *Subordinate* net, the basic colors a, c, r represent the tasks that are executed autonomously (a), in collaboration (c) or as a request from the environment (r). The number of these tasks is fixed in advanced, e.g. when a net token is created; this is represented by the constants na, nc, nr. The unlabeled transition u performs the autonomous tasks while the ones labeled as c and $\bar{r}$ are used for horizontal (R4) and vertical (R6) synchronization resp. In the environment, transitions labeled as r allow a regulated movement of an agent (e.g. those situated at *L1* – R4) or enforce specific forms of collaboration (e.g. between agents at *L2* and *L3* – R6). The unlabeled transition imposes no restriction for the autonomous migration (R4). An agent is moved back to *Subordinates* by means of transitions labeled as e and $\bar{e}$, for vertical synchronization. The latter transition has an inhibitor arc[2] that enables the firing once all tasks have been completed (R7).

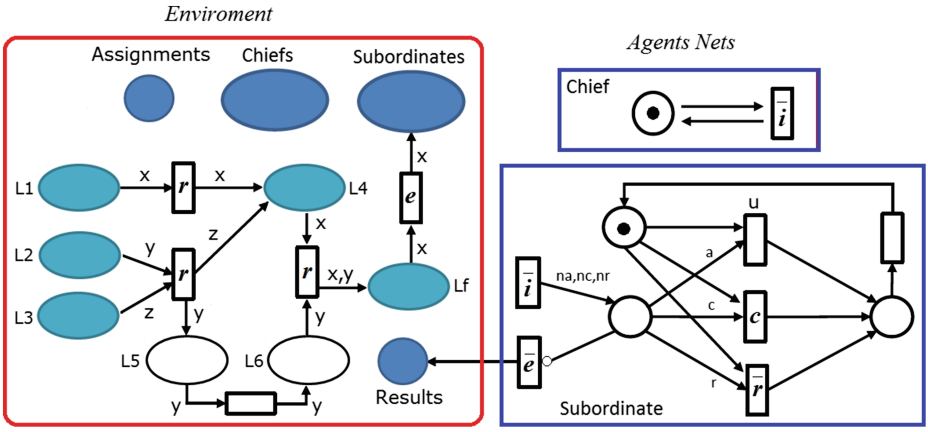


Fig. 2. Environment and agents nets for the emergency management scenario.

3.2 Modeling the Protocols

To illustrate the construction of an element net from a protocol we use a lightweight language (resembling e.g. [19,22]) that combines mobility, collaboration and concurrency, but not communication. We use three primitives **empty**, **get** and **put** for testing emptiness, getting and putting resources at shared locations

[2] An inhibitor arc tests the absence of tokens at the input place. It is represented using a line with a circle instead of an arrow on the transition side.

resp. Furthermore, we use procedures to describe cooperative tasks between heterogeneous agents. A protocol definition has the form $pn(\widetilde{x}) :: = A_1 \vee A_2 \vee \ldots \vee A_n$. where each A_i is obtained from the next rules:

$$A \rightarrow \mathbf{get}[(r_1, l_1), \ldots, (r_k, l_k)] + \mathbf{empty}[l_n, \ldots, l_m]; F$$
$$F \rightarrow \mathbf{get}[(r_1, l_1), \ldots, (r_k, l_k)] + \mathbf{empty}[l_n, \ldots, l_m] \mid \mathbf{put}[(r_1, l_1), \ldots, (r_k, l_k)] \mid \rho[a_1, \ldots, a_n][\lambda_1, \lambda_2](\widetilde{v}) \mid F_1; F_2 \mid P(\widetilde{v}) \mid (P_1(\widetilde{v}_1) \parallel P_2(\widetilde{v}_2))$$

In the above rules, l_i denotes a shared location, r_i a resource (or a list of resources) and a_i an agent variable. Here, a resource may be either a data value or a data/agent variable. The protocol body is a non-deterministic choice of several alternatives. Each alternative is guarded by the combination of the **get** and **empty** primitives (though they can be used separated). The suffix of the alternative may be a sequential composition (denoted as $F_1 \ ; \ F_2$) of primitives, procedures, protocol calls and the parallel composition. A procedure for a collaboration task operates over a set of agents that are initially synchronized using the label λ_1. This label can be interpreted as a request for collaboration. An additional synchronization label λ_2 should be provided if the task requires a response or an acknowledgement of completion. Procedure and protocol calls may have input and output parameters that allow the exchange of data during a synchronization. For the parallel composition, denoted as $P_1(\widetilde{v}_1) \parallel P_2(\widetilde{v}_2)$, we assume that the sets of output variables occurring in $\widetilde{v}_1$ and $\widetilde{v}_2$ are disjoint. A protocol for our running example can be defined as:

$$\begin{aligned} &\mathit{manageEmergency}() :: = \mathbf{empty}[\mathit{Assignments}] \vee \\ &\quad \mathbf{get}[(as, \mathit{Assignments}), ([s_1, s_2, s_3], \mathit{Subordinates}), (ch, \mathit{Chiefs})]; \\ &\quad \mathit{distributeTasks}[ch, s_1, s_2, s_3][i](as); \\ &\quad \mathbf{put}[(ch, \mathit{Chiefs}), (s_1, L_1), (s_2, L_2), (s_3, L_3)]; \mathit{manageEmergency}(). \end{aligned}$$

This protocol terminates when the *Assignments* location is empty. Otherwise, the details of an assignment, a *Chief* and three *Subordinate* agents are chosen to continue the protocol. The agents are synchronized using the *distributeTasks* procedure and then moved to specific locations in the environment (R2). The recursive call allows proceeding with the remaining assignments.

Figure 3 depicts a PN-based semantics for the protocol definitions. We remark that colored ellipses denote shared places in *SN*. As shown in Fig. 3a, each protocol produces an element net in the model. The net is constructed from the nets of each alternative using the standard pattern for the non-deterministic choice. The net of an alternative has two special uncolored places, a source and a sink, denoted as *In* and *Out* resp. The protocol net fuses the sources and the sinks of the alternative nets and the resulting *Out* place is linked to a sink transition labeled for vertical synchronization. Each protocol net has also a unique special place *CP* (*collaboration point* [29], here represented with a double line) where agents engaged in the protocol interact. The initial marking of the element net has an uncolored token at the resulting *In*. Therefore, it may enable the first transition (corresponding to the guard) of any of the alternative nets.

Figure 3b illustrates how the net for the **get+empty** combination is constructed. A single transition links the *In*, *Out* and *CP* places and also all the

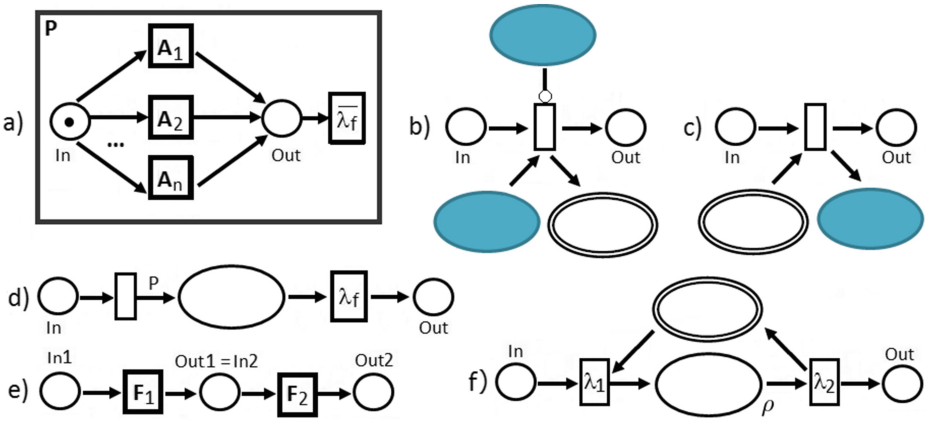


Fig. 3. PNs for translating a protocol definition.

shared places specified in the primitives. This transition gets enabled when there is a token at *In*, the agents involved in the **get** reach the corresponding locations and there is no token at the locations of the **empty**. When it fires the agents nets are transported from the shared locations to *CP* and a token is added to *Out*. Note that the movement to the collaboration point may be logical not physical. It prevents the agents to migrate to other locations and also avoids other protocols to use them. Furthermore, it allows them to collaborate using horizontal synchronization. An agent is *moved* back from *CP* to a location in the environment using the **put** primitive, whose net is depicted in Fig. 3c. We have omitted the arc inscriptions for these subnets but they must match the parameters of the primitives.

The net for a protocol call (Fig. 3d) creates a protocol net instance. The net token is consumed using vertical synchronization once it terminates the execution (with an uncolored token at the sink). The net for the sequential composition fuses the *Out* place of the left-hand side with *In* place of the right-hand side (Fig. 3e). The net for a collaboration procedure is shown Fig. 3f. The agents that commit to initiate the procedure are synchronized using a vertical step labeled as λ_1. The collaboration group is then isolated at an auxiliary place ρ until the completion of the task. Once all agents have finished, they are moved back to *CP* by a vertical step labeled as λ_2. However, if the response or acknowledgment is not required, then the place ρ and the second transition should be omitted (see the transition labeled as i in Fig. 4 corresponding to *distributeTasks*). The net for the parallel composition can be easily obtained by modifying the arc inscriptions of the net in Fig. 3d. In this case, the output arc from the first transition should be labeled as $\{P_1, P_2\}$ in order to create an instance of each protocol call. Besides, the inscription of the input arc of the labeled transition must remove two net tokens. The element net obtained from the *manageEmergency* protocol is represented in Fig. 2.

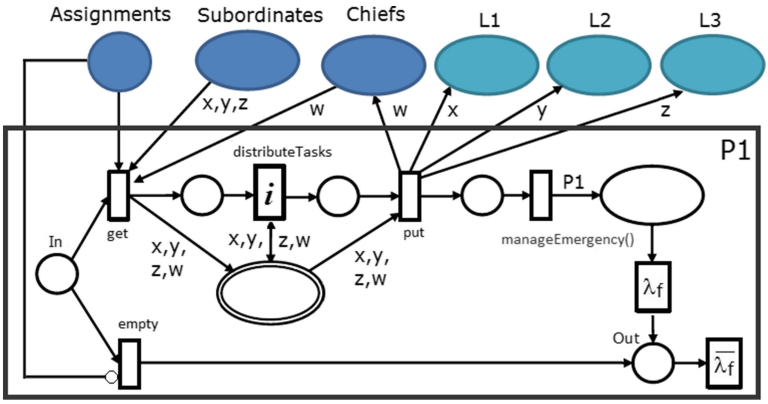


Fig. 4. PN representation of the *manageEmergency* protocol.

We point out that the PN semantics leaves aside the issues concerning the input and output parameters in procedures and protocols. Therefore, it models all possible execution paths of a protocol call. This is appropriate at first steps of design to analyze all possible undesired behaviors of the system. Note that the NPN model for our running example does not consider the requirement R3 concerning the passing of instructions from chiefs to subordinates. The details arising from direct exchange of messages and data between the agents may be added at later stages using e.g. synchronous channels for communication.

3.3 Extending the Cross-Organizational Workflow

A cross-organizational workflow can be modeled as a NPN where each local workflow is an element net, ending with a sink transition labeled for vertical synchronization [21]. An additional element net AC is used to encode the asynchronous communication relations between the workflow nets (WF-nets). Synchronous and asynchronous communications are modeled by means of labels for horizontal synchronization. The system net may be defined using a source and a sink (uncolored places) and an intermediate place for storing the WF-nets and a single AC net token (see Fig. 5, upper subnet).

Our model brings about two important benefits for modeling the integration of a MAB environment with a cross-organizational workflow. Firstly, each organization may have shared locations at the environment in order to exchange its resources with other organizations. Shared places of different organizations may be interconnected allowing the migration of agents from one organization to another. The local workflows may import and export the resources at these places; hence, external agents may enter, leave, execute tasks and navigate along a workflow path. In addition, complex forms of collaboration can be defined as indirect protocols and linked with tasks in the WF-nets. The latter can be achieved by replacing the transition corresponding to the task with the subnet associated to a protocol call (omitting the *In* and *Out* places) [27]. As we

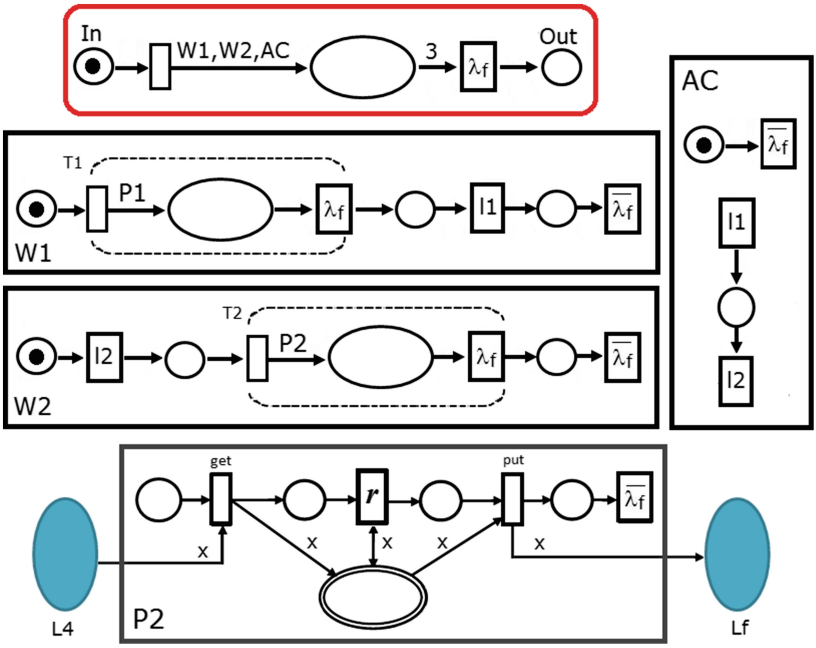


Fig. 5. Element nets in a cross-organizational workflow with indirect protocols.

explained above for the parallel composition, several instances of protocols may be related to a single task without increasing the size the workflow.

Figure 5 illustrates how this integration can be modeled for our running example. The first transition in the upper subnet triggers the net tokens belonging to the cross-organizational workflow. The first task in the WF-net *W1* is a call to *P1*, the element net corresponding to the *manageEmergency* protocol. When the protocol terminates, the transitions labeled as $l1$ (at *W1* and AC) fire simultaneously. This indicates to other WF-nets that the distribution of the assignments has been completed (R8). By the AC relation between $l1$ and $l2$, all enabled transition labeled as $l2$ may fire (R9). In particular, the firing of the first transition at *W2* triggers an indirect protocol (*P2*) that uses any agent reaching to *L4*. The protocol requests the agent to execute a task and then moves it to *Lf*. Once all WF-nets reach their final state, they are consumed (as well as AC) using vertical synchronization.

4 Verifying the Model Using SPIN

Ensuring the validity of the properties of a cross-organizational workflow (e.g. soundness) is a crucial issue, especially when the organizations share a common environment and resources. The problem is more complex when the resources are autonomous and mobile agents. The model we presented in the previous section

has the drawback that the NPN approach lacks for verification tools. The results reporting the verification of these or similar nets deal restricted subclasses and usually use a translation into the language of a model checker [2,4,26]. In this section we briefly describe the main features of a general translation from NPNs into PROMELA, the language of SPIN model checker. SPIN is one of the most successful tools in verifying software protocols [11]. Due to space limitations, we outline the representation of net-typed places, the translation for the element nets and how the synchronization steps are performed. The detailed description of the translation can be found in [28]. At the end of the section we explain how to use SPIN for verifying important properties of the model.

The PROMELA program for a NPN is composed of a `proctype` definition for each element net and the `init` process as the system net. Shared places are declared as global variables while non-shared ones are local to the process definition of the element net. Each net token is a process and a net place is represented as channel that holds the information about the net tokens it hosts. The channel also allows the exchange of messages between the processes in order to execute a synchronization. Each message consists of five fields. The first holds the instantiation number (`_pid`) of the net token process sending or receiving the message. The second and the third fields are the label and the identity number of the transition that is enabled, resp. The fourth bit field defines the type of the message: a synchronization request (0) or a response (1). In a response message, the last field indicates whether or not the net token is consumed after the step. Additional fields may be used depending on the application, e.g. for encoding the net type or exchanging data between net tokens processes.

The general structure of the translation is shown in Fig. 6[3]. The behavior of the system net is simulated via a loop where each option corresponds to the firing of a transition. The code for the firing follows the usual pattern for translating PNs into PROMELA [10,11]. The expression `enableTest_t` denotes the condition for checking the existence of the required tokens at each input place. The instructions for removing such tokens are denoted as `consumeActions_t`. In both cases, the code depends on the input arc inscriptions. The sequence of instructions for adding the tokens specified in each output arc to the corresponding output place is denoted as `produceActions_t`. We distinguish the case when a net token is transported since consuming a net token implies the termination of the corresponding process. As in [26], priorities are used to avoid the interleaving of firings not belonging to a step. But here, different levels of priorities (from 1 to 6) are used to reduce the size of the state space.

The behavior of an element net is simulated using two nested loops. The inner loop includes an option for each transition that may execute either an autonomous firing or a request for synchronization. The former applies to unlabeled transitions and transitions labeled for vertical synchronization in parent nets. It is translated as in Fig. 6 (lines `29-32`) if the transition does not produce net tokens. Otherwise, the firing is split into two parts and the creation of the new processes is performed outside the inner cycle (see Fig. 7, lines `6-8,26,27`).

[3] Some details have been omitted for the sake of readability and compactness.

```
 1  typedef NetPlace { chan d = [MaxMsg] of {byte,byte,byte,bit,bit} }
 2  chan gbChan = [MaxMsg] of {byte, chan, byte, byte, bit};
 3
 4   /* Auxiliary Code, Shared Places, Element Nets */
 5
 6 proctype EN_i(chan pc){
 7 /* Non-Shared Places and Arc Variables */
 8 atomic{ /* Initial Marking */ }
 9 do:: do:: d_step{ /* autonomous firing  */ }
10       :: d_step{ /* synchronization request */
11              enableTest_t && ! pc??[eval(_pid),L(t),_,0,0] ->
12                   pc!_pid,L(t),t,0,0 }
13       :: d_step{ /* horizontal synchronization */ }
14      od
15      unless atomic{ gbChan ?? [eval(_pid),pc,lt,it]
16                  ||  pc ?? [eval(_pid),lt,it,1,rm]       ->
17        if:: /* synchronization firing */
18             lt==L(t) && enableTest_t ->
19             consumeActions_t; produceActions_t;
20        fi;
21        if:: rm -> break ::else -> set_priority(_pid, 1) fi }
22 od;
23 /* Stop children nets */  }
24
25  init(){
26  /* Non-Shared Places and Arc Variables */
27  atomic{ /* Initial Marking */ }
28  do:: atomic{
29           enableTest_t ->
30              set_priority(_pid, 6);
31              consumeActions_t; produceActions_t;
32              set_priority(_pid, 1) }
33 od }
```

Fig. 6. General structure of the PROMELA model for a NPN.

This is due to the fact that SPIN `run` statement cannot be used inside a `d_step`. The remaining transitions must send a synchronization request to the parent net once they are enabled (Fig. 6, lines `10-12`). For such transitions, the firing occurs as soon as the process receives the response message, as shown in Fig. 6, lines `15-19`. Here, any enabled transition with the same label may be chosen for firing. If the net was transported, the response message is received via a global channel `gbChan`, along with a new `pc`. If the net token was consumed by the parent net then, after the firing, the outer loop is broken and the active children nets are removed (Fig. 6, lines `21-23`). In both cases, the parent net is responsible for transporting (`transpNet`) or removing (`consNet`) the child net messages before sending the response.

```
1  proctype W2Net(chan pc){
2  bit pIn=1, p1; NetPlace p2; bit pOut;
3  do::
4   do:: d_step{ pIn>0 && ! pc??[eval(_pid),7] ->      /* tl2 req */
5           pc!_pid,7 }
6     :: d_step{ p1>0  ->                              /* tu1 frg-1*/
7           set_priority(_pid, 6);
8           p1--;  gbChan ! _pid,8,pc; }
9     :: d_step{  p2.d ?? [_,4] ->                     /* tlf frg */
10          set_priority(_pid, 6);
11          recMsg(p2.d, nt,4); consNet(p2.d,nt);
12          gbChan ! nt,4,p2.d; set_priority(nt, 5);
13          pOut++;
14          set_priority(_pid, 1) }
15     :: d_step{ pOut>0 && ! pc ?? [eval(_pid),4] ->   /* t_lf req */
16          pc ! _pid,4; }
17     :: d_step{ pc??[eval(_pid),7]  &&
18                c_expr{numMsg(qptr(PW2Net->pc-1),7)>=2} ->
19          set_priority(_pid, 6);
20          pc??eval(_pid),7; gbChan ! _pid,7,pc;       /* hor sync */
21          recMsg(pc, nt,7); gbChan ! nt,7,pc;         /* ar(l2)=2 */
22          set_priority(nt, 4); }
23   od unless atomic{ gbChan ?? eval(_pid),lt,pc ->
24       if:: lt==7 ->                                  /* tl2 frg */
25            pIn--; p1++;
26         :: lt==8 ->                                  /* tu1 frg-2*/
27            nt = run P2Net(p2.d); p2.d ! nt,255;
28         :: lt==4 -> pOut--; break                    /* t_lf frg */
29       fi; set_priority(_pid, 1) }
30 od }
```

Fig. 7. PROMELA translation for WF-net *W2* in Fig. 5.

In this translation, the inner loop also includes an option for each horizontal label occurring in the element net. This option checks if the number of requests at `pc` fulfills the arity of the label. If so, the request messages from the participant nets are removed and the response messages are sent. Unlike in [26], here there is no restriction on the number of net tokens that can be involved in a synchronization. This is achieved by embedding C code into the PROMELA model (Fig. 7, lines `17-22`). The above general schema can be refined in several ways to improve the resulting program [28]. For example, depending on the structure of the NPN, some fields of the `NetPlace` may not be necessary. Furthermore, all response messages may be sent via `gbChan`. Figure 7 shows a refined translation for the WF-net *W2* in Fig. 5. Appendix A includes the code for two other element nets and parts of the system net of our running example. The entire program can be found at https://www.dropbox.com/s/ojfv8dcalh7dfpw/exMAEnvIOWF.pml.

The PROMELA translation can be used for investigating some properties of the NPN-model. Since SPIN provides support only for weak fairness, a full verification for a property may also be guaranteed if the NPN is terminating. While termination is not required for the soundness of the local WF-nets and the cross-organizational workflow, it is a fundamental requirement in many practical cases involving agents and interaction protocols. In order to test this property with SPIN, we may first investigate the absence of infinite recursive or unbounded sequences. This can be done by marking the valid `end` states of the model and using SPIN option for a safety verification. This way, all the sequences exceeding the bounds of the model will lead to an invalid state. The infinite cycles in the net can be detected in a second phase, using an accept label in front of the `init` loop and SPIN search for acceptance cycles. The NPN terminates if both verifications are completed without errors. If so, SPIN may report unreachable states due to transitions or net tokens that are never used. This information may also help to improve the model.

The final configuration of the system net (in particular the environment) can be investigated using reachability conditions encoded as LTL properties. Conditions involving the internal state of the agents nets should be specified using `never`-claims, in order to access the local places. For our running example, for instance, we may be interested in verifying that from an initial configuration of the environment, the cross-organizational workflow reaches its final state (a black dot at *Out*) and also all subordinate agents are able to return to the original location. An example of such initial configuration (hereafter called *sound*) is a marking with three agents at *Subordinates*, an agent at *Chiefs* and a single token at *Assignments*. We assume that each subordinate agent may execute four tasks: one autonomously, one in collaboration and two by request. We also assume that the collaboration task requires three agents; thus the arity of c is 3. In the next, we refer to this configuration as $C1$. Due to the structure of our NPN-model, we can verify that $C1$ is sound using the next LTL property:

```
<>[]( Out==1 && Assig==0 && len(L1.d)==0 && len(L2.d)==0 && len(L3.d)==0
        && len(L4.d)==0 && len(L5.d)==0 && len(L6.d)==0 && len(Lf.d)==0 )
```

SPIN (version 6.3.2) took 2.82s for a safety verification of the model (running in a notebook Intel Core I3, 2.4 GHz, 4 Gb RAM). It took 7.71s for verifying the above property for $C1$. In both cases we use bit state hashing. A sound configuration turns up unsound just with a slight modification. For instance, $C1$ is unsound if it is defined with an additional non-autonomous task for one of the agents or if we change the arity of c to 2 instead of 3. In both cases, SPIN found a counterexample for the above property in 0.003s using the standard search for acceptance cycles. The error trial shows that the cross-organizational workflow terminates but a subordinate agent cannot complete the assignment. Thus, it cannot move back to its initial location.

Another property that can be easily studied with SPIN is boundedness. This property can be used e.g. for predicting the maximal number of agents that may visit a location. It can be verified by means of assertions on the variables for places. The assertions should be placed at the end of the firing of each transition with an incident arc to the required places. Then, the safety verification will

report any sequence violating the boundedness conditions. The assertions and also the LTL properties can be simplified by adding auxiliary variables to the model. For instance, for a net place, an additional variable may be used to keep track of the number of net tokens[4]. For our running example, if we use an additional global variable `countSub` for counting the net tokens at *Subordinates*, the above property can be defined as `<>[]( Out==1 && Assig==0 && countSub==3 ).`

5 Conclusions

This paper presented a NPN-based model for analyzing cross-organizational workflows over multi-agent based environments. The model makes a clear distinction between the environment, the agents nets, the local workflows and the protocols for collaboration. This approach keeps simple the environment and the internal structure of the agents while the complex behavior is captured by the protocols and the workflows specification. Therefore, it is well-suited to analyze agent-supported business process management systems. The resulting models are more flexible and easier to simulate and verify than those arising from the agent-driven approach. If NPN-model does not involve recursive nets, it may be simulated using tools such as MASGAS and RENEW and later integrated with multi-agent architectures such as JADE and MULAN as done in [8,15] resp. Otherwise, it can be translated into PROMELA and simulated using SPIN model checker.

The PROMELA translation proposed in this paper is the first dealing with general NPNs (with non-restricted synchronization, arbitrary levels and recursion). It can be useful for testing the NPN-model and verifying properties related to termination, reachability and boundedness. SPIN is very effective for finding counterexamples to invalid properties. Nevertheless, due to the state-explosion problem, large nets cannot be fully verified. Therefore, the translation should be applied to a reduced model of the cross-organizational workflow and the multi-agent environment. The abstract model should preserve the main properties of the original one (e.g. the soundness of the local workflows), while focusing on those tasks involving the resources at the environment and the indirect protocols. The translation may also help in the construction of executable prototypes since it provides insights on how to deal with the synchronization of agents. The key idea is that each location provides a communication channel that is shared by all agents situated at the location. Therefore, the communication media changes each time the agent changes its host location. This allows a flexible and location-transparent form of collaboration, without fixing in advance the participants.

The use of SPIN is limited by its weak fairness model and its restriction to LTL properties. But the ideas behind the translation can be applied to other model checkers used in MAS verification (e.g. JFP) and combined with other initiatives such as those in [3,23]. A promising research line towards the latter is the combination of model checking tools (as proposed in [12]) in order to

[4] The number of net tokens at a place may not be equal to the length of the channel since the channel may also contain several request messages.

verify direct and indirect protocols. Our future work will address these and other open verification issues. We are currently working on an improved version of the translation and a plug-in for integrating SPIN with a visualization tool such as Renew. We are also interested in investigating reduction strategies that can be effective for model checking these nets.

A PROMELA Translation for Other NPN Components

In the next we include the PROMELA translation for the *Subordinate* net in Fig. 2 and the protocol net $P2$ in Fig. 5. The `proctype` definition of the former uses additional input parameters for the constants na, nc, nr. Besides, the colored place $p2$ is unfolded into three uncolored places $p2a, p2r, p2c$. The labels are encoded using integers e.g. $\bar{i} = 1, \bar{e} = 2, \bar{r} = 3, \bar{\lambda}_f = 4, c = 5$ and $l2 = 7$. The `rmvConflict` macro is used to deal with conflicts with labeled transitions. The `recMsg` macro chooses a request message with a given label from a channel. Some parts of the translation for *SN* (`init`) are also included.

```
proctype SubNet(chan pc; byte na,nc,nr){
bit p1=1; byte p2a, p2r, p2c;  bit p3;
do::
  do:: d_step{ p1>0 && p2a>0 ->                              /* u1 frg*/
         set_priority(_pid, 6);
         p1--; p2a--;
         rmvConflict(5); rmvConflict(3); p3++;
         set_priority(_pid, 1) }
    :: d_step{ p3>0 ->                                       /* u2 frg */
         set_priority(_pid, 6);
         p3--; p1++;
         set_priority(_pid, 1) }
    :: d_step{ ! pc??[eval(_pid),1] ->                       /* t_i req */
         pc!_pid,1 }
    :: d_step{ p1>0 && p2a==0 && p2c==0 && p2r==0 &&
               ! pc??[eval(_pid),2] ->                       /* t_e req */
         pc!_pid,2 }
    :: d_step{ p1>0 && p2r>0 && ! pc??[eval(_pid),3] ->      /* t_r req */
         pc!_pid,3 }
    :: d_step{ p1>0 && p2c>0 && ! pc??[eval(_pid),5] ->      /* tc req */
         pc!_pid,5 }
    :: d_step{ pc??[eval(_pid),5]  &&
              c_expr{numMsg(qptr(PSubNet->pc-1),5)>=3} ->
         set_priority(_pid, 6);
         pc??eval(_pid),5; gbChan ! _pid,5,pc;               /* hor sync */
         recMsg(pc, nt,5); gbChan ! nt,5,pc;                 /* ar(c)=3  */
         set_priority(nt, 4);
         recMsg(pc, nt,5); gbChan ! nt,5,pc;
         set_priority(nt, 4); }
  od unless atomic{ gbChan ?? eval(_pid),lt,pc ->
      if:: lt==1 ->                                          /* t_i frg */
```

```
          p2a = p2a+na; p2r = p2r+nr; p2c = p2c+nc;
          rmvConflict(2);
       :: lt==2 -> Results++;                              /* t_e frg */
       :: lt==3 && p1>0 && p2r>0 ->                        /* t_r frg */
          p1--; p2r--; rmvConflict(5); p3++
       :: lt==5 && p1>0 && p2c>0 ->                        /* tc frg */
             p1--; p2c--; rmvConflict(3); p3++
       :: lt==255 -> skip;                          /* transp w/o sync */
      fi; set_priority(_pid, 1) }
od }

proctype P2Net(chan pc){
NetPlace CP; bit pIn=1, p1, p2, p3, pOut;
do::
  do:: d_step{  pIn>0 && L4.d ?? [_,255] ->                /* get */
        set_priority(_pid, 6);
        pIn--; p1++;
        recMsg(L4.d, nt,255); transpNet(L4.d,CP.d,nt);
        CP.d ! nt,255; gbChan ! nt,255,CP.d;
        set_priority(nt, 3);
        set_priority(_pid, 1) }
    :: d_step{  p1>0 && CP.d ?? [_,3] ->                   /* tr frg */
        set_priority(_pid, 6);
        p1--; p2++;
        recMsg(CP.d, nt,3); gbChan ! nt,3,CP.d;
        set_priority(nt, 5);
        set_priority(_pid, 1) }
    :: d_step{ p2>0 && CP.d ?? [_,255] ->                  /* put */
        set_priority(_pid, 6);
        p2--; pOut++;
        recMsg(CP.d, nt,255); transpNet(CP.d,Lf.d,nt);
        Lf.d ! nt,255; gbChan ! nt,255,Lf.d;
        set_priority(nt, 3);
        set_priority(_pid, 1) }
    :: d_step{ pOut>0 && ! pc ?? [eval(_pid),4] ->         /* t_lf req */
        pc ! _pid,4; }
  od unless atomic{ gbChan ?? eval(_pid),4,pc ->
        pOut--; break }                                    /* t_lf frg */
od }

init{
NetPlace pw1; bit pwIn=1;
atomic{ /* Initial Marking */
  nt = run SubNet(Subs, 1, 1, 2);  Subs.d ! nt,255;
  nt = run SubNet(Subs, 1, 1, 2);  Subs.d ! nt,255;
  nt = run SubNet(Subs, 1, 1, 2);  Subs.d ! nt,255;
  nt = run chiefNet(Chiefs);     Chiefs.d ! nt,255; }
do
  /********* Environment *********/
  :: ...
```

```
  /********* Cross-organizational Workflow *********/
  :: atomic{  pwIn>0 ->                                        /* u1 frg */
       set_priority(_pid, 6);
       pwIn--;
       nt = run W1Net(pw1.d); pw1.d ! nt,255;
       nt = run W2Net(pw1.d); pw1.d ! nt,255;
       nt = run ACNet(pw1.d); pw1.d ! nt,255;
       set_priority(_pid, 1) }
  :: d_step{  c_expr { numMsg(qptr(Pinit->pw1.d-1),4)>=3 } ->
       set_priority(_pid, 6);                                  /* tlf frg */
       recMsg(pw1.d, nt,4); consNet(pw1.d,nt);
       gbChan ! nt,4,pw1.d; set_priority(nt, 5);
       recMsg(pw1.d, nt,4); consNet(pw1.d,nt);
       gbChan ! nt,4,pw1.d; set_priority(nt, 5);
       recMsg(pw1.d, nt,4); consNet(pw1.d,nt);
       gbChan ! nt,4,pw1.d; set_priority(nt, 5);
       pwOut++;
       set_priority(_pid, 1) }
od }
```

References

1. Arbab, F., Aştefănoaei, L., de Boer, F.S., Dastani, M., Meyer, J.-J., Tinnermeier, N.: Reo connectors as coordination artifacts in 2APL systems. In: Bui, T.D., Ho, T.V., Ha, Q.T. (eds.) PRIMA 2008. LNCS (LNAI), vol. 5357, pp. 42–53. Springer, Heidelberg (2008)
2. Barkaoui, K., Hicheur, A.: Towards analysis of flexible and collaborative workflow using recursive ECATNets. In: ter Hofstede, A.H.M., Benatallah, B., Paik, H.-Y. (eds.) BPM Workshops 2007. LNCS, vol. 4928, pp. 232–244. Springer, Heidelberg (2008)
3. Bordini, R., Fisher, M., Visser, W., Wooldridge, M.: Verifying multi-agent programs by model checking. J. AAMAS **12**(2), 239–256 (2006)
4. Chang, L., He, X.: A model transformation approach for verifying multi-agent systems using SPIN. In: Proceedings of the ACM Symposium on Applied Computing, pp. 37–42(2011)
5. Chang, L., He, X., Shatz, S.M.: A methodology for modeling multi-agent systems using nested Petri nets. Int. J. Softw. Eng. Knowl. Eng. **22**(7), 891–925 (2012)
6. da Silva, F.S.C., Venero, M.L.F., David, D.M., Saleemb, M., Chung, P.W.: Interaction protocols for cross-organisational workflows. Knowl.-Based Syst. **37**, 121–136 (2013)
7. Dworzański, L., Lomazova, I.: CPN tools-assisted simulation and verification of nested Petri nets. Autom. Control Comput. Sci. **47**(7), 393–402 (2013)
8. Flores-Badillo, M., Padilla-Duarte, M., López-Mellado, E.: Modeling and simulation of mobile agents systems using a multi-level net formalism. In: Gelbukh, A., Reyes-Garcia, C.A. (eds.) MICAI 2006. LNCS (LNAI), vol. 4293, pp. 1128–1138. Springer, Heidelberg (2006)
9. Flores, M., Padilla, M., López, E.: Modeling and simulation of workflow processes using multi-level petri nets. In: Proceedings of the EOMAS, pp. 50–63 (2008)

10. Gannod, G.C., Gupta, S.: An automated tool for analyzing Petri Nets using SPIN. In: Proceedings of the 16th IEEE International Conference on Automated Software Engineering, pp. 404–407. IEEE Computer Society (2001)
11. Holzmann, G.J.: The SPIN Model Checker: Primer and Reference Manual. Addison-Wesley, Boston (2003)
12. Hunter, J., Raimondi, F., Rungta, N., Stocker, R.: A synergistic and extensible framework for multi-agent system verification. In: AAMAS, pp. 869–876 (2013)
13. Jensen, K.: Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Springer, New York (1992)
14. Jiao, W., Zhou, M., Wang, Q.: Formal framework for adaptive multi-agent systems. In: IEEE International Conference on Intelligent Agent Technology, pp. 442–445 (2003)
15. Kissoum, Y., Maamri, R., Sahnoun, Z.: Modeling smart home using the paradigm of nets within nets. In: Ramsay, A., Agre, G. (eds.) AIMSA 2012. LNCS, vol. 7557, pp. 286–295. Springer, Heidelberg (2012)
16. Kissoum, Y., Sahnoun, Z., Barkaoui, K.: An approach for testing mobile agents using the nets within nets paradigm. In: 3rd International Conference on Research Challenges in Information Science, pp. 207–216 (2009)
17. Lomazova, I.A.: Nested Petri Nets - a formalism for specification and verification of multi-agent distributed systems. Fundamenta Informaticae **43**(1–4), 195–214 (2000)
18. Marchese, M., Vaccari, L., Trecarichi, G., Osman, N., McNeill, F., Besana, P.: An interaction-centric approach to support peer coordination in distributed emergency response management. J. Intell. Decis. Technol. **3**(1), 19–34 (2009)
19. Miller, T., McBurney, P.: Using constraints and process algebra for specification of first-class agent interaction protocols. In: O'Hare, G.M.P., Ricci, A., O'Grady, M.J., Dikenelli, O. (eds.) ESAW 2006. LNCS (LNAI), vol. 4457, pp. 245–264. Springer, Heidelberg (2007)
20. Mislevics, A., Grundspenkis, J.: Integrating workflow-based mobile agents with cloud business process management systems. Int. J. New Comput. Architectures Appl. **2**(4), 511–530 (2012)
21. Prisecaru, O., Jucan, T.: Interorganizational workflow nets: a Petri net based approach for modelling and analyzing interorganizational workflows. In: EOMAS, pp. 64–78 (2008)
22. Robertson, D.: Multi-agent coordination as distributed logic programming. In: Demoen, B., Lifschitz, V. (eds.) ICLP 2004. LNCS, vol. 3132, pp. 416–430. Springer, Heidelberg (2004)
23. Singh, M.: Semantics and verification of information-based protocols. In: AAMAS, pp. 1149–1156 (2012)
24. Stuit, M., Szirbik, N.B.: Towards agent-based modeling and verification of collaborative business processes: an approach centered on interactions and behaviors. Int. J. Coop. Inf. Syst. **18**(3–4), 423–479 (2009)
25. Telang, P., Singh, M.: Specifying and verifying cross-organizational business models: an agent-oriented approach. IEEE T. Serv. Comput. **5**(3), 305–318 (2012)
26. Fernández Venero, M.L., da Silva, F.S.C.: On the use of SPIN for studying the behavior of nested Petri nets. In: Iyoda, J., de Moura, L. (eds.) SBMF 2013. LNCS, vol. 8195, pp. 83–98. Springer, Heidelberg (2013)
27. Venero, M.L.F., da Silva, F.S.C.: Modeling and simulating interaction protocols using nested petri nets. In: Counsell, S., Núñez, M. (eds.) SEFM 2013. LNCS, vol. 8368, pp. 135–150. Springer, Heidelberg (2014)

28. Venero, M.L.F., da Silva, F.S.C.: A general translation from nested Petri nets into PROMELA. CoRR, abs/1403.7991 (2014)
29. Wong, D., Paciorek, N., Walsh, T., DiCelie, J., Young, M., Peet, B.: Concordia: an infrastructure for collaborating mobile agents. In: Rothermel, K., Popescu-Zeletin, R. (eds.) MA 1997. LNCS, vol. 1219, pp. 86–97. Springer, Heidelberg (1997)
30. Yan, Y., Maamar, Z., Shen, W.: Integration of workflow and agent technology for business process management. In: 6th International Conference on Computer Supported Cooperative Work in Design, pp. 420–426 (2001)
31. Zambonelli, F.: Abstractions and infrastructures for the design and development of mobile agent organizations. In: Wooldridge, M.J., Weiß, G., Ciancarini, P. (eds.) AOSE 2001. LNCS, vol. 2222, pp. 245–262. Springer, Heidelberg (2002)

Modeling and Visualization of Urban Planning and Building Development Processes for Local Government of Small Settlements

Vojtěch Merunka[1,2(✉)] and Iveta Merunková[3]

[1] Department of Information Engineering, Faculty of Economics and Management, Czech University of Life Science Prague, Prague, Czech Republic
vmerunka@gmail.com
[2] Department of Software Engineering in Economy, Faculty of Nuclear Sciences and Physical Engineering, Czech Technical University in Prague, Prague, Czech Republic
[3] Department of Landscape Architecture, Faculty of Agrobiology, Food and Natural Sources, Czech University of Life Sciences Prague, Prague, Czech Republic
merunkova@af.czu.cz

Abstract. The actual and specific problem in the Czech local government of small settlements is very low level of participation of citizens in these small villages caused by the complexity and also time varying form of law and statutory regulations. All this affects the passivity of local citizens. Our paper presents the original computer-aided organizational modeling and simulation of relevant socio-technical processes of urban planning and building development. During our experiment we performed initial analysis and subsequent two statistical surveys, which confirmed us our hypothesis about the usefulness of our approach, because most people have confirmed to us that now they have a higher level of knowledge that will allow them to participate more in the future. As one of the dimensions of quality of life is also self-realization and participation of local people, we expect an improvement of the quality of life in general. Our approach is based on the original type of process maps, which describe the legislation, visualize it and simulate it. This gets local people a better understanding of their life situations and causes the effect of better participation and subsequent improvement of the quality of life.

Keywords: Local government · Small settlements · Computer-aided modeling and visualization · Business process · Urban planning · Building development · Knowledge transfer

1 Introduction

Nowadays we have to solve many problems related to the small settlement development and expansion, landscape care and over-all efforts to improve the quality of life and the level of democracy while preserving the conditions of the sustainable development (addressing living standard, cultural and historic value, agricultural and industrial production, transport infrastructure construction,

J. Barjis and R. Pergl (Eds.): EOMAS 2014, LNBIP 191, pp. 59–73, 2014.
DOI: 10.1007/978-3-662-44860-1_4

tourism potential, etc.). Technophobia of local people is here the significant factor for growing of this problem, because it is strongly contrasting with incoming investors and external people penetrating the rural area using good ICT (especially GIS and project management software) knowledge.

Business process models show and animate (when they are simulated) the collaboration of more participants within the solved system. We need this approach for simulation, validation and verification the real world problems. This issue is stressed in specific areas of technical systems analysis and design in area of agriculture, landscape management and country planning. A very important purpose of such a business model is to create and simulate an interconnected complex system where local actors, citizens, regional government, various interested organizations and partners and other participants mutually communicate. In addition to that, business process models are also the foundation of subsequent system modeling activities of software engineering, organizational design and management consulting. Typical way of performing these activities is to start directly with drawing process diagrams just during the initial interviews. But in this paper, we present the idea, that for better modeling, we need to use a specific textual technique, which helps us to recognize, define and refine our initial set of business process participants and their properties before the graphical business process model is assembled.

2 Motivation

There is very low level of knowledge in the area of participation at the processes of country planning and building development. Everybody together with many political declarations by the European Union. like Aarhus agreement and European Regional and Spatial Planning Charter by the European Council agree that computer technology can solve the problem of low community participation of people, which decreases the quality of life of these people.

Expected output of business process modeling and simulation activities is presentation of information and data in a form that can be directly used as a tool for knowledge improvement or implementation some system or organizational change in the spirit of management consulting. However, this is not the easy case; there are following issues described by [1,5]:

1. *oversimplification* - while trying to at least finish business and organizational model we are forced simplify the problem being modeled and
2. *inability* - some important details cannot be expressed because of the method being poorly used.

The practical impact of this wrong inequality between people is the *urban sprawl* as it is stressed by Frumkin in [4].

Urban sprawl (see Fig. 1) is a phenomenon that emerged in the last decades in the advanced industrial countries (USA, France and Great Britain) and recently also in our country. Inhabitants of affected settlements usually perceive the urban sprawl positively at first, mainly because of the lobbying. It can be described as

Fig. 1. Urban sprawl example.

an uncontrolled expansion of certain kind of urban build-up into the free landscape caused by favorable land prices, demand for cheap but modern estates, etc. Duany [3] writes about harmful absorption of original small settlement structures, which causes following negative effects:

1. Pawning of infrastructure development of the original settlement. New inhabitants fulfill themselves and shop only at the place of their work in a metropolis and the settlements are just a kind of sleeping accommodation for them. New inhabitants' lack of interest in contributing to the settlement development leads to misusing of democratic principles of the self-administration against the original local inhabitants. This leads to the rise of social segregation between the original and the new inhabitants.
2. Urban sprawl causes disruption of the cultural and historical value of the settlement, disruption of the ecological stability of the area, destruction of the public transport, loss of touristic attractiveness etc.
3. Loss of the quality agricultural soil.

The cause of the urban sprawl in the small settlement development is the fact that the elected technophobic members of local administrations (e.g. mayors and clerks) are not aware in all the details of GIS, law, state and local administration agenda and their effects on living in the settlements. They don't know how to use fully the technology in favor of the settlements and usually depend on a misleading interpretation provided by their governing bodies and more often by another subjects (usually privately involved in the process in question and thus biased).

3 Our Project

One of the actual and specific problems of the Czech landscape development is very low level of participation of citizens from small settlements in rural areas. We tried to model and simulate process-based knowledge in area of urban planning and building development in order to use organizational modeling and simulation as an instrument for strengthening participation of local people in agendas related to their real life situations.

We analyzed the legislation and local officials' knowledge related to the processes and agendas of the urban planning of the landscape areas and small settlements with regards to the new housing and building law and regional management trends in the European Union. Our project consisted of these parts:

1. Initial analysis of the problem. In order to formulate our hypothesis, we performed preliminary research on the documentation of past projects.
2. Statistical survey from 463 people (e.g. 8 % of the total population) in 13 small settlements in the Central Bohemian area. This survey get the knowledge about the level of participation. This survey was based on questions inspired by the CMM-based approach in services as it is described in [2]. Our questions were focused to support the hypothesis that the level of process knowledge is at the lowest level of maturity (*initial level*).
3. Modeling of the selected process-oriented knowledge of 40 agendas related to the urban planning and building development on the local level. The result was 40 visualized computer models of these processes showing detailed participation and mutual communication of human actors in the form of special and simplified combination of UML state-diagrams and activity-diagrams.
4. Created models (e.g. process diagrams) were used in the training of 57 people from the same villages as those 463 in the previous survey. In this phase we tested the improvement of their knowledge and behavior.

4 Where We Did Our Project

Our project has been performed in 13 small settlements (villages) of the Central Bohemia. (See Fig. 2) There villages were of two little bit different groups. First group (11 villages) belonged to localities very distant from the metropolis of Prague, which lies in the middle of the field. Second group (2 villages, but about one half of the population because of different size of these villages) belonged to areas within driving distance from Prague and therefore more vulnerable to the *urban sprawl* effect. We expected to find differences in the participation of people in both groups, but this has not been confirmed. The identified differences were minimal, only about at the level of statistical error.

5 Our Approach to the Process Visualization and Simulation

We improved and modified Business Object Relation Modeling (BORM), which is an approach to both process modeling and the subsequent development of

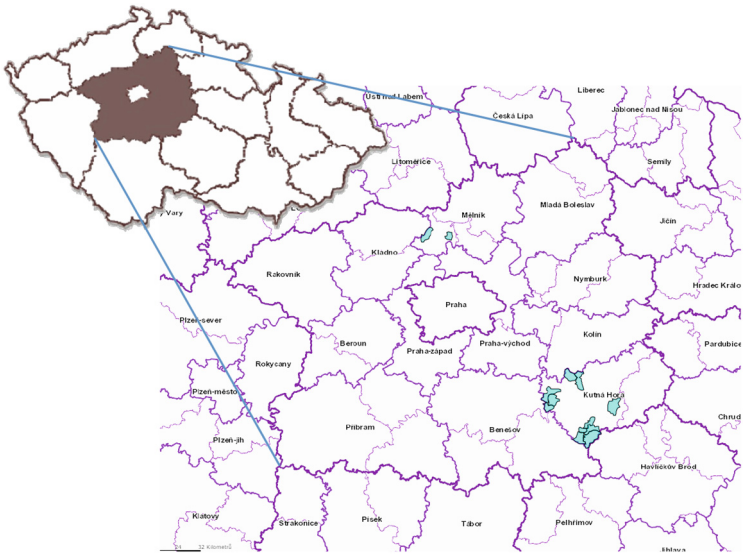


Fig. 2. Localities

information systems. It provides an approach that facilitates the description of how real business systems evolve, change and behave. BORM - Business Object Relation Modeling was originally developed in 1993 and was intended to provide seamless support for the building of object oriented software systems based on pure object-oriented languages, databases and distributed environments. Subsequently, it has been realized that this method has significant potential in business process modeling and other related business issues [6]. Our extension of this method for better modeling of local processes was firstly published at the HAICTA 2013 conference and used in the E.U. research workshop: Grundtvig project - Citizens of Mountainous Regions and Technophobia in Athens 2013.

Our approach is based on process models and their visual simulation. This helps the officials (especially in the smallest settlements) to clarify the legislation and shows them possible ways of its usage. Our models and their visual simulation show how it can be used to improve the process of decision-making on the level of mayors and local administrations. It offers the possibility to model and simulate real life situations in small settlements. The example at the Fig. 3 shows our business object diagram of a process of starting urban plan. Our modeling software shows the concrete simulation step. A diagram is a visual representation of object associations and communications in a particular process. Our notation is the re-used UML notation from the state diagram, activity diagram and sequence diagram UML but combined and simplified into the only one new diagram that shows the process as object-oriented participants in the form of mutually communicating Finite-State-Machines. Moreover, we can use a visual simulator in order to animate these processes and evaluate them.

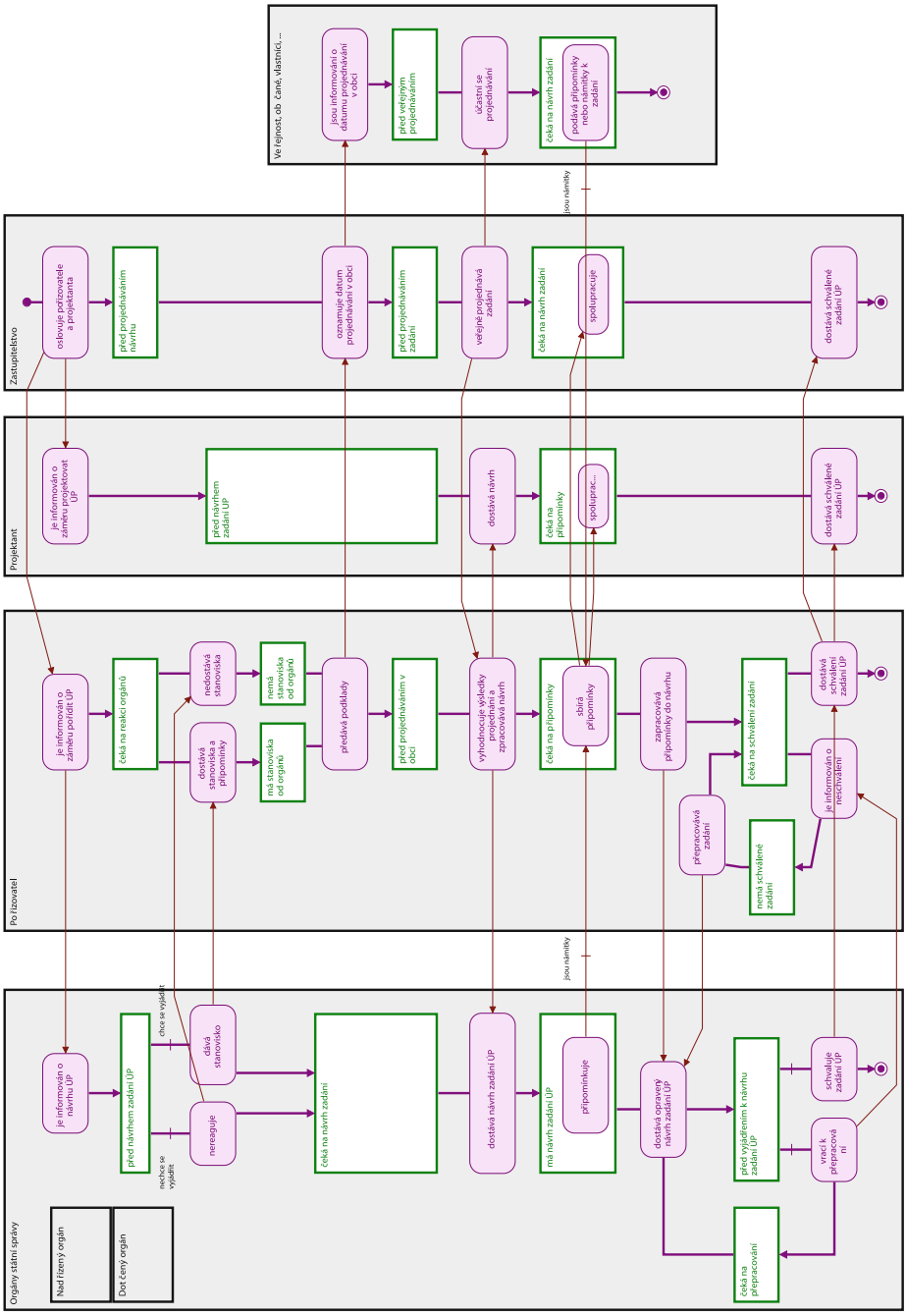


Fig. 3. Urban planning process.

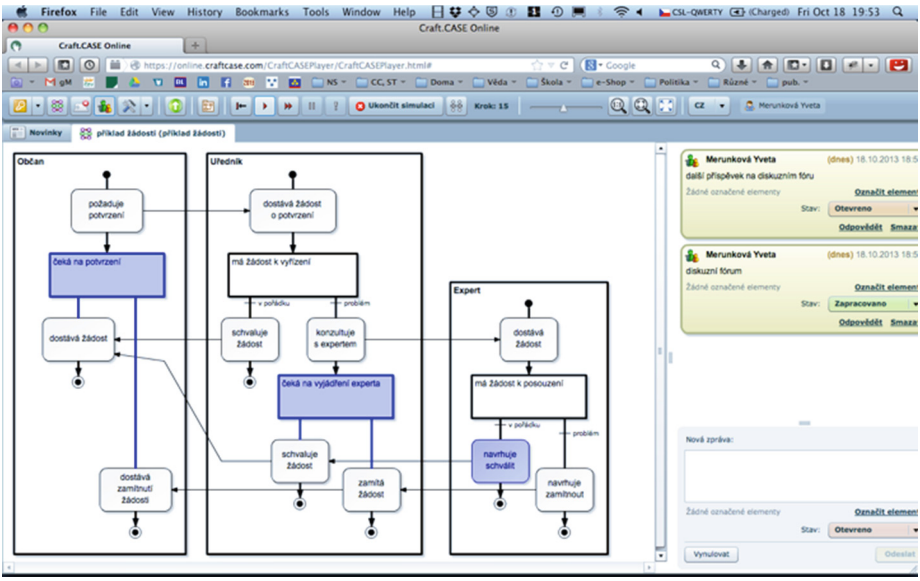


Fig. 4. Social network as communication of users during the process simulation.

Our simulator has included the communication module inspired by Facebook-like chatting. them within a group of users (Fig. 4).

Our BORM innovation is based on the reuse of old thoughts from the beginning of 1990 s regarding the description of object properties and behavior using finite state machines (FSM) and modeling the process-based knowledge as the communication of FSM, each representing a process participant. In this approach, states and situations are stressed in contrast to activities as it is in standard process diagrams. The first work expressing the possible merge of OOP (Object-Oriented Paradigm) and FSM was the Shaler's and Mellor's book in 1992 [7]. One of the first best books speaking about the applicability of OOP to the business modeling was written by Taylor in 1995 [8]. These works together with our practical experience is why we believe that the business requirement modeling and simulation and software modeling could be unified on the platform of OOP and FSM, where objects (e.g. process participants described as Mealy-type FSMs) are interconnected via messages (as it is in OOP) together in order to realize some business process. Furthermore, we introduced two textual tools, which we call modeling cards and scenarios.

Each modeled process participant has its own modeling card. (see example in Fig. 5) It is a special kind of small structured textual table describing in boxes participant's name, related legislation, documentation and responsibilities.

Scenario is also a very similar structured textual table, which refines the process. (see example in Fig. 6) Scenarios are written in a specific tabular form, that always includes at least following columns:

participant	
ACQUIRER	
states (phases in time)	activities
• Is asked for acquisition UP by municipality. • Is waiting for expression from the surrounding villages on the proposed assignment of UP. • Has completed proposal of UP. • Is informed by municipality about approval to start the UP.	• Creates the UP proposal and is responsible for its negotiation. • Submits the UP proposal to the concerned authorties and incoporates comments and opinions. • Cooperates with the municipality and collaborates with the projectant to present the proposal at public hearing in the village.
legislation	documentation
• Must have a certificate of special competence. • Law 183/2006 novelised 350/2012 valid from 1.1.2013, §44, §46 • Law 312/2002 novelised 46/2004	• Proposal of the urban plan. • Announcement of the village about entering. • Observations and discussion results.

Fig. 5. Participant modeling card example: country planning participant - Acquirer.

process	
ASSIGNMENT OF A URBAN PLAN	
how process begins	participants in this process
The municipality does not have a urban plan, but it needs it for the investment or development subsidies. Ot the urban plan is required to create the conditions for building and taking into account the sustainable development of the locality.	**municipality** **citizens in the village** **planner** **acquirer** **commercial entities** **relevant government authorities**
how process ends	
Entering the urban plan is approved and discussed by all participants and the UP project can begin.	
legislation	documentation
• Law 183/2006 novelised 350/2012 and valid from 1.1.2013, §44, §46 • Law 312/2002 novelised 46/2004	• Proposal of the urban plan. • Announcement of the village about entering. • Observations and discussion results.

Fig. 6. Scenario modeling card example: Assignment of an urban plan.

Table 1. Initial analysis results.

Locality	Requirements	Comments and protests (I.)	Comments and protests (II.)
Rašovice	0	1+0	0+0
Vavřinec	0	1+0	0+0
Vlastějovice	0	0+1	0+0
Dolní Pohleď	0	0+0	0+0
Rataje nad Sázavou	0	0+0	0+0
Horka	7	0+1	0+0
Soběšín	0	0+0	0+0
Slavošov	0	1+0	0+0
Podveky	4	1+0	0+0
Pertoltice	0	1+0	0+0
Červené Janovice	3	1+1	0+0
Nelahozeves	13	1+1	0+0
Dřínov	0	0+0	0+0

- Initiating situation, which is a brief and accurate verbal description of the beginning of the scenario and includes any inputs or entry conditions.
- Verbal description of the process itself.
- Set of participants, which is the set of those subjects (e.g. participants) of the system, which are required for the process.
- Result, which is a verbal description of the end and outputs of the scenario.
- Related legislation and documentation.

Once modeling cards and scenarios have been determined, it is good idea to validate them via an interview-driven process visualization during interview with selected stakeholders.

6 Results

6.1 Initial Analysis of the Problem

we performed preliminary research on the documentation of past projects in this phase of our projects. The results were alarming. As it is seen from Table 1, the local people did not participate at all in the second phase of the agendas, where they have the last opportunity to comment or protest anything. They participated only sporadically in the first phase. Just as bad, it was also in the collection of requirements when agendas have been started.

The alarming situation is best seen in the Fig. 7, which shows the overall average rate of participation of local citizens in those 40 agendas of their village development.

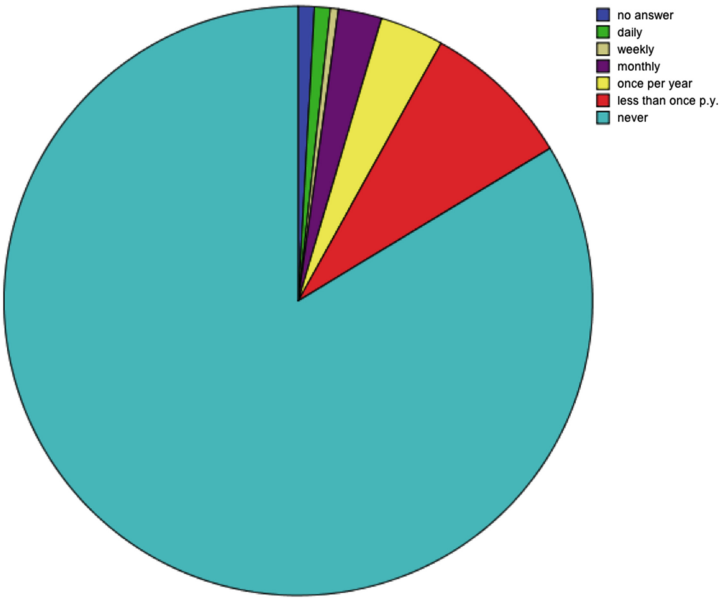


Fig. 7. Overall participation.

6.2 Statistical Survey

First survey was based on the questionnaire having 40 questions about details of processes in four areas of participation of the public and addressed to 463 people. These 40 questions were of 4 following groups:

1. New urban plan for a village.
2. Change of an old urban plan of a village.
3. Building management of a village.
4. Building permission.

The questions were about the knowledge of particular documentation and internal situations inside of these processes and about existing participation in these situations expressed in an ordinal scale (*daily, weekly, monthly, once per year, less than once per year, never*) or (*good, fair, something, very little, nothing*). For reason of mapping participation of people, we stressed the detail of particular processes. This is about concrete states and situations, where one can comment, ask or consult something in the process. This is very important, because legislation precisely defines situations, where concrete process participants have chance to participate in these processes.

Collected results were very alarming:

1. About 81 % people have no knowledge about these procedures.
2. About 84 % of people never participated in these processes.

3. About 10 % of people participated in these processes only once per more years.
4. In the result, only about 6 % of people participated on these processes at least once per one year.

6.3 Application and Verification

Based on these alarming results, we tried to apply our method and performed series of method presentations, training and semi-structured questionnaire meetings with 57 people collected by the non-discriminating snow-ball technique. After these workshops, 95 % of people evaluated our approach as a useful tool for breaking technophobia barriers of local officials in order to improve quality of local life via making better opportunities for local people to negotiate with outside interests, which often misused their low knowledge for private interests having the negative urban-sprawl impact on the countryside and rural landscape.

First set of questions have had the same aim as the previous survey. The aim was to determine and confirm the level of current knowledge by yet another statistical method in order to be sure about the results. In addition, we confirmed that the actual process knowledge was at the lowest level of the CMM scale - *initial level* only. We asked people about these questions before our method demonstration. The following figures provide the results of the percentage reduction of the statistical results obtained. Each first light gray column represents the first group of municipalities, the other dark gray column of the second group of villages, and the third black column is the total result from both groups. Figure 8 on the following page shows the results.

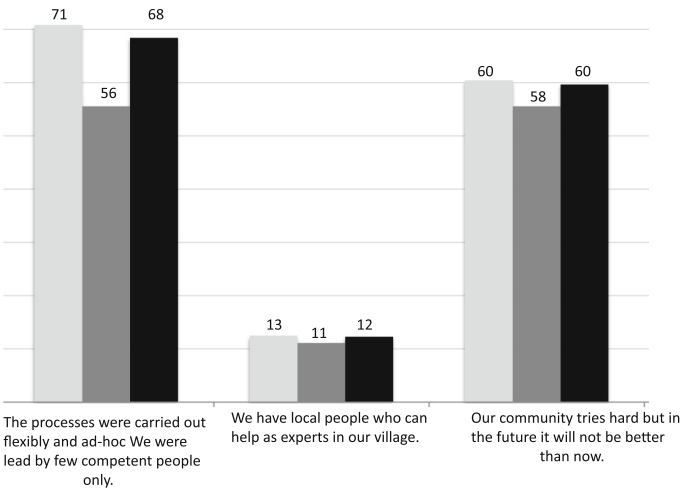


Fig. 8. Confirming the CMM initial level.

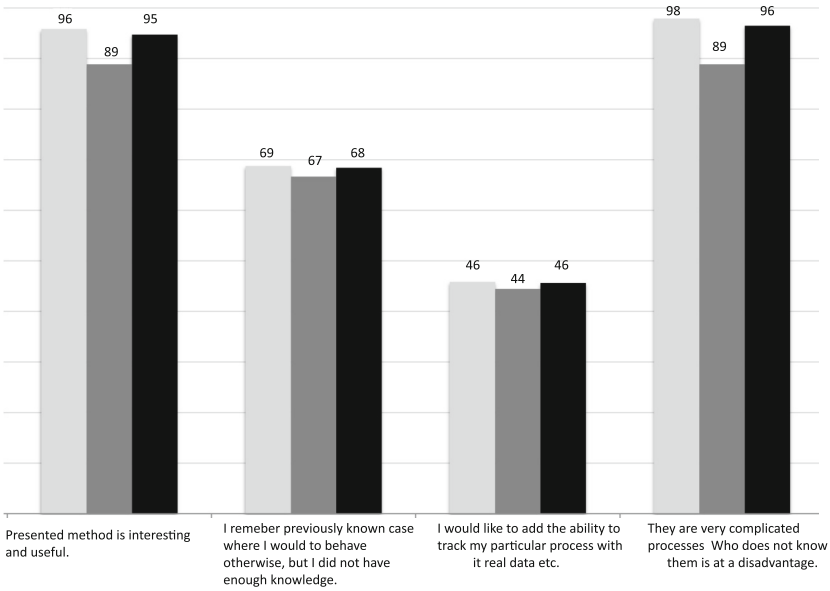


Fig. 9. Method evaluation.

Second set of questions was focused to the method. We asked people about these questions after our method demonstration. We got encouraging results, see Fig. 9.

Finally, we asked people about improvement of their knowledge and about their evaluation of past projects from the perspective of the newly acquired knowledge. The result is at Fig. 10. We also hereby confirm that we have leave the basic level of procedural knowledge according to the CMM approach to the third level (*defined*) out of total five levels, which defines this method. We consider it for success in the context of the environment.

The contingency between the answer about the efficiency of our new method and the estimated better participation was 0.64 (of possible interval between 0.0 and 1.0). Furthermore, based on the new knowledge, 67 % of tested people expressed the fact, that they were manipulated by external investors and lobbyists in the past due to their low level of knowledge. The detailed results are in Table 2.

Of course, when introducing our approach, the target staff is typically not educated in any computer science-related techniques. (Even if they are people from small settlements) On the other hand, the process-mapping phase must be performed quickly. This is why the analysis team does not have any time for detailed modeling courses such as the explanation of all aspects of used method with consequences into software engineering. Long time training to work with a modeling tool is also inappropriate. We have had time only for a very little introductory session on how to read our computer models. Here we felt the big

Table 2. Reduced results of semi-structured survey from 57 people.

Previous process knowledge at the level 1. of CMM [2]	We performed processes in ad-hoc manner, we were led by clerks, we did not manage our processes	39	68 %
	We have one or more experienced person in our village, who knows how to help us	7	12 %
	We do not expect better situation to the future, regardless our municipal representatives try hard	34	60 %
Method testing, understanding, process visualization	It is possible to understand this process visualization method in about 5-20 min	49	86 %
	Presented approach is interesting and useful	54	95 %
	Based on my new knowledge, I remember one or more events in the past, when I behaved wrong	39	68 %
	I would like to add new features into the visualization software. (documentation tracking, personalized workflow, ...)	26	46 %
	These processes are very complicated. Who does not know them, that one has a significant disadvantage	55	96 %
Improved process knowledge after usage the visualization method at the level between 2 and 3 of CMM [2]	Presented method is beneficial, I want to use it, I have better knowledge	54	95 %
	I will participate and forecast these processes better, I can plan better my activities	51	89 %
	I was informed in a misleading way by clerks, lobbyists or other people in the past	38	67 %
	I want more widely and more frequently to participate in these processes for my benefit and also for benefit of my community	35	61 %

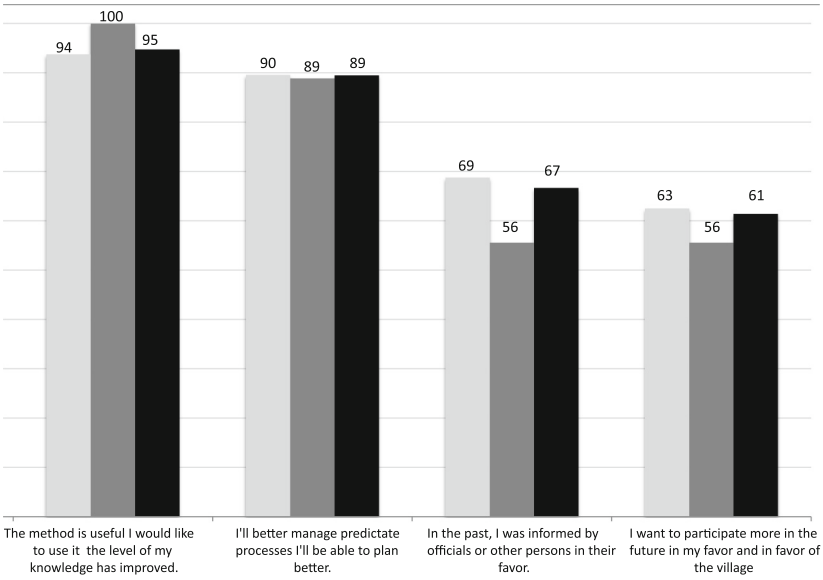


Fig. 10. New knowledge.

advantage of our approach, because this very introductory session typically took only 20 min of demonstration how it works. Almost all the people were able to start their work after this short introduction.

Finally, we recognized that the ICT equipment is not a big barrier for introducing our method as we expected on the assumption that the people were local from small settlements.

We prepared our models in the professional modeling application Craft.CASE and then transferred to the environment of freeware viewer and simulator, which is fully accessible via fast internet connection. This was not a problem, because the Central Bohemian region, where we did our project, is due to the relative proximity of Prague (capital city of the Czech Republic) very well equipped with high-speed Internet.

7 Conclusion

In this paper we presented our own project of modeling and subsequent visualization and simulation of process-based knowledge. The goal of this project was improvement of the quality of life by increasing the public life participation in small settlements in the Czech Republic via computer-based visualization of relevant law-based socio-technical processes (agendas) of urban development and country planning.

Our project described in this paper was about the organizational modeling and simulation as a tool for improvement the decision-making on the level of local

citizens. This idea of visualized and simulated real life situations was proven by a statistical survey. Data has been collected in questionaries from 463 people. Another survey was the practical evaluation of our method. This has been performed with 57 people. The results are valuable for the subsequent development of supporting information systems addressing better life situations of local people. We recognized the process-knowledge improvement from the *initial* level of CMM to the level three (*defined*) of CMM.

About the future work, we initiated cooperation with the Czech-Greece Chamber of Commerce in order to use our approach for the improvement of local people participation in development projects in the Czech Republic and Greece.

Acknowledgement. The authors would like to acknowledge the support of the research grant SGS14/209/OHK4/3T/14 and NAKI MK-S-3421/2011 OVV.

References

1. van der Aalst, W.M.P.: Business Process Simulation Revisited, keynote speech at the EOMAS workshop 2010 (2010). http://www.eomas.org. Accessed 10 April 2011
2. Christiansson, M.T.: A common process model to improve eService solutions - the municipality case. In: Proceedings of the 11th European Conference on eGovernment – ECEG 2011, Ljubljana, Slovenia (2011)
3. Dualny, A.: Suburban Nation the Rise of Sprawl and the Decline of the American Dream. North Point Press (2001) ISBN 978-0865476066
4. Frumkin, H., et al.: Urban Sprawl and Public Health: Designing, Planning, and Building for Healthy Communities. Island Press (2004) ISBN 978-1559633055
5. Igen, D., Hulin, C.L.: Computational modeling of behavior in organizations - the third scientific discipline. American Psychological Association, Washington DC (2000) ISBN 1-55798-639-8
6. Knott, R.P., Merunka, V., Polak, J.: The BORM methodology: a third-generation fully object-oriented methodology. In: Knowledge-Based Systems Elsevier Science International New York (2003) ISSN 0950–7051
7. Shlaer, S., Mellor, S.: Object Lifecycles: Modeling the World in States. Yourdon Press (1992) ISBN 0136299407
8. Taylor, D.A.: Business Engineering with Object Technology. John Wiley (1995) ISBN 0471045217

Enterprise Modelling
Formal Foundation

Enterprise Architecture: A Formalism for Modeling Organizational Structures in Information Systems

Alexander Lawall(✉), Thomas Schaller, and Dominik Reichelt

Institute for Information Systems at Hof University,
Alfons-Goppel-Platz 1, 95028 Hof, Germany
{alexander.lawall,thomas.schaller,dominik.reichelt}@iisys.de

Abstract. In this paper we give an overview on our approach to the field of organizational modeling that is mainly driven from the viewpoint of organization theory. This approach was proposed within the scope of the development of a workflow management system together with a big German energy provider and a case study with a major German insurance company. The examples used in the article are motivated by the insurance company domain.

Keywords: Organizational meta-model · Organizational model · Identity management · Workflow task assignment · User and rights management

1 Introduction

1.1 Research Subject

Looking at the various applications and systems needed for running a business one thing gets obvious: in almost every application there is the need to maintain a model of the organization structure, of the roles and the actors in order to define access rights or assign tasks to employees (in case of a workflow management system). These redundancies lead to a great maintenance overhead that – even for small businesses – can grow to a great burden. This problem can be avoided by deploying one logically central component offering this service to other applications. This component is called an organization server[1].

One important interface the server offers is a formal query language[2]. As a simplified example an expression could look like "clerk(claims department). HiringYear>10"[3]. On the basis of such an expression clients are now able to specify access rights or task assignments according to the real world needs. Let us

[1] Microsoft's Active Directory could be interpreted as an incarnation of such a server.
[2] The formal language is described in [LSR13a].
[3] We are looking for all clerks working in the claims department that have been working for the company for more than ten years.

J. Barjis and R. Pergl (Eds.): EOMAS 2014, LNBIP 191, pp. 77–95, 2014.
DOI: 10.1007/978-3-662-44860-1_5

Fig. 1. Task assignment based on the formal language

examine a simple policy definition scenario first. In Fig. 1, expressions are used to assign actors to tasks. The general rule is that all Managers that have been working for the company over half a year can be assigned to the task. The "OR"-term of the expression defines an additional rule by referring to all lawyers. At the time a user would like to execute the task, the client application passes the language expression to the organizational server. The server resolves the expression to a subset of matching employees. After that this set is passed back to the calling application (client). The client will grant access to the task if the user is an element of the returned subset. The case of permission rights is very similar. For each secured data object, a set of persons who have access rights are assigned to it. These so called actors can be easily specified using a language expression. At the moment a user wants to access the data object, the expression is handed to the organizational server. The server returns a set of employees satisfying the specification. The user has access if he is member of the result set of actors.

1.2 Related Work

There are a several approaches discussing the field of access control from the viewpoint of security administration. Only a few have their seeds in organization theory and are motivated from the workflow management discussion.

Lee, Kang and Hur describe in [LKH11] an interesting approach to overcome some problems with changes in organizational structures and workflows. The approach extends Role-based Access Control (RBAC) with organizational elements. The elements are restricted to organizational-units and roles which are combined with the users of RBAC concept. One key contribution is the assignment of access rights by users including an inheritance hierarchy.

Lawall et al. [LSR13b,LSR14] describe a formal language to formulate constraints (predicates) that can be assigned to relations. They distinguish between contexts (e.g. "PurchaseOrder") with or without parameters (e.g. "damage = 500") handed by an application (e.g. workflow management system) from a client and attributes of actors (e.g. "HiringYear"). This can be used to constrain relations (e.g. deputy relations).

Reference [CG13] describes algorithms to solve mathematically formulated workflow satisfaction problems. It includes the definition of constraints for the assignment of users to workflows. The proposed formal language includes the separation of duty concept (also addressed in [LSR13b]), as well as a formalism to specify that tasks must be assigned to the identical user.

The traditional RBAC is extended with organizational elements and an assignment language from roles to tasks is introduced in [WTG09]. The authors state that an organizational model is very important for workflows. The model of organizations is limited to structural, deputy and reporting relations. In reality, however, more than just these relations are present in companies. The organizational-unit formalized in this paper consists of different subtypes (i.a. organization, department and team). It limits the types of organizational-units to a fixed number. Organizational elements are exclusively resolved by means of predefined functions. It is not possible to simply follow relations within the organizational model. This causes maintenance effort if a new functionality (i.e. relation type) is needed. Beside the creation of the relation, the lookup mechanism for this relation must be implemented.

An ontological interpretation for the ARIS organization modeling language is proposed in [SAG10]. The paper shows a way to identify inappropriate elements of the language and gives recommendations for improvements. The assignment of persons to functional-units is limited in a way that persons can only have one function. If a person acts in different roles, an additional role must be created and assigned. In this approach, two different types (function and role) are required to specify the real scenario. Furthermore, attributes are limited to a predefined set of types (e.g. location). Thus problems occur if new attribute types are needed. In [SAG10], only human actors (persons) are allowed in the formalism.

Oh and Sandhu respectively Jing, Cai and Bu present in [OS02,JCB11] a model of a role administration system derived from organization theory. As will be shown in the following, there are some similarities but also major differences between their approach and the meta-model presented in this paper. The aforementioned papers describe hierarchical structures per dimension (e.g. product, organization or workstation). Each of these dimensions is structured using groups. A group can include parts from different dimensions as needed. The assignment of elements to tasks is not possible in a declarative way. It is defined at "compile-time". In reality, exceptions occur and therefore evaluation at "runtime" is essential in workflows.

Reference [DCs09] compares organizational models of three countries. The case studies discuss organizational structures in companies situated in Canada, China and Russia. Their analysis reveals different structures. For all countries, the functional organization structure is the most common, followed by the divisional. Additionally, matrix, network and virtual organizational structures are used in companies. Therefore, a meta-model that allows modeling such structures is required, e.g. for its application in workflows.

Schaad, Moffet and Jacob describe in [SMJ01] a case study, the role-based access control system of Dresdner Bank working with a role-based policy resolution server.

In [Seu01], Seufert gives a holistic overview from the access control perspective. Liu et al. in [LJRC08] present an analysis of the security perspective based on intelligent planning.

2 Motivation

2.1 Insights

Let's have a look at a real world scenario. A claims department usually has a manager, a number of clerks and a lawyer. Generally the lawyer is the deputy of the department head, cf. Fig. 2.

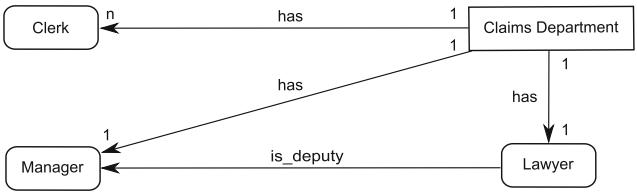


Fig. 2. Claims department in general

We examined two concrete departments: one responsible for "Car Damages" the other responsible for "House Damages". Compared to the general structure and policies we observed some differences (cf. Fig. 3). At "Car Damages" there was an additional secretary position. In absence of the manager, organizational tasks were assigned to the secretary position. There was a change in the deputyship between the department head and the lawyer as well. Byron[4], the lawyer, had been working in the department for only three weeks and therefore was not very experienced. The clerk Winter has been working in the department for over ten years. Based on that constellation the department head Smith decided that Winter should be his general deputy. Hinton was as well a deputy for Smith but only depending on some constraint information like the cash value of a claim for instance (constrained deputy relation in Fig. 3).

Looking at this two departments we also found an interesting mutual deputyship between the lawyers of the two departments (cf. Fig. 3). This observation gets important when thinking about dividing the organization system into types or classes on the one hand and instances on the other. Please note, that the relationships defined until now are specified on different levels of abstraction (positions and actors).

2.2 Lessons Learned

This section describes additional observations concerning real world organization policies[5].

[4] We are using fantasy names.
[5] A complete overview can be found in [Sch98].

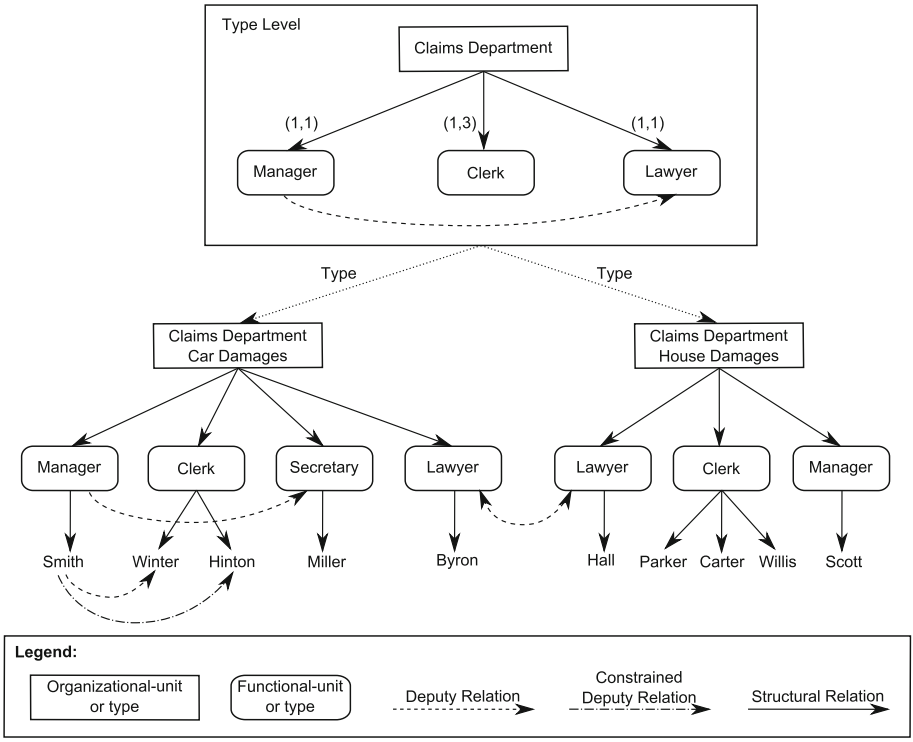


Fig. 3. Type and instance level of the example, adapted from [LSR13a, Fig. 3]

Knowledge Hierarchy. As we have seen there are different levels of organizational knowledge. On the top level general structural assertions like "a department consists of one to three clerks" are dominant. We call this level the *type* or *template* level. Knowledge on this level is based on experience and is changed seldom as time goes by. Looking at real world departments – we will call them *instances* – things become more concrete and specialized. There are concrete positions and the relationships between them. Finally, actors are assigned to the concrete positions. The organizational structures on this level are changing more frequently according to the demands of the daily business.

Relationships. An organization structure is formed by elements and relationships between them. It is important to realize the existence of several relationship types like "is_part_of", "is_deputy", "is_supervisor", "reports_to" and so on.

Positions are abstractions of persons (actors) having a defined skill set fulfilling specific tasks. These abstractions help defining a more stable model of the organization that is independent from employee turnover. Relationships can be defined between abstract positions or on the concrete actor level.

Relationships are rarely of a general nature. As discussed in our example, relationships depend on specific constraint information like the cash value of a car claim. Even the "is_deputy"-relationship can depend on projects or products if you think in the terms of a matrix organization. They can also be only valid for a fixed time period.

Multidimensional Organizations. Business organizations are multidimensional. Even in organizations that – at first glance – are structured hierarchically, there are structures belonging to the so called secondary ("shadow") organization comprising committees, commissions, boards and so on. The positions and functions of the secondary organization are assigned to the employees. This leads to a multidimensional organization in every case. We emphasize this point because there are some theories and approaches that are specific to hierarchical structures.

3 Resolving Expressions on the Organizational Model

Based on the organizational model, a resolution engine is proposed that implements a special architecture and a unique algorithm for the resolution of expressions. This engine is part of a greater system, the IT landscape of the company. ERP, workflow management or database management systems can be other components of the landscape. Each of these components uses mechanisms to map actors to tasks of processes or to permissions on data objects. Instead of maintaining such assignments for each system individually by total enumeration, we propose using an organizational server. This organizational server contains the organizational model. A formal language is used to formulate expressions that define the assignments. Clients send these expressions as requests to the organizational server and retrieve sets of actors as reply (cf. Fig. 4). The server offers a versatile interface consisting of only one function *dispatch* that returns a subset of the actors maintained within the organization model.

A major benefit of this approach is that the server forms the result set based on the current organizational model. This means that if actors change functions or relations, these changes have an immediate impact on the client systems. The language expressions remain unaltered. Before the organizational change, "Manager(*)" yields (according to Fig. 3) the actors Smith and Scott. If Scott leaves the company and is replaced by Willis, the model is changed. Now, the same expression evaluates to Smith and the new manager Willis.

An informal overview on the functionality of the proposed server based on the introduced scenario could be assumed as the following. A workflow management system passes the expression "Manager(Claims Department Car Damages)" to the organizational server. By traversing the organizational graph in Fig. 3, the algorithm moves to the department "Claims Department Car Damages" looking for a position "Manager". After that, the algorithm determines all the actors assigned to that position, finding manager Smith. If Smith is on the job, his identification is handed back to the workflow management system and the search

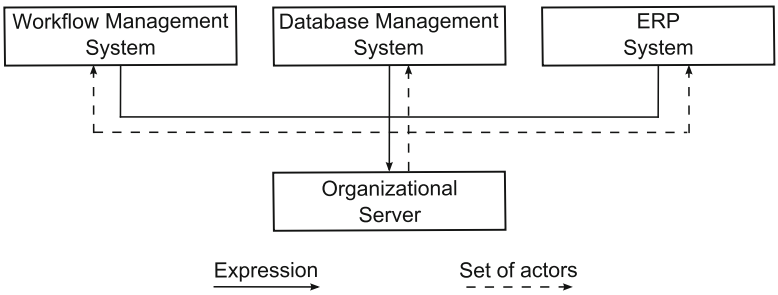


Fig. 4. Architecture

ends. In case that Smith is not available (e.g. through vacation or sickness) the algorithm searches for deputy relations between Smith and other actors. Obviously, there are two relations. If Winter **and** Hinton appear in the search result depends on the constraint on the relation to Hinton and whether they are on the job. In case of an empty set, the algorithm moves to the functional-unit manager looking for a deputy relation and finds the functional-unit secretary assigned to Miller. If Miller is on the job, her identification will be returned to the workflow management system. If not, the algorithm has the alternative of determining a valid deputy on the type level. Let us assume that the department is linked to the department type as depicted in Fig. 3. Within this type, the algorithm finds the lawyer as a deputy. It moves back to the instance "Claims Department Car Damages" and checks if there is a functional-unit with this name and an actor assigned to that functional-unit that is available. If Byron is on the job, his identification is returned. Otherwise, the lawyer of the "Claims Department Car Damages" has a two-way deputy relation with the lawyer of the "Claims Department House Damages". If this functional-unit has an actor assigned to itself and the actor is available, the algorithm will hand back his identification (here Hall, the lawyer of the "Claims Department House Damages"). Otherwise the returned set is empty. In this case, the workflow management system has to postpone the execution of the task.

4 Formal Specification

The formalization of the organizational model is described using relations and integrity constraints. In the following we present a simplified model of our approach.

Within our meta-model an organization is a tuple O = ($name$, $\mathcal{DOM}$, $\mathcal{ORG}$, $\mathfrak{R}$, $\mathcal{REL}$) where $name$ denotes the modeled organization. The remaining symbols have the following semantics:

4.1 Domains $\mathcal{DOM}$

$\mathcal{DOM} = \{\mathcal{BEZ}, T, ID, \mathcal{RN}, \mathcal{ATT}, \mathcal{W}, \mathcal{P}\}$ is a set of domains consisting of the subsets:

- $\mathcal{BEZ}$ an organization specific set of terms describing the building blocks of the organization, like *claims department*, department *head* and so on,
- T denotes a set of time values, like "May 19th 2010 08:00:00".
- ID a set of abstract identifiers.
- $\mathcal{RN}$ denotes a set of relationship names, *deputy* or *reports_to* for instance.
- $\mathcal{ATT}$ a set of attributes used to detail the elements of our model. Attributes are mapped to model elements using the function $val : \mathcal{ATT} \rightarrow \mathcal{W}$ that assigns a value $w \in \mathcal{W}$ to each $a \in \mathcal{ATT}$.
- $\mathcal{P}$ denotes a set of predicates like *(ActualYear - HiringYear) > 10*.

4.2 Organization Elements $\mathcal{ORG}$

The set $\mathcal{ORG} = \mathcal{OE} \cup \mathcal{F} \cup OE \cup F \cup A$ comprises all the building blocks of an organization on the type as well as on the instance level. The elements of $\mathcal{ORG}$ represent the nodes of the resulting organization graph.

- $\mathcal{OE}$ denotes the set of organizational-unit types, like departments or working groups.
- $\mathcal{F}$ is the set of functional-unit types, like the "manager", "lawyers" and so on.
- OE represents the set of organizational-units, like departments, committees, teams and so on. As already explained, organizational-units can have a relation to a type. The total function $type_{OE} : OE \rightarrow \mathcal{OE} \cup NULL$ returns the specific type for every organizational-unit.
- F is the set of functional-units, like positions or roles. There also exists a type function that is almost defined in the same manner as described above. The type of a functional-unit is returned by the function $type_F F : F \rightarrow \mathcal{F} \cup NULL$. $\Gamma_s^E \subset OE \times (OE \cup F)$ denotes the is_part_of-relation between organizational- and functional-units. Γ_s^E on the one hand describes the mapping of functional-units to organizational-units. On the other hand the hierarchy between orga-nizational-units can be modeled. When focusing on the organizational-units the relation $\Gamma_s^{E'} = \Gamma_s^E \rhd OE$ has to be irreflexive and cycle-free. $\Gamma_s^{E''} = \Gamma_s^E \rhd F$ has to be surjective.
- R^F denotes a set of user-defined relations. All members $r \in R^F$ have the structure $r \subset (F \times F)$ and are irreflexive.
- The set A denotes the actors: employees (users) and the computer systems. We explicitly model these computer systems because they can carry out tasks and therefore need permissions. $\Gamma_s^{FA} \subset F \times A$ is a relation and describes the assignments of employees to positions. As seen before, there is also a user-defined set of relationships R^A. All relations $r \in R^A$ have the structure $r \subset (A \times (A \cup F))$. Further on, for all $r \in R^A$ the condition $\forall r \in R^A : [(x, y) \in r \rightarrow x \neq y]$ holds, meaning that every relation $r \in R^A$ is cycle-free.
- Additionally every element of $\mathcal{ORG}$ is described as tuple $(id, name)$, with $id \in ID$ and $name \in \mathcal{BEZ}$.

4.3 Set of Relations $\mathfrak{R}$

$\mathfrak{R}$ denotes a set of relation sets and is defined as $\mathfrak{R} = \mathfrak{R}^{\Upsilon} \cup \mathfrak{R}^{A}$, with:

- $\mathfrak{R}^{\Upsilon}$ denotes the set of relations defined between the types of organizational- and functional-units. $\mathfrak{R}^{\Upsilon}$ is defined as $\mathfrak{R}^{\Upsilon} = \Gamma_s^{\Upsilon} \cup \mathfrak{R}_b^{\Upsilon}$, with:
  - $\Gamma_s^{\Upsilon} \subset \mathcal{OE} \times (\mathcal{OE} \cup \mathcal{F})$ is the "is_part_of"-relation on the type level. Concerning the structure between the elements of $\mathcal{OE}$, there are some restrictions. Let's say $\Gamma_s^{\Upsilon'} = \Gamma_s^{\Upsilon} \rhd \mathcal{OE}$[6]. An organizational-unit type can not be his own successor. $\Gamma_s^{\Upsilon'}$ therefore has to be irreflexive and cycle-free. Let's have a look at the functional-unit types. Obviously the relationship between $\mathcal{OE}$ and $\mathcal{F}$ can be described as $\Gamma_s^{\Upsilon''} = \Gamma_s^{\Upsilon} \rhd \mathcal{F}$. Since organizational-unit types combine functional-unit types, $\Gamma_s^{\Upsilon''}$ has to be total. On the other side, every $f \in \mathcal{F}$ has to be linked to an organizational-unit type $o \in \mathcal{OE}$. $\Gamma_s^{\Upsilon''}$ therefore has to be surjective.
  - As explained above, there is the need for a flexible integration of new relation-types into the model. Therefore we define a set of relation-types $\mathfrak{R}_b^{\Upsilon}$. Every relationship $\Gamma_b^{\Upsilon} \in R^{\Upsilon}$ has the structure $\Gamma_b^{\Upsilon} \subset (\mathcal{F} \times \mathcal{F}) \times \mathcal{P}$ and can further be constraint using predicates to restrict the set of valid functional-unit types and therefore the set of valid users[7]. Please note that Γ_b^{Υ} is used as variable. The relations between the functional-unit types, that can expressed using the term $dom(\Gamma_b^{\Upsilon})$, are irreflexive. Defining a deputyship between one node and itself is not very meaningful for instance. Concerning the predicates we additionally postulate that each $\Gamma_b^{\Upsilon} \in \mathfrak{R}_b^{\Upsilon}$ has to be a function, assigning each ordered pair $(f, f) \in \mathcal{F} \times \mathcal{F}$ a unique predicate $p \in \mathcal{P}$.
- $\mathfrak{R}^{A}$ defines several relations between organizational- and functional-unit types as well as actors. We declare $\mathfrak{R}^{A}$ as $\mathfrak{R}^{A} = \mathfrak{R}^{E} \cup \mathfrak{R}^{FA}$, with:
  - $\mathfrak{R}^{E} = \Gamma_s^{E} \cup \mathfrak{R}_b^{E}$ the set of relations between organizational- and functional-units, with:
    - $\Gamma_s^{E} \subset OE \times (OE \cup F)$ denotes the "is_part_of"-relation between organizational- and functional-units. On the one hand the relation describes the functional-units belonging to an organizational-unit, on the other hand the organization structure between the units themselves. Let $\Gamma_s^{E'} = \Gamma_s^{E} \rhd OE$ denote the structure between the organizational-units. According to our description $\Gamma_s^{E'}$ has to be irreflexive and cycle-free. In the same manner and similar to our definition of $\Gamma_s^{\Upsilon''}$,$\Gamma_s^{E''} = \Gamma_s^{E} \rhd F$ has to be total and surjective.
    - $\mathfrak{R}_b^{E}$ is a set of user-defined relations. Every single relation Γ_b^{E} within $\mathfrak{R}_b^{E}$ has the structure $\Gamma_b^{E} \subset F \times F$ and is irreflexive.
  - $\mathfrak{R}^{FA}$ denotes a set of relations between functional-units and actors. $\mathfrak{R}^{FA}$ is defined as $\mathfrak{R}^{FA} = \Gamma_s^{FA} \cup \mathfrak{R}_b^{FA}$, with:

[6] The operator $\rhd$ is defined as $((\Gamma \subset A \times B) \rhd (C \subset B)) := \{(x, y) \in \Gamma \mid y \in C\}$.
[7] Please take a look at the relation Γ_s^{FA} and its according constraints.

- $\Gamma_s^{FA} \subset F \times A$ describes the function assignments of the actors. We demand that every actor is named to at least one function.
- $\mathfrak{R}_b^{FA}$ a set of user-defined, irreflexive relations Γ_b^{FA} having the structure $\forall \Gamma_b^{FA} \in \mathfrak{R}_b^{FA} : \Gamma_b^{FA} \subset A \times (A \cup F)$. For every $\Gamma_b^{FA} \in \mathfrak{R}_b^{FA}$ the condition $\left[(x, y) \in \Gamma_b^{FA} \rightarrow x \neq y\right]$ holds.

All elements of our model can be detailed using time constraints and attributes.

4.4 Additional Relations $\mathcal{REL}$

$\mathcal{REL}$ consists of several relations and is defined as $\mathcal{REL} = \{\Gamma_{Time}, \Gamma_{\mathcal{ATT}}, \Gamma_{Card}, \Gamma_{val}, \Gamma_{Name}\}$, with:

- $\Gamma_{Time} \subset \left(\mathcal{ORG} \cup \mathfrak{R}\right) \times \left(T \times T\right)$ describes the duration of validity of every single organizational element in our model. Γ_{Time} therefore has to be a total relation.
  The two functions *start* and *stop* denote the birth and death of an organizational element. We define these functions as:

$$start : \left(\mathcal{ORG} \cup \mathfrak{R}\right) \rightarrow T, \text{ with the semantic } start(x \in (\mathcal{ORG} \cup \mathfrak{R})) = dom(ran(x \lhd \Gamma_{Time}))$$
$$stop : \left(\mathcal{ORG} \cup \mathfrak{R}\right) \rightarrow T, \text{ with the semantic } stop(x \in (\mathcal{ORG} \cup \mathfrak{R})) = ran(ran(x \lhd \Gamma_{Time}))$$

  It is obvious that the following constraint should hold: $\forall x \in \left(\mathcal{ORG} \cup \mathfrak{R}\right) : start(x) \leq stop(x)$. In order to define an existence ad infinitum we introduce the symbol "$*$" concerning the value of the *stop* function.
- $\Gamma_{\mathcal{ATT}} \subset \left(\mathcal{ORG} \cup \mathfrak{R}\right) \times \mathcal{ATT}$ assigns attributes to our organizational elements. $\Gamma_{\mathcal{ATT}}$ is a surjective relation. Thus every attribute can only be assigned to one organizational element.
- $\Gamma_{Card} \subset \Gamma_s^{\Upsilon} \times (\mathbb{N}_o \times \mathbb{N}_o)$ assigns cardinalities to our "is_part_of"-relation between organizational- and functional-unit types. Γ_{Card} is a total and unique relation.
  As abbreviations we define the functions *min* and *max*, with

$$min : \Gamma_s^{\Upsilon} \rightarrow \mathbb{N}_o, \text{ with the semantic } min(r \in \Gamma_s^{\Upsilon}) = dom(ran(r \lhd \Gamma_{Card}))$$
$$max : \Gamma_s^{\Upsilon} \rightarrow \mathbb{N}_o, \text{ with the semantic } max(r \in \Gamma_s^{\Upsilon}) = ran(ran(r \lhd \Gamma_{Card}))$$

  Additionally we demand $\forall r \in \Gamma_s^{\Upsilon} : min(r) \leq max(r)$.
- Via $\Gamma_{val} \subset \mathcal{P} \times (F^{\Upsilon} \cup A)$ each predicate, its typed functional-unit and actors is assigned. The predicate *true* holds for all typed functions and actors and we define $\forall x \in F^{\Upsilon} \cup A : \quad (true, x) \in \Gamma_{val}$.
- $\Gamma_{Name} \subset \left(\mathfrak{R}_b^{\Upsilon} \cup \mathfrak{R}_b^{E} \cup \mathfrak{R}_b^{FA}\right) \times \mathcal{RN}$ assigns names to our user-defined relations. None of these relations should be nameless. Γ_{Name} therefore has to be total and unique. As already mentioned, $\Gamma_s^{\Upsilon}, \Gamma_s^{E}$ and Γ_s^{FA} denote "is_part_of"-relations. A specific naming of these relations is therefore unnecessary.

4.5 Policy Resolution

As discussed in Sect. 3 our organizational server offers an interface with a very small footprint, consisting only of the function `dispatch`. The function returns a subset of the actors fulfilling the language expression in conjunction with conditions. These conditions are formulated via the following parameters expected by the function.

- a set of organizational-unit names $oe_{bez} \in \mathcal{BEZ}$,
- a set of tuples consisting of attributes and corresponding values $attr_{oe} \subset (\mathcal{BEZ} \times \mathcal{W})$ belonging to defined organizational-units,
- a set of functional-unit names $f_{bez} \in \mathcal{BEZ}$,
- a set of tuples consisting of attributes and corresponding values $attr_f \subset (\mathcal{BEZ} \times \mathcal{W})$ belonging to functional-units,
- a set of actor names $a_{bez} \in \mathcal{BEZ}$,
- a set of tuples consisting of attributes and related values $attr_a \subset (\mathcal{BEZ} \times \mathcal{W})$, belonging to actors,
- the name of a relation $rel \in \mathcal{RN}$ and
- a set of tuples consisting of attributes and corresponding values $attr_{rel} \subset (\mathcal{BEZ} \times \mathcal{W})$, belonging to that relation.

Algorithm 1 (dispatch)

```
(1)  funct dispatch(oe_bez ⊂ BEZ, attr_oe ⊂ (BEZ × W),
(2)     f_bez ⊂ BEZ, attr_f ⊂ (BEZ × W),
(3)     a_bez ⊂ BEZ, attr_a ⊂ (BEZ × W),
(4)     rel ∈ RN, attr_rel ⊂ (BEZ × W)) ⊂ A
(5)     begin
(6)        var ⟨oe ⊂ OE; f ⊂ F; a ⊂ A⟩;
(7)        / * First we have to determine the existing organizational elements * /
(8)        oe := (OE ▷ oe_bez);
(9)        f := (F ▷ f_bez);
(10)       a := (A ▷ a_bez);
(11)       if (oe ≠ {} ∨ attr_oe ≠ {})
(12)          then return   GetATbyOE(oe, attr_oe, f, attr_f, rel, attr_rel)
(13)       fi
(14)       if (f ≠ {} ∨ attr_f ≠ {})
(15)          then return   GetATbyF(f, attr_f, attr_a, rel, attr_rel)
(16)       fi
(17)       if (a ≠ {} ∨ attr_a ≠ {})
(18)          then return   GetAT(a, attr_a, rel, attr_rel)
(19)       fi
(20)       return{};
(21)    end
```

The execution of `dispatch({Claims Department Car Damages},{},{Clerk},{},{},{damage sum < $100},{},{})` is equivalent to search all clerks in the organizational-unit *car damages* that are authorized to sign claims with a damage

sum lower than 100 dollars. The execution will lead to a call of the `GetATbyOE`-function. Based on the values of *oe*, f and a `GetATbyOE` determines the organization elements fulfilling the remaining conditions specified in $attr_{oe}$, $attr_f$ and so on.

Algorithm 2 (GetATbyOE)

(1) funct GetATbyOE($oe \subset OE, attr_{oe} \subset (\mathcal{BEZ} \times \mathcal{W}), f \subset F,$
(2) $attr_f \subset (\mathcal{BEZ} \times \mathcal{W}), rel \in \mathcal{RN}, attr_{rel} \subset (\mathcal{BEZ} \times \mathcal{W}),$
(3) $attr_a \subset (\mathcal{BEZ} \times \mathcal{W})) \subset A$
(4) begin
(5) var $\langle f^{'} \subset F;\ oe_{successor}, oe^{'}_{successor} \subset OE;$
(6) $o_p \subset OE \times OE\rangle;$
(7) / * o_p denotes the vector corresponding to oe * /;
(8) $o_p := \{(x,y) | x \in oe \wedge y \in OE\};$
(9) $oe_{successor} := dom\left(\left(\left(\left(\Gamma_s^{E^{'}}\right)^*\right)^T\right) \circ o_p\right);$
(10) if $(oe_{successor} = \{\})$
(11) then $oe_{successor} := OE;$
(12) fi
(13) $oe^{'}_{successor} := GetOrgElements(oe_{successor}, attr_{oe});$
(14) if $(f = \{\})$
(15) then
(16) / * determine all functional-units of the organizational-units * /
(17) $f^{'} := ran(\{oe^{'}_{successor}\} \lhd \Gamma_s^E) \cap F;$
(18) return GetATbyF($f^{'}, attr_f, attr_a, rel, attr_{rel}$);
(19) else
(20) / * which of the preselected functional-units belong to * /
(21) / * the organizational-units determined in $oe^{'}_{successor}$? * /
(22) $f^{'} := ran(\{oe^{'}_{successor}\} \lhd \Gamma_s^E \rhd \{f\}) \cap F;$
(23) return GetATbyF($f^{'}, attr_f, attr_a, rel, attr_{rel}$);
(24) fi
(25) return $\{\}$;
(26) end

Line 8 describes the declaration of a relational vector. The calculation of all successors in line 9 is obtained by multiplying the transpose of the irreflexive closure of $\Gamma_s^{E^{'}}$ with our vector o_p. After that we select the appropriate organizational-units.

In the case of the functional-unit set f passed to our function is empty, the algorithm selects all functional-units of the calculated transitive-reflexive closure and calls the function `GetATbyF` (line 14). If f != {} the relevant functional-units are selected in dependency on the specified organizational-units. After that the function `GetATbyF` is called (line 19).

Algorithm 3 (GetATbyF)

(1) funct GetATbyF($f \subset F, attr_f \subset (\mathcal{BEZ} \times \mathcal{W}), attr_a \subset (\mathcal{BEZ} \times \mathcal{W}),$
(2) $rel \in \mathcal{RN}, attr_{rel} \subset (\mathcal{BEZ} \times \mathcal{W})) \subset A$
(3) begin

```
(4)      var ⟨f', f'' ⊂ F; a ⊂ A⟩;
(5)      if (attr_f = {})
(6)        then attr_f := ATT;
(7)      fi
(8)      f' := f;
(9)      if (f' = {})
(10)       then f' := F;
(11)     fi
(12)     f'' := GetOrgElements(f', attr_f);
(13)     a := ran(f'' ◁ Γ_s^FA);
(14)     return  GetAT(a, attr_a, rel, attr_rel);
(15)   end
```

Function `GetATbyF` determines the set of functional-units fulfilling the attributes passed over as parameters first. After that corresponding actors are selected and handed over to `GetAT`.

Algorithm 4 describes the core idea of our system. The passed parameters are being processed from the actor level up to level of organizational- and functional-unit types. Individual rules therefore have a higher priority than policies specified on the more abstract type level.

Algorithm 4 (GetAT)

```
(1)  funct GetAT(a ⊂ A, attr_a ⊂ (BEZ × W), rel ∈ RN, attr_rel ⊂ (BEZ × W)) ⊂ A
(2)    begin
(3)      if (attr_a = rel = attr_rel = {})
(4)        then return  a;
(5)      fi
(6)      /* exit for recursions */
(7)      if (attr_a ≠ {} ∧ (rel = attr_rel = {}))
(8)        then
(9)              if a = {} then a := A fi
(10)             return  GetOrgElements(a, attr_a);
(11)     fi
(12)     if rel ≠ {}
(13)       then
(14)             if a = {} then a := A fi
(15)              var ⟨Γ_x, Γ_x' ∈ R_b^FA; y ⊂ (F ∪ A)⟩;
(16)             /* is there a user-defined relation fulfilling the attributes? */
(17)             Γ_x' := dom(Γ_Name ▷ rel) ∩ R_b^FA;
(18)             Γ_x := GetOrgElements(Γ_x', attr_rel);
(19)             if Γ_x ≠ {}
(20)               then
(21)                    var ⟨z ⊂ A ∪ F; w, y ⊂ A⟩;
(22)                    z := ran(GetOrgElements({a} ◁ Γ_x, attr_rel));
(23)                    /* select the actors according to z
(24)                    w := z ∩ A;
(25)                    /* select the actors according to Γ_s^FA */
(26)                    y := ran({{z} ∩ F} ◁ Γ_s^FA);
```

```
(27)                    return  w ∪ y;
(28)                else
(29)                    var ⟨Γ_x^F, Γ_x^F' ⊂ ℜ_b^E⟩;
(30)                    Γ_x^F' := dom(Γ_Name ▷ rel) ∩ ℜ_b^E;
(31)                    Γ_x^F := GetOrgElements(Γ_x^F', attr_rel);
(32)                    if Γ_x^F ≠ {}
(33)                      then
(34)                          var ⟨a' ⊂ A⟩;
(35)                          / * select all actors connected to the functional-unit * /
(36)                          a' := ran({ran(Γ_x^F)} ◁ Γ_s^FA);
(37)                          return  GetAT(a', attr_a, {}, {});
(38)                       else / * lookup in the type level * /
(39)                          var ⟨Γ_x^F, Γ_x^F' ⊂ ℜ_b^Υ⟩;
(40)                          Γ_x^F' := dom(Γ_Name ▷ rel) ∩ ℜ_b^Υ;
(41)                          Γ_x^F := GetOrgElements(Γ_x^F', attr_rel);
(42)                          if Γ_x^F ≠ {}
(43)                            then
(44)                                var ⟨f', f'' ⊂ F; a', a'', a''' ⊂ A⟩;
(45)                                / * 1. address all true relationships * /
(46)                                f' := dom(type_F^Υ ▷ ran(dom(Γ_x^F ▷ {wahr})));
(47)                                a' := ran(f' ◁ Γ_s^FA);
(48)                                / * 2. address false relationships * /
(49)                                / * 2.1 address typed functional-units * /
(50)                                f'' := dom((type_F^Υ ▷ ran(dom(Γ_x^F ⊳ wahr)))
(51)                                ∩
(52)                                ran(ran(Γ_x^F ⊳ {wahr}) ◁ Γ_val));
(53)                                a'' := f'' ◁ Γ_s^FA;
(54)                                / * 2.2 address direct relationships * /
(55)                                a''' :=
(56)                                (dom(type_F^Υ ▷ ran(dom(Γ_x^F ⊳ {wahr}))) ◁ Γ_s^FA)
(57)                                ∩ran(ran(Γ_x^F ⊳ {wahr}) ◁ Γ_val));
(58)                                return  GetAT(a' ∪ a'' ∪ a''', attr_a, {}, {});
(59)                             else
(60)                                / * no corresponding relationship found * /
(62)                                return  {};
(63)                          fi
(64)                    fi
(65)            fi
(66)     fi
(67)  end
```

Line 3 describes the trivial case. If the only parameter is a set of actors the return value will be the same set.

In case that attributes are handed over (that have to be fulfilled by the respective actors) and no user-defined relation was defined (line 7), the algorithm determines all actors with a fulfilling attribute set. If a is empty, all actors (line 9) are used within the search (line 10).

The core concept of the algorithm starts with line 12. The following user-defined relations have to be resolved:

- The relation between actors and functional-units ($\mathfrak{R}_b^{FA}$),
- The relation between functional-units ($\mathfrak{R}_b^{E}$) and
- The relation between functional-unit types ($\mathfrak{R}_b^{\Upsilon}$)

If no corresponding relation on the actor level can be found (line 12), the algorithm checks for a connection between two functional-units ($\mathfrak{R}_b^{E}$) in order to retrieve the attached actors. If no match is possible, the algorithm searches for a relation on the more abstract type level (line 40). If a relation can be found these policies are used for retrieving matching actors on the instance level. Lines 45 to 58 show the semantics of the predicates $p \in \mathcal{P}$ of the relation $\Gamma_b^{\Upsilon} \in \mathfrak{R}_b^{\Upsilon}$. All not predicate constrained relations are evaluated (line 45), first. After that the algorithm examines the constrained relations (line 48). The resulting actor set is used for a recursive call of Algorithm 4.

Algorithm `GetOrgElements` selects those elements fulfilling a defined attribute set *attr* from the set of organizational-units and relations.

Algorithm 5 (GetOrgElements)

```
(1)  funct GetOrgElements(k ⊂ ORG ∪ REL, attr ⊂ (BEZ × W)) ⊂ ORG ∪ REL
(2)    begin
(3)      var ⟨x ⊂ ORG ∪ REL⟩;
(4)      x := {};
(5)      foreach y ∈ k do
(6)        if CheckAttributes(y, attr)
(7)          then
(8)                x := x ∪ y;
(9)        fi
(10)     od
(11)     return x;
(12)   end
```

The following algorithm checks if an attribute of set *attr* is mapped to an organizational element or relation $k \in \mathcal{ORG} \cup \mathcal{REL}$.

Algorithm 6 (CheckAttributes)

```
(1)  funct CheckAttributes(k ∈ ORG ∪ REL, attr ⊂ (BEZ × W)) ⊂ B
(2)    begin
(3)      var ⟨attr_k ⊂ ATT⟩;
(4)      if (attr = {})
(5)        then return true;
(6)        else
(7)              attr_k := ran(k ◁ Γ_ATT);
(8)              if (dom(attr) ⊂ ran(attr_k))
(9)                then
(10)                    foreach y ∈ attr do
(11)                      if (ran(y) ≠ val(attr_k ▷ dom(y)))
```

```
(12)                    then return  false;
(13)                fi
(14)            od
(15)            return  true;
(16)        else return  false;
(17)    fi
(18)  fi
(19)  end
```

If the required attributes and attribute values are mapped to an organizational element or relation, function `CheckAttributes` returns $true \in \mathbb{B}$, or otherwise $false \in \mathbb{B}$. $\mathbb{B} = \{true, false\}$ denotes the set of boolean values.

5 Implementation

The formalism discussed in this contribution is implemented in a prototype. Figure 5 depicts part of this implementation – the graphical user interface (GUI). It contains a *model* editor, a *search* area, a *tree-navigation* as well as an *attribute pane* and a *relation list* for a selected organizational element.

The *model editor* provides a graph-based view on the organizational structure. Organizational elements are represented as nodes and their relations as edges. It provides means to navigate the model by centering on selected nodes. As the central component of the user interface, it is discussed below in more detail.

The *search area* can be used to retrieve a list of organizational elements. It has two modes of operation:

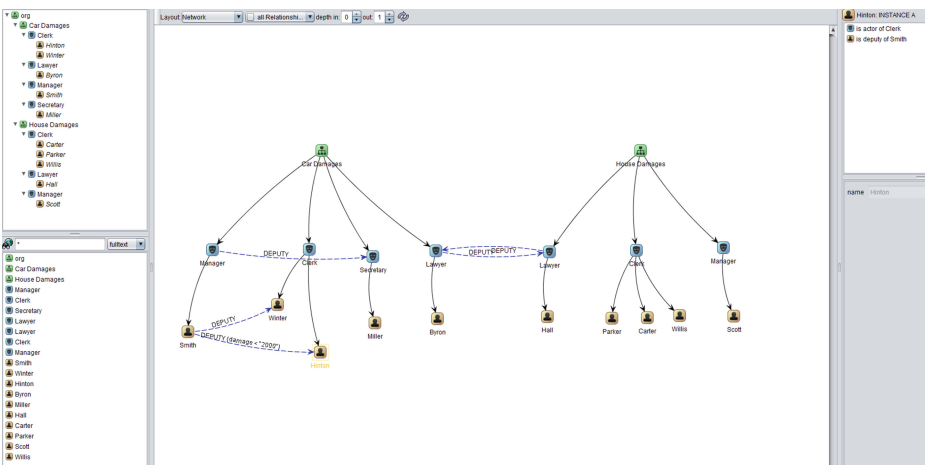


Fig. 5. Screenshot: implementation of the $\mathcal{C} - \mathcal{ORG}$ GUI

1. It provides a simple text index search for attribute values, e.g. entering "Wi*" will yield Winter and Willis.
2. It can also be used to evaluate language expressions based on the approach described in Sects. 3 and 4.5. An expression is entered and the result set for the current state of the organizational model is shown.

The *tree-navigation* projects the concrete organizational structure on a tree. Consequently, entities are duplicated in the projection if they can be reached on different paths.

The *attribute pane* in the bottom right section shows the attributes of the currently selected node or relation. It allows a quick modification, e.g. the assignment of a predicate to a relation.

The *relation list* lists all relations of the currently selected node, independent from the relation-types hidden in the model editor. This allows access to connected nodes and significantly reduces the time required to alter existing relations.

For quick access, elements can be dragged from any of the outer GUI sections and dropped into the model editor. If the elements have existing relations to the nodes already shown in the model editor, these relations will be shown as well. Otherwise, the elements are represented as unconnected nodes.

Figure 6 provides an enlarged view of the model editor[8]. Users perform most modifications of the organizational model via this component. In addition to navigating the model, they can create, modify and delete organizational elements and their interconnections.

It contains the model with the desired[9] relations. The editor also shows concrete constraints (predicates) on relations, e.g. the deputy relation with *damage* $<$ *"2000"* between *Smith* and *Hinton*.

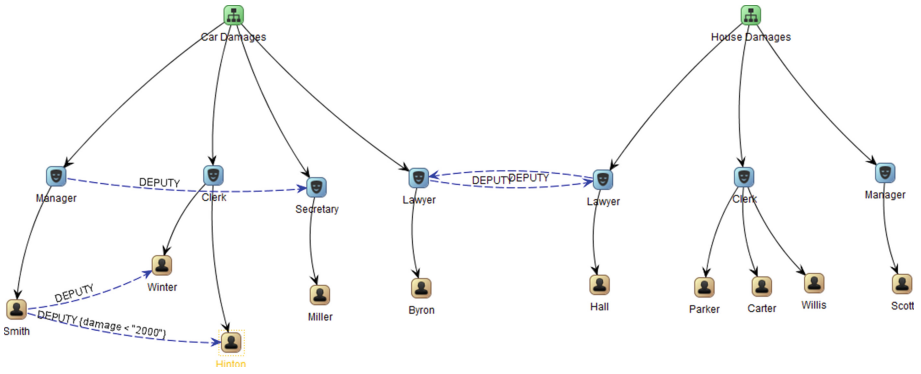


Fig. 6. Model region of the implementation

[8] It shows the example model (cf. Fig. 3). The type level is hidden.
[9] The relation-types to be shown can be selected.

In addition to the user interface, the implementation provides a service that can accept language expressions from client systems. This interface is based on the Representational State Transfer (REST) paradigm.

6 Conclusion and Further Research

Various organizational structures, such as hierarchical, matrix, tensor, and network structures, can be formally expressed by the introduced meta-model. A concrete organizational model of a company, institute or university etc. can be stored in the organizational server. All different systems of the IT landscape can use the model for the definition of access rights or task assignments. The formal language enables this mapping of actors.

The concept is not limited to the discussed monolithic systems. Further research concentrates on permissions for cloud resources and the substitution of mailing lists by addressing recipients using language expressions in conjunction with the organizational model.

References

[CG13] Crampton, J., Gutin, G.: Constraint expressions and workflow satisfiability. In: Proceedings of the 18th ACM Symposium on Access Control Models and Technologies, SACMAT '13, pp. 73–84. ACM, New York (2013)

[DCs09] Denis, P., Chun-sheng, S.: Comparing the organization structure of management in canadian, russian and chinese enterprises. In: International Conference on Management Science and Engineering, ICMSE 200, pp. 859–866 (2009)

[JCB11] Jing, M., Cai, H., Bu, F.: Flexible organization structure-based access control model and application. In: 2011 IEEE Asia-Pacific Services Computing Conference (APSCC), pp. 1–8 (2011)

[LJRC08] Liu, Q., Jiang, Y.-F., Rao, D.-N., Chen, X.-D.: Research of security analysis of arbac policy based on intelligence planning. In: 2008 International Conference on Computational Intelligence and Security, pp. 257–262 (2008)

[LKH11] Lee, J., Kang, S., Hur, S.: Access control using extended role graph corresponding to organizational hierarchy. In: 2011 First ACIS/JNU International Conference on Computers, Networks, Systems and Industrial Engineering (CNSI), pp. 468–473 (2011)

[LSR13a] Lawall, A., Schaller, T., Reichelt, D.: Integration of dynamic role resolution within the S-BPM approach. In: Fischer, H., Schneeberger, J. (eds.) S-BPM ONE 2013. CCIS, vol. 360, pp. 21–33. Springer, Heidelberg (2013)

[LSR13b] Lawall, A., Schaller, T., Reichelt, D.: Who does what - comparison of approaches for the definition of agents in workflows. In: 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), vol. 3, pp. 74–77, November 2013

[LSR14] Lawall, A., Schaller, T., Reichelt, D.: Cross-organizational and context-sensitive modeling of organizational dependencies in $\mathcal{C}-\mathcal{ORG}$. In: Nanopoulos, A., Schmidt, W. (eds.) S-BPM ONE 2014. LNBIP, vol. 170, pp. 89–109. Springer, Heidelberg (2014)

[OS02] Oh, S., Sandhu, R.: A model for role administration using organization structure. In: Seventh ACM Symposium on Access Control Models and Technologies, pp. 155–162. ACM Press (2002)

[SAG10] Santos, P.S., Almeida, J.P.A., Guizzardi, G.: An ontology-based semantic foundation for organizational structure modeling in the aris method. In: 2010 14th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW), pp. 272–282 (2010)

[Sch98] Schaller, T.W.: Organisationsverwaltung in CSCW-Systemen. Dissertation Universität Bamberg (1998)

[Seu01] Seufert, S.: Die Zugriffskontrolle - Eine Bestandsaufnahme relevanter Ansätze und deren Weiterentwicklung zu einem Konzept für die Ableitung von Zugriffsrechten aus der betrieblichen Organisation, Ph.D. thesis (2001)

[SMJ01] Schaad, A., Moffett, J., Jacob, J.: The role-based access control system of a european bank: a case study and discussion. In: Proceedings of the Sixth ACM Symposium on Access control models and technologies, pp. 3–9. ACM Press (2001)

[WTG09] Wang, T., Tan, Q., Guo, Y.: Enterprise organization oriented workflow task assignment language. In: International Conference on Advanced Computer Control, ICACC '09, pp. 79–86 (2009)

Business Rules, Constraints and Simulation for Enterprise Governance

Amjad Fayoumi[1(✉)] and Pericles Loucopoulos[2]

[1] Horizon Digital Economy,
University of Nottingham, Nottingham NG7 2RD, UK
amjad.fayoumi@nottingham.ac.uk
[2] Harokopio University of Athens, Athens, Greece
p.loucopoulos@hua.gr

Abstract. This paper aims to describe the main challenges to enterprise governance and proposes a hybrid approach based on business rules, constraints modelling and simulation. This approach can help to identify the high-level governance requirements and relate strategic, operational and IT governance aspects together. One of the important values of this approach is aligning governance knowledge in order to improve rules tracking between context and governance requirements thus catering for early identification of conflicts and other risk issues.

Keywords: Enterprise governance · Rule based modelling · Semantic of business vocabulary and business rules · System dynamic modelling · Hybrid enterprise rule approach

1 Introduction

Nowadays, enterprise work becomes increasingly complex due to the blurring of the boundaries of an enterprise's activities, influencers and impact, coupled with the economic and virtual business models in a globalised world. There are large volumes of information, knowledge and experience in either tacit or explicit forms that shape the business activities within any enterprise. Failure in understanding social and contextual forces may cause a failure in performing business activities and in building information systems supporting these activities [1]. Baxter and Sommerville [2] argued that IT projects often fail because they do not recognise the social and organisational complexity of the environment. To allow for better understanding of these issues, tools that help us to recognise and facilitate such complexity are needed. However, to design these efficiently, the alignment of the different scales in the complex systems (global, national, organisational, group and individual) is crucial. This alignment can help in moving from understanding the environment, to designing operations, policies and information systems. Morabito et al. [3] recommended that organisations should consider several stages of alignment; these alignments should consider the main knowledge ontologies which are the dynamic, static, social and intentional aspects as described and discussed in [4]. In order to do so, it is very important to codify the enterprise and its context knowledge, and to develop formal models that allow efficient

J. Barjis and R. Pergl (Eds.): EOMAS 2014, LNBIP 191, pp. 96–112, 2014.
DOI: 10.1007/978-3-662-44860-1_6

analysis, simulation mapping and implementation through the whole enterprise scale. Rule-based modelling is one of the well-regarded enterprise modelling techniques for codifying knowledge and it has very strong relations to the governance mechanisms in the enterprise. Recently, fact-oriented modelling was also proposed as a way to analyse business requirements toward information system development. Furthermore, business rules have been investigated widely with respect to the business processes, but in terms of overall enterprise governance, there is still more work to do. We believe that it is necessary to add the capability to understand the enterprise's behaviour under specific governance assumptions, thus to investigate the impact of each event/behaviour on the entire enterprise. We also believe that there is need for full alignment of business activities with internal and external events. In [1] two important principles were discussed and are relevant to this research: (1) intertwines requirements and contexts and (2) evolve designs with ecologies. An important research question was asked: how can we detect and deal with new types of indeterminism when dealing with a multitude of business rules? [1].

In this paper, we investigate how rules-based modelling can help enterprises to analyse and design their governances at all levels. We propose a hybrid approach based on facts, rules, decision and dynamic modelling. These aspects are semantically aligned and help to govern business operations. Section 2 discusses the main governance influencing factors in contemporary enterprises. Section 3 provides a review of the rule-based modelling. Section 4 provides a conceptual model of the proposed approach in terms of an overall meta-model and a set of methodological observations. Section 5 applies these to a case study and briefly discusses the evaluation of the approach. Then the paper concludes with Sect. 6 and includes any reflections and plans for future work.

2 Factors Affecting Enterprise Governance

With reference to the ISACA framework [5], factors influencing the enterprise are classified into two major categories: (1) Environmental, which is further divided into (a) internal environment and (b) external environment; (2) Capability of the organisation with the focus of the ISACA mainly on IT-based capabilities. However, in this section we elaborate more on the first category, which is focused on the concerns of the enterprise as a whole, rather than IT systems alone. Based on an intensive literature review and twenty years industrial experience, many enterprise influencing factors were recognised and discussed.

External Factors: It has been argued that the enterprise's activities evolve within an evolving context [1]. The environment's rules are rapidly changing and many factors should be considered when designing enterprise governance systems in order to avoid inconsistency, conflicting or wrong assessments, and to increase the operational performance. Despite the fact that many enterprises now operate globally, different types of influencing factors are affecting the enterprise; some of these factors are external such as: the global and regional economy, resources, consumer perspective/demand, purchasing power, competitors, business partners, ecology, regulating bodies, etc. both of the Industry's policies and the country's policies might enforce new levels of

restrictions on the enterprise activities in specific industries, e.g. strong regulations were applied to the healthcare and medicine industries to enforce safety and eligibility of the enterprise operations. Also, the understanding of a country's culture and norms in which the enterprises operate or target to operate is considered an important key success factor.

Internal Factors: External factors and organisational internal factors are intertwining and balancing, consequently this has a crucial impact on the enterprise activities and on the performers. Some of the recognised internal influencing factors are the common norms and value systems. One of the important reported internal factors is the divisional and inter/intra-organisational requirements conflict where holding power will force the agenda on other stakeholders and then possible compliance issues may appear. Equilibrium fulcrum can be found in balanced management styles (heterogeneous of top-down and bottom-up) which will impact on performance, knowledge sharing and governance mechanism inside the organisations. Employees' motivations, goals, expectations, capabilities, skills and background play a crucial role in the enterprise performance. Emergent technologies will probably disturb the market's direction and the customer demand, thus, understanding and adapting the new technology can increase competitive advantages. Also, the technology used by the enterprise can influence the activities as well as facilitate, assess and help in decision-making. Ordinary technical components (legacy systems) are not enough to build intelligent enterprises in the information era. However, privacy and security of information is an open issue and is considered as an important aspect of the enterprise governance since many enterprises have lost and are still losing millions of pounds due to the lack of technical and ISs' security. Access control policies, cyber protection and mature information systems' constraints can reduce such risks and increase agility in response to access change.

Enterprises need to focus on enforcing governance constraints rather than thinking about how to prevent operation errors. Managers can engage stakeholders in designing and planning for governance in order to minimise conflict and incompatibility from different viewpoints. Deciding the core business activities and supportive business activities will help to decide how to manage the enterprise with the focus on delivering/making value. Enterprise performance should have clear measures against objectives so that organisations can decide if the performance is suitable for the aimed growth. Also, what is the control level that the stakeholders feel comfortable to deal with within the designed governance system (policies and rules) and how much they feel it is well-suited to their daily activities requirements; this needs to be continuously assessed during the governance's design and operation. One of the techniques used by enterprises to deliver enforcement and manage relation with consumers is Service Level Agreement (SLA). Other related work can be found in methods of business impact analysis (BIA) and enterprise risk management (ERM) [6] to insure business continuity. However, many organisations experienced failure in governance system implementation due to the negative user experience and lack of compliance.

As discussed in this section, various internal and external factors influence the enterprise activities and the underlying information systems' requirements. Enterprise responsiveness becomes a crucial factor for enterprise success, and in order to increase

the ability to respond to different types of influencers in the required time and optimum response, the enterprise needs to capture and classify the contextual and environmental knowledge. Strategic thinking and decisions will be made based on intensive contextual awareness and analysis. Figure 1 shows a conceptual illustration of the enterprise governance influencing factors.

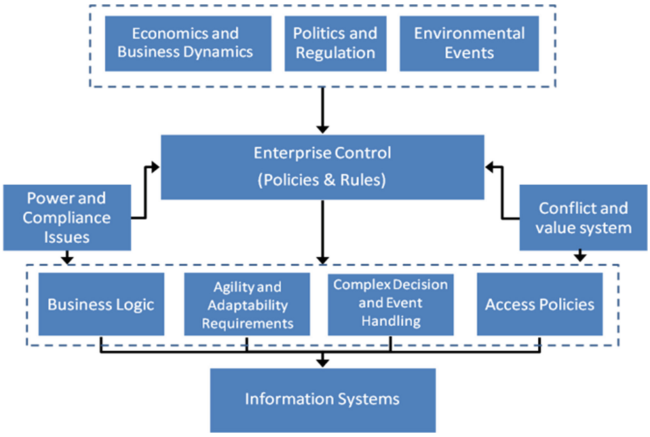


Fig. 1. Enterprise governance factors

3 Related Works

In this section, we discuss how business/IT alignment can strengthen enterprise governance. We are going to focus on rule-based modelling in order to improve alignment and offer a basis for enterprise governance. Later, we will discuss how simulation using system dynamics modelling can further enhance the governance by exploring different potential scenarios of enterprise operations under different inter and intra various circumstances.

There is number of Industrial standards that are strongly relevant to enterprise governance, such as COSO, balanced scorecard, ISO 9001, generally accepted accounting principles (GAAP), Control Objectives for Information and related Technology (COBIT), IIA Professional Practices Framework and others. The Chartered Institute of Management Accountants (CIMA) have proposed an enterprise governance notion [7] that is formed by corporate governance, business governance and enterprise assets. Their governance framework is based on the CIMA Strategic Scorecard. Since there are a number of enterprise governance standards, most of these standards do not communicate the value of business/IT alignment; they disregarded how different levels of governance policies and rules are aligned together. Therefore, we refer back to this paper's aim, and this section discusses the methods that can enhance business/IT alignment and improve traceability and analysis toward mature enterprise governance practices.

3.1 Rule-Based Modelling

Rule-based modelling approaches were introduced in the early 70s [8]; many rule-based approaches were proposed from that time and were based on the natural language

approach [9], decision tables [10, 11], visual notations [12, 13] and software code constrains [14, 15]. Nowadays, the rule engines are a part of what is called the Business Rules Management System (BRMS). The BRMS software is aimed at focusing on the whole rule cycle from defining, modelling, deploying and execution. Some BRMS's provide facilities to monitor and maintain the business rules; the BRMS is composed of (1) rules repository to store business rules and logic, (2) tools, mostly visual, to allow users (technical and business) to define and modify rules and decision logic, (3) rule engine to execute the rules in the runtime environment.

The BRMS standards also vary and many have been proposed and still have different quality perspectives. Some of the well-known standards are RuleML, RIF, SBVR, SRML, PRR and R2ML; a systematic review and analysis of the most common rule standards are presented in [16, 17]. Also some constraints-based software and system development languages can be found in programming languages such as object-oriented programming languages, the OCL and Database constraints. Nevertheless, a number of new rule-based languages were also proposed recently such as the contract representation language CSL/GCSL. This language was developed in the SPEEDS project [18]. The current trend focuses on moving the control from the software level to the business level to increase agility and responsiveness. Business rules tools have been developed for modelling, managing, execution and deployment of the business rules; for example, BRMS such as Drools© JBoss Rules, OpenL Tablets, OpenRules, IBM© ILOG, Pegasystems and Sparkling Logic Smarts.

3.2 Simulation Using System Dynamics

A system dynamics modelling approach provides "essential insight into situations of dynamic complexity", particularly when testing whether real systems are viable [18]. System dynamic modelling is concerned with the representation and modelling of dynamic behavioural aspects of the system components by presenting a simulation of the evolution of behavioural aspects over time, which can be controlled, by either events or assumptions. It aims to develop an in-depth understanding of the targeted system's reflexive behaviour by delivering a reflective system based on the analyst's mental model expansion, using the system's thinking mechanisms including feedback loops among system components. Also, it helps in representing the system's behaviour patterns over time, even in the most complex systems. What makes using system dynamics different from other approaches to studying complex systems is the use of feedback loops and stock and flow mechanism to control variables dynamics. These elements help to describe how even seemingly simple systems display baffling non-linearity. System dynamics modelling is one of the important research and simulation methods for analysing the dynamics in organisations. This is because it is particularly suitable for quantifying qualitative, intangible and 'soft' variables involved in human and social systems [19] to produce insight that helps decision makers in designing business activities. System dynamic modelling is expected to be optimal for prediction business (strategic and operational) scenarios, which is mainly exploratory and quantifiable. This approach has been successfully used in many management, engineering and social disciplines. However, this approach does not imply how the system elements

should be reconfigured in order to produce the desired result, which will rely on other techniques or methods, historical data and analyst experience.

System dynamics modelling can be applied to strategy simulation and to operational simulation and technical simulation [19]. Several scholars have successfully applied system dynamics modelling to enterprises planning [20, 21], strategy design [21, 22] and supply chain risk [23]. Also for evaluating traffic safety policy [24], the impact of policies change on operational models [25] and organisational change to improve the performance in the public sectors [26]. System dynamics was used also in planning for large scale sports events [27], in requirements engineering [28] and many others.

To conclude, to achieve effective enterprise governance, enterprises need to consider a hybrid analysis and design approach. There is a wide spectrum of analysis and design approaches and languages, some are more business-oriented and others are IT-oriented, which have also been developed based on different technologies, or mechanisms. Thus, different qualities and capabilities are associated with these approaches. In our approach, we focus on a selective number of tools to align and classify governance variables based on the source and impact/enforcement level of these policies and rules and how they might influence the enterprise. We see rules' sources classified to internal and external, also we see how some rules can be automated and some are not. However, we have no intention of developing a new language; an approach based on selective of tools to align governance factors, forecasting the impact and reflect that to ISs development is the goal of this paper.

4 Developing a Hybrid Enterprise Governance Approach

Knowledge should be managed (tacit or explicit) and the technical and social practices helped to improve the management of knowledge and are likely to implement or improve internal process and efficiency. Most of a human's knowledge is communicated and represented by natural language. Natural language is a basis for requirement gathering, and all business concepts documented using natural language as 'explicit knowledge', or in form of reflection and experience in the human mind 'tacit knowledge', which can be codified verbally/written using natural language. Thus, research in business and computer studies have focused on natural language as a key part of understanding business and the environment. The business research area has mainly focused on business taxonomy, concepts and definitions. Work in computer science has covered ontological development, natural language processing, fact-based modelling, textual models and the semantics of business vocabulary and business rules. We are aiming to use a simple structure of linguistic requirements in order to improve communications, alignment and planning. Organisations have many business terms and concepts that need to be defined. These concepts consist of vocabulary and connecting the concepts using ontological relations [29]. These factors can be used to describe business rules and policies using quantifiers, logical operations and logical models to provide what are called business facts [30, 31]. The concepts and facts will also help to define services and describe sets of business operations and activities used in the process model. The aim of the ontological (vocabulary, concept and fact) model is to link the higher-level domain constructs with description of the rules that could

influence the lower operational levels and vice versa; for example, external facts, events and rules imposed by other systems, e.g. government, ecology, cultural norms, etc. and the internal rules imposed from inside the system, how they are semantically linked and how they influence each other. What can an enterprise do to mitigate the risk of change?

Organisations need to be keen to make the right decision and in order to do so, they need to consider a wide spectrum of artefacts and the ways these artefacts connect. Business forecasting is a critical practice, but it is unlikely to always be accurate. The methods used in decision-making should be comprehensive and consider the ripple effect of the internal or external environment artefacts. Business rules can be applied to any business division. Different business functions require different types of analysis and simulation in order to support business activities design decisions. Here, we focus on simulation to support strategic decision for operation governance. The approach proposal distinguishes between two important activities: the input and the output of the assessment. The assessment should consider internal and external factors, some of which are influencers, others are influenced by these influencers and those should be linked to the particular drivers of making each assessment. Usually the drivers of the enterprise are the goals, objectives or strategy. Key Performance Indicators (KPIs) are the measurement elements used to assess against the designed business [32], where the assessment should measure the influences and influencers of the drivers and operations or any of the enterprise aspects. Assessment in the research will focus on using System Dynamics (SD) for quantitative assessment for linear and discrete simulation based on the stocks and flows model [19]. Examples of internal influencers which can be under the influence of each other are: resources, infrastructure, habits, management style, assumptions, corporate value and rule compliance. While external influencers/influences could be and are not limited to customers, government, economy, partners, competitors, environment, suppliers, technology and ecology as discussed in Sect. 2. The assessment output will show the strengths and weaknesses of the enterprise, and will inform the policies/rules design. It could also show the impact in terms of rewards/opportunities that the enterprise could experience by going in a certain direction. Alternatively, this could be a risk/threat that it needs to avoid and take into consideration. The assessment could also show up issues related to specific directions or options; thus, the enterprise will end up with a recommendation as to how to make the decision. The assessment could be qualitative or quantitative, based on the nature of the assessed artefact itself. Figure 2 shows the developed meta-model for our proposed solution.

SBVR is a controlled natural language specification for defining business vocabulary and business rules initially invented by the business rule group, and later the Object Management Group (OMG) adopted the specification and improved the SBVR formalisation [31].

SBVR is based on ideas rather than writing formal business specifications then transferring it to software code; we can write executable formal business specifications. This obviously will bridge the gap between business and IT people and offer to business people the opportunity to write, modify and update business applications without the need to change the code or application structure. Therefore the approach's steps will start by defining the business vocabulary using the SBVR, these vocabularies in addition to a set of quantifiers and qualifiers will construct the business policies and

rules. Later, organisational, contextual and environmental facts and events need to be captured in order to classify and assess the impact on the enterprise activities.

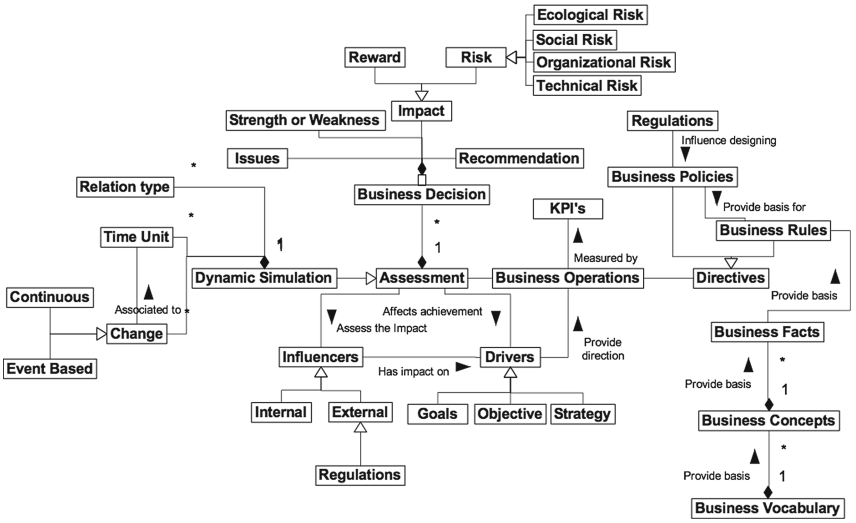


Fig. 2. Enterprise alignment for governance meta-model

The basis for developing mature business policies and business rules is accomplished. The enterprise will develop a set of structural and behavioural rules aimed at providing guidelines, controlling humans' and systems' behaviour, mitigating risks and handling different types of events. Many events and behaviours cannot be easily predicted or understood, therefore, dynamic simulation to assess the impact and understand the enterprise behaviour under specific circumstances can offer an in-depth insight to support decision-making and policies/rules design. This insight will be translated into decision tables to combine a set of events/rules for complex decisions and the level of enforceability in the decision tables is very high. The possible automation in information systems and software development can be done using either Commercial Off-The-Shelf (COTS) rule software packages or by transforming decision tables and SBVR to UML/OCL [14] software components in case of structural rules and for XML-based formats for rule engines execution in case of behavioural rules.

Here, we suggest a new approach to tackle the governance alignment problem from its root. The basis of business language needs to be captured and defined clearly, so that sharing common understanding among all stakeholders and providing foundation for governance design will be the first step of the approach. One of the reasons for inventing the SBVR specification is to allow business people to define their business vocabulary and business concepts. The defined business elements will later act as a foundation for building fact models. Considering a number of variants enterprise facts, these facts are mostly constructed or influenced by the events which need to be identified and modelled clearly, to build a strong justification for building business

policies and rules. Thanks to SBVR, we can still define and model these aspects by using it. After all, the suggested business setting under different circumstances needs to be simulated using the system dynamics model. System dynamic will provide a forecasting model of how influencers and events can affect the enterprise. Simulation of different scenarios will offer the necessary insight for business managers to identify the suitable decision to be made in each particular case. Thus, decision tables will capture and identify the complex decisions that need to be taken in each scenario. Finally, the rules and decision will be mapped to information system models and inherent constraints that are need to be embedded within the software architecture using combination of UML and OCL constraints, which can be serialised as XML Schema Definition (XSD). Figure 3 represents the suggested modelling approach for enterprise governance.

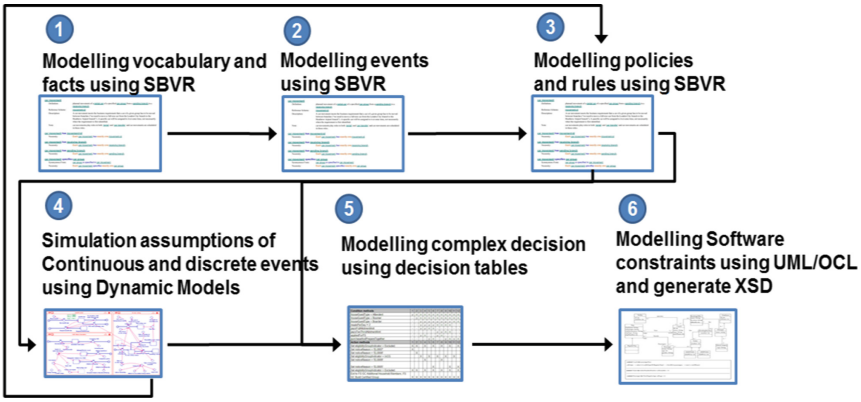


Fig. 3. Hybrid enterprise governance processes

5 A Case Study

A logistics company operating in Iceland is looking to improve their governance and reduce business risks (financial, workplace, CRM) of shipping services. The change in the ecological station will influence directly or indirectly the company. Natural disasters such as earthquakes, volcanoes and hurricanes also have similar possible disaster effects on this company. Safety and hazard mitigation is one of the top requirements of most of the enterprises that operate in logistic sector. After the events of 9/11 in 2001, many countries have had to focus much more attention on their national and homeland security. The Iceland volcano in 2010 nearly blocked all air flights around Europe, which caused losses of more than one billion pound for the airline industry alone. Government regulations may bind the company activities and could add additional operational costs.

Automating business activities will obviously enrich the control and analysis. Simply put, the automated process allows managers to monitor the rules' and policies' compliance as well as giving early notification of any violation or event that may have

occurred. In order to implement successful automation of the rule system, the analysis and design processes are needed in order to improve the alignment among enterprise levels and divisions. The tracking and alignment will increase the governance and control over the business activities. However, in order to plan the activities in a proper manner, a top-down alignment approach to the proposed framework has to be applied. Some of the rules cannot be automated, which are mainly related to human manual activities. Therefore, manual checks need to be strictly governed by supervisors. We found that in enterprise, the most influential factors are either contextual or social, where the enterprises are forced to react and evolve with social, market and economical changes. The consideration of a wider risk scope led to enterprise maturity increasing. One weaknesses of the wider consideration of external influencers is represented in the cost of time of the analysis and the enterprises need to decide the value of investing in increasing enterprise maturity and risk assessment. In the following, the approach is elaborated in terms of different project activities:

I. Modelling vocabulary and facts using SBVR

In this section, we define the main concepts and factual knowledge within Iceland as a market. Table 1 shows a sample of vocabulary and facts of the case study:

Table 1. Business vocabulary and facts

Vocabulary	*V1: Market: The targeted economical and commercial area of the company services*
Facts	F1: Market *has* segmentations. F2:Market must *have* analysis report. F3:Iceland *is a* Market. F4:Iceland *has a* volcano. F5:Iceland's volcano *has* erupted *in* 2010

II. Modelling events using SBVR

Here, we define the related market events, Table 2 presents the number of the business and environmental events that may occur:

Table 2. Business events

Environmental Events	*EE1: It is possible that Iceland volcano can erupt* anytime. *EE2: It is possible that the heavy snow falling to block the roads.* *EE3: It is possible that the fog reduces the sight of drivers on the road.*
Business Events	BE1: Customers *place* PO *to buy* products or services. BE2: Customer *pay* down payment. BE3:It is possible that each employee driver *to oversight* at least one *of the* health and safety policies

III. Modelling policies and rules using SBVR

Policies and rules designed in order to respond, avoid or mitigate particular environmental and business events and/or activities are presented in this section. Table 3 present the number of the identified country policies (where the enterprise operates), industry polices (logistic industry) and particular business rules:

Table 3. Business policies and rules

Country Policy	*CP1: It is obligatory that each logistic company pays taxes equal to 18% of their annual revenue.*
Industry Policy	IP1: Logistic companies must *ensure the* quality of services *according to* annex H "the service provider responsibility" *of the* logistic industrial practices report .
Rules	R1: Company must add new policy *to activate* possible delay in delivery time notification if there is *an* environmental event. R2: It is obligatory that *the* company *implements the* health and safety policy. R3: Customer must send the PO before *processing the* shipping . R4: *50%* down payment must be received before *processing the* shipment. R5: it is obligatory that the supervisor *check the* drivers compliance *to* health and safety policy every time the driver *go on their* delivery pick up.

After defining the business vocabulary, events, rules and policies, it is very important to ensure the aspects are aligned and correspond to each other. The alignment will also help in the early identification of any analysis gaps. Alignment between the basic business concepts brought the required insight, as shown in the following Fig. 4:

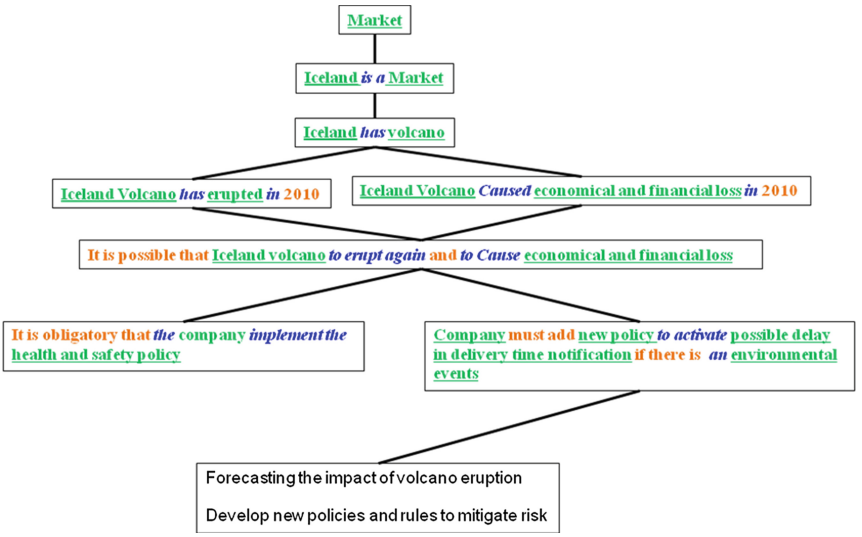


Fig. 4. Aligning context for enterprise governance

IV. Simulation using the system dynamics modelling

When the knowledge is on the edge of human awareness, deciding the required simulation to push the insight boundaries become important. Table 4 present the simulation required to be conducted:

Table 4. Simulation required

Simulation needed for decisions	*SD1: Forecasting the impact of volcano eruption*
	SD2: Forecast the impact of country and industrial policies on business

The dynamic model in Fig. 5 was built in order to measure the impact of any of the environmental events on the enterprise revenue; simulation assumptions were applied to generate the insights. We ran the simulation model several times to measure the impact belonging to the different conditions, therefore to decide the desired settings the enterprise would like to maintain. The dynamic model has informed the enterprise about the potential negative impacts of the external events on business activities.

We understand that the total impact of risk can influence the enterprise profitability, this risk can be identified as a summation of operating cost and the potential loss divided by confidence level about this loss and should subtract the confidence about potential profit from that total, and the total will be added to governance cost as a new element of the operating cost. The following equation was developed and used for calculating enterprise profitability risk:

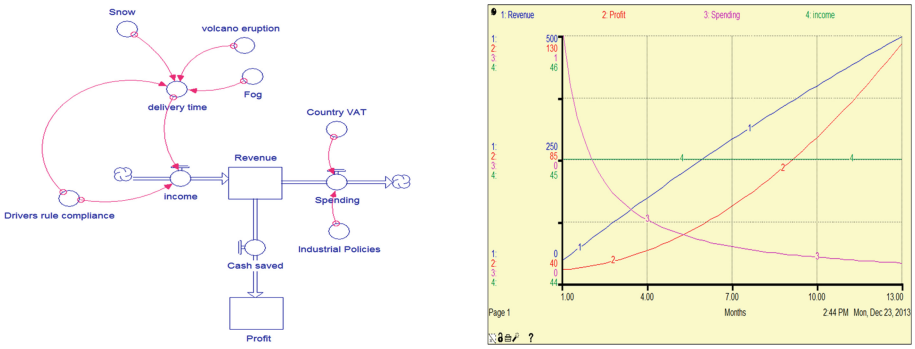


Fig. 5. Dynamic modelling and simulation

$$Enterprise\ Profitability\ Risk = \sum\nolimits_T [Operation\ Cost + \left(\frac{Potenital\ Loss}{Confidance\ Level} - \frac{Potenital\ Profit}{Confidance\ Level}\right) + Governance\ Cost] \quad (1)$$

After the simulation, the potential impact becomes clearer since the simulated assumptions have tested several scenarios of environmental events, some of the simulation scenarios have tested the impact of one event e.g. the driver did not comply with safety rules. Other simulation scenarios have tested the composite events impact such as the impact of the weather being snowy and foggy at the same time. The impact value of these events was qualitatively acquired from the company's staff based on their practical experience. In-depth insights were obtained based on the previous simulation, and this can act as a foundation for future enhancement of the simulation model and assumptions. Here we present a qualitative classification of the events impact on business and operation as shown in Table 5 below:

Table 5. Internal and external events and their impacts

Event	Potential impact	Impact level
Volcano eruption	Full destruction for the logistic and shipping operation	High
Fog	Delay in delivery, potential road accident	Medium
Snow storm	Delay in delivery, potential road accident or problems in vehicle	Medium
Driver violate rules	Road accidents, potentially life threatening	Medium to high
Clerk violate rules	Procedural problem may cause economical loss	Low

V. Evaluate complex decisions using decision tables

After understanding the potential impact and its level, we should associate the impact to each particular business case, operation or event. The decision tables combine several rules and events, when the decisions needed to be made based on complex situations, decision table can reduce the complexity of the case and provide accurate judgments based on the true conditions in the time unit as presented in Table 6.

VI. Modelling software constraints using UML and OCL

The modelled rules should be instantiated in the developed information systems. The business aspects that can be governed by the IS will be implemented using a combination of UML and OCL. The following structural model (Fig. 6) corresponds to the case study requirements and generates the XSD serialisation.

In this initial study, historical data was used to support simulation assumptions. For evaluation purposes, a qualitative evaluation is the most suitable method at this stage. Stakeholders were asked to express their opinions regarding the strengths and weaknesses of the suggested approach, focusing on the issues of ease of use, comprehensiveness and the depth of insight provided to better design enterprise activities and information systems. The stakeholders were aware of the inverse relationship between maturity and flexibility and between quality and time/speed. Suggestions and recommendations for improvements from the case study are discussed.

Stakeholders confirmed that this approach is easy to use in its structured natural language part (SBVR), while they found it quite challenging to use system dynamics

Table 6. Decision table

PO received?	50 % Down payment received?	Shipping confirmed?	Any environmental event?	Any impact of the event?	Decision
Yes	No	–	–		Do not process order
Yes	Yes	No	–	–	– Confirm the shipping address with customer – Then click internal proceeds to confirm shipping
Yes	Yes	Yes	No	–	Process and shipping
Yes	Yes	Yes	Yes	No	Process and shipping
Yes	Yes	Yes	Yes	Yes	– Activate delay in shipping policy – Inform customer about potential delay – Calculate potential impact on business

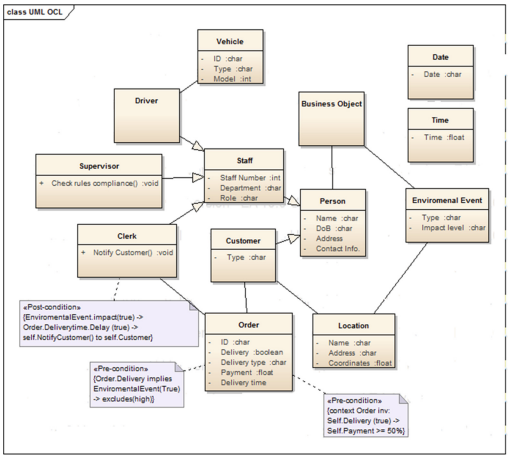


Fig. 6. Implementation model using UML and OCL and XSD

(SD) models. We believe that considering an easy to use simulation tool is still a potential option for future work. Different types of facts, rules and events were also captured in the case study, an insight was obtained base on their alignment through the approach we proposed. It might be useful to build a fully automated tool for software, and systems integration with the proposed approach; for example, the data in the tables needs to be normalised and in executable form, so the simulation dynamic model can import the data easily, also to export the simulation result to enterprise information systems. One of the common mistakes in implementing governance is that many enterprises rely completely on technology and ISs to understand the details of how enterprises operate and how they should operate, rather than guiding the technology to enable the strategy. This causes less agility, longer development life cycles and the failure of IS initiatives. It is the responsibility of analysts to bring the enterprise and its context's complexity and dynamics understanding to ISs implementation continuously. A mixture of agile ISs platform and analysts' effort for thinking, analysis and simulation will remain in high demand.

6 Conclusions and Future Work

The development of rules has been used for both for the purpose of business and IT analysis. To this end, the rule-based modelling has been presented to offer a structured method to simplify the complexity of enterprise work and provide efficient analysis and design mechanism. Still, there is little literature discussing the rules for holistic complex enterprise modelling, questions like how the higher level description of the rules could influence the lower levels and vice versa not yet answered. For example, the internal rules forced from inside the system, and external rules forced from the other systems (i.e. government, ecology, cultural norms, etc.), what is the semantic relation among them and what the organisation can do to mitigate the risk of the change.

In this paper, we presented a hybrid rule-based approach that can handle and align different types of rules for enterprise governance analysis and design. It is easy to use for both business and IT people and offers better alignment and insight. Compared to most of the previous approaches, the approach proposed in this paper provides a unique process of classifying, defining and implementing enterprise rules' systems. First, we focused on capturing the essence of knowledge that is influencing the enterprise activities by defining external and internal enterprise related vocabulary and facts. Also, classifying rules based on their source and nature in order to identify the potential impact (high, medium and low) and the suitable rule/action that the enterprise can take in reacting to each rule or event. The approach is also considered unique as it focuses on expanding the mental model of understanding patterns of behaviour using dynamic simulation. This will help in identifying the influencers' impacts to offer more insight that can help towards producing better internal policies, rules and decisions. In summary, the approach will offer the following capabilities:

1. Defining domain and context knowledge.
2. Electing and defining rules.

3. Classifying and aligning rules.
4. Documenting rules.
5. Simulating the rules' impact.
6. Implementing the rules in software and ISs.

Future work can go in two directions: (1) adding social/stakeholders analysis stage to this approach, where rules cannot be automated and stakeholders have greater influence on compliance and performance, (2) working on separating the rules' concerns, what should be implemented in the structural model (UML/OCL) and what should be defined in the rule management systems using some COTS rule engines to maintain business agility.

References

1. Jarke, M., Loucopoulos, P., Lyytinen, K., Mylopoulos, J., Robinson, W.: The brave new world of design requirements. Inf. Syst. **36**(7), 992–1008 (2011)
2. Baxter, G., Sommerville, I.: Socio-technical systems: from design methods to systems engineering. Interact. Comput. **23**(1), 4–17 (2011)
3. Morabito, J., Sack, I., Bhate, A.: Organization Modeling: Innovative Architectures for the 21st Century. Prentice Hall, Upper Saddle River (1999)
4. Jurisica, J., Mylopoulos, J., Yu, E.: Ontologies for knowledge management: an information systems perspective. Knowl. Inf. Syst. **6**(4), 380–401 (2004). doi:10.1007/s10115-003-0135-4. Springer, London
5. ISACA: The Risk IT Framework (2009). www.isaca.org/riskit
6. Kildow, B.A.: A supply chain management guide to business continuity. AMACOM Division of American Management Association (2011)
7. The Chartered Institute of Management Accountants (CIMA), "The CIMA Strategic Scorecard" (2005). www.cimaglobal.com/cps/rde/xbcr/SID-0AAAC544-0F6A361A/live/tech_dispap_CIMA_strategic_scorecard_0305.pdf
8. Codd, E.F.: Derivability, redundancy, and consistency of relations stored in large data banks. Research Report RJ599, IBM Corporation, San Jose, CA, USA, 19 August 1969
9. Linehan, M.: Ontologies and rules in business models. In: VORTE 2007, The 3rd International Workshop on Vocabularies, Ontologies and Rules for the Enterprise, Annapolis, MD (2007)
10. Vanthienen, J., Wets, G.: Integration of the decision table formalism with a relational database environment. Inf. Syst. **20**(7), 595–616 (1995)
11. Vanthienen, J., Mues, C., Wets, G., Delaere, K.: A tool-supported approach to inter-tabular verification. Expert Syst. Appl. **15**(3–4), 277–285 (1998). doi:10.1016/S0957-4174(98)00047-5
12. Wan-Kadir, W.M.N., Loucopoulos, P.: Relating evolving business rules to software design. J. Syst. Architect. **50**(7), 367–382 (2004)
13. Kardasis, P., Loucopoulos, P.: Expressing and organising business rules. Inf. Softw. Technol. **46**(11), 701–718 (2004)
14. Cabot, J., Pau, R., Raventós, R.: From UML/OCL to SBVR specifications: a challenging transformation. Inf. Syst. **35**(4), 417–440 (2010)
15. Rosca, D., Greenspan, S., Wild, C.: Enterprise modeling and decision-support for automating the business rules lifecycle. Autom. Softw. Eng. **9**(4), 361–404 (2002)

16. Goedertier, S., Haesen, R., Vanthienen, J.: EM-BrA2CE v0.1: a vocabulary and execution model for declarative business process modeling, SSRN 1086027 (2007)
17. Muehlen, Z.M., Indulska, M.: Modeling languages for business processes and business rules: a representational analysis. Inf. Syst. **35**(4), 379–390 (2010)
18. Contract Specification Language, D.2.5.4. Israel Aerospace Industry (2008). http://www.speeds.eu.com/downloads/D_2_5_4_RE_Contract_Specification_Language.pdf
19. Sterman, J.G.: Business Dynamics: Systems Thinking and Modeling for a Complex World. Irwin/McGraw-Hill, Boston (2000)
20. Bianchi, C.: Improving performance and fostering accountability in the public sector through system dynamics modelling: from an 'external' to an 'internal' perspective. Syst. Res. Behav. Sci. **27**(4), 361–384 (2010)
21. Barnabè, F.: A "system dynamics-based Balanced Scorecard" to support strategic decision making: Insights from a case study. Int. J. Prod. Perform. Manag. **60**(5), 446–473 (2011)
22. Bianchi, C., Winch, G., Cosenz, F.: Sustainable strategies for small companies competing against multinational giants. In: Acere-Diana Conference, Perth (2012)
23. Ghadge, A., Dani, S., Chester, M., Kalawsky, R.: A systems approach for modelling supply chain risks. Supply Chain Manag. Int. J. **18**(5), 523–538 (2013)
24. Goh, Y.M., Love, P.E.: Methodological application of system dynamics for evaluating traffic safety policy. Saf. Sci. **50**(7), 1594–1605 (2012)
25. Saleh, M., Oliva, R., Kampmann, C.E., Davidsen, P.I.: A comprehensive analytical approach for policy analysis of system dynamics models. Eur. J. Oper. Res. **203**(3), 673–683 (2010)
26. Bianchi, C., Bivona, E., Cognata, A., Ferrara, P., Landi, T., Ricci, P.: Applying system dynamics to foster organizational change, accountability and performance in the public sector: a case-based Italian perspective. Syst. Res. Behav. Sci. **27**(4), 395–442 (2010)
27. Beis, D.A., Loucopoulos, P., Zografos, K.G.: PLATO helps Athens win gold: olympic games knowledge modelling for organizational change and resource management. Interfaces **36**(1), 26–42 (2006)
28. Loucopoulos, P., Prekas, N.: A framework for requirements engineering using system dynamics. In: 21st International Conference of the System Dynamics Society, New York City (2003)
29. Karpovic, J., Nemuraite, L.: Transforming SBVR business semantics into Web ontology language OWL2: main concepts. In: Information Technologies, 27–29 April 2011, pp. 231–238 (2011)
30. Malik, N.: Enterprise Business Motivation Model (EBMM), V. 4 (2011). http://motivationmodel.com/wp/downloads/
31. OMG: Semantic of business vocabulary and business rules (SBVR) 1.0. (2008). Retrieved from Object Management Group http://www.omg.org/spec/SBVR/1.0/
32. Fayoumi, A., Yang, L.: SBVR: knowledge definition, vocabulary management and rules integration. Int. J. E-Bus. Dev. (IJED) (2012). ISSN-P: 2225-7411, ISSN-E: 2226-7336

The Prefix Machine – A Formal Foundation for the BORM OR Diagrams Validation and Simulation

Martin Podloucký and Robert Pergl(✉)

Department of Software Engineering, Faculty of Information Technology,
Czech Technical University, Prague, Czech Republic
{martin.podloucky,robert.pergl}@fit.cvut.cz

Abstract. Business Object Relation Modelling (BORM) is a method for systems analysis and design that utilises an object oriented paradigm in combination with business process modelling. BORM's Object Relation Diagram (ORD) is successfully used in practice for object behaviour analysis (OBA). OBA has found its firm place for visualisation and simulation of processes, however several ontological flaws were identified and there seems to be missing a strong formal foundation that would enable correct reasoning about the models. In this paper, we propose a sound formal foundation for BORM'S ORD. Based on this formal foundation (which we call "the prefix machine"), we get not only to a precise behaviour specification, but it also offers some interesting means of process analysis.

Keywords: Prefix machine · BORM · OR diagrams · Process simulation · Process analysis · Formal foundations

1 Introduction

1.1 Motivation

Business Object Relation Modelling (BORM) is a complex method for systems analysis and design that utilises an object oriented paradigm in combination with business process modelling. It is not probably necessary to introduce this method at EOMAS, so we will save precious space and dive into the problem directly. The diagram notation is not very complex and will be probably clear from the figures. Readers interested in more detail will find everything necessary in the references.

The authors of this paper have been using BORM successfully in practice for several years both in management consulting practice and research projects. Unfortunately, it seems there is no foundational paper that would explain the simulation semantics and rules in detail, which limits the use of BORM for mere diagramming and – inaccurate and ontologically not correct – simulations that are provided by the Craft.CASE tool [7]. We think that BORM has bigger potential for rigorous process analysis, as we intend to show in this paper.

J. Barjis and R. Pergl (Eds.): EOMAS 2014, LNBIP 191, pp. 113–131, 2014.
DOI: 10.1007/978-3-662-44860-1_7

1.2 Goals

In this paper we introduce a new formal model based on basic mathematical formalisms (sets, mappings, relations), graph theory and boolean algebra, that models the behaviour of BORM Object Relation Diagram (ORD). The model was designed with the following subgoals in mind:

- Simple, clear and sound formal model that would allow for formal foundations for conclusive BORM ORD behaviour interpretation.
- Having a formal description that serves as a specification (and probably also as a construction) for developing software for BORM ORD validations and simulations.
- The model should describe ideally all behaviour of BORM ORD that is commonly agreed upon.
- The model should not allow ontologically extravagant interpretations (i.e. interpretations that are not aligned with our perception of the process modelling domain).

1.3 Structure of the Paper

In Sect. 2 we briefly introduce BORM's Object Relation Diagram, being the focus of our study. In Sect. 3 we sum up the conclusions made in our previous paper [10] together with the current state of our work. In Sect. 4 we present our formal foundations of ORD. We begin with a simplification of the ORD model and we gradually define and describe the *prefix machine* and its execution model. In Sect. 5 we show an example to illustrate usage of the prefix machine. We also briefly describe process simulation and analysis based upon the prefix machine semantics.

The rest of the paper follows a common structure: Discussion, Related work, Conclusions and future work.

2 Object Relation Diagrams

Given that we set out to give a formal description of OR diagrams, we start by a very brief description of the basic concepts of this modelling notation together with minor changes to the meta-model we performed. A more thorough description of OR diagram notation can be found in our previous paper [10] and originally in the work introducing the object behavioural analysis [6].

Since processes are the major theme of our work, it is appropriate to explain our use of the term, in particular given the amount of definitions available in the literature. For the purposes of this work, we stick to the simple, practically-oriented definition provided by ISO 9000:2000:

Definition 1. Process is a set of interrelated or interacting activities that transforms inputs into outputs.

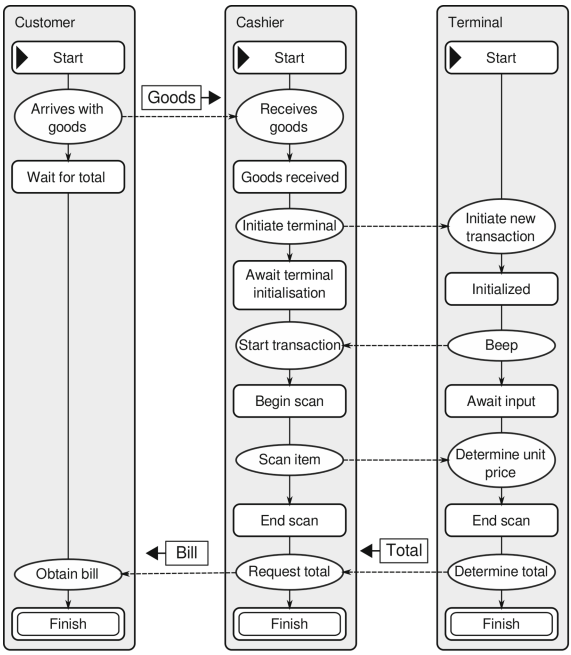


Fig. 1. A sample Object Relation Diagram. Inspired by [6].

To describe such a process, the BORM methodology uses the OR diagram. Figure 1 shows an ORD example taken from [6]. It describes a process of interaction among a customer, a cashier and a cash desk terminal.

The customer, cashier and terminal are so called *participants* of the process. Participants are basically state machines consisting of *states* – represented as rectangles – and transitions between them, represented as arrows. On each transition, there is an *activity*, represented as an ellipse. An activity represents an output of the process, which may simply be a task that needs to be done by the participant in order to advance to the next state. Activities from different participants can be connected by so called *communications*. These allow participants to communicate with each other. Communications are channels through which *data flows* can be sent.

For our purposes, we make one syntax change to the original BORM notation: We do not use extra symbols for start states and end states, rather we have switched to a notation more consistent with the definition of finite state machine: a start and a finite state are ordinary states distinguished only by an attribute. Graphically, we use a triangle symbol for a start state and a double line for an end state, as may be seen in Fig. 1 and the following ones.

3 Current State

In our previous paper [10], we started to explore the major issue of the BORM OR diagrams which is the lack of solid formal foundations. This issue becomes most apparent when one would like to actually execute the process defined by an OR diagram for the purpose of simulation or even orchestration. The simulation or execution of processes defined by ORD is therefore the main challenge of our work. As we have already mentioned in our previous work [10], there is no canonical definition of how the ORD process should by executed yet. Hence, we stand at the point where we need to investigate the semantics of the OR diagram and develop our own solution to this problem.

3.1 Simultaneity and Dependency Principles

When trying to identify the most fundamental principles underlying the execution of the OR diagram [10], we formulated the **dependency principle** and the **simultaneity principle**. As these principles have been already thoroughly discussed [10], we mention them here again only briefly.

Principle 1. The **simultaneity principle** states that no participant can in fact split itself into multiple instances and actually do several tasks in parallel.

This principle states that even though any participant may *be* in several states at once, no participant can actually *perform* several tasks at once. The parallel branches in ORD have, therefore, ontologically this meaning: The activities belonging to different branches do not depend on each other. From that follows that such activities can be done *regardless of order*, which allows one to perform them *virtually* in parallel. Therefore, if a participant is required to do activities in parallel, the actual meaning is that he can choose to do them in any order desired, or switch between doing them as wanted.

The statement that activities belonging to different branches do not *depend* on each other brings us to the *dependency principle*.

Principle 2. The **dependency principle** states that a task A may require other tasks to be completed before A itself can be completed.

The terms “interrelated” and “interacting” in Definition 1 imply the fact that often several tasks have to be completed prior to completing another task. The rules that determine on which tasks the task A depends may be quite complex. Let us have a set of tasks $\{X, Y, Z\}$. For example, the task A may require a completion of *exactly two* tasks from this set. Thus, we need a sufficiently expressive system for specifying such dependency conditions. We have already introduced such a system [10] which utilises boolean algebra and is called **input and output conditions**.

3.2 Input/output Conditions

To be able to express the dependency principle in an ORD, we propose to attach an input condition to each state:

Definition 2. Input condition of a state is a boolean expression whose variables are the transitions ending in that state. It specifies that the execution of the process cannot advance further from the given state until its input condition is met, i.e. until the corresponding boolean expression is evaluated as being true.

Similarly, we propose that each state also should have an output condition.

Definition 3. Output condition is a boolean expression whose variables are the outgoing transitions from the given state. It specifies admitted combinations of branches into which the process execution may split itself from this state.

Figure 2 shows an example of input and output conditions allowing precisely two distinct paths through the participant's state graph. State A has an output condition which says (using the classical boolean XOR operator) that exactly one of two possible transitions may be chosen to continue forward. The state D has, in turn, an input condition saying that exactly one branch is allowed to complete.

Having arrived at this point we have introduced all the necessary concepts from our previous work [10]. The main challenge now is to put these concepts on a solid basis and develop a formal framework for the execution of processes defined by OR diagrams. We need to precisely define the behavioural aspect of the OR diagram based on input/output conditions and the simultaneity principle. As explained in our paper [10], neither Mealy's machines nor Petri nets provide a suitable framework to begin with. Therefore, we need to develop our own formal machine, which would be capable of expressing the desired semantics of OR models regarding their simulation and execution. We will call this machine **the prefix machine**.

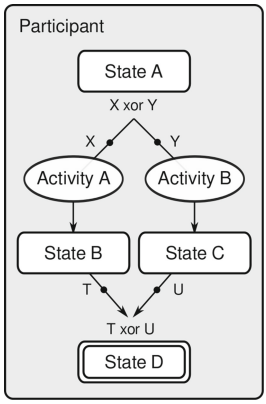


Fig. 2. Sample input and output conditions.

3.3 Simplified OR Model

As the current OR diagram does not have any formal model, we have to start from scratch and develop our own, maybe redefining some portions of the ORD along the way. Here starts our journey towards the prefix machine. In this section, we start a rather informal description of the prefix machine making ourselves ready for the more formal next section.

The prefix machine aims at formalizing a merge of both Mealy's machine and Petri net and overcomes the issues described above and in the paper [10]. The machine is based on directed graphs of states (similarly to the Mealy's machine), and its execution uses tokens (a trait typical for Petri nets). To describe the necessary concepts using the graph theory, we use the standard terminology [1].

The prefix machine is a somewhat simplified model aiming at capturing the essence of the "future" OR diagram while omitting unnecessary details and complexities from the current OR diagram. The prefix machine should then evolve into a formal basis upon which the future revised OR diagram should be based.

The prefix machine is basically structured as a directed graph. The correspondence between the current ORD and a directed graph is rather straightforward.

1. States correspond to vertices.
2. Transitions along with their respective activities correspond to edges.

We call the resulting digraph edges *transitions* as well, reducing, in essence, activities to transitions between states. At the same time, transitions represent dependencies between states, and thus express the dependency principle. For the sake of simplicity, we diverge from the current OR diagram structure a little by changing three important concepts.

1. We place communications between states instead of activities (recall that activities are no longer represented as vertices).
2. As it is convenient to have only one kind of edges in a directed graph, we replace communications in the model by transitions. We assume that in order for a communication to happen, both its source and target states have to be visited by their respective participants. This actually makes these two states dependent on each other, so we can *represent each communication with two opposite transitions.*
3. We do not allow cycles in the model, i.e. we discuss loop-less processes only.

None of the first two changes diminish the expressive power of the formal model in any way. The third change, on the other hand, certainly diminishes the expressive power and we are prepared to address this issue in our further research.

Notice, however, that by replacing communications with transitions, we already introduced cycles of length 2. A general directed graph can be transformed into an acyclic graph (to model a process) simply by contracting all strongly connected components into single vertices. Thus, after the aforementioned cycles removal, the source and target state of each communication collapses into a single *communication state.* Incidentally, such a result corresponds exactly to the intended meaning:

The act of communication between two participants in an OR diagram may happen just when both of them are visiting their respective communicating states.

Notice also that neither participants nor data flows are included in the simplified OR model. At this level of abstraction it is immaterial for us which task is executed by whom and what exactly is being sent along the communications lines.

4 The Prefix Machine

Now we are ready for the precise formal definition of the prefix machine. From now on, we use Greek letters such as φ, ψ to denote boolean expressions. We work rather extensively with them, with a special kind – **positive boolean expression**, in particular.

Definition 4. A boolean expression of at least one variable which evaluates to false when all of its variables are false, is called **positive**.

Let V be an arbitrary finite set. We denote by E_V the set of all positive boolean expressions whose variables are elements of V. Notice, that E_V does not contain formulas with no variables – the truth constant $\top$ and the false constant $\bot$. We deal with them separately. The rationale behind positive boolean expressions and constants will become clear later on, when we discuss the prefix machine execution. Now we have the tools for the long awaited formal definition of the prefix machine ready.

Definition 5. Prefix Machine G is a 5-tuple (S, T, K, e^-, f), where

- S is a set of digraph vertices representing **states**,
- $T = \{(x, y) \mid x, y \in S\}$ is a set of oriented edges representing **transitions**,
- $K \subseteq S$ is a set of **communication states**,
- $e^- : S \to E = E_T \cup \{\top\}$ is the mapping of states to their input conditions,
- $f \in S$ is the **terminal state**.

According to the above explanation, E_T is the set of all positive boolean expressions with variables drawn from T. The set of all possible input conditions E also includes the $\top$ constant. This is merely for convenience and it is justified further below.

To explain the notion of the terminal state, let us first consider the following notation. For a given state x we denote the number of its ingoing transitions by $\deg^-(x)$ and the number of its outgoing transitions by $\deg^+(x)$. The terminal state f is the one and only state for which $\deg^+(f) = 0$. On the other hand, there may be several states for which $\deg^-(x) = 0$.

Definition 6. A state $x \in S$ for which $\deg^-_G(x) = 0$ is called an **initial state**. Let us denote I_G the set of all initial states of G.

Definition 7. The condition e^-_f assigned to the terminal state is called the **terminal condition**.

Continuing the prefix machine definition, we add some further restrictions to it. For a state $x \in S$, let us denote its **outgoing transitions** as $t_x^1, \ldots, t_x^n$ and its **ingoing transitions** as $u_x^1, \ldots, u_x^m$.

The following statements hold:

$$\forall x \in S \, \exists \varphi(u_x^1, \ldots, u_x^n) \in E : \; e^-(x) = \varphi \tag{1}$$

$$\forall x \in K : \; \deg^+(x) = \deg^-(x) = 2 \tag{2}$$

$$\forall x \in K : \; e^-(x) = u_x^1 \wedge u_x^2 \tag{3}$$

Formula (1) says that there is an input condition for each state of G and all ingoing transitions of x are variables of that condition. This is where the constant $\top$ comes in handy since it is the only valid choice for the input conditions of the initial states which do not have any ingoing transitions. Formula (2) ensures that there are exactly two ingoing and two outgoing transitions for each communication state. This fact reflects the nature of a communication state: a communication state was created by merging two states from two different participants in the OR model. Thus, there must be exactly two paths going through each communication state, and each path has been taken by a different participant. Formula (3) ensures that both of those participants must be present in the communication state in order to advance further.

The definition of the prefix machine is now complete. Nevertheless, we need to define another important concept. Similarly to the mapping of the input conditions e^-, we define the mapping $e_G^+ : S \to E$ of **output conditions**.

However, this mapping is not a part of the prefix machine definition, since it is completely derived from the machine's structure and properties. The mapping is derived as follows:

- $e_G^+(f) = \top$
- $\forall x \in K : \; e_G^+(x) = t_x^1 \wedge t_x^2$
- $\forall x \in S \setminus \{K \cup \{f\}\} : \; e_G^+(x) = t_x^1 \vee \cdots \vee t_x^n$,

There are also several important facts implied by the definition of the prefix machine that are worth mentioning.

$$\forall x \in I_G : \; e_G^-(x) = \top \tag{4}$$

$$\forall x \in S : \; \deg_G^-(x) = 1 \Rightarrow e^-(x) = \mathrm{id}_u \tag{5}$$

Formula (4) says that when a state x is initial, there is no other possible choice for its input condition than the expression $\top$. Formula (5) says that when the state x has only one ingoing transition u, again, there is only one possible choice for its input condition – and that is a positive expression consisting of one and only one variable u. There is precisely one such expression for each variable u and it is the identity denoted by id_u.

4.1 Prefix Machine Examples

We can see two examples of prefix machines in Fig. 3. States are represented as rectangles and transitions as arrows. Let us first look at the machine G_1. We can see a sample input condition $u_1 \oplus u_2$ attached to the state z. Here we use the symbol $\oplus$ to denote logical non-equivalence (the same operator as XOR) and u_1, u_2 to denote ingoing transitions to the state z. Notice also the terminal state depicted as a crossed rectangle. Here we have omitted the obvious constant and identity input conditions which are attached to the initial states and states with only one ingoing transition. This prefix machine represents a participant, who needs to take one simple decision and the input condition ensures that they cannot choose to go both ways simultaneously.

The machine G_2 is a bit more complex. It actually represents two participants. This is clear from the fact that it contains two initial states. It contains one communication state depicted by a dashed rectangle which represents the situation where these two participants need to communicate with each other. Notice also the terminal condition attached to the final state, which says that both of these participants need to successfully finish. Examples of how such machines relate to OR diagrams and how exactly they are executed are presented in Sect. 5.

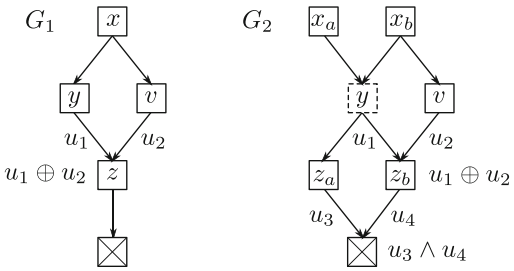


Fig. 3. Examples of prefix machines.

4.2 Prefix Machine Execution

Let us now proceed to the formal definition of a process simulation, as defined by the prefix machine. Let $G = (S, T, K, e^-, f)$ be a prefix machine. We put $\bar{S} = S \setminus \{f\}$.

Definition 8. State visitor is a mapping $g : \bar{S} \to \{\varnothing, \bullet, \circ\}$ with the following properties:

$$\forall x \in I_G : \; g(x) = \circ \tag{6}$$

$$\forall x \in \bar{S} : \; g(x) \neq \varnothing \Rightarrow \forall y((y, x) \in T \Rightarrow g(y) \neq \varnothing) \tag{7}$$

The purpose of the mapping g is to assign a **token** to each state of the machine with the exception of the terminal state. We distinguish between two types of tokens – *positive* (○) and *negative* (●).

Definition 9. We say that a state $x \in \bar{S}$ is **positively visited** or **having a positive token** if $g(x) = ○$.

Definition 10. We say that a state $x \in \bar{S}$ is **negatively visited** or **having a negative token** if $g(x) = ●$.

Definition 11. We say that a state $x \in \bar{S}$ is **unvisited** or **having no token** if $g(x) = \varnothing$.

We use the tokens as boolean variables with ○ representing the value *true* and ● representing *false*. By assigning a positive token to the state x, we express the situation where x's participant has already passed through x. A negative token means that the x's participant will never pass through x. No token means that in the current execution step, it has not been yet decided whether the state will ever be visited or not. Now we can define the notion of *configuration*.

Definition 12. The structure $C = (G, g)$ is called a **configuration** of G.

We can now interpret the above properties in the following way. Property (6) says that all initial states are always visited in all configurations of G. Property (7) ensures that when the state x is visited, all states on the path from the initial state to x are visited as well. The name *prefix* machine comes from this very property.

Sample prefix machine configurations can be seen on the left side of Fig. 4. White, grey and black circles represent unvisited, positively visited and negatively visited states respectively. Of course, there are many other configurations possible in this case. However, not all of them are interesting and useful here. Let us, therefore, define two special kinds of useful configurations.

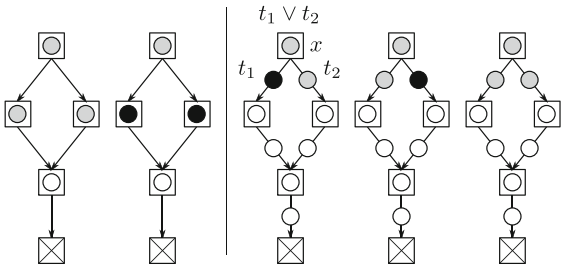


Fig. 4. Sample configurations of a simple prefix machine.

Definition 13. Configuration $C = (G, g)$ is called an **initial configuration** if $g(I_G) = \{\circledcirc\} \wedge g(\bar{S} \setminus I_G) = \{\varnothing\}$.

Definition 14. Configuration $C = (G, g)$ is called a **terminal configuration** if $g(\bar{S}) = \{\bullet, \circledcirc\}$.

From the above definitions we can see that each prefix machine G has precisely one initial configuration and possibly several terminal configurations. Now we would like to define relationships between different configurations of the prefix machine G. We do this by defining a mapping h called a *transition visitor*.

Definition 15. Mapping $h : T \to \{\varnothing, \bullet, \circledcirc\}$ is called a **transition visitor** if

$$\forall (x, y) \in T : \; h((x, y)) = \varnothing \Leftrightarrow g(x) = \varnothing \tag{8}$$

$$\forall x \in \bar{S} : \; g(x) \neq \varnothing \Rightarrow g(x) = e^{+}(x)[h] = e^{-}(x)[h] \tag{9}$$

This mapping is also required in order to evaluate the input and output conditions. We need to assign logical values to transitions, since they represent variables of those conditions. Thus, the symbol $\varphi[h]$ denotes the value of the condition φ under the mapping h (recall that we are using the tokens as logical variables). Now we can interpret the above formulas as follows. Formula (8) says that a transition is visited if and only if it is going out of a visited state. Formula (9) ensures that the token on the state x indeed corresponds to evaluation of both the input and output condition of x.

If there exists h that meets the properties (8) and (9) for a given configuration, it is called the **reachability witness**. Let's now denote $V_h = \{t \in T | h(t) \neq \varnothing\}$ the set of transitions visited by the reachability witness h. Now we are prepared for the following definitions.

Definition 16. Let C_1, C_2 be configurations of a prefix machine G. We say that the configuration C_2 is **reachable from** the configuration C_1 if C_2 is reachable by the witness h_2, C_1 is reachable by the witness h_1 and the following conditions hold

$$V_{h_1} \subseteq V_{h_2} \tag{10}$$

$$\forall x \in V_{h_1} \cap V_{h_2} : \; h_1(x) = h_2(x) \tag{11}$$

Definition 17. We say that the configuration of a prefix machine G is **reachable** if it is reachable from the initial configuration of G.

There may be more than one reachability witness for a configuration. The right side of Fig. 4 shows three possible witnesses for one initial configuration. This is allowed by the output condition of the state x.

Let's denote $\mathcal{R}_G$ the set of all reachable configurations of the machine G. By the properties of the $\subseteq$ relation we observe that the reachability relation on

$\mathcal{R}_G$ is reflexive, transitive and antisymmetric. This implies that the relation is a partial ordering of $\mathcal{R}_G$ with one minimal element – the initial configuration, and possibly several maximum elements – the terminal configurations. When we view this partial ordering as a directed graph we call this structure a **configuration graph**.

4.3 Process Simulation

Simulation of a process represented as prefix machine is done simply by traversing successive configurations of its configuration graph. Each path from the initial configuration to a terminal configuration is then called a **simulation course**. The configuration graph thus represents all possible simulation courses of a process and as such it can be used not only for process simulation, but also for process analysis.

5 Application and Examples

In the previous section, we developed a formal framework such, that it can serve as a base for sound definition of process simulation. This section presents two demonstrative examples illustrating the use of the framework to simulate and analyse processes in BORM. These examples also help to clarify the rather abstract and formal definitions presented in the previous section.

5.1 Example 1

As a first example, in Fig. 5, we can see a participant that needs to make a decision. This situation is modelled by the current OR diagram on one side and by the prefix machine on the other. The input condition at the terminal state z states that exactly one branch is allowed to be completed.

Let us look at all possible configurations of this machine pictured in Fig. 6. Configurations labelled as D are possible, but not reachable: there is no valid transition visitor satisfying all the required properties for any of them. On the

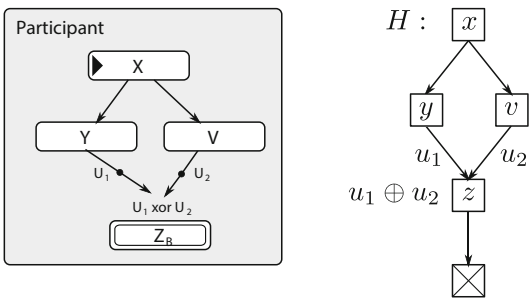


Fig. 5. A model of a simple decision participant using the prefix machine.

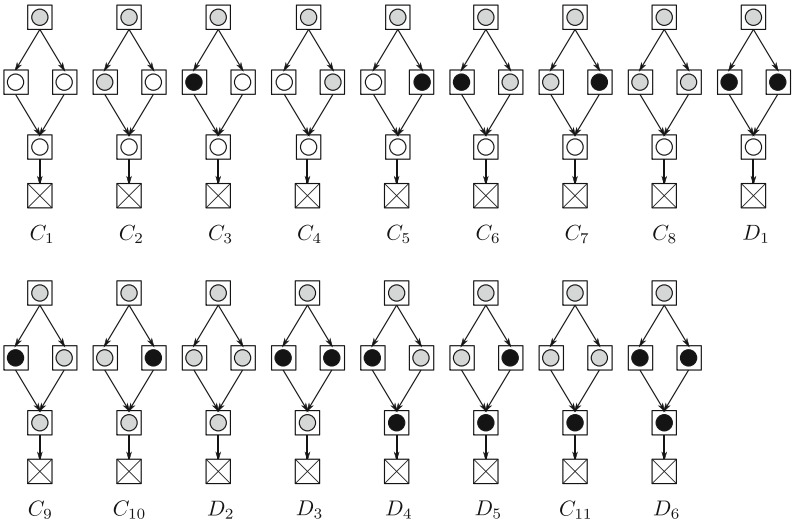


Fig. 6. All possible configurations of the machine H.

other hand, all the C configurations are reachable. In Fig. 7 we see all the reachable configurations with their respective transition visitors. Figure 8 then shows the resulting configuration graph.

Notice that the configuration C_{11} in Fig. 7 is terminal, but however, its joining state z is visited negatively and therefore the process arrives at the terminal state also negatively. Arriving at a terminal state may be interpreted as *achieving a specific goal*. Hence, the simulation course ending in configuration C_{11} did not reach the goal of the process. In the real world situation, we would say that the execution of the process has failed. The configuration graph in Fig. 8 even shows that when the simulation reaches the configuration C_8, it is doomed to eventually end up in C_{11} – and thus to fail. We use the *terminal condition* on the terminal state f to capture this notion formally. As the definition for the prefix machine says, all the transitions going into the terminal state are variables of the terminal condition. Logical values for these variables are provided by the transition visitor h. Therefore, the terminal condition can be evaluated only at a terminal configuration. In other words, the condition is evaluated at the end of the execution and states exactly which combinations of the terminal tokens represent a successfully completed execution of the process.

Now we have the tools to define the so called *failed* configurations.

Definition 18. We say that the **configuration is failed** when one of the following conditions holds.

- The configuration is terminal and the terminal condition evaluates to false.
- All paths from the configuration lead to failed configurations.

Finding failed configurations is very useful in a process execution analysis. Constructing a configuration graph and identifying the failed configurations

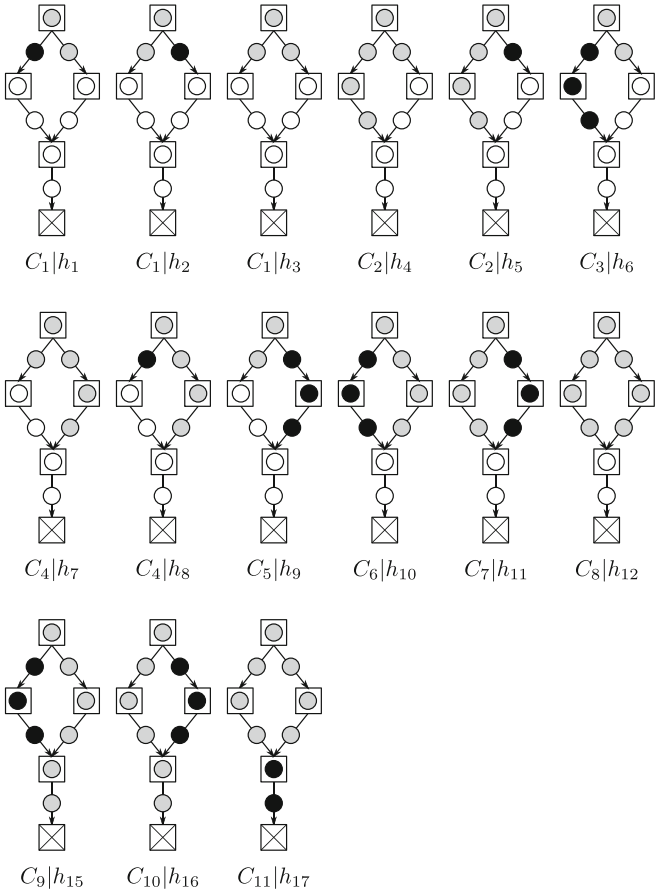


Fig. 7. All reachable configurations of the machine H with their respective transition visitors.

allows us to see valid scenarios of the process. When looking at the configuration graph in Fig. 8, we see all the failed configuration grayed out. Hence, we can easily see all the valid paths through the configuration graph.

Depending on our overall goals, we may consider the process successful even when just *some* of the participants reach terminal states (typically, the "customer" of the process). The purpose of the terminal condition in the prefix machine definition is precisely to allow such freedom in specification of the process goals. Since there is a straightforward mapping from the process machine states to participants, construction of the terminal condition can by done easily.

5.2 Example 2

The second example is a bit more complicated because it shows how communication states function. Figure 9 shows a model of two communicating participants.

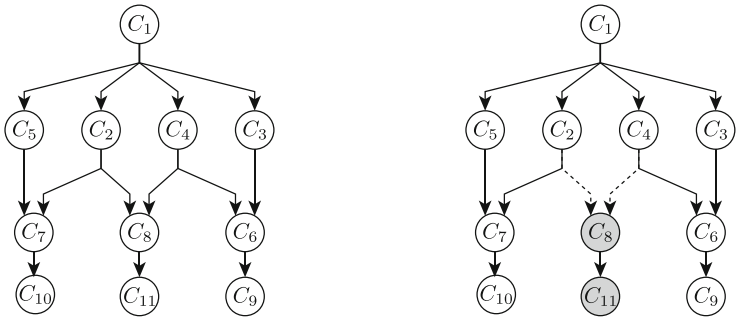


Fig. 8. The complete configuration graph of H (left). Failed configurations highlighted (right).

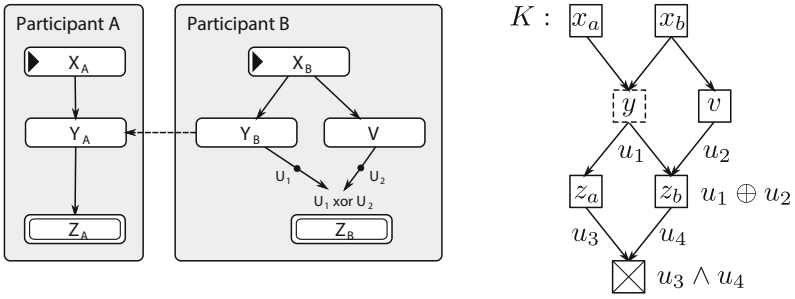


Fig. 9. A model of two communicating participants using the prefix machine.

Notice how the two communicating states from the OR diagram transform into one state in the prefix machine. As this example contains more states, its number of reachable configurations starts to grow large. Figure 10 tries to show all the 19 reachable configuration along with their 24 different transition visitors for an illustration purpose. It is now even more practical to look at the resulting configuration graph in Fig. 11.

You can notice an interesting thing there. It seems, that most of the reachable configuration are actually failed. This is due to the rather restrictive terminal condition seen in Fig. 9. This condition requires that both of the participants finish successfully. Notice, however, that when the participant B chooses to avoid communicating with the participant A by choosing to go to the other possible branch at state x_b, participant A cannot finish successfully. This is because A has no option to avoid communicating like participant B. This is a nice example of the process analysis framework. We have discovered that even though participant B seems to have a choice at state x_b, he actually has only one option in order for the whole process to succeed.

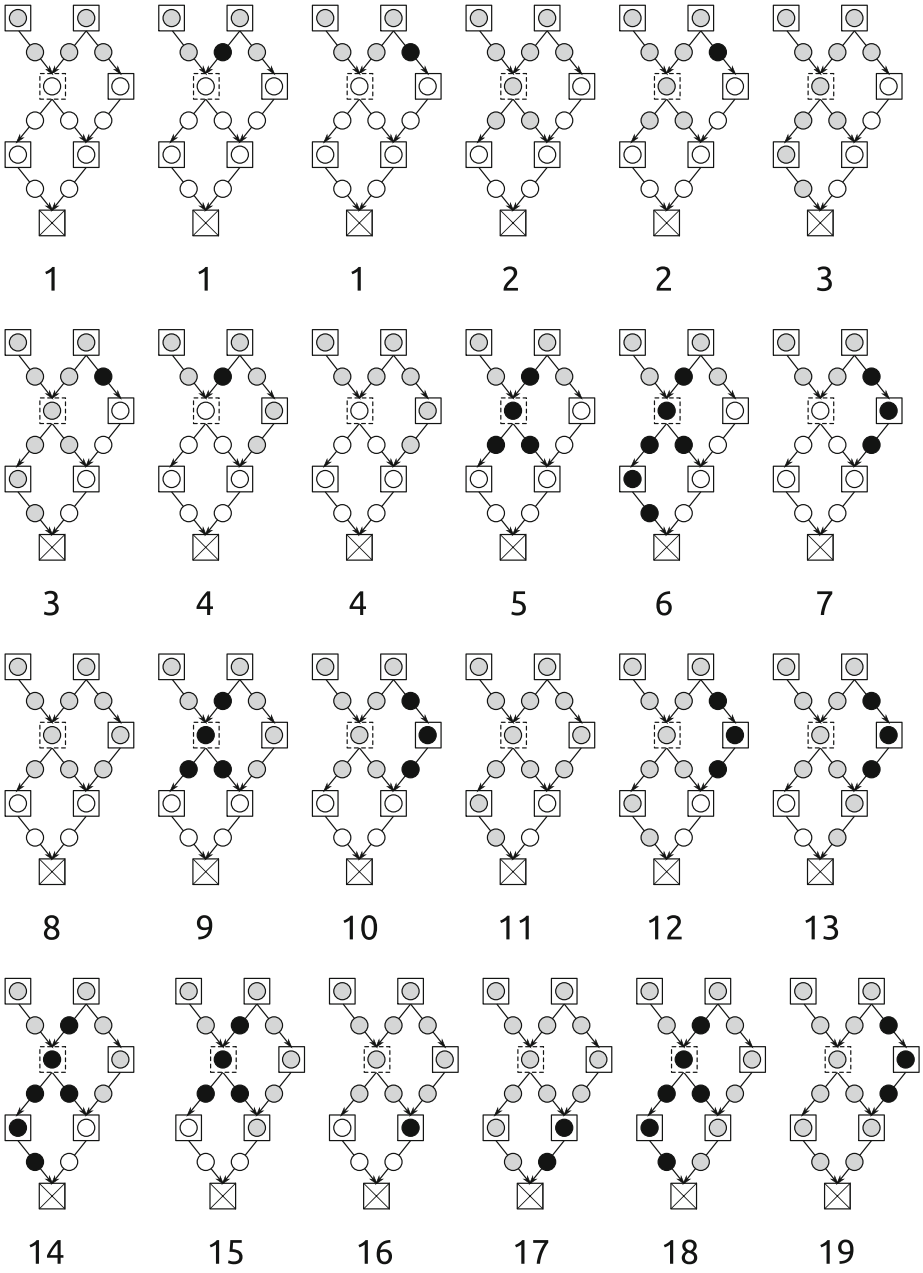


Fig. 10. All reachable configurations of the machine K with their respective transition visitors.

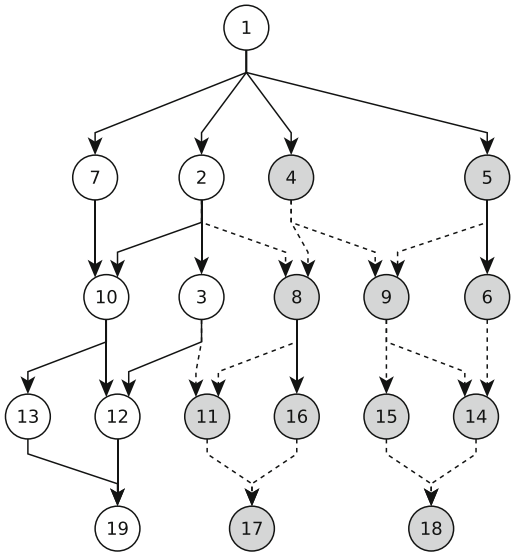


Fig. 11. The complete configuration graph of K with failed configurations highlighted.

6 Discussion

The previous section showed an example of how the prefix machine is used for simulation of BORM processes and how it can be applied to a process analysis.

The concept of failed configurations is a very useful tool in process analysis, especially if a suitable software were available. Such software tool may be used to algorithmically construct the configuration graph of a given process and, ultimately, to allow the user to inspect it closely or, simply, to learn more about the whole process. Such software, for example, could be used to identify all the courses doomed to fail and to illustrate them graphically; this, then, would identify in a neat way exactly those decisions in the process that lead to an inevitable failure.

It is not without interest that more accurate state output conditions (representing branching conditions) may be actually inferred from input conditions in the prefix machine. These inferred output conditions would allow only non-failing paths through the configuration graph when simulating the process. This opens another option for a process modelling tool which would enable the user to assign not only input, but also output conditions to every state. Then, an algorithm can be used to check whether those output user-specified conditions actually correspond to valid choices in the process by comparing them to the inferred output conditions.

7 Related Work

As we stated above, we are not aware of any systematic effort to build formal foundations of BORM OBA. Given that, our work is very likely quite novel for BORM. However, looking at model execution, simulations and behaviour analysis from a broader perspective, we may identify other attempts similar to ours – and at these, we want to look at now. There are generally two complementary approaches:

1. Start with a formal apparatus and build a practically applicable domain-oriented method and/or tool.
2. Start with a method used in practice and upgrade it into a simulation-able or an executable model.

Starting with the first type of approach, Brand's and Zafiropulo's Communicating Finite State Machines are an example. Their purpose was to design communication protocols [3]. The authors took the finite state machines (FSM) theory and upgraded it consistently for modelling several together-bound FSMs. Another example of such an approach is Pattavita's and Trigila's proposal to combine the FSM with Petri nets for modelling communicating processes [8]. Another example is the Yasper tool for workflow modelling and analysis [5]; it is based on Petri nets enriched by several practical concepts from the domain of process analysis (hierarchies, choices, roles and others).

The second mentioned approach, i.e. to upgrade an existing method is exemplified by our work. Kindred spirit to ours is Barjis: he proposed a method for developing executable models of business systems. Barjis' method is based on the DEMO method [4]. To make the static DEMO models executable, Barjis proposed a transformation into Petri nets [2]. His insight has been recently followed by, for instance, Vejrazkova and Meshkat [11].

8 Conclusion and Future Work

In this paper, we proposed a formal foundation for BORM Object Behaviour Analysis and Object Relation Diagrams (ORD). We call the resulting formalism "*the prefix machine*" (Sect. 4). This machine not only formally defines the behaviour of ORD, but it also enables the user to perform *automated process analysis* (Sect. 5) by finding valid and failed process scenarios.

With our prefix machine, we are now able to analyse the behaviour of OR diagrams, as we showed in the example and as we discussed above. We also proposed a couple of practical cases ready to implement in Sect. 5. However, we still miss a complete picture of the process in its discrete steps. So, to fill that gap, as a first step, we try to describe a *complete visualisation of the ORD behaviour* through the process. Another area for future work is to look at a formal model of cycles omitted by the prefix machine.

In order of the prefix machine to be useful and usable, we need a tool support. We plan to implement the prefix machine into the OpenCASE [9] tool.

Acknowledgements. This paper was written with the support of the SGS14 grant no. 103/OHK3/1T/18. The authors would also like to express their very great appreciation to Oskar Maxa for his insightful ideas and valuable contribution to this work.

References

1. Bang-Jensen, J., Gutin, G.Z.: Digraphs: Theory, Algorithms and Applications. Springer Monographs in Mathematics. Springer, London (2009)
2. Barjis, J.: Developing Executable Models of Business Systems. INSTICC - Institute for Systems and Technologies of Information, Control and Communication, Setubal (2007)
3. Brand, D., Zafiropulo, P.: On communication finite-state machines. J. ACM **30**(2), 323–342 (1983)
4. Dietz, J.L.G.: Enterprise Ontology: Theory and Methodology. Springer, Berlin (2006)
5. van Hee, K., Oanea, O., Post, R., Somers, L., van der Werf, J.: Yasper: a tool for workflow modeling and analysis. In: Sixth International Conference on Application of Concurrency to System Design, 2006. ACSD 2006, pp. 279–282, June 2006
6. Knott, R., Merunka, V., Polák, J.: Process modeling for object oriented analysis using BORM object behavioral analysis. In: 4th International Conference on Requirements Engineering, 2000. Proceedings, pp. 7–16 (2000)
7. Merunka, V.: Object-oriented process modeling and simulation - BORM experience. Trakia J. Sci. **8**(3), 71–87 (2010)
8. Pattavina, A., Trigila, S.: Combined use of finite-state machines and petri nets for modelling communicating processes. Electron. Lett. **20**(22), 915–916 (1984)
9. Pergl, R., Tůma, J.: `OpenCASE` – a tool for ontology-centred conceptual modelling. In: Bajec, M., Eder, J. (eds.) CAiSE Workshops 2012. LNBIP, vol. 112, pp. 511–518. Springer, Heidelberg (2012)
10. Podloucký, M., Pergl, R.: Towards formal foundation for the BORM OR diagrams validation and simulation. In: Proceedings of the 16th International Conference on Enterprise Information Systems, pp. 315–322 (2014)
11. Vejrazkova, Z., Meshkat, A.: Translating DEMO models into petri net. In: Barjis, J., Gupta, A., Meshkat, A. (eds.) EOMAS 2013. LNBIP, vol. 153, pp. 57–73. Springer, Heidelberg (2013)

Enterprise Optimisation

Simulation-Based Cyber-Attack Assessment of Critical Infrastructures

Marlies Rybnicek, Simon Tjoa(✉), and Rainer Poisel

Insitute of IT Security Research, St. Pölten University of Applied Sciences,
Matthias Corvinus Street 15, 3100 St. Pölten, Austria
{marlies.rybnicek,simon.tjoa,rainer.poisel}@fhstp.ac.at
http://english.fhstp.ac.at

Abstract. Nations, more than ever, depend on the correct functionality of critical infrastructures. In order to deliver their services, critical infrastructure providers often rely on information technologies. Thus, cyber attacks can lead to severe impacts within a nation's critical infrastructure landscape causing deep scars to health, safety and economic wealth. To provide the demanded service level of critical infrastructures and to reduce the impacts of disruptions and unavailability of components during attacks, it is essential to have a comprehensive understanding of the linkages between providers on the one side and to have the capabilities to identify vulnerabilities of systems and their consequences if exploited on the other side. Therefore, in this paper, we present a agent-based modeling and simulation approach facilitating the assessment of critical infrastructure entities under attack. To demonstrate the capabilities we further provide a motivational example how our approach can be used to perform simulation-based evaluation of cyber attacks. We further provide an overview of our simulation prototype.

Keywords: Cyber attacks · Critical infrastructures protection · Agent-based modeling and simulation · Anticipation games · Distributed denial of service

1 Introduction

Our personal, social and economic welfare heavily depends on services provided by critical infrastructures [1]. The German Federal Office for Information Security [2] defines the term *Critical Infrastructure* as "... organizational and physical structures and facilities of such vital importance to a nation's society and economy that their failure or degradation would result in sustained supply shortages, significant disruption of public safety and security, or other dramatic consequences" [2].

The far-reaching implications of malfunction and disruptions in the past demonstrated the vulnerability of critical infrastructures. Occurred events showed the extent of impacts caused by a wide variety of threats, such as targeted attacks

J. Barjis and R. Pergl (Eds.): EOMAS 2014, LNBIP 191, pp. 135–150, 2014.
DOI: 10.1007/978-3-662-44860-1_8

(e.g. cyber-attacks on critical infrastructure [3,4]), failures (e.g. major blackouts [5]) or natural disasters (e.g. earthquakes and floods [6]).

To be prepared for the multitude of threats and hazards, it is essential to minimize exposure and to improve resilience and resistance of critical organizations. But this requires to set the right priorities for securing critical infrastructures. In order to meet this objective it is crucial to have a clear understanding of the dependencies and interdependencies between individual infrastructures.

This is an extremely difficult task as critical infrastructures became extremely interweaved among each other. Furthermore, the fast technological progress and the growing complexity make it even harder to protect critical infrastructures. Another challenge of today's critical infrastructure providers are owed by multitude of public and private organizations [7] which have different organizational cultures, strategies and risk appetites.

Min et al. [8] highlight in their paper the importance of analyzing interdependencies and impacts of disruptions of critical infrastructures. The authors claim that building the models is a challenging task as "... (i) data acquisition is difficult; (ii) each individual infrastructure is complicated; (iii) infrastructures are evolving; (iv) governing regulations are changing; and (v) model construction is jointly performed by government agencies, academia, and private industries".

Additionally different types of interdependencies [9] between critical infrastructures have to be considered. According to Rinaldi et al. [9], interdependencies can be categorized into cyber, geographic, physical and logical interdependencies. *Cyber* interdependencies comprise all interdependencies between (information) infrastructure components which are connected via electronic/informational links. Interdependencies based on geographic proximity are categorized as *geographic* interdependencies. Infrastructures are *physical* interdependent if the rely on physical outputs of other infrastructures. All interdependencies that cannot be categorized by the before-mentioned categories can be classified as *logical* interdependencies.

In course of this paper we want to concentrate on cyber attacks which pose an enormous risk to services of critical infrastructures relying on information and communication technologies. One reason for the exposure is the increasing use of IT services provided by external providers. The Information Security Breaches Survey 2012 [10] confirms the trend that organizations outsource more and more business processes over the internet. Thus, it is no big surprise that experts agree that it is vital for organizations to assess, analyze and evaluate their dependencies. In a recent report, analysts of Chatham House [11] underline the necessity to deeper analyze "dependencies and vulnerabilities hidden in other organizations" [11].

Large-scale denial of service attacks and network infiltrations [12] and recent cyber-attacks (e.g. on water and energy systems) [13] impressively demonstrated that no industry is immune against cyber-criminalists and cyber-terrorists. According to an evaluation of cyber-attack statistics [14], Distributed Denial of Service (DDoS) attacks are identified as an extremely serious challenge for ICT dependent critical infrastructures. Distributed Denial of Service (DDoS) is

a special form of DoS attacks where an attacker uses a multitude of computers, called zombies and/or bots. If a botnet is large enough to consume the resources of the target-system, the system under attack collapses and its legitimate users are prevented from using it [15].

The major contribution of this paper is the introduction of a approach to simulate critical infrastructures and their interdependencies. The intention for the creation of such a model was to ease the participation of critical infrastructure suppliers by providing them a model where they can choose the level of granularity according to their business requirements and company structure. A main emphasis thereby was to enable the interaction between agents of different modeling layers.

To manage the complexity when assessing the risks and impacts of different scenarios, simulation techniques are used. For our simulation approach, as mentioned above, we specialized in cyber interdependencies and the impacts focusing on Distributed Denial of Service (DDoS) attacks on critical infrastructure sectors. To cope with this complexity we decided to use an Agent-based Modeling and Simulation. The behavior of the agents is described by using the game theoretical approach of Anticipation Games.

The remainder of this paper is structured as follows: In Sect. 2, the requirements of simulating critical infrastructures are described. Section 3 introduces our generic approach to simulate critical infrastructures and their interdependencies. Furthermore, we present the design of our prototype. In Sect. 5 we outline our conclusion and declare future work.

2 Requirements for an Approach Enabling Simulation-Based Cyber-Attack Assessment of Critical Infrastructures

A frequently mentioned key objective when performing critical infrastructure modeling and simulation is to observe the dynamic effects caused by attacks, malfunctions, interruptions or disruptions of systems of critical infrastructure providers. The usage of simulation techniques enables analysts to improve their understanding about cascading and domino effects. Reference [16] As the services provided by critical infrastructures are often unique, it can be observed that they are highly interconnected and interdependent amongst each other. That is the reason why the unavailability or improper functionality of one critical service can lead to devastating consequences.

Besides the value for risk-assessment, modeling and simulation can be used to gain a better understanding on the security situation of complex and interweaved systems. Therefore it can be used to continuously improve the planning of redundancies and incident response strategies as well as to facilitate the planning of exercises (e.g. scenario simulation exercises).

Derived from the above-mentioned aims, the succeeding requirements can be identified for a generic approach to model and simulate critical infrastructures. In order to enable the consideration of different viewpoints (e.g. service provider

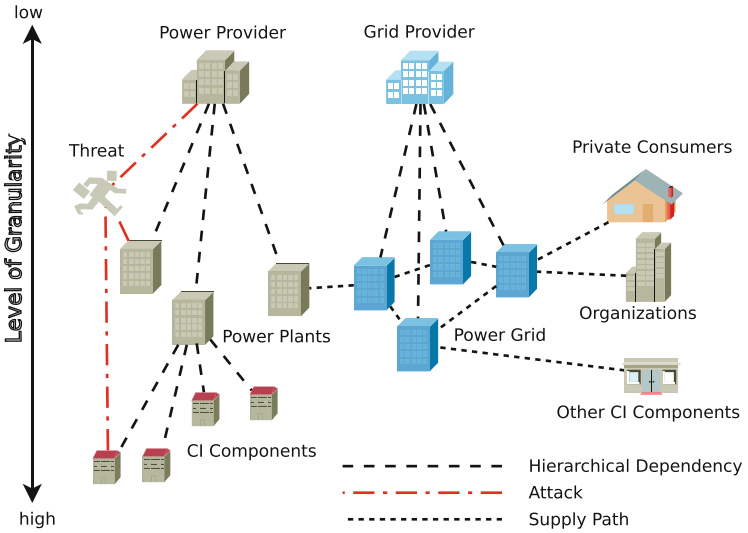


Fig. 1. Level of granularity

sector), the modeling and simulation approach has to be flexible with respect to the modeling level of granularity, see Fig. 1. Furthermore, as critical infrastructure providers have different fields of operations (e.g. water supply, ICT) the modeling approach must be capable to provide a generic way to model such services and underlying systems as well as their connections.

For a critical infrastructure simulation it is crucial to consider different impact categories. From the definitions introduced in Sect. 1 we derived the following impact areas caused by critical infrastructure failures: National economy, public order and social welfare, environment, national security/defense and health & safety [17,18]. Failures concern a lot of different parties, private users, providers or other critical infrastructures.

For the characterization of security requirements of a critical infrastructure provider we chose to implement well-known and widely used concepts and paradigms of security and dependability. One of the most popular approaches for defining security requirements is the CIA triad [19]. CIA stands for the three security goals confidentiality, integrity and availability defined by ISO 17799 respectively 27002 [20]:

- **Confidentiality** - ensuring that information is accessible only to those authorized to have access.
- **Integrity** - safeguarding the accuracy and completeness of information and processing.
- **Availability** - ensuring that authorized users have access to information and associated assets when required.

Although the CIA triad provides a solid foundation for the analysis of various threat scenarios, we decided to implement additionally the concept of security and dependability as described by Avizienis et al. [21–23]. Dependability is described as "... the ability to avoid service failures that are more frequent and more severe than [...] acceptable" [23]. In addition to the adapted attributes confidentiality, integrity and availability, the authors [21] introduce the extended attributes reliability, safety and maintainability to seek security of systems:

- **Reliability** - safeguarding the continuity of service.
- **Safety** - is the non-occurrence of catastrophic consequences on the environment.
- **Maintainability** - is the ability to undergo repairs and evolutions.

More attributes which could be used to further specify business and security attributes can be derived from the SABSA taxonomy of ICT business attributes [24,25].

3 An Approach Simulating Critical Infrastructures Under Cyber Attacks

In this section we introduce our model for analyzing critical infrastructures and their interdependencies. The herein presented approach pursues the goal to provide critical infrastructure providers a tool to model as well as to simulate threats and hazards. In order to manage the complexity we decided to use the Agent-based Modeling and Simulation approach. For modeling the behavior of agents an extended version of Anticipation Game [26–28] is used. Interdependencies within critical infrastructures and to external organizations are acquired by applying techniques described in the German BSI Standard for Business Continuity [29].

3.1 Agent-Based Modeling and Simulation

Agent-based Modeling and Simulation [30] provides the capabilities to model and simulate complex systems and is applied in many different areas like Biology, Medicine, Physics, Chemistry, Security, Social and Economic Modeling and range from minimalistic models to large scale decision support systems [31]. Each entity of a complex system can be described within an Agent-Based Modeling and Simulation framework as an individual agent. Every agent has its own behavior, goals and experience, its attributes (e.g. name, coordinates, security and dependability attributes) further specify the entity. Therefore this approach is most suitable to simulate heterogeneous individuals like humans, institutions, organizations or even critical infrastructures including their full diversity.

In our simulation we differentiate between three types of agents: Critical infrastructure agents, Consumer agents and Threat agents.

- Consumer agents: These agents model consumers and are demanding the services of a critical infrastructure providers. Consumer agents help to gain a clearer understanding about the impacts caused by interruptions or malfunctions of critical infrastructures and to get more information about changes in the service demand. In our case, a consumer agent can be a private entity (e.g. household) or a business entity.
- Critical infrastructure (CI) agents: CI agents represent entities of a critical infrastructure and face threats which are declared as agents too. Every threat agent is able to attack critical infrastructure components leading to a decrease of their dependability and/or security depending on the countermeasures of the CI-agent.
- Threat agents: These threats attack certain attributes of an agent which impacts attributes of services and agents. In our simulation Threat agents are used to perform cyber attacks, such as DDoS attacks.

According to Macal et al. [30] a typical Agent-based Model consists of three elements: a set of agents, their attributes and behavior, a set of agents relations based on an underlying topology of connectedness to define how and with whom agents interact and an agent environment.

In our simulation we assume that a critical infrastructure which is vital to a nations well-being because of provision of goods, funding, public health, safety and so on are dependent on or provide services. In order to specify service parameter required by an agent we use attributes (e.g. availability). Additionally, every agent aims to achieve its predefined objectives, e.g. an energy provider delivers the service energy to other Critical infrastructures or Consumer agents.

The behavior of agents is expressed using a rule set and a set of strategies based on Anticipation Games which are introduced in the succeeding Sect. 3.2 and can be dynamically adapted during the simulation according to learning rules. This enables the agent to react and interact autonomous with its environment, influences other agents and in turn are able to learn based on gained experience to be better suited to their environment. These is essential for simulating cyber attacks as an interaction between Critical Infrastructures and Threat agents. Relationships within or between CI-agents and Customer agents are defined based on an adapted version of the [29].

3.2 Game-Based Implementation of Agent Behavior

For modeling the behavior of agents, game theory is used [26–28,32–34]. The strategic interactions among the CI, Consumer and Threat agents are defined as an adapted Anticipation Game. Bursztein [26–28] introduced Anticipation Games in order to combine attack graphs and game theory. This technique enables the simulation of realistic and dynamic real-time behavior of two players as well as their interactions. NetQi which is a complete implementation of Anticipation Games by Bursztein [26–28] demonstrated the suitability of the approach in the field of service failure analysis.

We assume, in course of this paper, that CI-agents as well as Consumer agents under attack are acting as a defender team. After detecting an attack, defender agents use all possible and available defense mechanisms to protect against the attack. Auxiliary defense mechanisms differ from cyber attack to cyber attack. To get closer to reality not all defend mechanisms are available to every defender. Based on the level of knowledge, experience, money and credits, defender act different.

For the description of a Distributed Denial of Service (DDoS) attack from the attacker and defender view, an attack/defend graph is used. Based on [35], we categorize DDOS attacks into bandwidth and resource depletion. In order to defend against these attacks, several defense mechanisms such as DDoS Mitigation Service, Active Load Balancing, Throttling or Dropping Requests and Traffic Pattern Analysis can be deployed. Figure 2 visualizes the attack/defend graph which has been used to describe the actions of attackers and defenders. More information about attacks as well as possible prevention and defense mechanisms are highlighted by [35,36].

One or more malicious DDoS agents (the attacker team) are able to attack critical infrastructures or consumers anytime at any location. The behavior of defender and attacker agents depends on the type of attack, the detection rate and the available countermeasures which are applied. Furthermore, the behavior is influenced by predefined strategies (e.g. aggressiveness of attacker). The notation of strategies S is based on the work of [26–28].

$$S : (name, player, objectives, objectivesorder, constraints, location)$$

Interactions of attack and defender are defined by means of timed decision-rules following the form [28]:

$$\Gamma_x : PreF \overset{\Delta,p,a,c,e}{\rightarrow} P$$

F is a set of preconditions, Δ the time needed to execute an action. P identifies the player and a the chosen action. The parameter c provides information about the cost of the chosen action. We extended the rules by a novel parameter e for experience. By means of this parameter it is possible to differ between unskilled and professional attackers and defenders.

A game starts at a discrete point in time. The attacker agent has a set of actions at this time. The defender agent is able to recognize an attack depending on the probabilistic detection rate of an attack. After the detection of the attack a defender can choose a response out of the defender's actions. The game then moves forward to another state. An attacker and defender payoff function, as well as a state transition function, is defined for each combination of possible strategies.

4 Realization

In the following, we briefly outline how our approach can be implemented in a real world setting. As outlined in Fig. 5 we use two views to improve maintainability and usability of the approach.

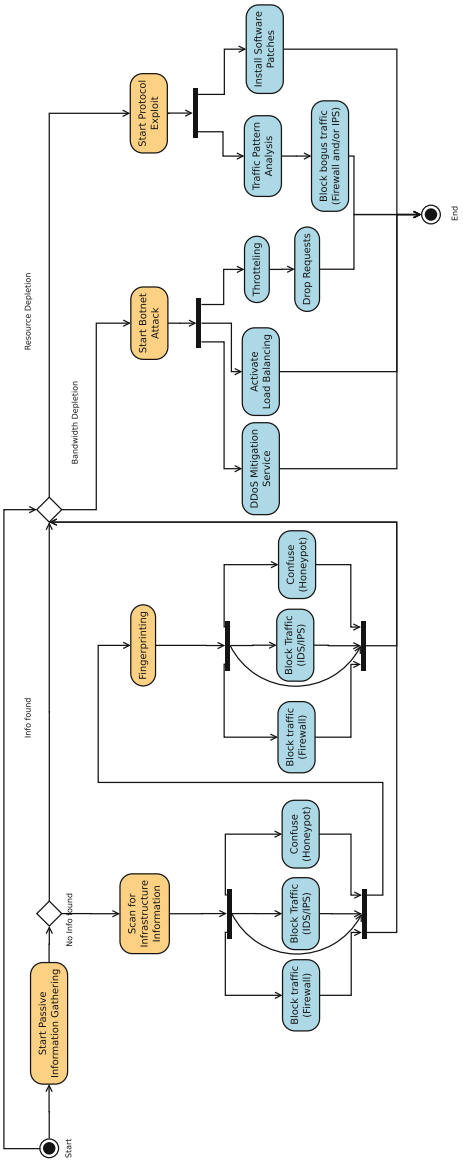


Fig. 2. DDoS attack and defend graph

The infrastructure view is used to get a deeper understanding on an organizational level (micro dimension) about the dependencies of individual critical infrastructure providers. The data expressing the dependencies between internal and external services can be acquired using a business impact analysis based

approach. For our realization we aligned our acquisition process to the German BSI-Standard 100-4 [29].

As suggested in the standard, within the first step of the analysis the scope is determined. After this initial step a damage assessment takes place. Within the assessment phase, each organization uses applicable damage categories, such as impacts on reputation/brand, financial performance, safety, environmental damage or compliance to legal and regulatory requirements. The outcome of the assessment is structured information how impacts evolve over time if a certain process or service is interrupted or disrupted. Then important recovery parameter such as RTO (recovery time objective) or MTPD (maximum tolerable period of disruption) are identified. The succeeding steps further refine the analysis by providing information about dependencies of processes, priority and criticality of processes. In the final step the resource requirements for normal and emergency operations over time are determined.

On a higher abstraction layer (macro dimension) the damage categories which are determined by the impacts of the individual providers are different. In a report for the European Commision [37] the following four hazard categories - to assess impacts of critical infrastructure disruptions - have been presented: (1) number of human casualties, (2) restricted ability of governments and public authorities to act, (3) economic losses and (4) public effects. Figure 3 shows example descriptions of the proposed categories.

Hazard Category / Aspect	negligible	low	medium	large	catastrophic
Human Casualties	No consequences	Slightly reduced well-being	Few casualties	Many casualties or some fatalities	Many fatalities
Restricted governmental ability to act	Slight restrictions	Few reductions	Limited restrictions	Considerable restrictions	Inability to act
Economic damages	Up to € 10 Mio	€ 10 - 100 Mio	€ 100 Mio - 1 Bil	€ 1 - 10 Bil	More than € 10 Bil
Public effects	Slight restrictions	Few reductions	Local riots	Regional loss of public order	Nation wide loss of public order

Fig. 3. Hazard categories as introduced by [37]

The attacker/defender view represents the attack and defense strategies for certain scenarios. As a motivational example, in the following we will shortly outline how Distributed Denial of Service attacks on ICT infrastructures can be simulated using our approach. The reason why we decided to demonstrate the application of our approach with this particular example is that DDoS poses a serious threat for ICT dependent critical infrastructures which can have serious external and internal impacts on critical infrastructure sectors.

4.1 Prototypical Implementation

In this section we summarize the implementation of our prototype. Our prototype consists of four parts: the input component, the simulation engine, the result database, and the visualization framework. Using this approach our framework aims at supporting the simulation of critical infrastructures which are compromised by cyber attacks.

Input Component. By using this component, general and specific data can be defined using Excel spreadsheets.

General scenario data describes parameters like database connection parameters, the number of simulation rounds, or parameters concerning agents that exist in every simulation. Scenario specific data consists of agents that are involved in the actual simulation, such as critical infrastructure components or agents that represent attackers or defenders in cyberspace.

In order to process Excel spreadsheets, the OpenL library [38] is utilized. This library allows to define the logic of involved agents both in kind of business logic (in case of simple logic) or directly in JAVA (in case of complex logic). The overall structure of configuration spreadsheets is optimized by separating relevant documents across several files. Therefore it is possible to present only the amount of information to people defining relations between involved agents that is necessary for them to understand their part of the simulation. Relevant spreadsheets are referenced in the main configuration spreadsheet.

Simulation Component. The proof-of-concept prototype of our framework is based on the MASON JAVA-library [39,40] which provides "...a fast, easily extensible, discrete-event multi-agent simulation toolkit" [39]. In our simulation this library offers efficient management of the round-based application flow as well as basic functionality, such as simulation data types. Using MASON, registered agents are invoked once per round in random order [41]. In the following the data structures involved in simulations are described. Figure 4 shows a UML-class diagram of agents available to simulations.

The `SimObject` class is the base class of all simulation entities. It does not contain any logic that is relevant to the actual simulation. Instead, each `SimObject` has a name attribute. Further, it contains a location attribute which can be utilized by the simulation in order to find agents based on their geographical position.

The `Agent` class is derived from the `SimObject` class. It extends existing attributes (name, location) with path-information which, together with its name, results in an identifier that is unique for every simulation. Furthermore, `Agent` objects consist of `SimAttribute`s which in turn consist of an arbitrary name and a floating point value. All `Agent` objects are organized in a tree-like structure ("Hierarchical Dependencies"). The tree of agents can be traversed from each `Agent`. Our framework provides convenience functions for that purpose. In addition to that, `Agent`s of the same hierarchical level can be organized in networks. This allows to implement path-finding algorithms such as the Dijkstra algorithm in order to determine whether different `Agent`s can communicate

with each other. In our simulation we further utilize the JGraphT library [42] which allows to determine if a communication path can be established between arbitrary `Agent`s. Using this approach, data exchange scenarios as given in e. g. telecommunications or IT infrastructures can be simulated effectively.

Figure 1 outlines examples of hierarchical relationships. In our model, each `Agent` additionally has a reference to its parent `Agent`. All `Agent`s that do not have an explicit parent `Agent` are assigned the generic root `Agent`. The tree-like organization of `Agent`s allows to group `Agent`s based on certain criteria. For example, complex infrastructures such as data centers consist of a myriad of devices like computers and network components. In order to determine the `Agent`s which are part of a particular data center, those `Agent`s that have the data center as parent agent, are selected.

In order to effectively implement Anticipation Games in an agent-based simulation, the `Agent` class has further been extended. With `Agent`s having their own logic which can be implemented by users of our framework, `AgAgent`s have a final logic which implements Anticipation Games. Each `AgAgent` is assigned a set of rules which are maintained by the so called `RuleManager`.

During each round, depending on its budget (simulation money), an `AgAgent` chooses one of the available rules and executes it. It is possible that running a rule lasts for several simulation rounds. The decision on which rule to take, depends on the `Strategy` assigned to an `AgAgent`. `Strategy` objects make their decisions based on a scoring that determines how relevant a rule is for the current situation. An example for a cyber-attacker would be to apply those rules first that fit the available budget and cause the highest damage to its oponent. The `CiAgent` class extends the `AgAgent` class by introducing three standard `SimAttribute`s including convenience getter- and setter-methods: confidentiality, integrity, and availability.

The relation between different `CiAgent`s according to the German BSI-Standard 100-4 [29] is described by contingency tables. Based on these definitions, the propagation of states (e. g. damages) is accomplished by a specialized Anticipation Rule which has a cost of 0 (zero). Thus, it is applied to each agent it is assigned to each round.

Results Component. At the time of writing, during each step of the simulation specific data structures are fed with current values of involved agents' attributes. These data structures can then be queried by the visualization component to graphically represent the results of each simulation step.

In the future, we intend to further enhance the support for storing simulation results in databases with GIS support (e.g. PostGIS [43], or PERST [44]). This would allow for the implementation of analysis software using different technologies or programming languages. Furthermore, this approach would allow to implement filtering mechanisms and thus to reduce the complexity as well as the amount of data that would have to be processed by the visualization or reporting component.

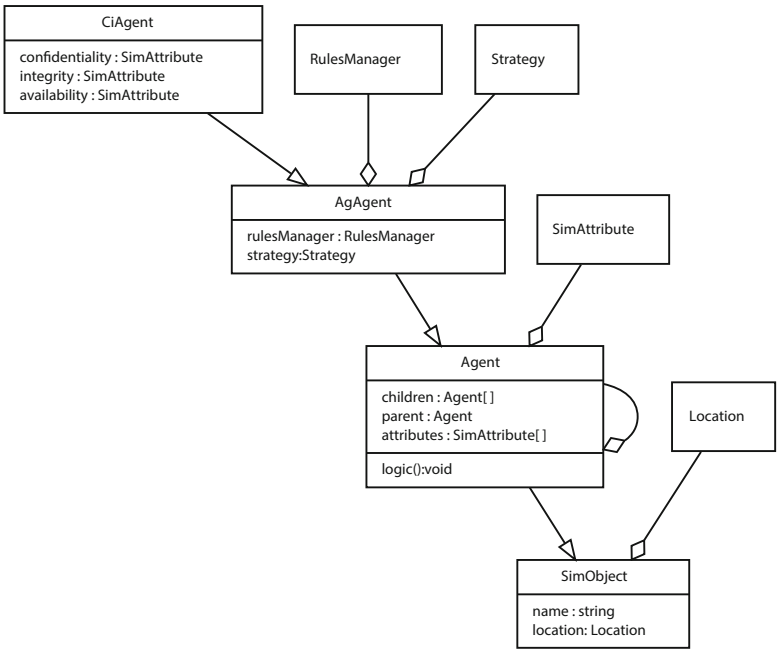


Fig. 4. Overview of involved types of agents

Visualization and Reporting Component. Using our framework results from simulations can be visualized in two ways: by using existing GUI-libraries such as the SWING framework or by using web-based GUIs that obtain their data from the results component. Chart libraries such as JFreeChart [45] allow for basic reporting functionality. In our prototype we visualize the chronological sequence of `SimAttribute` values as two-dimensional chart ("XY-Chart") with the horizontal axis representing time and the vertical axis representing the actual value.

In the future we intend to provide users of our framework with extensive reporting functionality such as provided by the iReport library [46].

5 Conclusion and Outlook

Our social and economic life depends on reliable functionality of critical infrastructure components. Through the tight interdependencies and interactions between critical infrastructure services provided by public and private, the disruption of a critical service can lead to far-reaching consequences caused by domino and cascading effects. For this reason, it is vital to create new possibilities to get a deep understanding of the influence of interdependencies from a micro and macro social-economical perspective.

Therefore in this paper we initially elaborated the requirements for a simulation-based assessment approach. We then introduced our generic approach

to model and simulate critical infrastructure entities. The specific feature of our approach is the flexibility in choosing the level of granularity and the convenient way to define dependencies. In order to find broad acceptance amongst critical infrastructure providers we decided to align our approach with well-known and widely accepted business continuity standard. For the simulation of interactions between attackers and defenders we adapted the Anticipation Game approach of [26–28].

Figure 5 schematically outlines how our approach is structured. The left view (infrastructure view) holds information about the individual critical infrastructure, their dependencies and the services they offer. The information required in this view can be extracted to a great extend from business continuity tools such as business impact analysis. The right view (attacker/defender view) represents the capabilities, behavior and strategies of defending and attacking parties.

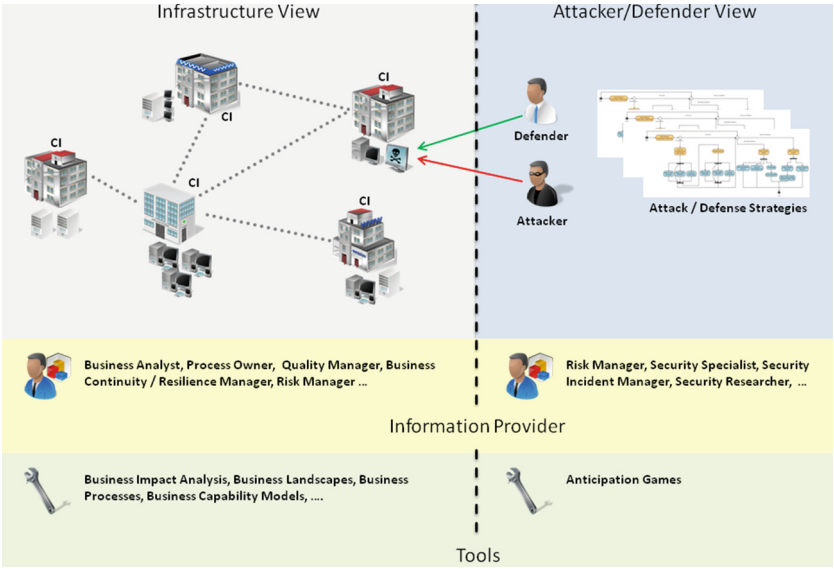


Fig. 5. Overview of simulation entities

To demonstrate how our approach can be put into effective practice, we elucidated the implementation of our proof-of-concept prototype. It consists of four parts (input component, simulation component, results component, and visualization component) which together represent a solid framework supporting the process of simulating critical infrastructures. Software components implemented for this project are designed to be reusable also in complex setups as given when simulating the behavior of large critical infrastructures which are compromised by cyber attacks.

In the near future we intend to extend our simulation to support distributing parts of the simulation to different nodes on a computer network. This approach

allows to provide users with privacy settings. Secret information such as relation parameters from users of the system could be concealed from other users of the system but still being able to simulate complex critical infrastructure setups.

Acknowledgments. This work has been supported by the Austrian Research Promotion Agency (FFG) under the Austrian Security Research Programme KIRAS.

References

1. Mansfield, N.: Development of policies for protection of critical information infrastructures. Technical report, Organisation for Economic Co-operation and Development (OECD) (2007)
2. German Federal Office for Information Security: Recommendations for critical information infrastructure protection (2013)
3. Symantec: Symantec intelligence quarterly report: Q4 2010 - targeted attacks on critical infrastructure. Technical report, Symantec (2010)
4. Mandiant: Mandiant intelligence center report - apt1: Exposing one of china's cyber espionage units. Technical report, Mandiant (2013)
5. Public Safety Canada: Ontario-U.S. power outage - impacts on critical infrastructure (2006). http://www.publicsafety.gc.ca/prg/em/ia06-002-eng.aspx. Accessed: 16 May 2012
6. Centre for Natural Hazard Research: Types of hazards. http://www.sfu.ca/cnhr/types.html. Accessed: 16 May 2012
7. Hellström, T.: Critical infrastructure and systemic vulnerability: towards a planning frame. Saf. Sci. **45**, 415–430 (2007)
8. Min, H., Beyeler, W., Brown, T., Son, Y., Jones, A.: Toward modeling and simulation of critical national infrastructure interdependencies. IIE Trans. **39**(1), 57–71 (2007)
9. Rinaldi, S.M., Peerenboom, J.P., Kelly, T.K.: Identifying, understanding and analyzing critical infrastructure inderdependencies. IEEE Control Syst. Mag. **21**, 11–25 (2001)
10. Potter, C., Waterfall, G.: Information security breaches survey 2012. Technical report, PwC (2012)
11. Cornish, P., Livingstone, D., Clemente, D., Yorke, C.: Cyber security and the uk's critical national infrastructure. Technical report, Chatham House (2011)
12. Baker, S., Filipiak, N., Timlin, K.: In the dark - crucial industries confront cyberattacks. Technical report, McAfee - Center for Strategic International Studies (2011)
13. Obama, B.: Taking the cyberattack threat seriously (July 2012)
14. Hackmageddon.com (2013). http://hackmageddon.com/2012-cyber-attacks-statistics-master-index/. Accessed: 20 February 2013
15. CERT CC: Denial of Service Attacks (1999). http://www.cert.org/tech_tips/denial_of_service.html. Accessed: 20 February 2013
16. George Mason University: The CIP Report, August 2010. http://cip.gmu.edu/archive/CIPHS_TheCIPReport_August2010_CIPHSUpdate.pdf. Accessed: 16 May 2012
17. Boin, A., McConnell, A.: Preparing for critical infrastructure breakdowns: the limits of crisis management and the need for resilience. J. Contingencies Crisis Manage. **15**(1), 50–59 (2007)

18. Moteff, J., Parfomak, P.: CRS Report for Congress - Critical Infrastructure and Key Assets: Definition and Identification. Technical report, Congressional Research Service (2004). Accessed: 16 May 2012
19. Harris, S.: CISSP All-in-One Exam Guide, 5th edn. Mcgraw-Hill Professional, New York (2010)
20. ISO/IEC: ISO/IEC 27002:2005 Information technology - Security techniques - Code of practice for information security management (2005)
21. Laprie, J.C.: Dependable computing: concepts, limits, challenges. In: 25th IEEE International Symposium on Fault-Tolerant Computing, Pasadena, CA, USA, pp. 42–54. IEEE (1995)
22. Avizienis, A., Laprie, J.C., Randell, B.: Fundamental concepts of dependability. Seven **1145**, 7–12 (2001)
23. Avizienis, A., Laprie, J.C., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Dependable Secure Comput. **1**(1), 11–33 (2004)
24. Sherwood, J., Clark, A., Lynas, D.: Enterprise security architecture. Technical report, SABSA Institute (2009)
25. Sherwood, J., Clark, A., Lynas, D.: Enterprise Security Architecture: A Business-Driven Approach. CRC Press, San Francisco (2005)
26. Bursztein, E.: NetQi: a model checker for anticipation game. In: Cha, S.S., Choi, J.-Y., Kim, M., Lee, I., Viswanathan, M. (eds.) ATVA 2008. LNCS, vol. 5311, pp. 246–251. Springer, Heidelberg (2008)
27. Bursztein, E.: Extending anticipation games with location, penalty and timeline. In: Degano, P., Guttman, J., Martinelli, F. (eds.) FAST 2008. LNCS, vol. 5491, pp. 272–286. Springer, Heidelberg (2009)
28. Bursztein, E.: Multiple-sites defense strategy. Technical report, LSV, ENS Cachan, CNRS (2009)
29. BSI-Standard 100–4: Business Continuity Management (2008)
30. Macal, C.M., North, M.J.: Tutorial on agent-based modelling and simulation. J. Simul. **4**(3), 151–162 (2010)
31. Allan, R.: Survey of agent based modelling and simulation tools. Engineering **501**, 57–72 (2009)
32. Liu, D., Wang, X., Camp, L.J.: Game theoretic modeling and analysis of insider threats. Int. J. Crit. Infrastruct. Prot. **1**, 75–80 (2008)
33. Grossklags, J., Christin, N., Chuang, J.: Secure or insure?: a game-theoretic analysis of information security games. In: Proceedings of the 17th International Conference on World Wide Web, pp. 209–218. ACM (2008)
34. Boehmer, W.: Dynamic systems approach to analyzing event risks and behavioral risks with game theory. In: 2011 IEEE Third International Conference on Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third International Conference on Social Computing (SocialCom), pp. 1231–1238 (2011)
35. Specht, S., Lee, R.: Distributed denial of service: taxonomies of attacks, tools, and countermeasures. In: Proceedings of the 17th International Conference on Parallel and Distributed Computing Systems, pp. 543–550 (2004)
36. Mirkovic, J., Reiher, P.: A taxonomy of DDoS attack and DDoS defense mechanisms. SIGCOMM Comput. Commun. Rev. **34**(2), 39–53 (2004)
37. Gottwald, S.: Studyon critical dependencies of energy, finance and transportinfrastructures on ict infrastructure. Technical report, European Commission (2009)
38. OpenL Tablets: Business Friendly Rules (2013). http://openl-tablets.sourceforge.net/. Accessed: 14 March 2013

39. Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., Balan, G.: MASON: a multi-agent simulation environment. Trans. Soc. Model. Simul. Int. **82**(7), 517–527 (2005)
40. George Mason University: MASON (2012). http://cs.gmu.edu/eclab/projects/mason/. Accessed: 26 July 2012
41. Luke, S.: Multiagent simulation and the MASON library, August 2011. http://cs.gmu.edu/eclab/projects/mason/manual.pdf
42. Naveh, B.: Contributors: JGraphT (2013). http://jgrapht.org/. Accessed: 15 March 2013
43. Refractions Research: PostGIS, March 2013. http://www.postgis.org/. Accessed: 15 March 2013
44. mcobject: Perst - an open source, object-oriented embedded database, March 2013. http://www.mcobject.com/perst. Accessed: 15 March 2013
45. Object Refinery Limited: JFreeChart (2013). http://www.jfree.org/. Accessed: 15 March 2013
46. JasperSoft: iReport Desinger (2013). http://community.jaspersoft.com/project/ireport-designer. Accessed: 15 March 2013

Emergency Response Planning Information System

Victor Romanov(✉), Ilya Moskovoy, and Margarita Onokhova

Information Systems in Economics and Management Department,
Russian Plekhanov University of Economics, 117997 Moscow, Russia
{victorromanovl, marktravkin}@gmail.com,
onokhova@list.ru

Abstract. In case of spill, an essential point of restoring consequences of the unexpected situations in oil transportation system is appropriate planning to provide the sequence of adequate actions (plans) and support these actions by actors (personnel) and by resources. The plan may needs coordination with local, municipal, territorial, regional or federal departments and includes means of ground and air transportation, logistics support, spill reagent supply and repair equipment delivery.

The task of automated planning, which is classic in the domain of the Artificial Intelligence, becomes more and more urgent due to manifold oil leak cases. This problem is not new, the research is continuing for more than 50 years. Predicate logics, situation calculus, dynamic and integer programming models STRIPS, PDDL languages are applied. This problem is especially urgent for Russian petroleum and gas sectors of economy since the number of the oil leaks reached 28 thousand per year.

Keywords: Oil logistics · Oil spill · Pipeline rupture · Rapid response system · Situation calculus planning · Automated planning and unexpected situations

1 Introduction

Russian oil and gas sector (OGS) is one of the most important elements in the economy of the country and a significant part of the world's power supply system. Oil and gas sector is not only the main source of power and energy supplies but also a key element in the social system, providing stable social and economic status of the country. Capital investments in oil and gas sector made by all sources of funding account for nearly a third of the overall investment volume. From 2000 to 2012, oil production has increased by 150 %, which is nearly 200 million tons, which means that 2 million barrels of oil are being produced every day (Fig. 1) [1].

Oil production revenue accounts for 50 % of Russian Federation's budget, 1/3 of the country's GDP and 2/3 of the country's export (Fig. 2) [2].

The power strategy of Russian Federation for the period until 2020 implies annual oil production of 450–520 million tons. The country's energy sources are exported to a number of countries of near- and far-abroad. Germany, France, Italy, Poland, China, the Netherlands, Kazakhstan, Belarus, Ukraine among others, is the import buyers of

J. Barjis and R. Pergl (Eds.): EOMAS 2014, LNBIP 191, pp. 151–170, 2014.
DOI: 10.1007/978-3-662-44860-1_9

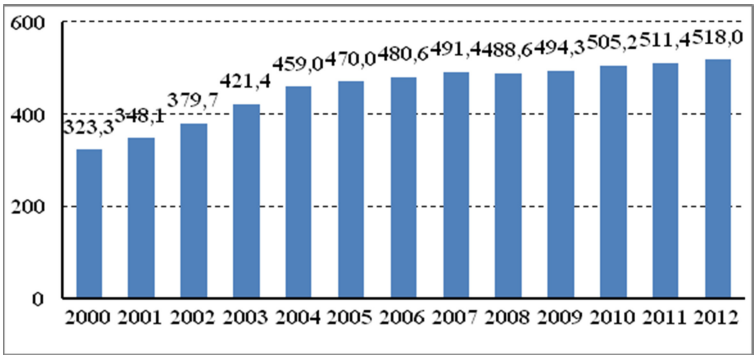


Fig. 1. Oil production growth (million tons)

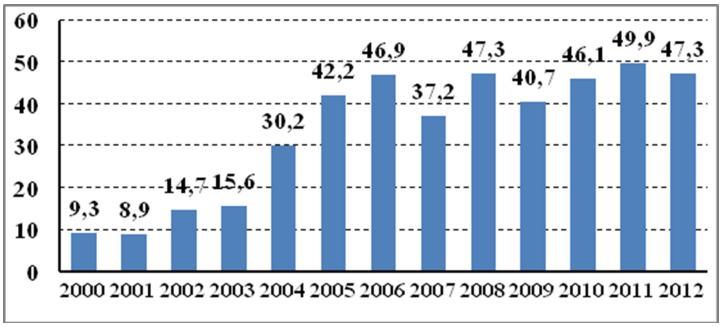


Fig. 2. Oil production revenue in the budget of Russian Federation

Russian oil products. Complicated oil logistics system is used in order to provide efficient delivery. Energy sources are delivered to the consumers by the pipeline systems and by marine terminals with the use of oil tankers, train with tank cars or automobiles. Oil logistics provides the delivery of oil products from the extraction stations to its consumers, which is based on the system of pipelines covering thousands of kilometers, and working without a break. Such complex system demands constant supervision and maintenance.

Oil logistics implies coordination and managing of oil transfer equipment along with unification of scientific, technological, economical and productive potentials of oil production facilities. It also deals with overall development of long distance main pipelines, which means the establishment of centralized managing, financial, operating, and investment systems.

2 Normative Documents of the Response on the Oil Spill

Fuel and energy sector supports country's economy; however it still has certain flaws, including the following:

- High degree of wear of fixed assets (in the oil industry, almost 60 % in the refining industry, −80 %) [3].
- Sometimes organizations do not have enough forces and means for liquidation of possible accidents associated with oil spills. At the same time, due to inadequate actions, costs of elimination of oil spill consequences increase many fold [4].
- Each year, no more that 2 % of the pipelines are repaired and replaced, and they are the most vulnerable according to statistics.

Oil and petroleum products are among the most widespread pollutants. Surface water of land - rivers and lakes in Russia is contaminated by oil everywhere to some extent. Petroleum products are found in almost any body of water (even in Lake Baikal). Crude oil enters waters mainly in the production, transportation and handling, primarily - as a result of leakage from pipelines [5].

Almost all oil companies inform the public, in one form or another, of the breakthroughs that have occurred in pipelines operated by them, and of the amount of oil leakage. However, in general, data on the number of pipeline ruptures in company reports are not fully represented, or are presented in a form that prevents estimating the total state of affairs with breakthroughs and making comparisons between other companies. In 2009, the total number of breakthroughs was about 26,000 in 2010, their number increased and reached 28 thousand (Fig. 3). Every year, several million tons of oil is poured into the environment in Russia [5].

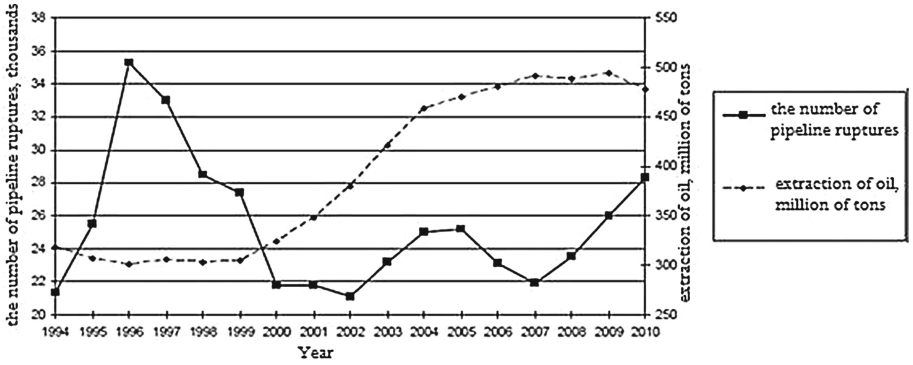


Fig. 3. Number of pipelines raptures in Russia

State agencies in Russia pay great attention to the consequences of such situations; they established very strict deadlines for oil spill response [6–11]. Depending on the amount of oil spilled and the territory, affected oil spills can be classified into the following groups of importance: minor local, major local, territorial, regional and federal.

- Oil spill of minor local importance involves no more than 100 tons of oil spilled and does not fall outside the oil station.
- Oil spill of major local importance involves from 100 to 500 tons of oil spilled and does not fall outside the perimeter of the municipal entity or the oil spill which involves up to 100 tons of oil and falls outside the borders of the oil station.

- Oil spill of territorial importance involves from 500 to 1000 tons of oil spilled and does not fall outside the perimeter of the constituent territory of the federation. The oil spills involving up to 100 tons of oil, which affect a body of water or seacoasts, are also applied to this category.
- Oil spill of regional importance involves from 1000 to 5000 tons of oil or a spill of from 500 to 1000 tons of oil, which falls outside the perimeter of the constituent territory of the federation. Oil spills of up to 100 tons of oil that may affect the water bodies and pollute the territory or the seacoasts of adjoining constituent territories of the Russian federation.
- Oil spill of federal importance involves either more than 5000 tons of oil or more than 100 tons of oil, which may get into the inland waters or the seacoasts. Oil spills, which fall outside the perimeter of the Russian Federation and oil spills spreading to adjoining states, are also related to this category.

Depending on the size of an oil spill in the sea, there are the following emergency situations:

- of local importance – oil spill involving the minimum amount of oil spills (which is estimated by the dedicated federal agency of the executive power in environmental protection field) up to 500 tons of oil;
- of regional importance – oil spill involving from 500 to 5000 tons of oil;
- of federal importance – oil spill involving the amount of oil which exceeds the limit of 5000 tons.

While planning the procedures aimed at the elimination of an oil spill incident it should be taken into account, that there local, regional and federal emergency scales which demands different amount of resources and staff necessary for successful elimination.

Oil spills, entailing a negative impact on the environment and violation of conditions of the population are considered emergencies according to Resolution of RF Government of 13 September 1996 No. 1094 «On classification of natural and man-caused emergencies» and require organization to eliminate them in accordance with the current legislation in this area of the Russian Federation.

Stringent standards for responding to such situations are provided by Russian Federation Government Resolution № 240 of 15.04.2002 “On organization of measures to prevent and eliminate oil spills and oil products in the Russian Federation”: when you receive a message about the oil spill and oil spill containment, localization time should not exceed 4 h – for the spill in the waters, 6 h for the spill on the ground since the discovery of oil spill or from the moment when the information about the spill is received.

The procedure for the rapid development of plans for responding to emergencies and a system of mutual exchange of information between organizations - participants of liquidation of oil spill was established. The plans developed must take into account:

- the number of forces and means necessary to deal with emergencies,
- whether the existing on-site power and resources are enough for the tasks of elimination and the need for professional rescue - rescue teams;
- the organization of interaction of forces and means;

- a system of mutual exchange of information between organizations - participants of liquidation of oil spill;
- priority actions upon receiving the signal of the emergency;
- geographical, navigational - hydrographic, meteorological and other features of the area of oil spill;
- provision for public safety and health care;
- schedule of operations to eliminate oil spills;
- organization of materials - technical, engineering and financial support of spill combating operations.

Events on localization and liquidation of oil spills are considered complete after the compulsory performing the following steps

- cessation of the discharge of oil and petroleum products;
- collecting the spilled oil and petroleum products to the highest attainable standard of due specification using special techniques;
- hosting of the oil and oil products collected for subsequent disposal, to prevent secondary pollution of production facilities and the environment.

The work can be considered completed when the permissible level of residual oil content and oil (or their transformation products) are achieved in soils, sediments of water bodies.

List of mandatory information for reporting of oil spills and oil in the territorial bodies of the Ministry of Natural Resources of the Russian Federation includes the following data:

- Date, time and place of oil spill;
- The source of pollution;
- Cause oil spill;
- Type and approximate quantity of spilled oil and petroleum products;
- Area of pollution;
- Purpose and type of use of the contaminated area (water area);
- Hydro meteorological setting;
- Threat of oil entering surface or groundwater;
- For industrial sites: threat of oil entering adjacent territories;
- Ability or inability to eliminate pollution by themselves in terms stipulated by the plan;
- Measures undertaken;

Stated above leads to the conclusion about necessity of creating a rapid response information system and decision support for combating pollution.

3 Existing Information Systems Oil and Gas Transportation Monitoring in Russia

We may study the projects of the information support systems in the leading facilities of oil and gas sector of the country. Integrated solutions for automation of gas

transporters and gas producers have been worked out in JSC 'Gazprom'. It provides 24/7 operations control, which is control, planning, calculations and overall performance optimization of the gas transmission network and its separate facilities, including the network of gas pipelines. Operations control is based on a 4-level hierarchical scheme which reflects the organizational structure of JSC 'Gazprom': Central production and dispatch department (CPDD, Moscow) – Operations and dispatch service of subsidiary companies – operations control center of the affiliates (main-pipeline field-operation department and others) – operations control stations on production facilities and sites [12] (Fig. 4).

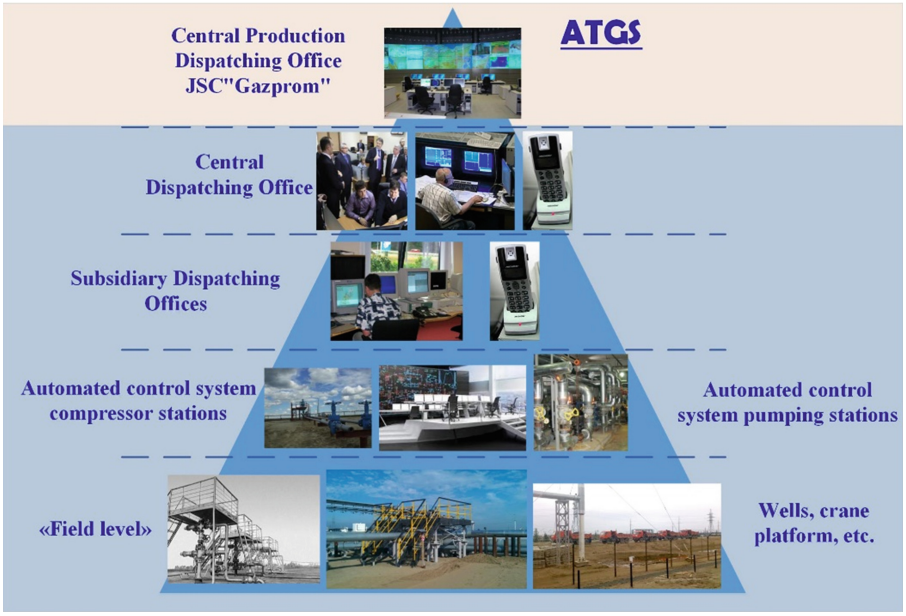


Fig. 4. Structure of the integrated managing system

Operations control managers of all 4 levels (except for the lowest one – local systems) can monitor the process online using the managing (SCADA – Supervisory Control and Data Acquisition). Operations control station of line control provides centralized data, which are collected by sub-system of online monitoring from local automatic systems, and creates informational base for solving problems on higher levels of control.

Automatic managing systems of JSC 'Surguneftegaz', such as 'OKO' informational system, are aimed at control and managing of the technological processes of oil drilling, producing, preparing and pumping. In order to provide automated accounting of the volume of gas shipped to the outside consumers, the data from gas disposition terminal is now being collected with the use of GSM [13].

'OKO' information system allows the online control of all the main technological processes, including drilling, producing, preparing, oil and gas transportation,

producing and consuming of the electrical energy (Fig. 5). The core of the information-computer complex of 'Surgutneftegaz' is the Central Computing complex (CCC). Corporate network for the transmission of data is based on the hierarchical and geographical structure of the company, and involves more than 300 facilities in West and East Siberia, Moscow, Saint Petersburg, on the North-West of Russia and in Krasnodar Krai. Specialists working with JSC 'Surgutneftegaz' monitor 'OKO''s working capacity 24/7.

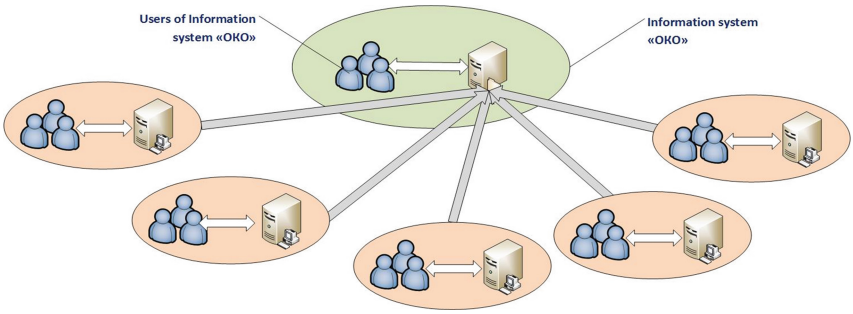


Fig. 5. Information system «OKO»

In order to develop current fields and find new ones, relevant cartographic information is needed which is being collected by means of aerial photography, satellite observations and laser scanning. One of the most relevant types of cartographic information necessary for dealing with a number of production problems are materials collected from large-scale aerial photography (Fig. 6).

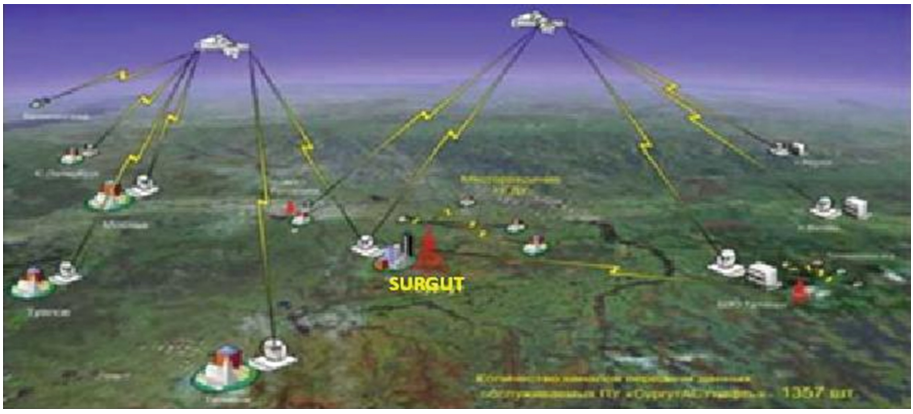


Fig. 6. Corporate network of JSZ 'Surgutneftegaz'

Literature [14] features the following data on Russian systems of monitoring oil and gas pipeline ruptures and registration of oil spills:

Parameter system «LeakSPY» (JSX «Energoavtomatica»)
Parameter system 'LeakSpy' is the first Russian system diagnosing oil spills from main oil pipelines using mathematical model. They control more than 16000 kilometres of main oil pipelines. 'LeakSpy' is a fully complete program package, which features a mathematical model of online situation of the pipeline.

«Appius LD» (engineering company «Kombit»)
«Appius LD» system collects, generalizes and analyzes all available technological data and, as a result, presents an emergency warning of the location of the oil spill. The operator is able to analyze the technological data presented by the system and take the final decision.

«Infra acoustic system of the pipeline monitoring» (LLC «TORI»)
If there is a possibility of an oil spill, it is located, and the system specifies the intensity of the spill. The results are presented online on a computer screen of the controlling operator of the pipeline, defining geographical position, technological schemes and altitude of the pipeline above the sea level.

From this brief review, one can see an abundance of the means of collecting and processing information on the operation of equipment and systems for the transportation of energy. Not being connected as a whole unified complex, all this set unfortunately does not fully satisfy the regulatory requirements in terms of speed of development of planning documents and in terms of closed loop information gathering and implementation. Once again, this underlines the importance of developing automation systems for planning operations of elimination of unexpected events in the oil logistics.

4 Related Works

Earlier, we have discussed the principles of construction rapid response system for oil logistics [18]. Now in this paper, we will consider the problem of the automated planning algorithm development and implementation.

Automated planning task, that is classic in domain of artificial intelligence, become more and more actual. Therefore, this task is not new: the research in this domain is continuing for more than 50 years with application of formal logic, dynamic and integer programming, STRIPS [15] and PDDL [17] languages.

Main task in the elimination of unexpected situation in the oil transportation system is adequate planning to ensure the creation of a sequence of actions (plan) and their support with staff and resources. Plans may include the use of ground and air modes of transport, repair and logistics means, delivery of reagents to neutralize the impact of oil spills on the environment. Depending on the size of the leak and the contaminated area, additional efforts may be required to coordinate with regional and federal agencies.

Development of such an information system should be based on a theory that includes the following concepts: the actor or agent (person or device), system status,

situations, actions. The conditions must be defined that make possible the execution of action, description of the changes arising from the implementation of actions, description of rules of transition from one state to another, as well as the properties of the system, which remain unchanged. Theory must define the concept of purpose and a plan as a sequence of elementary steps (actions), leading from the initial state to target.

Automated planning is associated with the development of automated procedures to generate and perform inferences to implement operations aimed at achieving a particular goal. The problem of automated planning is to develop a computer program, in which the plan is a sequence of actions that transforms an object from an initial state to a desired goal state. Planning is a generalization of the concept of scheduling. Planning is NP-complete problem, and at the same time PSPACE-complete [18].

When simulating the process of planning, we assume that there is a real object, of which the information system may receive data in real time from the staff as well as from the instruments, sensors, aerial data or systems, satellite surveillance, and that it is possible to make operational requests to obtain relevant information and perform scheduled operations.

Planning is the process by which the agent is looking for a way of action to achieve a certain goal. Here, the aim is understood as some formula, which should take the value "true". Planning is considered as the process of finding a sequence of actions that will transform the initial state of the object at a given target state. In formalizing the approach will be based on STRIPS [15].

5 Planning Formalization and Algorithm Development

5.1 Planning Task Formalization

The plans space consists of a set of all possible plans. The plan is presented by partially ordered set of actions. Given a language of mathematical: logic L-propositional calculus. The statement f in the language L is called fluents, a set of statements is denoted F. Let these statements describe the state of an object, for example: {"oil leak", "fire on the pipeline"}. All possible combinations of statements (fluents) form a set called "Boolean" and is denoted 2^F, thus the set of states - is a subset of 2^F, $S \subseteq 2^F$; in other words, each set of statements in the language L corresponds to some state.

We have the initial state of the object $s_0 \in S$ and a set (or one) final (target) states $s_g \subseteq S$ (there may be more than one target state). Further, we have some set of actions A. Action a ∈ A has a name, can have parameters a (ξ), where ξ - parameters, as well as a pre-condition pre (a), the positive post-condition add (a) and negative post-condition del (a).

For example, pre («start extinguishing the fire") = "closed plug", i.e. to extinguish fire you have to close the plug first. Action is called applicable in a situation S, if the precondition is satisfied. Postcondition defines the result of the action, i.e. describes the change in the situation.

As the situation is a set of true propositions, i.e. statements are true in this situation; the effect of the action is changing the truth value.

Positive postcondition adds a new situation and a true statement to the sense of statements, for example: add («to deliver fire truck on fire place") = "fire brigade delivered to the place of fire", this action can imply a change in the truth-value from 0 to 1.

Example of the del (a) the following: del («start extinguishing the fire") = "there is a fire on the pipeline." As the result of this situation is the quenching of fire, the approval of a pipeline fire is removed from the description of the situation or of its truth-value changes from 1 to 0.

Thus the action (action)-a collection of the following elements:

{a (ξ), pre (a), add (a), del (a)}. Thus the action of *a(ξ),* together with elements of *pre(a), add (a), del (a)* defines the rules of the system transition from one state to another.

In the simplest variant plan - it is a linearly ordered sequence of actions $\Pi = < a_1, a_2 \ldots a_n >$. Plan to resolve the problem situation (state s0 we understand how problematic situation) - is a plan in which, as a result of an action sequence $< a_1, a_2 \ldots_{an} >$, the object transitions from s_0 to target state s_g, i.e., passes the sequence of states $< s_0, s_1 \ldots s_n >$, and relevant actions $< a_1, a_2 \ldots a_n >$.

We define the function γ as a transition from state s to state s ' under the influence of action *a: s' = γ (s, a).* Then plan to resolve the problem situation is a sequence of actions and conditions such that $s_1 = \gamma (s_0, a_1)$, $s2 = \gamma (s_1, a2) \ldots s_n = \gamma (s_{n-1}, a_n)$ and $s_n = s_g$. Simply speaking, a plan to resolve problematic situations is a sequence of actions and states that leads from the original state to a destination. We call the plan that contains a minimal amount of action in sequence: "The optimal length".

In its simplest form, the planning system is based on the following assumptions:

1. System is static – no exogenous events that can change the state of the system.
2. System is fully observable – i.e. the results of the states observation coincide with the states themselves.
3. System is determined in the sense that the results of the action are uniquely defined and unique in advance.
4. System is finite – with a finite set of states and actions.
5. System is static – the behavior of the system is determined by the rules of transition states, and external events have no effect on its behavior.
6. Solution to the problem of planning is a finite linearly ordered sequence of actions.
7. Actions have no duration in the state space.
8. System does not allow for changes in the composition and transition from state to state in the planning process.

In [16] it is proposed to use integer linear programming method for finding the plan of the minimum length. However, the use of the finished software product type of ILOG CPLEX [21] has some drawbacks, such as impossibility to make algorithm modifications.

In this paper, in order to enhance the utility of the algorithm, we tried to eliminate the restrictions 6–8 by using branching, determined by business rules, the introduction of the cost and runtime duration of action, and possibility of interactive changes in the initial conditions and rules. We decided to stay on the interactive search algorithm on

the tree "first in depth." As the complexity of the algorithm is polynomial and increases with the states (the computation time and memory space), we believe that the algorithm is designed to solve problems of moderate dimension, which usually occurs in the early stages of the elimination of emergencies, just when the decision must be made especially quickly.

1. The program has a library of previous similar situations (case-based reasoning). It tries to find a similar situation in the library, by comparing the new input vector with the situations stored in the library repository by number of matching bits, and if the number of matched bits exceeds a certain threshold, it extracts the old similar situation from library.
2. The algorithm contains elements BRMS [18], i.e. the branch points (for example, depending on the volume of oil spilled), and the algorithm allows the operator to enter the data necessary to select the option of branching or remotely send a request and read the necessary data from a remote device that records a leak.
3. The algorithm is interactive in the sense that provides the user the ability to select data input method and criteria for selection of the optimal solutions.
4. If new data is received after the algorithm is initiated, there is the possibility to take into account the new circumstances.

For a model of the problem situation on the pipeline have been developed (presented here as truncated versions):

Set of propositions F– (Tables 1 and 2).
Set of actions – (Table 3).
Preconditions of the actions set pre(a) – (Table 4).
Positive post-conditions of actions add(a) – (Table 5).
Negative post-conditions of action del(a) – (Table 6).

5.2 Description of Algorithm

1. Define variables π - the current plan as a sequence of selected actions to this moment. Set up a variable Π, that will contain plan formed when the target vertex was reached.
2. Variable L is created, which stores the length of the generated plan, and n - count of the number of the acts committed. Counter value at the time of program start = 0. Variable (array) cs is created, which stores the states as they passed by algorithm.
3. Starting from the node s_0, scan the list of preconditions. The value of the variable L is high (much greater than all operations can be performed).
4. Form a list of active operations, i.e. actions that may be performed in this step (they satisfy the precondition in the state and set a value of the vector responsible for the passed state n = 0).
5. Save s_0 state in the cs variable.
6. Make depth first search, that is apply the fist action from the list of active operations.

Table 1. F-propositions set

Code	Content	Value in the initial state s_o	Value in the goal state s_g
f_1	The pipeline is in the working state	0	1
f_2	Oil through pipeline enters to customer	0	1
f_4	Operations Control Center received the message about the destruction of the pipeline due to explosion at the point with coordinates x_1, y_1	1	0
f_{16}	Repair crew is situated at the place permanent stationing	1	1
f_{17}	Repair crew is situated at the destruction place of the pipeline due to explosion at the point with coordinates x_1, y_1	0	1
f_{18}	The pipeline section is in the warehouse at the point with coordinates x_2, y_2	1	0
f_{19}	The pipeline section is at the destruction place of the pipeline due to explosion at the point with coordinates x_1, y_1	0	1
f_{28}	The pipeline plug at the destruction place of the pipeline due to explosion at the point with coordinates x_1, y_1 is closed	0	0
f_{29}	The pipeline plug at the destruction place of the pipeline due to explosion at the point with coordinates x_1, y_1 is opened	1	1
f_{30}	The destruction of pipeline eliminated	0	1
f_{32}	Replacing the damaged section of the pipeline performed	0	1
f_{44}	Repair crew returned at the place of permanent disposition	0	1
f_{46}	The customer is notified about oil supply suspension	0	1
f_{47}	The customer is notified about oil supply resumption	0	1
f_{57}	The customer is not notified about oil supply suspension	1	0
f_{58}	The customer is not notified about oil supply resumption	1	0
f_{59}	Quality of work complies with the environmental requirements of the standards	0	1
f_{50}	Oil comes to the customer	0	1
f_{60}	The pipeline section does not delivered from warehouse at the point with coordinates x_2, y_2 to the destruction place of the pipeline due to explosion at the point with coordinates x_1, y_1	1	0
f_{61}	Damaged pipeline section did not cut and changed by mean welding on the new one, just delivered	1	0
f_{63}	All works to eliminate emergency completed, pipeline again in working condition	0	1

Table 2. Truncated set of propositions *F*.

Code	s_0	s_g
f_4	1	0
f_{16}	1	1
f_{17}	0	1
f_{18}	1	0
f_{19}	0	1
f_{29}	1	1
f_{32}	0	1
f_{46}	0	1
f_{47}	0	1
f_{59}	0	1
f_{63}	0	1

Table 3. Truncated set of actions

a_1	Clouse pipeline plug
a_3	Direct repair crew with equipment to the emergency point
a_{12}	Notify the customer about oil supply suspension
a_{13}	Notify the customer about oil supply resumption
a_{14}	Monitor the quality of work and it correspondence to the ecological standards and requirements
a_{15}	Open the pipeline plug
a_{16}	Deliver the pipeline section from warehouse to the emergency place
a_{17}	Repair crew to cut damaged part of the pipeline and change it by welding to the new one just delivered from warehouse
a_{19}	Repair crew return back to the place of permanent disposition
a_{22}	Due to the completion of work to reintroduce pipeline into operation

Table 4. Precondition Set *pre(a)*

Condition	Action
$f_4 \wedge f_{29}$	a_1
$f_4 \wedge \neg f_{29}$	a_3
$f_4 \wedge \neg f_{29} \wedge f_{17}$	a_{12}
$f_{17} \wedge f_{19} \wedge \neg f_{32}$	a_{17}
f_{18}	a_{16}
$f_{17} \wedge f_{19} \wedge \neg f_{29} \wedge \neg f_{32}$	a_{17}
$f_{17} \wedge f_{19} \wedge \neg f_{29} \wedge \neg f_{32}$	a_{17}
$f_{29} \wedge f_{32} \wedge \neg f_{63}$	a_{15}
f_{32}	a_{14}
f_{46}	–
f_{47}	a_{22}
$f_{59} \wedge \neg f_{63}$	a_{24}
f_{63}	–

Table 5. Set of positive post-conditions *add(a)*

Actions	Propositions
a_1	f_{29}
a_3	f_{17}
a_{12}	f_{46}
a_{13}	f_{47}
a_{14}	f_{59}
a_{15}	f_{63}
a_{16}	f_{19}
a_{17}	f_{32}
a_{22}	f_{29}
a_{24}	f_{47}

Table 6. Set of negative post-conditions *del(a)*

Actions	Propositions
a_1	f_{29}
a_{12}	f_4
a_{16}	f_{18}
a_{15}	f_{17}

Table 7. F-difference

Code	s_0	s_g	$D = (\neg s_0 \wedge s_g) \vee (s_0 \wedge \neg s_g)$
f_4	1	0	1
f_{17}	0	1	1
f_{18}	1	0	1
f_{19}	0	1	1
f_{29}	1	1	0
f_{32}	0	1	1
f_{46}	0	1	1
f_{47}	0	1	1
f_{59}	0	1	1
f_{63}	0	1	1

7. Compute vector F–difference between current and goal states (Table 7).
8. Algorithm refers to F-difference and looks up in turn lines vector D, starting from the top, until it finds the first one with value 1. It selects a value in the code column and searches in the table pre (a) in an attempt to find an action that fulfilled the preconditions and that would eliminate the distinction between them.
9. The algorithm checks all the conditions on satisfiability in a given state, first action for which the precondition is satisfied, places to the list π, into variable L places and sums depending on the option, either one (the number of actions in the current plan), or the time or cost of operation).
10. The algorithm selects the action stored in π and accesses the tables add(a) and del (a) corresponding to the selected operator and produces post-condition operations on the current state.
11. The algorithm compares the newly obtained state s_1 to the target state s_g, by performing an equivalence negation. If the state does not match, repeat the steps above for the state s_1 and put again the founded action into π - current plan. With another founded action algorithm reiterates tables add(a) and del(a) operations.
12. Algorithm repeats the same sequence of actions has not yet come to a state that does not satisfy any preconditions s_n and the corresponding vector F-difference, which did not comply with any of the preconditions, hence none of action can be performed (dead-end node). Then the algorithm returns to the previous state s_{n-1} and the previous queue of action.
13. Returning to a previous state, the algorithm selects a new action and repeats all the cycles until it reaches the next dead-end node or target node. As a result, the algorithm retains all the plans ended achievement target node and matching these plans length (or cost/time). Completed blind nodes or nodes, all paths of which were passed are marked as traversed.
14. The algorithm works as long as have completed all the vertices, including the primary. At the final stage of the algorithm chooses the plan with the smallest length (time/cost).

Another version of the algorithm is represented in the form of flowchart. We divided it into two parts in order to simplify its reading (Figs. 7, 8).

5.3 Software Development

Software is designed in the development environment Eclipse. Main core of the program in C#, data format JSON. Program interface is designed with CSS. The interface frames are presented at Figs. 9, 10, 11.

The software has the ability to test the algorithm specifying parameter values and operating conditions can be used detailed debug mode, the best solution is based on the search criteria, optimal solution - it is possible to specify the amount of oil spilled out.

Program allows viewing and editing data in tables, actions, fluents and conditions with different statements description.

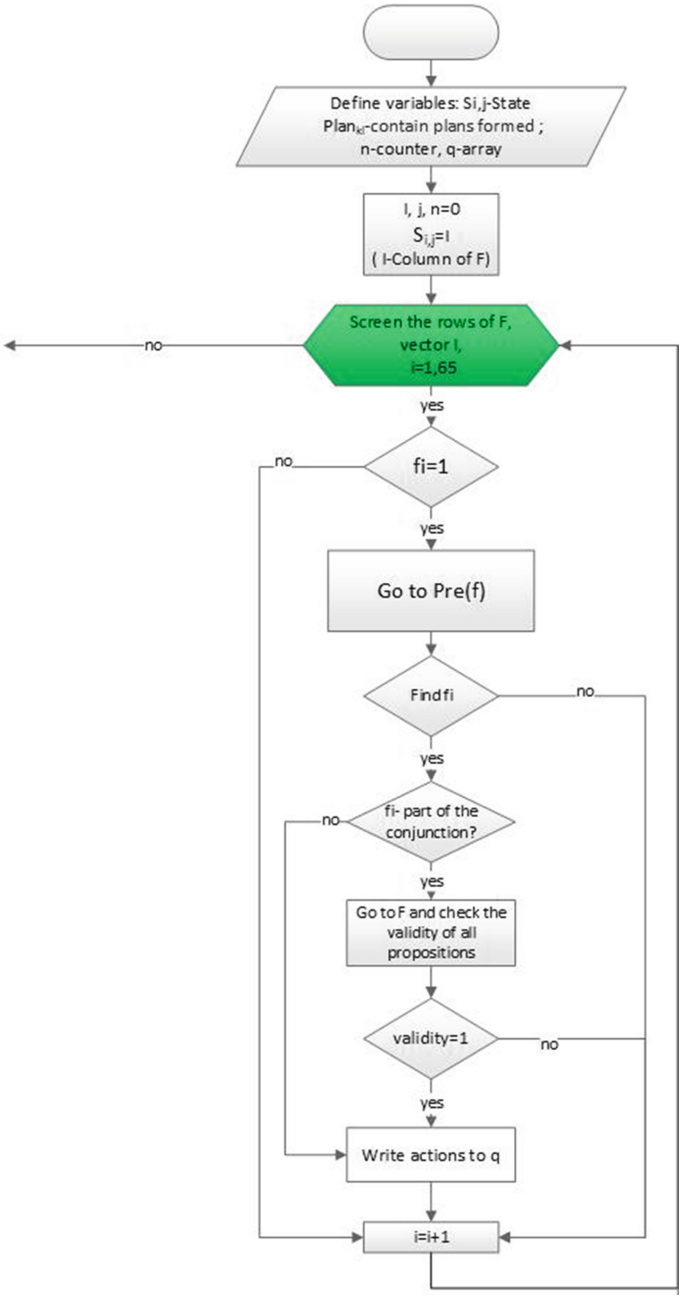


Fig. 7. Algorithm, part1

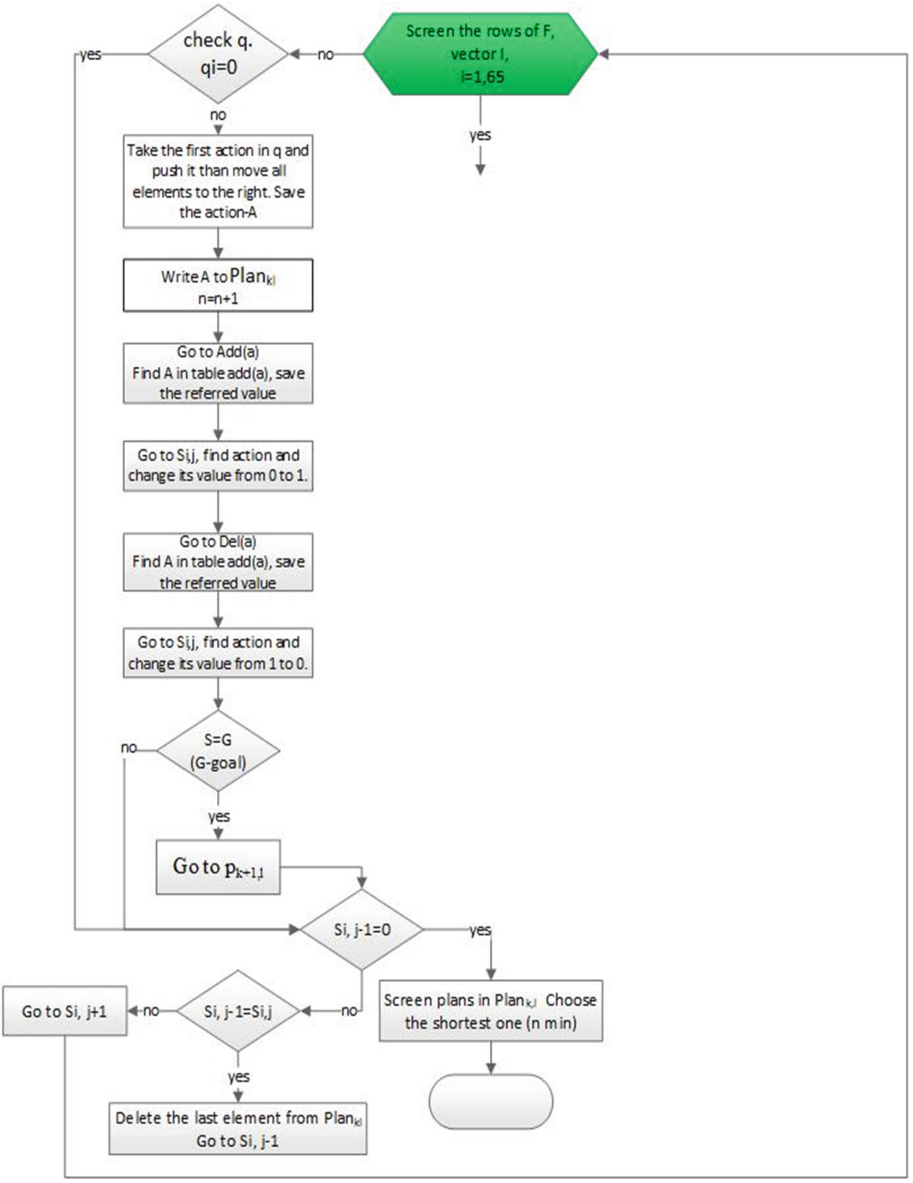


Fig. 8. Algorithm, part 2

Software allows you to view specific solutions step by step, as well as have the opportunity to play a decision on the steps, stopping at each stage to assess the correctness of the algorithm logic. Software based on described algorithm is in active development mode.

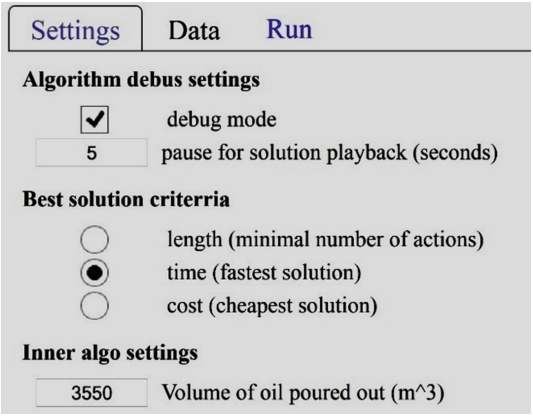


Fig. 9. Main Interface

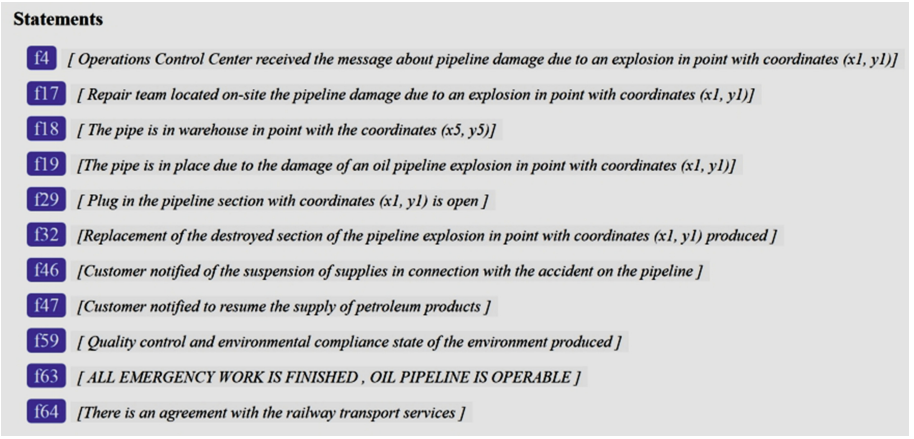


Fig. 10. Table –statements

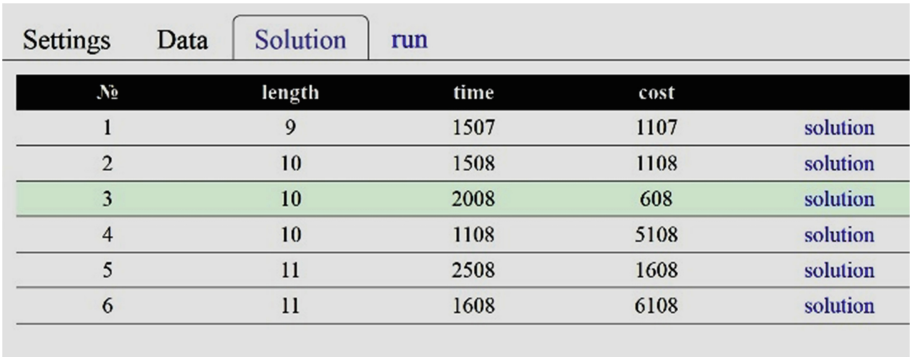


Fig. 11. Decisions algorithm criteria demonstration optimal solution by time.

6 Conclusion and Future Works

Basing on the existing solution of the automated planning problem, we developed more flexible algorithm and software that can be installed as a component of more general rapid response system, aimed on the oil logistics emergency situations The algorithm can generate plans with optimality criteria, chosen by user in dialogue process. We are planning to perform testing of algorithm for estimating its computational complexity and restrictions. We are confident that the implementation of the developed software we will significantly reduce economic losses and environmental damage.

References

1. Report of Minister of Energy A.V. Novak on the National Oil and Gas Forum, 2013 (in Russian). http://minenergo.gov.ru/press/doklady/14507.html
2. Ministry of Finance of the Russian Federation, and gas revenues of the federal budget (2012). http://www.minfin.ru/ru/reservefund/accumulation/
3. Russia's Energy Strategy until 2030 (approved by the RF Government Decree of 13 November 2009 N 1715-p) (in Russian)
4. Rybakov, S.N., Meyer, S., Tarasov, A.G.: Improving the legal regulation of the prevention, containment and elimination of oil spills (in Russian). http://cetek.ru/informaciya-i-analitika/sovershenstvovanie-regulirovaniya
5. Blokov, I.P.: Greenpeace Russia, Overview of pipeline breaks and volume of oil spills in Russia, 2012. 12 pp (in Russian). http://www.greenpeace.org/russia/Global/russia/report/Arctic-oil/Oil_spills.pdf
6. Russian Federation Government Resolution № 240 of 15.04.2002 "On organization of measures to prevent and eliminate oil spills and oil products in the Russian Federation". http://www.56.mchs.gov.ru/…/915165e3f2522f6121ebbc7999392c66.doc
7. Ministry of Emergency Situations Order number 621 of 28.12.2004 "On approval of rules for the development and approval of plans for prevention and liquidation of oil spills on the territory of the Russian Federation" (in Russian). http://www.atgs.ru/articles/?aid=df646f6976c5e2da72b421b82dc11cc9
8. Regulations on the classification of natural and man-made disasters, approved by the Government of the Russian Federation dated 13 September 1996 № 1094 (in Russian). http://www.mchs.ru/law/index.php?ID=4262
9. Basic requirements for the development of plans for the prevention and liquidation of emergency oil spills (admitted by. RF Government Decree of August 21, 2000 N 613). http://www.center-ps.ru/rostehnadzor/dev_document/plarn/
10. Russian Federation Government Resolution № 613 of 21.08.2000. (as amended. RF Government Decree of 15.04.2002 № 240) "On urgent measures for the prevention and liquidation of emergency oil spills" (in Russian). zakonprost.ru/content/base/39087
11. MNR Order number 156 of 03.03.2003 "Guidelines for the definition of low-level oil spill for assignment to emergency flood emergency" (in Russian). zakonprost.ru/content/base/part/311…копия
12. Integrated automation solutions for gas transportation and gas companies of JSC "Gazprom" (in Russian). http://www.atgs.ru/articles/?aid=df646f6976c5e2da72b421b82dc11cc9
13. National industrial magazine, "Oil & Gas Vertical" Surgutneftegas, № 17, 2012г., 100 pp. (in Russian)

14. Mishkin, G.B.: Summary of leak detection systems Russian producers. J. Young Sci. № **2. T.1. — C**, 41–47 (2011). (in Russian)
15. Fikes, R.E., Nilsson, N.J.: STRIPS: A new approach to the application of theorem proving to problem solving. Artif. Intell. **2**, 189–208 (1971)
16. van den Briel, M.H.L.: Integer programming approaches for automated planning. A Dissertation Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy Arizona State University (August 2008). http://rakaposhi.eas.asu.edu/menkes-dissertation.pdf
17. Fox, M., Long, D.: PDDL2.1: An extension of PDDL for expressing temporal planning domains. J. Artif. Intell. Res. **20**, 60–124 (2003)
18. Romanov, V., Moskovoy, I., Grigoryeva, K.: Response information system on oil logistics unexpected emergency situations. In: Proceedings of the 2013 Summer Simulation Multi-Conference (SummerSim'13), 7–10 July 2013, Toronto, ON, Canada

On Compatibility Analysis of Inter Organizational Business Processes

Zohra Sbaï[1(✉)] and Kamel Barkaoui[2]

[1] Université de Tunis El Manar, Ecole Nationale d'Ingénieurs de Tunis, BP. 37 Le Belvédère, 1002 Tunis, Tunisia
zohra.sbai@enit.rnu.tn

[2] Conservatoire National des Arts et Métiers, Rue Saint Martin, 75141 Paris, France
barkaoui@cnam.fr

Abstract. Distributed systems are promisingly used in context of cross organizational enterprises. To develop these systems, Web services form an essential and wide accepted technology because they provide a significant level of platform independency and autonomy. Services are generally designed to interact with other services to form larger applications. In order to interact correctly with each other, Web services have to be compatible. This include not only composability of the involved services but also the correct execution of the overall composite service. In this context, we suggest in this paper to study the compatibility of Web services in different aspects and to provide a formal approach to characterize and verify this property. This approach is straightforward since it combines Petri nets and model checking techniques.

Keywords: Model checking · Inter-organizational business processes · Open workflow-nets · Compatibility · NuSMV · CTL

1 Introduction

During the last years, the emergence of complexity in organizations increases the need for organizations to collaborate. In fact, cross-organizational companies are working to bring together common objectives. These companies invest in inter-organizational systems such as electronic data interchange, supply chain management and customer relationship management in order to improve the quality of inter-organizational relationships. To make these relationships more successful, information technology played an important role to integrate and automate inter-organizational business processes. Service Oriented Architecture (SOA) [1], as a realization of business process management, is keen on supporting these processes within enterprises and between commercial partners [2,3]. The tasks of these processes are performed by services. Today, companies use the Web as an effective and efficient means not only to sell products but also as a platform to provide these services [4].

J. Barjis and R. Pergl (Eds.): EOMAS 2014, LNBIP 191, pp. 171–186, 2014.
DOI: 10.1007/978-3-662-44860-1_10

Web services use is becoming increasingly important, especially in applications based on distributed systems. Their main purpose is to enable heterogeneous applications, different platforms and programming languages to communicate via internet by exchanging XML messages through the SOAP communication protocol. In the majority of service oriented scenarios, a single service cannot satisfy users' objectives. For instance, when a requested functionality cannot be offered by a single service, some of the existing services can be composed to provide the same functionality. Hence, Web Services Composition (WSC) drew the attentions of many researchers and industrials.

From a software engineering point of view, the construction of new services by composing existing services raises a number of challenges. The most important is the challenge to guarantee a correct interaction of independent, communicating pieces of software. In deed, due to the message sending nature of service interaction, many delicate errors might take place when several services are put together (unreceived messages, deadlocks, contradictory behaviors, etc.). So far, it is necessary to ensure the proper functioning of each service involved in the composition as well as their ability to be composed and their good communication and the validity of their messages exchange.

In this context, we are interested by the verification of Web services composition compatibility as a main feature to ensure a correct composition and to prevent eventual errors from occurring.

Due to the solid theoretical basis of formal methods, we choose to adopt a model checking [5] based approach to model and analyse WSC. In particular we propose first to model WSC by means of a Petri net class named open workflow nets. We model each Web service by an open workflow net possessing interface places used to communicate with other Web services. In this way, we guarantee the conversation between the Web services interacting with each other. The conversation considered here is involved through the two well known behaviors: operational and control. An operational behavior is a behavior specific to each partner according to its business logic. A control behavior describes the general behavior of any process related to composite Web services.

Then, aiming to detect and correct errors as soon as possible, we propose an analysis method of the overall Petri net modeling the WSC based on the well known model checker NuSMV [3]. This approach allows the simulation and verification of the behavior of a model at design time, helping thus the increase of service compositions correctness. Our method is based on two fundamental phases which are specifying the behaviors of the composition and formalizing the requirements to be verified. In the former phase, we begin by translating the WSC model, consisting on a composition of the different open workflow nets, to a corresponding SMV file specifying the different Web services and their conversation. In the latter phase, we exhibit how to formalize the requirements to be checked in Computation Tree Logic (CTL). The obtained formula together with the SMV model will be passed to NuSMV model checker which responds by yes if the requirements are verified by all the possible executions and by no if not. If the desired property are violated, we can show a counter example which

consists on an execution not satisfying the property, leading thus to study and correct the error's source.

The rest of this paper is organized as follows. Section 2 is dedicated to the presentation of our method of modeling Web services composition. The formal verification of the proposed model is then tackled in Sect. 3. The existing approaches of WSC modeling and verification are discussed in Sect. 4. We outline our approach and announce future directions in Sect. 5.

2 Formal Approach of WSC Modeling

We present in this section our method of WSC modeling and verification based on a fully automatic formal method which is model checking. Our choice is based on the ability of model checking methods to explore exhaustively all states of the model and to verify if a certain requirement is guaranteed in all possible executions. Moreover, This formal method is characterized by a strong point which consist on producing a counter example in case of error which facilitates the characterization of the error' source and its correction. Model checking is thus an automated verification technique for proving that a model satisfies a set of properties specified in temporal logic. It is situated at the design phase, allowing thus to find design bugs as early as possible and therefore to reduce the cost of failures. Several model checking tools, named model checkers, have emerged such as NuSMV [3], ProB [20], BLAST [16] and SPIN [17].

NuSMV is a model checker based on the SMV (Symbolic Model Verifier) software, which was the first implementation of the methodology called Symbolic Model Checking described in [15]. This class of model checkers verifies temporal logic properties in finite state systems with implicit techniques. NuSMV uses a symbolic representation of the specification in order to check a model against a property. Originally, SMV was a tool for checking CTL properties on a symbolic model. But NuSMV is also able to deal with LTL (Linear Temporal Logic) formulae and SAT-based Bounded Model Checking. The model checker allows to write properties specification in CTL or LTL and to choose between BDD-based symbolic model checking and bounded model checking [12].

We thus proposed a method for validating and verifying the composition of Web services using the NuSMV Model Checker. For this, we have to provide first an SMV specification of the WSC and than the requirements to be verified have to be formulated in LTL or CTL.

For the first stage, which is specifying WSC in SMV, on the one hand dealing directly with SMV is a task that is not wield by all process designers. In addition, there are no tools which permit to export or import SMV specifications for analysis. On the other hand, open workflow nets form a sub class of Petri nets which has been used successfully in WSC modelling. Hence, we propose to define a WSC model in terms of open workflow nets and then to translate this model in an SMV specification.

In this way, our approach has a double strength. In fact, we benefit first from the expressive power of Petri nets as well as the coverage of workflow and

service interaction patterns [2]. Second, we enhance the analysis of WSC, based on Petri nets modeling, by a model checking verification method, which ensures an exhaustive analysis among all the possible executions. We emphasize more the possibility of generating a counter example in case some requirement is not satisfied, and thereby knowing the source of the problem.

We summarize our approach in Fig. 1. We start by modeling WSC by a class of Petri nets named open workflow nets. Then, we translate the obtained Petri net to an SMV specification. This specification together with a property formulated in Computation Tree Logic (CTL) will be passed to NuSMV model checker.

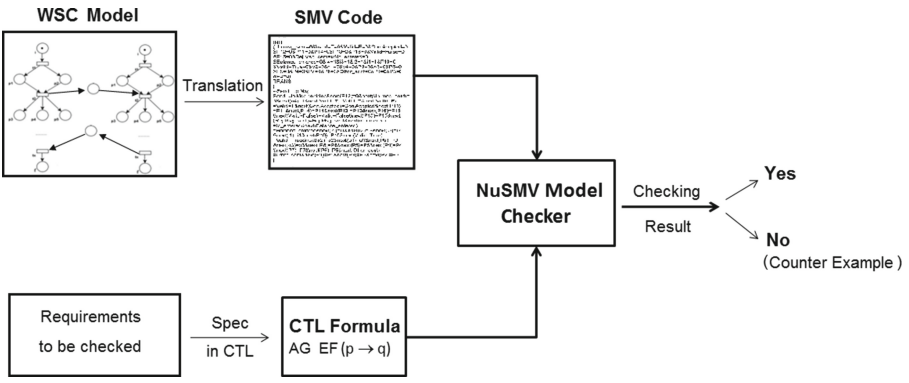


Fig. 1. Principle of model checking WSC

In the rest of this section, we present first the WSC model based on open workflow nets, then we expose a translation of this model to SMV.

2.1 WSC Model Based on Open Workflow Nets

As mentioned above, we begin with modeling the WSC by means of open workflow nets [22,23]. Open workflow nets are mainly an extension of workflow nets (WF-nets) to model workflow processes which interact with other workflow processes via interface places. Simple WF-net is a result of Petri nets' application to workflow management. The choice of Petri nets is based on their formal semantics, expressiveness, graphical nature and the availability of Petri nets based analysis techniques and tools.

A Petri net is a 4-tuple $N = (P, T, F, W)$ where P and T are two finite non-empty sets of places and transitions respectively, $P \cap T = \emptyset$, $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation, and $W : (P \times T) \cup (T \times P) \rightarrow \mathbb{N}$ is the weight function of N satisfying $W(x, y) = 0 \Leftrightarrow (x, y) \notin F$. If $W(u) = 1\ \forall u \in F$ then N is said to be ordinary net and it is denoted by $N = (P, T, F)$. For every node $x \in P \cup T$, the set of input nodes of x is defined by ${}^\bullet x = \{y | (y, x) \in F\}$ and the set of output

nodes is denoted by $x^\bullet = \{y|(x,y) \in F\}$. We refer the reader to [7], for more Petri nets notations used in this paper.

A Petri net which models a workflow process is said to be a workflow net (WF-net) [1]. A Petri net $N = (P, T, F)$ is a WF-net iff N has one source place i named initial place (containing initially one token) and one sink place f named final place. In addition to this characteristic, in a WF-net, every node $n \in P \cup T$ is on a path from i to f.

To communicate with other processes, a WF-net is augmented by interface places. These places will ensure the interaction between the different processes (said to be partners) involved in a composition. They can, for instance, be used for messages sending between partners or simply as routing conditions between tasks of different processes.

We are thus using open WF-nets (oWF-nets) [6,19], which generalize the classical WF-nets by introducing interface places for asynchronous communications with partners. Hence, we model WSC by a set of oWF-nets communicating via interface places. These places connect only transitions of different Web services.

Definition 1. *Un oWF-net N is a tuple (P, T, F, I, O) with:*

- *(P, T, F) is a WF-net with an extended flow relation F,*
- *I is a set of places representing input interfaces which are responsible for receiving messages from other processes: ${}^\bullet I = \emptyset$.*
- *O is a set of places representing output interfaces that are responsible for sending messages to other processes: $O^\bullet = \emptyset$.*
- *I, O and P are disjoint. I and O connect transitions of different partners.*
- *$F \subseteq ((P \cup I) \times T) \cup (T \times (P \cup O))$ is the flow relation,*

Now, we define the composite net obtained while composing the oWF-nets corresponding to different partners. Thus, the composed system N of *nbs* oWF-nets $N_1 ... N_{nbs}$ consists of all oWF-nets which share interface places, i.e. every place of N which is an input interface in a WF-net is also an output place in another oWF-net in the composition. Trivially, N can be seen as a Petri net with *nbs* input places and *nbs* output places. The marking of a composite net N is a vector of $\mathbb{N}^{(P \cup I \cup O)}$ such that for each place $p \in P \cup I \cup O$. Knowing that $M(p)$ is the number of tokens in p, M_p is used to denote a marking for which $M(p) = 1$ and $M(q) = 0\ \forall q \in (P \cup I \cup O) \setminus \{p\}$. The initial marking of the composite net N is $M_0 = \sum_{s=1}^{nbs} i_s$.

As a case study, we propose to present a Web portal treating the credit requests of the bank clients. Different steps are involved: a preliminary assessment by an employee, followed by an evaluation by a supervisor before offering the customer a credit agreement, if his demand is accepted.

At the beginning of the process, the client connects to the portal by entering his name and password. He is then invited to provide the desired amount of credit, guarantees and balance. The service checks the balance through a validation service, and if the balance is not validated, they asked him to provide it again. When the application is completely filled by the client, the service ranks the request in the list of tasks that bank employees have to perform.

Then an employee withdraws the request from the list of tasks and performs the evaluation.

The evaluation consists of two parts: a private part (exclusive use of the bank) and a public part available to the customer. The private evaluation includes a rating assigned to the client and other useful information. The public evaluation make the decision on the request, which is either the provision of the bank or the reasons for the refusal. The decision may be to reject the request, to accept it or to ask the customer to reformulate the request.

The overall process is modeled by the composite net drawn in Fig. 2 in which the gray circles represent the interface places.

2.2 SMV Translation

The Petri net model of WSC presented above will be translated to SMV code in order to be checked by NuSMV. A system description in the SMV input language is a collection of modules, each of which represents a component of the system. The overall behavior of the system is the behavior of the main module. Modules can be parameterized, so multiple instances can be generated with different parameters. A module description consists of the header $MODULE$ and the following declarations:

- VAR contains the declarations of the variables in the module;
- $ASSIGN$ contains assignments to variables;
- $DEFINE$ is used to make descriptions shorter and better readable;
- $SPEC$ gives a system specification as a CTL formula;
- $FAIR$ specifies a fairness constraint; and finally
- $TRANS$ and $INIT$ declarations which are an alternative to the $ASSIGN$ declaration and make it possible to specify the transition relation and the initial states directly as boolean expressions.

We present now the translation of a WSC model in SMV, we declare in the section VAR:

- nbs tables (a table PLS_s for each service s) which contain the marking of places of different services except the interface places,
- nbs tables (a table TRS_s for each service s) containing the transitions firing,
- a table I containing the marking of interface places of the composite net.

These tables will be initialized in the section $INIT$. All the elements of these tables will be initialized to 0 except those corresponding to initial places which will be initialized to 1.

After proposing a static description of the net in the SMV file, let us propose an implementation of its dynamic behavior. Each transition t of each oWF-net s may fire by consuming a token from each place and/or interface place of ${}^\bullet t$. This results in incrementing by 1 the cell of TRS_s corresponding to t, and adding a token in each place or interface which is in $t^\bullet$.

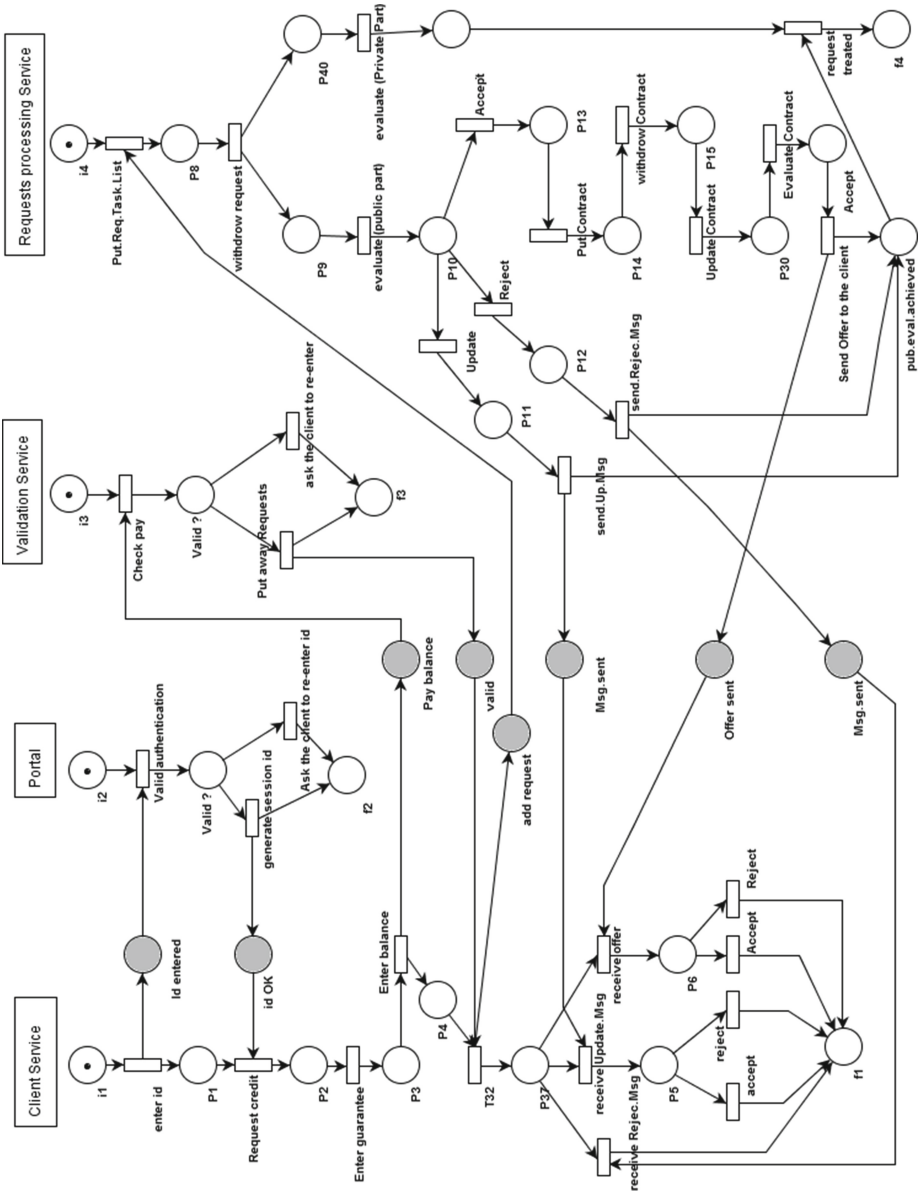


Fig. 2. The overall credit loan example of WSC

This is expressed in the $TRANS$ section. The actions of incrementing are ensured only if a condition on marking of pre places is ensured. This condition is expressed via boolean macros defined in section $DEFINE$. The states evolution is ensured only if the necessary macros return $True$ value.

An example of a simple WSC and its SMV code is given in Fig. 3.

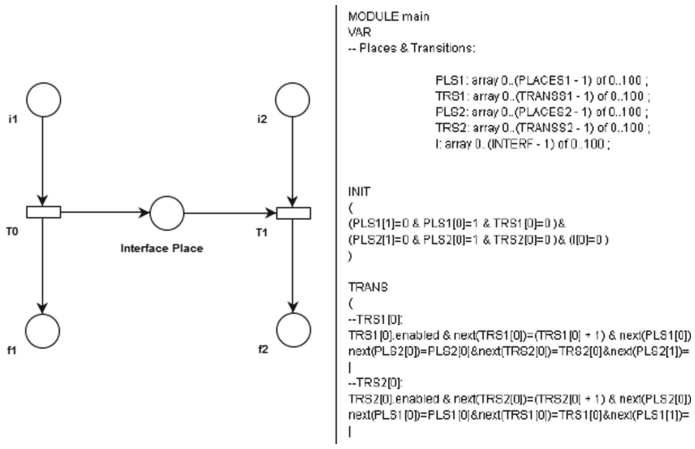


Fig. 3. A simple WSC and its SMV code

3 Compatibility Characterization and Analysis

Once the SMV model is ready, we pass to check if the requirements are satisfied by this model. It is to ask the question: does the system S satisfy the property P? Formally, this is expressed by the formal statement: $M_S \models \varphi_P$ where M_S is the system's model, φ_P is the property formula expressed in Temporal Logic and $\models$ is a satisfaction relation. NuSMV checks properties formulated with both Linear Temporal Logic (LTL) and Computation Tree Logic (CTL) [4]. Among these two well known temporal logics, we chose CTL since it permits the use of universal as well as the existential path quantifiers.

A CTL formula f is built from atomic propositions corresponding to variables, boolean operators such as $\neg$, $\wedge$, $\vee$, $\rightarrow$, and temporal operators. Each temporal operator consists on two parts: a path quantifier (A or G) followed by a temporal modality (F, G, X, U). The temporal operators are interpreted relative to a current state s. The path quantifier indicates if the temporal operator expresses a property that should hold on all paths starting at s (denoted by the universal path quantifier A), or at least on one of these paths (denoted by the existential path quantifier E).

A CTL formulae is inductively defined as follows:

$\varphi ::= \top|\bot|p|\neg\varphi|\varphi \wedge \varphi|\varphi \vee \varphi|\varphi \rightarrow \varphi|AX\varphi|EX\varphi|AF\varphi|EF\varphi|AG\varphi|EG\varphi|A[\varphi \sqcup \varphi]|E[\varphi \sqcup \varphi]$

Due to the expressiveness of the Computation Tree Logic, we can verify processes interactions and especially we can check that the conversations between different partners have been ensured as planned. But before checking the correctness of a composition, let us check if the processes involved in the composition are compatible.

From a behavioral point of view, two (or more) services are said to be compatible if they can interact correctly: this means they can exchange the same type of messages and the composite net does not suffer from the deadlock problem. This leads us to distinguish between a syntactic compatibility which concerns the verification of the interfaces conformance and a semantic compatibility which is related to check the absence of deadlocks. We investigate in this paper in the analysis of the semantic compatibility.

According to [9,11,21], the compatibility is closely related to the absence of deadlock in the composite system. They considered that two oWF-nets are compatible if they can reach their final states. Deeper, in [8], the authors defined three classes of compatibility: weak compatibility, compatibility and strong compatibility. The definition of each type is given as follows:

- Weak compatibility: A composite net N is weakly compatible if it doesn't suffer from deadlock, meaning N is deadlock-free.
- Compatibility: A composite net is compatible if N is already weakly compatible and furthermore, it verifies the proper termination.
- Strong compatibility: A composite net is strongly compatible if it is compatible and quasi-live.

Barkaoui et al. [8] structurally verified these three classes of compatibility while we propose an approach to formally verify them.

We focus here on expressing in CTL the three types of compatibility properties: weak compatibility, compatibility and strong compatibility.

Let us consider the following:

- nbs: is the number of services,
- nbp: is the number of places in a given service,
- nbt: is the number of transitions in a given service,
- nbi: is the number of interface places available in a composition,
- index-of-fs: is the index of the output place f_s of the service s.

3.1 Weak Compatibility

To assert weak compatibility of oWF-nets, we should just check the absence of deadlock in the composite net. The composite process is deadlock-free if there

is a transition allowed for any marking, except for the final marking, where the process is supposed to be achieved.

$$N \text{ is deadlock-free} \quad \Leftrightarrow \quad \forall M \in [M_0\rangle, \exists t \in T, M[t\rangle$$

Here, it suffices to verify that no deadlock is encountered until the final places are reached (i.e. will be marked). Reasoning in temporal logic, we can express this property by this assertion: "In all the executions, no deadlock is encountered until we reach a marking in which all final places are marked". So, in CTL, the expression of the weak compatibility is given as follows:

$$A(not\ deadlock\ U\ P)$$

Where P is a proposition on marking which asserts the following:

$$\overset{nbs}{\underset{s=1}{\&}}(PLS_s[index-of-fs] > 0)$$

And *deadlock* is a boolean variable defined in $TRANS$ section of the SMV specification which is only $True$ if all places and transitions cells remain unchangeable.

This states that for all the executions, from every state, we will attend a state in which all the final places of the involved services are marked. We are not interested, at this stage, if the other places (internal and interface places) are marked or not. However, in the next class of compatibility, we sink for a final marking which ensures a proper termination.

3.2 Compatibility

Having expressed the weak compatibility, we focus here on the expression of the property of proper termination in CTL. This property allows the process to complete its execution in any case, but at the time of termination, all places of oWF-nets must be empty except for the final places where each of which must have exactly one token.

Verifying the proper termination consists in checking the existence of a marking M for which all places are empty except the final ones. The expression of this property is given as follows:

$$\forall M \in [M_0\rangle : M(f_s) \geq 1\ \forall s \in \{1,..,nbs\} \quad \Rightarrow \quad M = f_1 + .. + f_{nbs}$$

In CTL, we can state it as: "a marking in which only each final place contains one token will always be potentially achievable". This property is formulated as follows:

$$AG\ EF\ P$$

Where P is a proposition on marking which states that:

$$\overset{nbs}{\underset{s=1}{\&}}\left((PLS_s[index-of-f_s] = 1) \quad \& \quad (\overset{nbp}{\underset{p=1}{\&}}(PLS_s[p] = 0))\right) \quad \& \quad \overset{nbi}{\underset{i=1}{\&}}(I[i] = 0)$$

This amounts to finding a state for which each final place of the processes involved in the composition contains exactly one token and all the other internal places and interface places are empty.

3.3 Strong Compatibility

We focus here on the property of quasi-liveness. Indeed, a transition is said to be quasi-live if, from the initial marking, it is crossed at least once. It is thus to check that no unnecessary activity is modeled. A Petri net is quasi-live iff all its transitions are quasi-live.

$$t \in T \text{ is quasi-live } \Leftrightarrow \exists\ M \in\ [M_0\rangle,\ \mathrm{M[t}\rangle$$

The expression of the quasi-liveness property in CTL is as follows:

$$\overset{nbs}{\underset{s=1}{\&}} (\overset{nbt}{\underset{t=1}{\&}} EG\ EF\ P_{ts})$$

Where P_{ts} is a proposition on the firing of the transition number t of the service number s. It is testable from the table TRS_s. Hence, P_{ts} states that:

$$TRS_s[t] > 0$$

Thus, the property of strong compatibility is ensured by the following three properties: deadlock-freeness, proper termination and quasi-liveness. Both properties: proper termination and quasi-liveness are also used to check the soundness of a composition of oWF-nets. Strong compatibility and soundness [28] are two properties very similar in their definitions. They differ only by a single property: soundness should check the termination while strong compatibility must verify the quasi-liveness.

Let us study the simple example of Fig. 4.

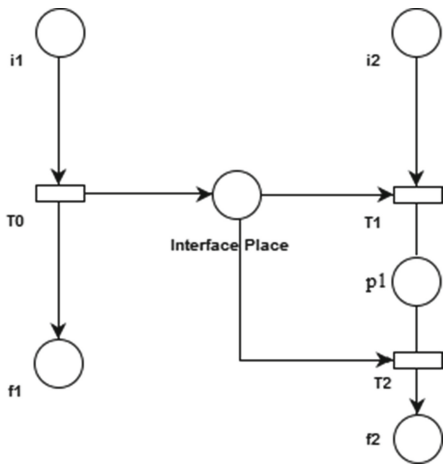


Fig. 4. A simple WSC not weak compatible

The WSC modelled in not weak compatible since a deadlock will be encountered after the execution of T_1, which prevents the execution of T_2 and thus the

final marking desired will not be reached. The Fig. 5 shows the result returned by NuSMV which contains a counter example.

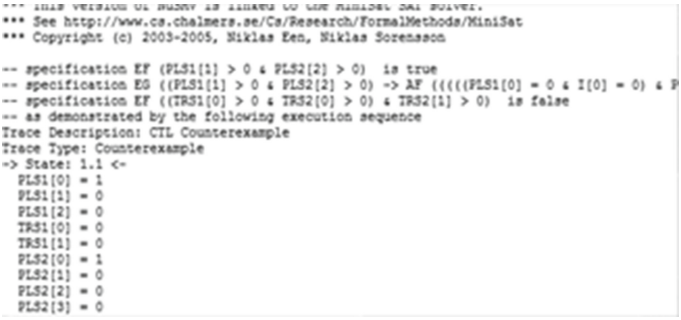


Fig. 5. A counter example generated by NuSMV

4 Related Work

The problem of service composition is approached by two orthogonal forces. On the one hand, most of the major industry partners offer process modeling and execution languages as BPEL. These languages allow programmers to implement complex Web services as distributed processes and to compose them in general. However, the definition of new processes that interact with existing products must be done manually, which is hard, takes a lot of time, and is considered as a source of errors.

On the other hand, a main problem with all of these industrial approaches is the verification of WSC correctness. For this reason, we have used the formal verification of composed Web services. Indeed, this robust technique ensures that computer system has no errors by exploring exhaustively all possible executions. The main advantage of the use of languages and models, which have clear and formal semantics, is the ability to use automatic tools in order to verify if a system matches some requirements and if it works as planned.

In case of modeling, we found in literature several works which dealt with formal modeling of WSC with a semantic based on transition systems (Petri nets, timed automata, process algebras, etc.). We refer reader to [24] for a survey on services composition approaches. Automata based models are more and more being used to formally describe, compose, and verify WSC. For instance, in [13] the authors introduce a framework to analyze and verify properties of WSC of BPEL processes that communicate via asynchronous XML messages. They start from translating BPEL processes to a particular type of automata, which will be them translated to Promela in order to be verified by the model checker Spin [17]. Petri nets are very popular in Business Process Management and related fields due to the variety of process control flows that they can capture. This motivates us to benefit from the Petri nets expressive power as well as the symbolic model checker NuSMV to propose an approach of formal analysis of WSC.

Let us now survey previous works which dealt with WSC compatibility analysis. van der Aalst et al. [2] considered that two services are compatible if their interfaces are also compatible and if in addition the composition does not suffer from any problem of deadlock. They also formalized other concepts related to the compatibility as strategy and controllability.

Bordeaux et al. [9] studied the verification of compatibility of Web services assuming that the messages exchanged are semantically of the same type and have the same name. They based their work on labeled transition systems (LTS) for the modeling of Web services. Three types of compatibility have been defined: the opposite behavior, unspecified reception and absence of deadlock.

Guermouche et al. [14] proposed an approach that allows the automatic verification of the compatibility of Web services taking into account their operations, the messages exchanged, the data associated with messages and time constraints. To check the compatibility of services using all of these properties, they proposed to extend the Web Services Timed Transition System (WSTTS).

Dumas et al. [10] have classified the incompatibility of Web services into two types: (1) Incompatibility of signatures (it occurs when a service request an operation from another service which can't provide it) and (2) Protocol incompatibility which occurs when a service A engages in a series of interactions with a service B, but the order which undertakes the service A is not compatible with the service B. Hence, they focused on the incompatibility of protocols in their article.

Tan et al. [29] proposed an approach that checks interface compatibility of Web services described in BPEL, and corrects these services if they are not compatible. To do this, they modeled the composition by SWF-nets, a subclass of CPN (Colored Petri Nets). Then they checked the compatibility of interfaces.

Olivia et al. [27] proposed a necessary condition for compatibility, based on the so-called state equation which consists on a linear algebraic necessary condition for states reachability.

These works allow either syntactical (checking interfaces) or semantic (verifying the absence of deadlocks) checking of the compatibility of the Web services, while we emphasize tho two types of compatibility. Besides, our approach is not limited to the verification of the absence of deadlock in a composite service, but we rather considered this property as a weak compatibility and we have verified other types of compatibility which are more exigent: compatibility and strong compatibility. We have also presented the relationship between compatibility aspects and the well known soundness property. Last but not least, none of the approaches mentioned above is based on the formal verification of compatibility while we have used this method in our approach. We mainly used the model checking formal method to check the semantic compatibility of the WSC.

5 Conclusion

Web Services composition is promisingly used to manage and automate cross-organizational business processes. It consists on a combination of existing Web services in order to produce a more complex and useful service. Although it can

greatly increase the reusability of service unit and reduce the coupling between software modules. However, the composition of Web services and the analyze and understanding of Web services based system with characters such as heterogeneity, distributed and loosely coupling are difficult tasks for software maintainers. In this context, we presented in this papier an approach of modeling and analysis WSC based on the well known symbolic model checker NuSMV, allowing thus the detection of eventual errors as early as possible in order to correct them before system implementation.

Our approach is summarized as follows. We first modeled the services interacting in a WSC with a Petri net class named open WF-nets, specifying thus the behavior of the different Web services as well as the communication between them. Then, we translated the composition of the obtained nets to SMV code which describes the WSC model in terms of states and the transition relation. In this SMV code, we used a set of arrays of integers to save the marking of places and a set of integer arrays to save the transitions firings. The transition relation is defined so as to act on the above arrays. Finally, we specified in CTL the requirements to be checked. These requirements are expressed in CTL formulae acting on propositions testing cell values of the various tables. The WSC model described in SMV will be checked by NuSMV against the requirements specified in CTL.

The results obtained in this paper are very promising since a negative result can be used to correct the model as early as possible and this by exploring the counter example which can be returned by NuSMV model checker in case a property is violated.

Actually, we proposed a platform that allows to the user to model any composition with oWF-nets. It allows the generation of the SMV code of a given modeling. This code is sent to the model checker NuSMV with properties of compatibility as well as soundness that should be checked. It finally returns the result which can be correct if the property is valid or a counter example if it is not. If the compatibility is valid, our platform generates an owfn file of the composition and we invoke the compiler oWFN2BPEL [26] which takes as input an owfn file and generates as output a BPEL [18] file.

We plan to extend this work by an automatic composition of Web services in case they are compatible and a Web services similarity search otherwise. In other words, in case a service N1 is not compatible with service N2, we suggest to look for replacing N1 by an other service allowing the same functionality as N1 and which is compatible with N2.

References

1. van der Aalst, W.: Verification of workflow nets. In: Azéma, P., Balbo, G. (eds.) ICATPN 1997. LNCS, vol. 1248, pp. 407–426. Springer, Heidelberg (1997)
2. van der Aalst, W.M.P., Mooij, A.J., Stahl, C., Wolf, K.: Service interaction: patterns, formalization, and analysis. In: Bernardo, M., Padovani, L., Zavattaro, G. (eds.) SFM 2009. LNCS, vol. 5569, pp. 42–88. Springer, Heidelberg (2009)

3. Cimatti, A., Clarke, E., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., Tacchella, A.: NuSMV 2: an OpenSource tool for symbolic model checking. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002. LNCS, vol. 2404, pp. 359–364. Springer, Heidelberg (2002)
4. Antonik, A., Huth, M.: Efficient patterns for model checking partial state spaces in ctl intersection ltl. Electr. Notes Theor. Comput. Sci. **158**, 41–57 (2006)
5. Baier, C., Katoen, J.P.: Principles of Model Checking (Representation and Mind Series). The MIT Press, Cambridge (2008)
6. Baldan, P., Corradini, A., Ehrig, H., Heckel, R.: Compositional modeling of reactive systems using open nets. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR 2001. LNCS, vol. 2154, pp. 502–518. Springer, Heidelberg (2001)
7. Barkaoui, K., Ben Ayed, R., Sbaï, Z.: Workflow soundness verification based on structure theory of petri nets. Int. J. Comput. Inf. Sci. (IJCIS) **5**(1), 51–61 (2007)
8. Barkaoui, K., Eslamichalandar, M., Kaabachi, M.: A structural verification of web services composition compatibility. In: Proceedings of the 6th International Workshop on Enterprise & Organizational Modeling and Simulation, EOMAS '10, pp. 30–41. CEUR-WS.org (2010)
9. Bordeaux, L., Salaün, G., Berardi, D., Mecella, M.: When are two web services compatible? In: Shan, M.-C., Dayal, U., Hsu, M. (eds.) TES 2004. LNCS, vol. 3324, pp. 15–28. Springer, Heidelberg (2005)
10. Dumas, M., Benatallah, B., Motahari Nezhad, H.: Web service protocols: Compatibility and adaptation. Institute of Electrical and Electronics Engineers (2008)
11. Foster, H., Uchitel, S., Magee, J., Kramer, J.: Compatibility verification for web service choreography. In: Proceedings of IEEE International Conference on Web Services, pp. 738–741 (2004)
12. Frappier, M., Fraikin, B., Chossart, R., Chane-Yack-Fa, R., Ouenzar, M.: Comparison of model checking tools for information systems. In: Dong, J.S., Zhu, H. (eds.) ICFEM 2010. LNCS, vol. 6447, pp. 581–596. Springer, Heidelberg (2010)
13. Fu, X., Bultan, T., Su, J.: Analysis of interacting bpel web services. In: Proceedings of the 13th International Conference on World Wide Web, pp. 621–630. ACM (2004)
14. Guermouche, N., Perrin, O., Ringeissen, C.: Timed specification for web services compatibility analysis. Theor. Comput. Sci. **200**, 155–170 (2008)
15. Henzinger, T., Nicollin, X., Sifakis, J., Yovine, S.: Symbolic model checking for real-time systems. Inf. Comput. **111**(2), 193–244 (1994)
16. Henzinger, T.A., Jhala, R., Majumdar, R., Sutre, G.: Software verification with BLAST. In: Ball, T., Rajamani, S.K. (eds.) SPIN 2003. LNCS, vol. 2648, pp. 235–239. Springer, Heidelberg (2003)
17. Holzmann, G.J.: The model checker spin. IEEE Trans. Softw. Eng. **23**(5), 279–295 (1997)
18. Huang, Y., Li, J., Dun, H., Wang, H.: Analyzing service composition patterns in bpel. In: Proceedings of the 2009 International Joint Conference on Artificial Intelligence, JCAI '09, pp. 623–627 (2009)
19. Karsten, S.: Controllability of open workflow nets. In: EMISA. LNI, pp. 236–249. Bonner Köllen Verlag (2005)
20. Leuschel, M., Butler, M.: Prob: A model checker for b. In: Araki, K., Gnesi, S., Mandrioli, D. (eds.) FME 2003. LNCS, vol. 2805, pp. 855–874. Springer, Heidelberg (2003)
21. Martens, A.: On compatibility of web services. In: Petri Net Newsletter, pp. 12–20 (2003)

22. Martens, A.: Analyzing web service based business processes. In: Cerioli, M. (ed.) FASE 2005. LNCS, vol. 3442, pp. 19–33. Springer, Heidelberg (2005)
23. Massuthe, P., Reisig, W., Schmidt, K.: An operating guideline approach to the soa. Ann. Math. Comput. Teleinformatics **1**(3), 35–43 (2005)
24. Maurice, B., Antonio, B., Stefania, G.: A survey on service composition approaches: From industrial standards to formal methods. Technical report 2006TR-15, Istituto, pp. 15–20. IEEE CS Press (2006)
25. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes, i. Inf. Comput. **100**(1), 1–40 (1992)
26. Niels, L.: owfn2bpel, translates a petri net model into an abstract bpel process (2008)
27. Oanea, O., Wolf, K.: An efficient necessary condition for compatibility. In: ZEUS. vol. 438, pp. 81–87. CEUR-WS.org (2009)
28. Sbaï, Z., Barkaoui, K.: Vérification formelle des processus workflow - extension aux workflows inter-organisationnels. Revue Ingénierie des Systèmes d'Information: Ingénierie des systèmes collaboratifs. **18**(5), 33–57 (2013)
29. Tan, W., Fan, Y., Zhou, M.: A petri net-based method for compatibility analysis and composition of web services in business process execution language. IEEE T. Autom. Sci. Eng. **6**(1), 94–106 (2009)

Recovering Traceability Links Between Code and Specification Through Domain Model Extraction

Jiří Vinárek[1,2], Petr Hnětynka[2(✉)], Viliam Šimko[1], and Petr Kroha[2]

[1] Institute for Program Structures and Data Organisation,
Karlsruhe Institute of Technology, Karlsruhe, Germany
{jiri.vinarek,viliam.simko}@kit.edu
[2] Department of Distributed and Dependable Systems,
Faculty of Mathematics and Physics, Charles University in Prague,
Malostranske namesti 25, Prague, Czech Republic
{vinarek,hnetynka,kroha}@d3s.mff.cuni.cz

Abstract. Requirements traceability is an extremely important aspect of software development and especially of maintenance. Efficient maintaining of traceability links between high-level requirements specification and low-level implementation is hindered by many problems. In this paper, we propose a method for automated recovery of links between parts of the textual requirement specification and the source code of implementation. The described method is based on a method allowing extraction of a prototype domain model from plain text requirements specification. The proposed method is evaluated on two non-trivial examples. The performed experiments show that our method is able to link requirements with source code with the accuracy of $F_1 = 58 - 61\,\%$.

Keywords: Specification · Requirements · Traceability · Domain model

1 Introduction

Requirements traceability is an extremely important aspect of software development [1]. Traceability itself has been defined in the paper [2] as "the ability to describe and follow the life of requirements, in both a forwards and backwards direction".

Efficient maintenance of traceability links between high-level requirements specification and low-level implementation is hindered by many problems (as also stated in the paper [1]). These problems include high manual effort of making the links up-to-date, insufficient tool support, etc. Keeping the links up-to-date is hard due to the evolving implementation as well as specification. As stated in the book [3], requirements specification cannot be understood as final and unchangeable, especially when incremental development is applied. On the other hand, as the specification commonly serves as a bridge between developers and stakeholders without technical background, it is vital to keep the specification

J. Barjis and R. Pergl (Eds.): EOMAS 2014, LNBIP 191, pp. 187–201, 2014.
DOI: 10.1007/978-3-662-44860-1_11

and implementation synchronized with correct traceability links. Also, this is important because the specification quite often serves as a base for decisions about software taken by the system stakeholders.

In this paper, we propose a viable method for automated recovery of links between specification and code. In particular, the method can recover traceability links between implementation classes and specification documents and also between classes and individual domain entities mentioned in the textual specification. The method is suitable especially for projects in a later stage of development. We do not assume that our method recovers all links, yet it may be useful as a starting point in this tedious process. Currently, we focus mainly on use-case specifications written in natural language and Java implementation code.

The method proposed in this paper is based on the tool [4] which is able to extract a prototype domain model from plain text employing statistical classifiers.

Evaluation of the method results is done using several example projects.

The paper is structured as follows. Section 2 presents the method we use as a basis. In Sect. 3, the core method is described and it is evaluated in Sect. 4. Section 5 discuses related work while Sect. 6 concludes the paper.

2 Domain Model Extraction

As a basis of our traceability method, we utilize the *Domain Model Extraction Tool* described in [4]. The tool extracts potential domain-model entities from text written in natural language (English). Input of the tool is a regular HTML document, and output is an EMF[1] model containing the derived domain entities linked to parts of the input text.

A domain model is a high-level overview of the most important concepts in the problem space. The domain model serves as a common vocabulary in the communication among technical and non-technical stakeholders throughout all project phases. This helps them come to an agreement on the meaning of important concepts. In [5] (p. 23), the domain model is defined as "a live, collaborative artefact which is refined and updated throughout the project, so that it always reflects the current understanding of the problem space".

The Domain Model Extraction Tool itself runs a deep linguistic analysis on the input text and then, using a set of statistical classifiers (Maximum Entropy models), it derives the prototype domain model. The linguistic pipeline employed is based on the Stanford CoreNLP framework[2]. The pipeline generates linguistic features such as identified sentences, dependency trees of words in each sentence, coreferences, etc. Most of the linguistic features are preserved and stored in the generated EMF model. The tool already contains a default set of classification models trained on several real-life systems. The training data consists of EMF domain models and HTML files linked together (a sequence of words is linked to a model element as depicted in Fig. 1); a link is encoded as an HTML anchor

[1] http://eclipse.org/emf/
[2] http://nlp.stanford.edu/software/corenlp.shtml

`<a href="#EntityName>multi-word term</a>` that carries the reference to a model element (domain entity) in the EMF model.

In detail, after running the *Domain Model Extraction Tool*, we obtain: (1) identified entities, (2) identified relations among entities, (3) links to the original text. It should be noted that we only focus on the entities in this paper and ignore the identified relations.

The tool achieves a classification accuracy of $F_1 = 76\,\%$ when classifying words that form a domain entity, and an accuracy of $F_1 = 88\,\%$ when identifying sequences of words that form an entity, i.e., identification of multi-word entities. Details about measurements and other details of the tool evaluation are available in the [4] (pp. 47–70). These measurements have been cross-validated on a simple book library system specification.

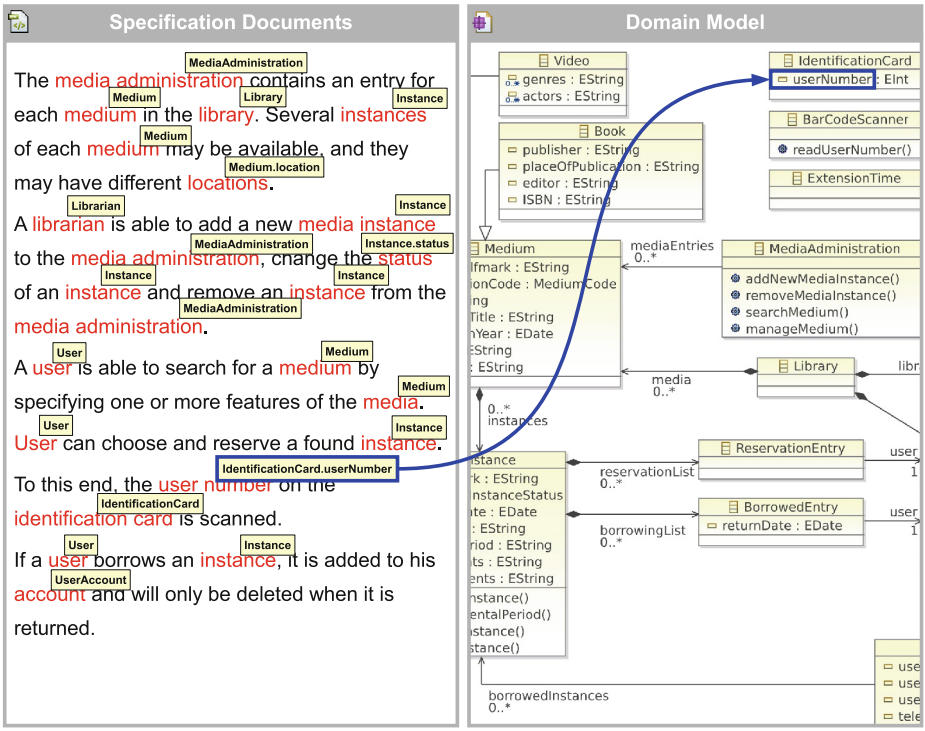


Fig. 1. An example of training data for the domain model extraction tool

3 Traceability Links Recovery

The pipeline of our method for recovery of traceability links is depicted in Fig. 2. First, a prototype domain model is extracted from the requirement specification (Sect. 3.1). Then, an implementation model from the Java source code is

extracted (Sect. 3.2). Finally, a similarity matrix is computed that assigns scores to the potential traceability links (Sect. 3.3). All of these stages are described in detail in the following sections.

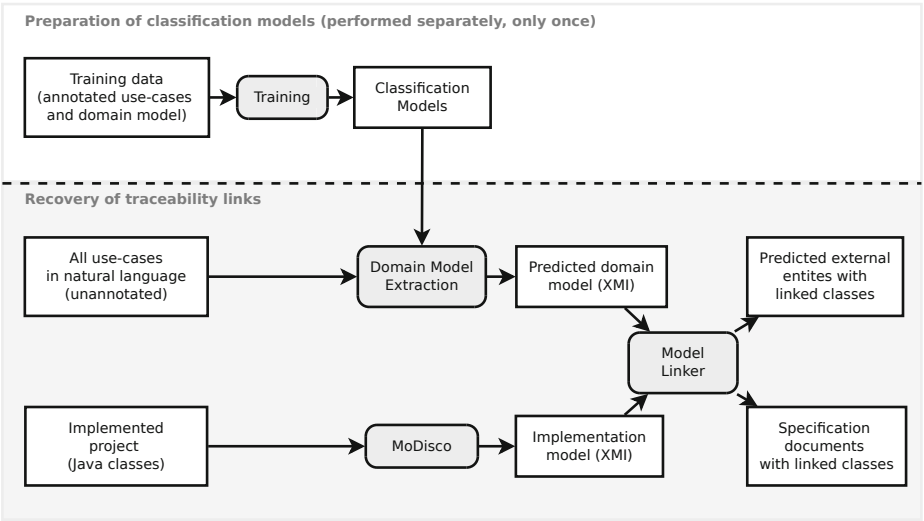


Fig. 2. Method pipeline

3.1 Extraction of a Domain Model

As a first step, we use the *Domain Model Extraction Tool* to predict domain entities out of text (an HTML document containing textual requirements specification). Output of the tool is an EMF model containing identified entities of the domain model and links to the specification.

The tool usually predicts more entities than would be predicted by manual inspection. To generate only a domain model, these false positives can be an issue. Nevertheless, for our method, they do not affect the final outcome as they are filtered out in the *linking phase* (Sect. 3.3).

The *Domain Model Extraction Tool* internally employs statistical models for classifiers. In a typical model extraction scenario, the training phase is omitted and the tool uses saved preconfigured statistical models. However, re-training the tool on the additional developed domain model and specification may improve precision of the model extraction.

3.2 Extraction of the Implementation Model

The implementation model of the project is reverse-engineered from the source files with the use of the MoDisco[3] framework. MoDisco is able to obtain a model

[3] http://eclipse.org/MoDisco/

from multiple sources (Java, JSP, XML, etc.) but currently only Java code is relevant for our method. Output of the MoDisco is an EMF model of the implementation project that can be easily queried.

3.3 Linking Phase

Linking phase consists of the following steps:

1. The model linker generates a similarity matrix, where the rows represent domain entities and columns represent classes/interfaces found in the implementation model. Each cell in the matrix contains a fractional number between 0 and 1 which represents a string similarity measure between the corresponding entity and class/interface (more details about the chosen string similarity measure are in Sect. 3.4).
2. Next, we further filter-out cells from the matrix that are lower than a given threshold value. The surviving entity-class pairs are taken as a result of our method (a particular example is in Fig. 7).
3. Finally, as the domain model entities "remember" from which words in the input document they were generated, these words are transformed into hyperlinks pointing to the particular sources files (using hyperlinks from the specification to the code is one of widely used techniques for the traceability links visualisation [6]). Classes from the predicted model that have no classes/interfaces from the implementation model assigned are rejected.

3.4 String Similarity Measure

As a particular string similarity measure we have adopted the *Jaro-Winkler* measure [7]. The *Jaro-Winkler* measure is an extension of the *Jaro* string comparator that produces a distance of two strings.

Roughly, the *Jaro* comparator works in three steps: (i) computes lengths of compared strings s_1 and s_2, (ii) computes number of *matching characters* m, (iii) finds number of *transpositions* t.

Two characters, each from a different string, are *matching* if they are the same, and they are not too "far" from each other in the strings (at most a half the length of the shorter of two strings). Position of each character from one string is compared with positions of all its matching characters from the other string, and the number of *transpositions* is the number of matching characters that are in different order. When the number of matching characters is zero, the distance is defined as 0; in other cases it is defined as follows:

$$Jaro(s_1, s_2) = \frac{1}{3} \cdot \left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{|t|}{2m}\right).$$

The *Jaro-Winkler* measure adds additional bonus to strings with a common prefix. The length of the prefix (labeled L) can reach at maximum 4 characters and the measure is defined as:

$$JaroWinkler(s_1, s_2) = Jaro(s_1, s_2) + \frac{L}{10} \cdot (1 - Jaro(s_1, s_2)).$$

Apart from the *Jaro-Winkler* measure, we also tried several other string similarity measures (Levenshtein distance, Jaro distance, Dice's coefficient). The *Jaro-Winkler* measure gave us the best results as it assigns higher score to the words with the same prefix. This fact suits our needs as giving common prefix to related classes is a common practice. Levenshtein and Dice coefficient measures perform poorly especially in cases when the two strings differ greatly in their length.

Disadvantage of the *Jaro-Winkler* measure is lack of the preference for strings with common suffix. This fact discriminates related entities with a particular naming convention. For example the words "Dispenser" and "SimCashDispenser" obtain low similarity score although they would be recognized as similar by a human analyst. To overcome this drawback, we proposed a modification of the measure, which adds additional bonus to the words with common suffix. The modified measure computes the *Jaro-Winkler* measure first (labelled JW) and then adds the suffix bonus. The bonus equals to the scaled ratio of the common suffix length (labeled S) and length of the first word (which is in our case used for name of the predicted class). We call it the *Boosted-Jaro-Winkler* measure:

$$BoostedJaroWinkler(s_1, s_2) = JW(s_1, s_2) + \frac{S}{|s_1|} \cdot (1 - JW(s_1, s_2))$$

A comparison of the mentioned measures on several string pairs (the pairs taken from the example used in Sect. 4) is shown in Fig. 3 (values range from 0 to 1; higher value means that the strings are more similar).

Entity from text / Class from code	Levenshtein	Jaro	Dice coefficient	Jaro-Winker	Boosted-JW
Dispenser / CashDispenser	0.69	0.79	0.80	0.79	1.00
Dispenser / SimCashDispenser	0.56	0.48	0.70	0.48	1.00
ProductAvailable / Product	0.44	0.81	0.57	0.89	0.89
ProductAvailable / ProductNotAvailableException	0.57	0.86	0.71	0.91	0.91
ProductAvailable / ProductTO	0.44	0.74	0.52	0.84	0.84
Scanner / ScannerController	0.41	0.80	0.57	0.88	0.88
Scanner / ScannerControllerEventHandlerIf	0.23	0.74	0.39	0.85	0.85
Scanner / ScannerControllerEventHandlerImpl	0.21	0.74	0.36	0.84	0.84

Fig. 3. String similarity on example string pairs

4 Test Data and Evaluation

To evaluate the described method, we used data from two software projects in our experiment: (1) the CoCoME [8] (Common Component Modeling Example) and (2) the ATM project[4]. In both cases, a textual specification together with a Java-based implementation was available.

[4] http://www.math-cs.gordon.edu/courses/cs211/ATMExample/

In specifications, we manually identified a set of entities (actors and external systems) that communicate with the described system. In the source files, we located Java classes representing these entities and created a so-called *gold set*, i.e., pairs connecting the specification entities with their implementation counterparts. The *gold set* is used for evaluation of the proposed method success rate.

4.1 CoCoME Project

The goal of CoCoME was to create a common example for evaluation of component-based frameworks. The specification of CoCoME defines a trading system used for handling sales in a chain of stores. Importantly, the specification tries to mimic a description of the system as delivered by a business company (as it could be in the reality), and as such it can be potentially incomplete and/or imprecise.

Primary reason for the use of CoCoME was the fact that it offers a real-life system with both the requirements specification (the use-cases) and a freely available[5] implementation, which typically is not very common.

The specification itself contains both functional and extra-functional requirements. Functional requirements are described in a form of high-level use-cases accompanied with sequence diagrams. Extra-functional requirements add timing, reliability and usage-profile-related constraints. Apart from the requirements, the CoCoME specification contains also architectural component model, deployment view and behavioral view of the described system—all these parts use structured text in conjunction with UML diagrams.

From the specification, we took the high-level use-cases describing communication between modeled system and involved actors. As a particular implementation of CoCoME, we took the reference JEE-based implementation provided together with the specification.

4.2 ATM Project

The second project used for evaluation describes an ATM system and it was originally developed for an object-oriented software development course. The course material shows the complete process of software system development from its initial requirement collection, analysis, and design to implementation. From the project deliverables, we leveraged initial requirements and use-cases together with system implementation.

4.3 Evaluation

We evaluated our method by setting the following goals for the evaluation:

G1 Find the best performing threshold value for filtering the similarity matrix.

[5] http://cocome.org/

G2 Compare the results obtained by our method in a fully automated scenario against a prepared baseline. The baseline consists of the domain entity list obtained by picking up all subjects and objects. When possible, we concatenated adjacent nouns (identified by the POS-tagger) to form an entity name.

baseline and **baseline-boosted:** In these scenarios, the domain entities from the baseline were used and string similarity was computed using the *Jaro-Winkler* and *Boosted-Jaro-Winkler* measures. These scenarios act as our *baselines*.

predicted and **predicted-boosted:** Here, the lists of entities were derived using the *Domain Model Extraction Tool* and for string similarity the *Jaro-Winkler* and *Boosted-Jaro-Winkler* measures were used.

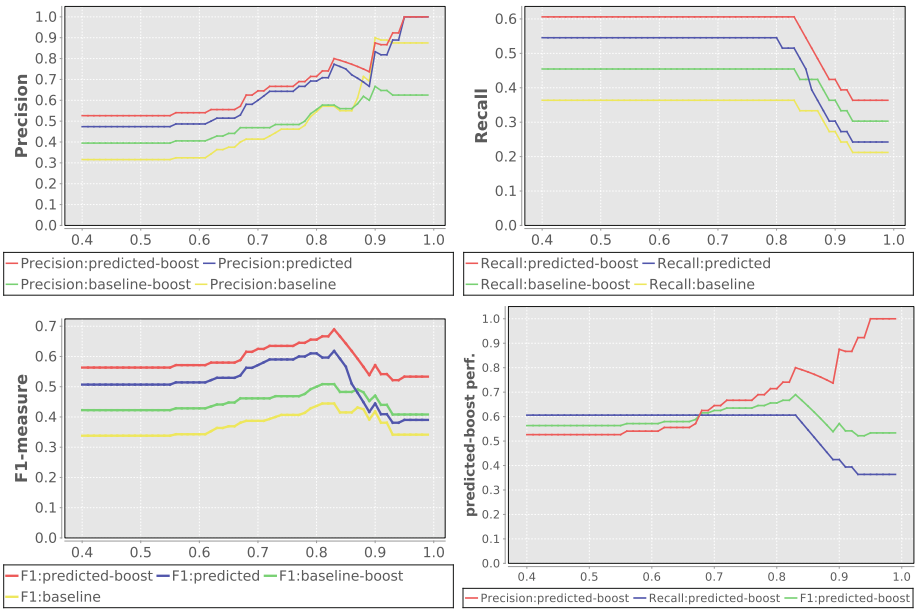
G3 Evaluate the ability of our method to find traceability links between classes and specification documents and compare it with state of the art methods (Vector space model and probability method as stated in the paper [9]).

Characteristics of the data used for evaluation and training are shown in Fig. 4. The CoCoME dataset consisted of 8 use-cases which were split into 23 traced documents; the ATM example contained the high level requirements document and 9 use-cases which were split into 25 traced documents. Models for the statistical classifiers were trained on independent specifications before evaluation, in order to prevent classifier over-fitting; the ATM example evaluation employed CoCoME and the Library system (a model bundled with the *Domain Model Extraction Tool*) as training data, while CoCoME used the ATM example and the Library system data. The Library system was used for training only (not for evaluation), and it is mentioned here just for the sake of completeness.

	CoCoME	ATM	Library system
Number of documents	23	25	1
Total number of words	2095	1945	1130
Total number of classes	118	38	n/a
Gold set – number of entities	23	21	41
Gold set – number of classes	86	33	n/a

Fig. 4. Characteristics of the data used for evaluation/training

Results for G1: To find the optimal threshold value, we executed the model linker multiple times for thresholds in the interval $(0.4, 1.0)$ and computed the accuracy. Results are presented in a form of **Precision**, **Recall**, and F_1-**measure** related to cut-off **Threshold**. Tables with the results are shown in Figs. 5 and 6. We can see from the F_1-**measure** diagram that the highest F_1 corresponds to the threshold value 0.83 for the ATM example and value 0.79 for CoCoME.

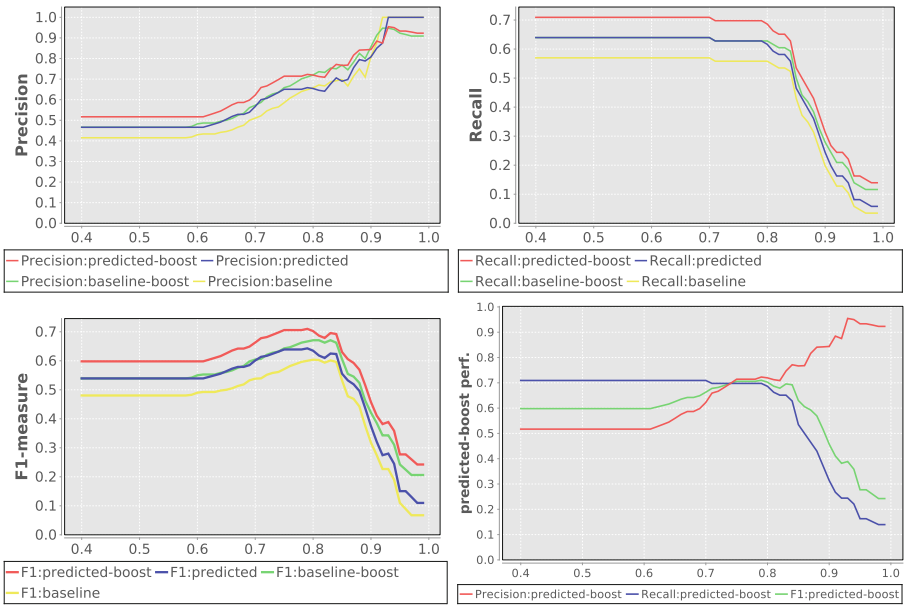


	Predicted-boost			Predicted			Baseline-boost			Baseline		
Threshold	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
0.40	0.53	0.61	0.56	0.47	0.55	0.51	0.39	0.45	0.42	0.32	0.36	0.34
0.45	0.53	0.61	0.56	0.47	0.55	0.51	0.39	0.45	0.42	0.32	0.36	0.34
0.50	0.53	0.61	0.56	0.47	0.55	0.51	0.39	0.45	0.42	0.32	0.36	0.34
0.55	0.53	0.61	0.56	0.47	0.55	0.51	0.39	0.45	0.42	0.32	0.36	0.34
0.60	0.54	0.61	0.57	0.49	0.55	0.51	0.41	0.45	0.43	0.32	0.36	0.34
0.65	0.56	0.61	0.58	0.51	0.55	0.53	0.44	0.45	0.45	0.38	0.36	0.37
0.70	0.65	0.61	0.63	0.60	0.55	0.57	0.47	0.45	0.46	0.41	0.36	0.39
0.75	0.67	0.61	0.63	0.64	0.55	0.59	0.48	0.45	0.47	0.46	0.36	0.41
0.80	0.71	0.61	0.66	0.69	0.55	0.61	0.56	0.45	0.50	0.55	0.36	0.44
0.83	0.80	0.61	0.69	0.77	0.52	0.62	0.58	0.45	0.51	0.57	0.36	0.44
0.85	0.78	0.55	0.64	0.75	0.45	0.57	0.56	0.42	0.48	0.55	0.33	0.42
0.90	0.88	0.42	0.57	0.83	0.30	0.44	0.67	0.36	0.47	0.90	0.27	0.42
0.95	1.00	0.36	0.53	1.00	0.24	0.39	0.63	0.30	0.41	0.88	0.21	0.34

Fig. 5. ATM example – The diagrams show accuracy (Y-axis) for different threshold values (X-axis). A subset of the results is also presented as a table with the best performing threshold value being highlighted.

Results for G2: The diagram denoted as **predicted-boost perf.** in Figs. 5 and 6 focuses only on the **best** scenario and shows all the measures together. The model linker executed on the CoCoME example with threshold set to value 0.79 returned 23 domain entities and 83 implementation classes (as seen in Fig. 7) from which 19 predicted and 60 implementation were recognized/classified correctly; executed on the ATM example with threshold set to value 0.83 returned 18 domain entities and 25 implementation classes (as seen in Fig. 8) from which 12 predicted and 20 implementation were recognized/classified correctly.

Results for G3: We traced the predicted entities returned from the model linker back to the specification documents. In this way, we got links between specification documents and implementation classes. Methods mentioned in [9] are able to recover traceability links with accuracy 34–53 % while our method is



	Predicted-boost			Predicted			Baseline-boost			Baseline		
Threshold	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
0.40	0.52	0.71	0.60	0.47	0.64	0.54	0.47	0.64	0.54	0.42	0.57	0.48
0.45	0.52	0.71	0.60	0.47	0.64	0.54	0.47	0.64	0.54	0.42	0.57	0.48
0.50	0.52	0.71	0.60	0.47	0.64	0.54	0.47	0.64	0.54	0.42	0.57	0.48
0.55	0.52	0.71	0.60	0.47	0.64	0.54	0.47	0.64	0.54	0.42	0.57	0.48
0.60	0.52	0.71	0.60	0.47	0.64	0.54	0.48	0.64	0.55	0.43	0.57	0.49
0.65	0.56	0.71	0.63	0.50	0.64	0.56	0.50	0.64	0.56	0.45	0.57	0.50
0.70	0.62	0.71	0.66	0.57	0.64	0.60	0.57	0.64	0.60	0.51	0.57	0.54
0.75	0.71	0.70	0.71	0.65	0.63	0.64	0.66	0.63	0.64	0.59	0.56	0.57
0.79	0.72	0.70	0.71	0.66	0.63	0.64	0.71	0.63	0.67	0.65	0.56	0.60
0.80	0.72	0.69	0.70	0.65	0.62	0.63	0.72	0.63	0.67	0.66	0.56	0.60
0.85	0.77	0.53	0.63	0.69	0.47	0.56	0.77	0.50	0.61	0.70	0.43	0.53
0.90	0.84	0.31	0.46	0.81	0.24	0.38	0.86	0.28	0.42	0.81	0.20	0.32
0.95	0.93	0.16	0.28	1.00	0.08	0.15	0.92	0.14	0.24	1.00	0.06	0.11

Fig. 6. CoCoME example – The diagrams show accuracy (Y-axis) for different threshold values (X-axis). A subset of the results is also presented as a table with the best performing threshold value being highlighted.

successful in 58–61 % according to F_1-measure. Figure 9 shows the dependency of the **Precision**, **Recall**, and F_1**-measure** related to cut-off **Threshold** used in the model linker.

5 Related Work

Probabilistic and vector space information retrieval techniques for traceability links are explained in the paper [9]. These approaches apply a text normalization procedures to both source code and software documents. Normalized documents are indexed and traceability links are estimated according to their similarity score. Contrary to that, our method goes in an opposite direction—it tries to

Entity from text	Class from code	Similarity
Button	OrderButton	1,00
Button	RefreshButton	1,00
CardReader	CardReader	1,00
CardReader	CardReaderControllerEventHandlerIf	0,86
CardReader	CardReaderControllerEventHandlerImpl	0,86
Cash	CashAmountEnteredEvent	0,84
CashBox	CashBox	1,00
CashBox	CashBoxClosedEvent	0,88
CashBox	CashBoxControllerEventHandlerIf	0,85
CashBox	CashBoxControllerEventHandlerImpl	0,84
CashDesk	CashDesk	1,00
CashDesk	CashDeskConnectorIf	0,88
CashDesk	CashDeskGUI	0,95
ChangeAmount	ChangeAmountCalculatedEvent	0,89
CreditCard	CredidCardScanFailedEvent	0,81
CreditCard	CreditCardPaymentEnabledEvent	0,87
CreditCard	CreditCardScanEvent	0,91
CreditCard	CreditCardScanFailedEvent	0,88
CreditCard	CreditCardScannedEvent	0,89
Database	DataIf	0,83
Database	DataIfFactory	0,80
Database	DataImpl	0,80
Enterprise	EnterpriseQueryIf	0,92
Enterprise	EnterpriseQueryImpl	0,91
Enterprise	EnterpriseTO	0,97
Enterprise	TradingEnterprise	1,00
Item	StockItem	1,00
LightDisplay	LightDisplayController	0,91
LightDisplay	LightDisplayControllerEventHandlerIf	0,87
LightDisplay	LightDisplayControllerEventHandlerImpl	0,86
Order	OrderEntry	0,90
Order	OrderEntryTO	0,88
Order	OrderTO	0,94
Order	ProductOrder	1,00
Pin	PINEnteredEvent	0,81
Printer	PrinterController	0,88
Printer	PrinterControllerEventHandlerIf	0,85
Printer	PrinterControllerEventHandlerImpl	0,84
ProductAmount	ProductAmountTO	0,97
ProductAmount	ProductBarcodeNotValidEvent	0,85
ProductAmount	ProductBarcodeScannedEvent	0,83
ProductAmount	ProductMovementTO	0,90
ProductAmount	ProductNotAvailableException	0,83
ProductDescription	ProductBarcodeScanEvnent	0,84
ProductDescription	ProductDispatcher	0,90
ProductDescription	ProductDispatcherIf	0,91
ProductDescription	ProductOrderDisplay	0,86
ProductDescription	ProductSupplierStockItemTableModel	0,82
ProductDescription	ProductSupplierTableModel	0,84
ProductDescription	ProductWithSupplierAndStockItemTO	0,82
ProductDescription	ProductWithSupplierTO	0,87
ProductItem	Product	0,93
ProductItem	ProductStockItemTableModel	0,87
ProductItem	ProductSupplier	0,88
ProductItem	ProductSupplierOrderTableModel	0,82
ProductItem	ProductTO	0,92
ProductItem	ProductWithStockItemTO	0,85
Report	EnterpriseStockReport	1,00
Report	MTDeliveryReport	1,00
Report	ReportTO	0,95
Report	Reporting	0,93
Report	ReportingIf	0,91
Report	ReportingImpl	0,89
Report	StoreStockReport	1,00
Sale	Sale	1,00
Sale	SaleFinishedEvent	0,85
Sale	SaleRegisteredEvent	0,84
Sale	SaleStartedEvent	0,85
Sale	SaleSuccessEvent	0,85
Sale	SaleTO	0,93
Stock	StockItemTO	0,89
Store	Store	1,00
Store	StoreDescr	0,90
Store	StoreIf	0,94
Store	StoreImpl	0,91
Store	StoreQueryIf	0,88
Store	StoreQueryImpl	0,87
Store	StoreQueryImplTest	0,86
Store	StoreTO	0,94
StorePressIdentifier	StoreWithEnterpriseTO	0,86
Transportation	TransactionContext	0,87
Transportation	TransactionContextImpl	0,85
Transportation	TransactionID	0,88

Fig. 7. CoCoME - predicted entity-class pairs with threshold value 0.79

synthesize a domain model from the given documents and match them with source code. It works at finer-grained level as it traces not documents as a whole but entities contained in them. Using statistical classifiers it can leverage semantic context of the document's words.

Probabilistic and vector space methods are also discussed in the paper [10]. In addition to [9], the paper proposes best practices for writing and structuring software artefacts (documentation, specification etc.) to improve automated traceability.

The method introduced in [9] is further extended in [11]. Its main contribution is utilization of a syntax tree derived from code. Identifiers found in code are converted into comment keywords based on their appearance in the syntax tree. Using this approach, the authors are able to match abbreviated identifiers or identifiers using synonyms to their documentation counterparts.

Entity from text	Class from code	Similarity
Account	AccountInformation	0,88
Atm	ATM	1,00
Atm	ATMApplet	0,84
Atm	ATMMain	0,87
Atm	ATMPanel	0,85
Bank	NetworkToBank	1,00
Bank	SimulatedBank	1,00
Card	Card	1,00
Card	CardPanel	0,89
CardReaderSlot	CardReader	0,94
CashDispenser	CashDispenser	1,00
CashDispenser	SimCashDispenser	1,00
Customer	CustomerConsole	0,91
Deposit	Deposit	1,00
Display	SimDisplay	1,00
Envelope	EnvelopeAcceptor	0,90
Inquiry	Inquiry	1,00
MoneyAmount	Money	0,89
Operator	OperatorPanel	0,92
Operator	SimOperatorPanel	0,83
ReceiptCash	Receipt	0,93
Startup	Status	0,89
Transaction	Transaction	1,00
Transfer	Transfer	1,00
Withdrawal	Withdrawal	1,00

Fig. 8. ATM - predicted entity-class pairs with threshold value 0.83

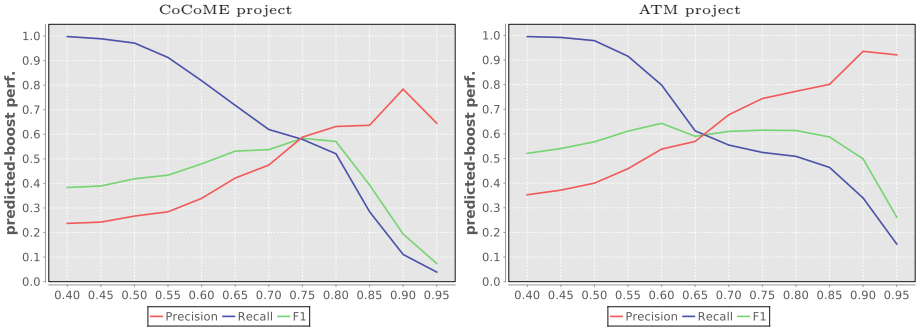


Fig. 9. Accuracy of tracebility link recovery between classes and whole specification documents. X-axis represents different threshold values, Y-axis represents accuracy.

Another approach extending the method from [9] is presented in [12]. It uses information retrieval techniques to obtain the traceability links between code and requirements. Consequently, information mined from software repositories (CVS/SVN) is used to rerank/discard retrieved links. Information can be mined from multiple sources and weights for the links may be assigned on a per-link basis.

An approach helping developers to maintain source code identifiers and comments consistent with high-level artifacts is presented in [13]. The proposed method computes the textual similarity between source code and related high-level specification documents and presents computed similarity to the developer.

Moreover, the method recommends candidate identifiers built from high-level artifacts. The approach is implemented as an Eclipse plugin called COde Comprehension Nurturant Using Traceability (COCONUT). The paper also reports on two experiments using COCONUT that evaluate the quality of the developed code. In contrast with the above mentioned approaches [9,10], the method uses the latent semantic indexing technique for document indexing, which gives more accurate results compared to the vector space method. Compared to our method the approach is more focused on interactive improvement of the source code and less on the automatic derivation of the traceability links.

A probabilistic approach to bridge the gap between high-level description of the system and its implementation is described in the paper [14]. The mentioned *cognitive assignment* technique has 2 phases—the *cognitive map derivation* and *concept assignment*. In the first phase, the system processes relevant project documents (specification, bug reports, etc.) authored by an expert engineer. In the second phase, a non-expert engineer uses queries to look for relevant pieces of code. The query together with the cognitive maps are transformed into a Bayesian network; it is used to classify the source code and relevant results are returned to the user. The method is implemented as an Eclipse plugin and compared to our method, it is more suited to interactive exploration of the software project and less applicable to automatic link derivation.

A method for automated traceability links retrieval using ontologies is explained in the paper [15]. The method processes source and target artifacts with linguistic tools and tries to map concepts extracted from sentences to the domain-specific ontologies. In case of unsuccessful mapping, it establishes similarity using generalized ontology which is composed of single words and very simple phrases. To compare with other methods, a disadvantage of the method is the need to create a domain-specific ontology to obtain more accurate results. The paper states that one of the authors spent two days to create a domain-specific ontology for 40 of the 158 source artifacts.

An approach targeted on linking the implementation source code with code snippets included in learning resources or supporting channels (e.g., bug trackers, forums) is presented in [16]. Authors identified sources of commonly found ambiguities used in code snippets and designed a pipeline to precisely locate the traced source code. Evaluated on several open-source projects, the method shows high *precision/recall* ratio (96 %). The technique is narrowly focused on the source code and does not take specification written in natural language into account.

An extensive survey and categorization of traceability-discovery techniques can be found in [17]. The survey encompasses 89 articles from 25 venues published between years 1992 and 2011. It defines 7 dimensions and their attributes for the feature location taxonomy. Our method would be classified for *Type of analysis* dimension as *Textual* as it uses NLP tools, *User input* would be *Natural Language Query* and *Source Code Artifact*. In *Data sources* dimension, it would fit into *Non-compilable* category. *Output* category defines granularity of the results, the method works on *File/class* level. *Programming language support*

dimension is in current phase restricted to *Java*. *Evaluation* dimension would be ranked as *Preliminary* as the method is evaluated on small data set. *Systems evaluated* would contain CoCoME.

The TraceLab project[6] [18] aims at providing an experimental workbench for designing, constructing, and executing traceability experiments, and for facilitating the rigorous evaluation of different traceability techniques.

Automated detection and classification of the non-functional requirements from both structured and unstructured documents is discussed in [19]. It describes a classification algorithm and evaluates its effectiveness on two datasets—requirements specification developed as a student term project and a large dataset from an industrial project. The method as well as our method uses machine-learning techniques to identify candidate entities. However, the method targets the non-functional requirements only and processes only specifications.

6 Conclusion and Future Work

We presented our method for recovering traceability links between a requirements specification and implementation. In particular, it can be used to recover links (1) between classes and domain entities or (2) between classes and specification documents as shown in the evaluation on two non-trivial examples. We compared the former with a baseline approach that considers all nouns in the text as potential entities. We showed that precision and recall is increased when the entities are extracted using our Domain Model Extraction Tool. When comparing the latter with existing probabilistic and vector space information retrieval methods (e.g., those presented in [9]), we showed that our method performs better with respect to the F_1-measure.

Currently, we plan to evaluate our method on several different case studies and examples to confirm performance of the method and to tune it. We also plan to evaluate our method on the same dataset with other mentioned methods for traceability recovery to obtain more accurate method comparison. Obtained values may be affected by the fact that a common specification does not contain high number of domain entities. This issue would be solved with evaluation of additional data sets. Method accuracy could be further improved by detection of common prefixes/suffixes used for domain entities (Java interfaces are frequently prefixed with "I", enums with "E", EMF models use "Impl" suffix for class implementation etc.). Their trimming would raise string-similarity values observed during linking phase.

Acknowledgments. This work was partially supported by the EU project ASCENS 257414, partially supported by the European Union Seventh Framework Programme FP7-PEOPLE-2010-ITN under grant agreement n°264840, and partially supported by Charles University institutional funding SVV-2014-260100.

[6] http://www.coest.org/index.php/tracelab/

References

1. Bouillon, E., Mäder, P., Philippow, I.: A survey on usage scenarios for requirements traceability in practice. In: Doerr, J., Opdahl, A.L. (eds.) REFSQ 2013. LNCS, vol. 7830, pp. 158–173. Springer, Heidelberg (2013)
2. Gotel, O.C.Z., Finkelstein, A.C.W.: An analysis of the requirements traceability problem. In: Proceedings of ICRE 1994. Colorado Springs, USA, April 1994
3. Larman, C.: Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Proces, 3rd edn. Prentice-Hall, Upper Saddle River (2004)
4. Šimko, V.: From textual specification to formal verification. Ph.D. thesis, Charles University in Prague, Faculty of Mathematics and Physics (2013)
5. Rosenberg, D., Stephens, M.: Use Case Driven Object Modeling with UML: Theory and Practice. Springer, New York (2007)
6. Li, Y., Maalej, W.: Which traceability visualization is suitable in this context? a comparative study. In: Regnell, B., Damian, D. (eds.) REFSQ 2011. LNCS, vol. 7195, pp. 194–210. Springer, Heidelberg (2012)
7. Winkler, W.E.: Overview of record linkage and current research directions. Research report series, Statistical Research Division, US Census Bureau, February 2006
8. Rausch, A., Reussner, R., Mirandola, R., Plášil, F. (eds.): The Common Component Modeling Example. LNCS, vol. 5153. Springer, Heidelberg (2008)
9. Antoniol, G., Canfora, G., Casazza, G., De Lucia, A., Merlo, E.: Recovering traceability links between code and documentation. IEEE Trans. Softw. Eng. **28**(10), 970–983 (2002)
10. Cleland-Huang, J., Settimi, R., Romanova, E., Berenbach, B., Clark, S.: Best practices for automated traceability. Computer **40**(6), 27–35 (2007)
11. Nagano, S., Ichikawa, Y., Kobayashi, T.: Recovering traceability links between code and documentation for enterprise project artifacts. In: Proceedings of COMPSAC 2012, Izmir, Turkey, pp. 11–18. IEEE, July 2012
12. Ali, N., Gueheneuc, Y., Antoniol, G.: Trustrace: mining software repositories to improve the accuracy of requirement traceability links. IEEE Trans. Softw. Eng. **39**(5), 725–741 (2013)
13. Lucia, A.D., Penta, M.D., Oliveto, R.: Improving source code lexicon via traceability and information retrieval. IEEE Trans. Softw. Eng. **37**(2), 205–227 (2011)
14. Cleary, B., Exton, C.: The cognitive assignment Eclipse plug-in. In: Proceedings of ICPC 2006, Athens, Greece. IEEE, June 2006
15. Li, Y., Cleland-Huang, J.: Ontology-based trace retrieval. In: Proceedings of TEFSE 2013, San Francisco, USA, pp. 30–36. IEEE, May 2013
16. Dagenais, B., Robillard, M.: Recovering traceability links between an API and its learning resources. In: Proceedings of ICSE 2012, Zurich, Switzerland, pp. 47–57. IEEE, June 2012
17. Dit, B., Revelle, M., Gethers, M., Poshyvanyk, D.: Feature location in source code: a taxonomy and survey. J. Softw. Evol. Process **25**(1), 53–95 (2013)
18. Dit, B., Moritz, E., Poshyvanyk, D.: A TraceLab-based solution for creating, conducting, and sharing feature location experiments. In: Proceedings of ICPC 2012, Passau, Germany, pp. 203–208. IEEE CS, June 2012
19. Cleland-Huang, J., Settimi, R., Zou, X., Solc, P.: The detection and classification of non-functional requirements with application to early aspects. In: Proceedings of RE 2006, St. Paul, USA. IEEE, September 2006

Choreography Modeling Compliance for Timed Business Models

Manuel I. Capel[1(✉)] and Luis E. Mendoza[2]

[1] Software Engineering Department, College of Informatics and Telecommunications, University of Granada, 18071 Granada, Spain
manuelcapel@ugr.es
http://lsi.ugr.es/~mcapel
[2] Processes and Systems Department, Simón Bolívar University, PO Box 89000, Caracas 1080-A, Venezuela
lmendoza@usb.ve
http://www.lisi.usb.ve/

Abstract. Business Process Modeling (BPM) is a conceptual activity for embodying the functioning and complex structure of any enterprise's business processes, so that these can be then analyzed and improved. A BP can be understood as a set of related, structured, interacting services driven by a *choreography* that is capable of giving complex functionality to customers. General *choreographies* of BP cannot be verified, specially *timed choreographies*, because implementations scarcely show the same behavior than the one initially specified according to business rules. We therefore propose here a formal semantics for a subset of BPMN, in order to check if a given choreography is *realizable*. This includes formalization of behavioral and temporal aspects of BPMN. A set of transformation rules for choreography diagrams into a timed process algebra is given. Therefore, we obtain an easy verification approach for choreography implementation models based on model–checking tools. In this way we can obtain advantage of the strengths that a formalization of behavioral and temporal aspects of BPMN will bring about, at design and implementation stages, to any model of interest.

Keywords: Business Process Modeling · Choreography · BPMN 2.0 · Transformation rules · Choreography modeling conformance · Timed business processes

1 Introduction

Business Process Model and Notation (BPMN) is now considered to be the standard modeling language for business process realm. BPMN uses a graphical notation, very similar to UML, and analogously to it their semantics is not formally defined. The referred models may lead to ambiguity and confusion with interpretation between technical and business users. Our proposal is mainly intended to propose a formal semantics to a set of BPMN modeling constructs, useful

J. Barjis and R. Pergl (Eds.): EOMAS 2014, LNBIP 191, pp. 202–218, 2014.
DOI: 10.1007/978-3-662-44860-1_12

to enforce any *choreography* to be *realizable*; and then, it introduces an easy approach to the verification of business processes that may use model–checking techniques. Any *choreography* is said to be realizable if all the interactions specified in BPMN 2.0 diagrams are equivalent to those that can be executed with the analogous service–description language implementation, such as WS–CDL for instance. In a longer term perspective, we also aspire to support business analysts and modelers to find a way to improve the quality of their business models.

Several authors have made progress in solving the *choreography realization problem*. The research work up until now can be divided into two categories, the first one focuses on business process model analysis, whereas the second one centres on obtaining a formal semantics of BPMN modeling entities. The main approaches of work in the first area can be found in [1] The principal challenge to this path of research is related with the necessity of transforming BPMN models into executable environments. Any BPMN transformation is more than just converting these models. We will not get significant results in improving BP models if task execution order and other influential elements from the environment are not considered during the analysis. With regard to the second group, there are formal methods proposed for verifying BPMN models based on Π-calculus [10] or Petri Nets [4], which can debug grammatical errors and can transform business processes diagrams (BPD) into BP Execution Language (BPEL) code [2,11]. In this second group, the central problem to tackle consists of proving the soundness of BPMN model transformation. In many cases, a model cannot be verified because its representation in an executable environment does not show the same *behavior* as observed in the original model, i.e., it does not react to external events in the same way.

The lack of verification exhibited by the aforementioned approaches, in our opinion, it is mainly due to semantic and syntactic problems caused by incorrect integration of (1) business properties formalization and (2) the corresponding BP–task model. We therefore propose here, (a) a set of transformation rules from a subset of BPMN temporal analysis constructs into Communicating Sequential Processes+Time (CSP+T) calculus, (b) an easy verification approach for task models designed with BPMN. Differently from other authors [5,17], our method intends to merge the verification process of dynamic properties described by a choreography specification with the design of any BP model. In this way, we can take full advantage of the strengths that a formalization of behavioral and temporal aspects of BPMN models can provide to the analysis, both at design and run–time. Moreover, our aim is to facilitate the description of a BP model as a collection of verified software components, thereby allowing their complete verification with state–of–the–art model checking tools.

2 BPMN Choreographies Formalization

Business Process Model and Notation (BPMN) [12] is a standardized graphical notation that provides modeling elements for Business Process (BP) description

with an emphasis on control–flow. A *Business Process Diagram* (BPD) is a specific type of flowchart that incorporates graphical constructs tailored to model BPMN elements, such as: (a) gateways, (b) events *start, stop*, (c) tasks, and (d) control flows. A BPD is easy to use and understand by non–technical personnel, usually management–oriented, for modeling BPs. Notwithstanding, a BPD offers the expressiveness necessary to model very complex BPs and it can be mapped to different business execution languages such as BPEL [2] or XLANG [19]. Figure 1 shows the main BPMN elements as they are represented in diagrams.

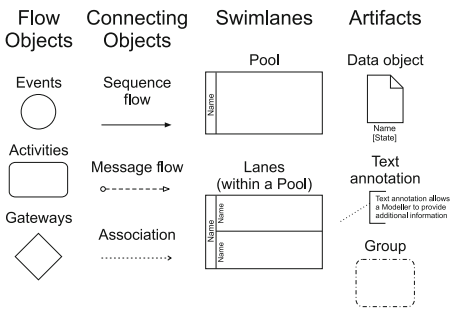


Fig. 1. Graphical representation of BPMN elements.

BPMN is a standard for the semi–formal specification of task workflows in business BP models and for describing the collaboration between services. BPMN 1.x only supported *choreographic* specifications through UML–like collaborative diagrams. The description of *conversations* between models' components was considered to suit better the low–level services languages such as WS–BPEL. However, "interaction–based" business models consider the description of "conversations" between peers as the basic building blocks of any BP system design, whereas the specification of interfaces has become secondary to the system's properties analysis. BPMN 2.0 supports now the collaboration between analysis entities in BP models, which brings forward a *choreographic* model based on peer interactions [16], instead of following a design model based on services orchestration [13].

BPMN 2.0 promotes a collaborative and abstract description of software systems that allows for focusing more on what services do in a composition than on how they do it. Interactions between system's components or *peers* should be more precisely described now than in "interconnected interface" models, within which the interactions are defined internally to each peer only. BPMN 2.0 advocates for the Web Service Choreography Description Language (WS–CDL), which better fulfills the requirements of choreography specifications due to their global perspective of the system. Interactions between peers are the basic *building blocks* of any WS–CDL specification, thus promoting the separation of concerns principle of Software Engineering. BPMN 2.0 Choreography Diagrams (BPMN-CD) describe one–way and two–way interactions between peers. In Fig. 2, the

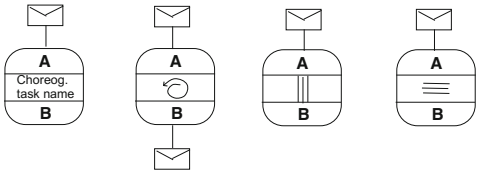


Fig. 2. Choreography diagrams.

peers A and B are represented by the upper and lower bands, respectively, of the participant tasks message–exchanging rounded box. Thus, there is a two way interaction described: peer A receives a message and peer B sends a return message. Tasks on the right of the above one include internal markers, such as the *standard loop* (the interaction is performed several times depending on a boolean condition), *multi–instance* parallel loops (the interactions are performed by several instances of the choreography task), which can execute the actions in parallel$\| \ |$ or sequentially $\equiv$.

2.1 Notation Formalization

The elements of our core subset of BPMN can either be: tasks, subprocesses, multiple instances, pools, or control gateways. These notational elements represent *states* and are linked by *Sequence*, *Exception* or *Message flows*. A *sequence flow* is used to show the order that BP states are achieved. *Exception* flows deal with events that occurs during a BP execution and redirects the normal flow. *Message* flows represent the communication between two asynchronous modeling elements in a BPD, usually BP–*participants*, which are prepared to send and receive messages. A BP flow is depicted by modeling the *Events* that occur to start the BP, the *activities* that are carried out inside the BP, and the outcome of the BP flow.

In Fig. 1, *tasks* are the lowest abstraction level components of a BP, i.e., any *Activity* that is not further decomposable into parts is considered a *Task*. Furthermore, an *Activity* in the flow can be considered a *Subprocess*, and thus graphically represented by another BPD connected via a hyperlink to a process symbol. Each task or subprocess can define multiple instances. Serial and parallel multiple instances are represented by specific state types in BPMN: *miseq* and *mipar*, which are meant to define a specific task repeated in sequence or composed in parallel, respectively. A *Pool* typically represents an *organization* or *business entity* and a *Lane* represents a *department* or a *business worker* within that organization, or other modeling entities like functions, applications, and systems. Both, pools and lanes, represent *BP participants*. Business decisions and flow branching are modelled using *Gateways*, which are similar to a decision symbol in a flowchart. The gateways are used to control how sequence flow interacts, by diverging and converging within a BP, and can define all the types of BP sequence flow behavior: branching (exclusive, inclusive, and complex decisions), merging, forking, and joining.

2.2 Denotational Semantics

The abstract syntax of a significative subset of BPMN using $\mathbb{Z}$ schemas and the set theory can be summarized as it follows:

$$Basic\_types \mathrel{\widehat=} [CName, \mathbb{P}\, Name, Task, Line, Channel, Guard, Message]$$

$$Subtypes \mathrel{\widehat=} [BName, PlName, InMsg, OutMsg, EndMsg, LastMsg]$$

$$InMsg, OutMsg, EndMsg, LastMsg : \mathbb{P}\, Message$$

$$BName, PlName, PName : \mathbb{P}\, Name$$

$$\begin{aligned} Type : {} & pgate \mid xgate \mid ogate \mid start \mid end\langle\langle\mathbb{N}\rangle\rangle \mid abort\langle\langle\mathbb{N}\rangle\rangle \mid task\langle\langle Task\rangle\rangle \\ & \mid link\langle\langle PName\rangle\rangle \mid bpmn\langle\langle BName\rangle\rangle \mid pool\langle\langle PlName\rangle\rangle \mid miseq\langle\langle Task \times \mathbb{N}\rangle\rangle \\ & \mid miseqs\langle\langle BName \times \mathbb{N}\rangle\rangle \mid mipar\langle\langle Task \times \mathbb{N}\rangle\rangle \mid mipars\langle\langle BName \times \mathbb{N}\rangle\rangle \end{aligned}$$

Each BPMN entity ([20]) has associated attributes describing its properties; for example the number of loops of a sequence multiple instance is recorded by the natural number in the constructor *miseq*. The type of a sequence flow or an exception flow is given by the following schema definition:

$$Transition \mathrel{\widehat=} [guard : Guard;\ line : Line]$$

and the type of message flow:

$$Message\_flow \mathrel{\widehat=} [message : Message;\ channel : Channel]$$

If the sequence flow has no guard or the message flow contains an empty message, then the values of Transition and Message_flow record the default values "*tt*" and empty respectively. There are five sets of message flows (*send, receive, reply, accept, break*) associated to the *state* entity, which are syntactically defined in the next type and whose function follows from their names,

$$\begin{aligned} State \mathrel{\widehat=} [{} & type : Type;\ in, out, error : \mathbb{P}\, Transition;\ exit : \mathbb{P}(\mathbb{N} \times Transition); \\ & send, receive, reply, accept, break : \mathbb{P}\, Message\_flow;\ link : \mathbb{P}(Transition \times \\ & Message\_flow);\ depend : \mathbb{P}(Message\_flow \times Message\_flow);\ loopMax : \mathbb{N}] \end{aligned}$$

Each state also incorporates the variable *loopMax* to limit the number of state instances that each process can invoke. The state's component *link* pairs the incoming message flow which initiates or interrupts the execution of the state with either an incoming transition or an exception flow. The component *depend* pairs each incoming message that initializes the state's execution with its corresponding outgoing message flow.

The formal semantics of BPMN abstracts (partial function *hide*) the internal flow of the modeling entity named *state* and only describes the sequence of initializations and terminations with the semantic function *bpmn*.

$$
\begin{array}{l}
[bpmn] \\
bpmn : \mathbb{P}\ Name \nrightarrow Local \nrightarrow Process \\
hide : \mathbb{P}\ Name \nrightarrow Local \nrightarrow \mathbb{P}\ Event \\
\hline
\mathsf{function\_name} = (Y \mid [\alpha Y]X) \backslash hide(\!|\ S\ |\!) \\
\mathsf{Where}\ X = \Box_i : \alpha Y \backslash hide(fin, abt) \bullet \\
(i \rightarrow X \,\Box\, fin \rightarrow null \,\Box\, abt \rightarrow stop) \\
\mathsf{And}\ Y = (\|_i : Process\_set \bullet \alpha(P_i) \circ P(i))
\end{array}
$$

The partial function *bpmn* maps a syntactic description of a BPMN diagram encapsulated by a *pool* or BPMN–*subprocess* into a parallel composition of CSP+T subprocesses, which correspond to the diagrams of the basic elements of BPMN shown in Fig. 2. The set αY represents the communication alphabet that includes all the messages types and events that may affect the execution of processes. These communications represent the events that a process P receives from its *environment* (made up of all the other processes in the system) or those events that occur internally, i.e., those which are not externally visible.

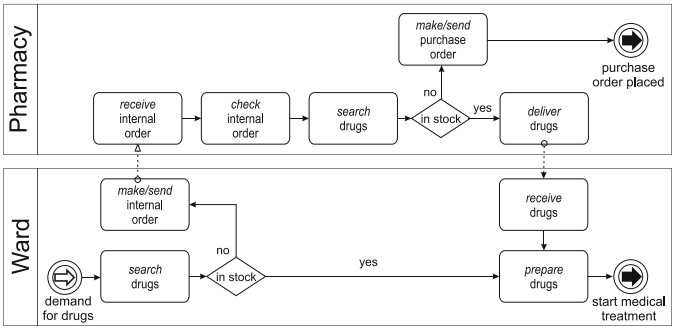


Fig. 3. Example of a BPMN diagram.

2.3 Example

Consider, for instance, the simple example of an Hospital Pharmacy logistic process shown in Fig. 3. The BPD depicts the message flows between two participants, the *Ward* and the *Pharmacy*, which are independent BP and may have been constructed separately. Clearly, the *synchronization* between both participants is a necessary behavioral property for successful collaboration.

For example, from the pharmacy participant's perspective, drugs delivering can be guaranteed by receiving a message from the ward participant prior a purchase order would be made or sent. From the ward participant's perspective, to be sure to have the required drugs to begin a medical treatment, a message to the pharmacy participant, with enough time in advance, must be sent. The graphical description should describe the interaction between the peers

(customer, ward, pharmacy, db) translated from the Hospital Pharmacy BPD shown in Fig. 3, which represent the behavior of tasks within the pools *Ward* and *Pharmacy*. In this specification, we can firstly see that the customer interacts with the ward (`demand for drugs`), then sends the prescription to the pharmacy (search drugs) and eventually receives a response if the drug is in stock. On the contrary, the pharmacy makes an order (purchase order) and when the drug is available, it delivers the drug to the ward, which goes into (`receive drugs`) state. In either case, the prescription will be prepared and given to the customer to (start medical treatment), thereby terminating the complete protocol.

3 CSP+T

CSP+T [21] is a real–time specification language which extends *Communicating Sequential Processes* (CSP) allowing the description of complex event timings, within a single sequential process, for use in the behavioral specification. The CSP+T language is defined by the rules,

$$
\begin{array}{ll}
SKIP & :\equiv \textit{success} \text{ (successful termination)} \\
STOP & :\equiv \textit{deadlock} \\
t_a.a \rightarrow P & :\equiv \text{a occurs at } t_a, \text{ then } P \text{ (prefix)} \\
t0.\bigstar \rightarrow \widetilde{P} & :\equiv (\bigstar \wedge s(\bigstar) = t0) \text{ then } \widetilde{P} \text{(instant.)} \\
t_a.a \bowtie v \rightarrow P & :\equiv (t_a.a \wedge s(a) = t_a) \text{ then } P \text{ (marker)} \\
P \fatsemi Q & :\equiv P \text{ (successfully) followed by } Q \\
P \sqcap Q & :\equiv P \text{ or } Q \text{ (non–deterministic)} \\
P \Box Q & :\equiv P \text{ choice } Q \text{ (external choice)} \\
P \backslash A & :\equiv P \text{ without } A \text{ (hiding operator)} \\
P \triangle Q & :\equiv P \text{ interrupted by } Q \\
\mathrm{I}(T, t_1).a \rightarrow P & :\equiv (t_a.a \wedge t_a \in [rel(t_1, v), rel(t_1 + T, v)]) \\
 & \quad \text{then } P \text{ (event–enabling interval)} \\
\mathrm{I}(T, t_1) \rightarrow \widetilde{P} & :\equiv t > rel(t_1 + T, v) \text{ then } \widetilde{P} \text{ (delay of T)} \\
P \| Q & :\equiv P \text{ in parallel with } Q \text{ (composition)} \\
P \mid [A] \mid Q & :\equiv P \text{ in parallel with } Q \text{ in alphabet } A \\
 & \quad \text{(alphabetized composition)} \\
P \,|||\, Q & :\equiv P \text{ interleave } Q \text{ (interleaving)} \\
\mathrm{I}(T_a, t_a).a \rightarrow P \mid [A] \mid & :\equiv P \| Q \text{ if } (a = b) \wedge (\mathrm{I}(T_a, t_a) \cap \mathrm{I}(T_b, t_b) \\
\quad \mathrm{I}(T_b, t_b).b \rightarrow Q & \quad \neq \emptyset) \\
 & \quad P \,|||\, Q \text{ if } (a \neq b) \wedge (\mathrm{I}(T_a, t_a) \cap \mathrm{I}(T_b, t_b) \\
 & \quad \neq \emptyset) \\
 & \quad STOP \text{ if } \mathrm{I}(T_a, t_a) \cap \mathrm{I}(T_b, t_b) = \emptyset \\
\mu\, X \bullet P & :\equiv \text{the process } X \text{ such that } X = P(X) \\
\Box_{i=1}^{m} : N \bullet P(i) & :\equiv i : N \rightarrow P(i) \text{ (external choice)} \\
\bigsqcap_{i=1}^{m} : N \bullet P(i) & :\equiv P((\tau -) action) \text{ (internal choice)} \\
|||_{i=1}^{m} : N \bullet P(i) & :\equiv i : N \rightarrow |||_{i=1}^{m} P(i) \text{ (indexed)} \\
\|_{i=1}^{m} [A] : N \bullet P(i) & :\equiv i : N \rightarrow \|_{i=1}^{m} P(i) \text{ (partial)} \\
\|_{i=1}^{m} : N \bullet A(i) \circ P(i) & :\equiv i : N \rightarrow \|_{i=1}^{m} A(i) \circ P(i) \\
 & \quad \text{(parallel combination of processes)}
\end{array}
\tag{1}
$$

A CSP+T process term $\mathcal{P}$ is defined as a tuple $(\alpha P, P)$, where $\alpha P = Comm\_act(P) \cup Interface(P)$ is the *communication alphabet* of P. CSP+T is

a superset of CSP, the latter being changed by the fact that traces of events become *pairs* denoted as $t.a$, where t is the time at which event a is observed. Given the communication alphabet Σ, $\mathcal{M}$ a set of marker variables, $\mathcal{T}$ a set of times instants, $\mathcal{I}$ a set of time intervals, $\mathcal{P}$ a set of process names, and the function $s(t_a.a)$ that returns the time at which symbol a occurs.

The event enabling interval $\mathrm{I}(T, t_a) = \{t \in \mathcal{T} \mid rel(t_a, v) \leq t \leq rel(t_a+T, v)\}$ indicates the time span where any event is accepted. $rel(x, v) = x + v - t_0, t_0$ corresponds to the preceding *instantiation event* ($\bigstar$), occurred at some absolute time t_0, and x is the value held in the *marker variable* v at that time. The time interval expression can be simplified to $\mathrm{I}(T, t_a) = [t_a, t_a + T]$ if the instantiation event, after which the event a can occur, corresponds to the origin ($t_0 = 0$) of the rt-clock. Where $a, \bigstar \in \Sigma$ (communication alphabet); $A, N \subseteq \Sigma$; $v \in \mathcal{M}$ (marker variables); $\mathrm{I} \in \mathcal{I}$ (time intervals); $P, Q, X, \widetilde{P} \in \mathcal{P}$ (process names); $t_0, t_a, t_1 \in \mathcal{T}$; and $T \in \mathbb{N}$ (time instants), and the function $s(t_a.a)$ which return the occurrence time of symbol a.

The specification in Fig. 3 can be directly translated into WS-CDL, but transforming the interactions that occur in the choreography still needs a lot of interpretation work. There is a gap to fill up between the specification expressed as a choreography diagram and its instantiation as a standard description language code. WS-CDL is more an implementation language than a specification notation for BPMN 2.0 choreographies. Therefore, we need an abstract notation that naturally supports the interaction relationships that occur between the peers of any BPMN 2.0 choreography diagram, and gives a direct mapping between those syntactical constructs and the corresponding ones in WS-CDL.

4 Reification of Choreographies

Any choreography can be considered as *realizable* if all the interactions that we have specified in the BPMN 2.0 diagram are equivalent to those that can be executed by the interacting peers when we implement the model in a service–description language, such as WS-CDL.

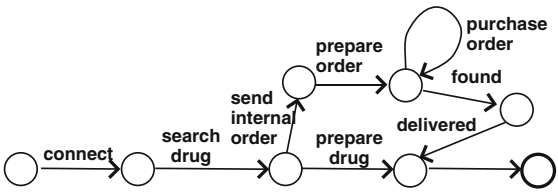


Fig. 4. LTS model of the Ward–Pharmacy example.

In general, a Labelled Transition System (LTS) [3] model can *reify* a choreography (see the one for the example in Fig. 4) and allows the verifier to check its realizability w.r.t. the model of the system composed of interacting peers.

If the two models mentioned above are *behaviorally* equivalent, it means that the peer generation exactly satisfies the BPMN communication requirements. On the contrary, if the peers do not generate the same interactions as the ones specified in the choreography, we can say that this choreography is unrealizable.

We use the concept of *trace refinement* [18] to check the realizability of choreographies, formally described as CSP+T process terms. A *trace* ($< \text{tr} >$) represents the sequence of indivisible or *atomic* interactions that each peer in a choreography must go through until reaching a final state. The set of all these interleavings is called the *behavior* of a choreograhy.

The transformation of choreographies into CSP+T process terms sets the ground for performing behavioral verification of constituent components of any BP model by using the FDR2 MC tool [6]. FDR2 checks the behavioral equivalence of two models written as CSP process terms through a refinement relationship between syntactical process terms [8].

4.1 Behavioral Equivalence

We place both, the choreography reification and the interacting peers model, in the same semantic domain, that of CSP+T process calculus. In this way we can take full advantage of the strengths that a formalization of behavioral aspects of BPMN models provide to the analysis, both at design and run–time.

The proposed procedure consists of the following integrated steps,

1. The choreography reification (LTS) derived is generated.
1.′ Transform the LTS into CSP+T.
2. The peers' behavior are extracted from of the initial BPMN–CD by the application of a proposed transformation rules set.
3. From the extracted interacting peers, the system model is built as a parallel composition of structured CSP+T process terms.
4. The choreography reification is *model–checked*. To prove behavioral equivalence with the peer-based and distributed system model.

4.2 BPMN to CSP+T Transformation

We need the semantic precision given by a formal language to the basic analysis entities in order to be able to correctly describe a fully executable *choreography diagram* (CD), such as the ones shown in Fig. 3. BPMN 2.0 defines advanced constructs, such as different types of OR decision gateways, multiple instances of tasks and subprocesses, which may be transformed into CSP+T process terms, thereby preserving the interaction information between the participant tasks. In Sect. 3, a semantic definition of CSP+T syntactical constructs can be seen.

There have been several proposals to formalize BPMN 2.0 non–temporal constructs with process algebras, [7,9,15], and the CCS–based notation proposed in [20] which we will follow in the sequel (Fig. 5).

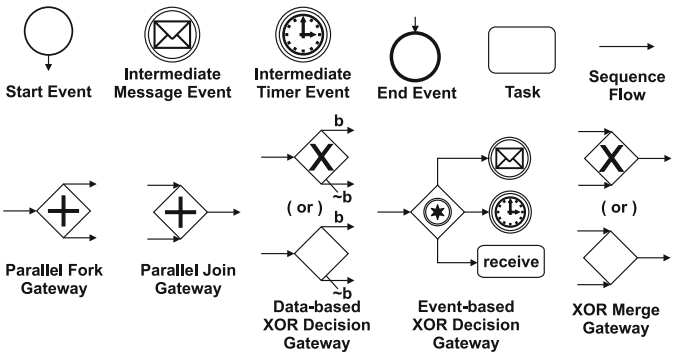


Fig. 5. BPMN elements extended graphical representation.

4.3 BPMN Temporal Extension

In many models of choreographies, constraints on time and resources appear that may cause the violation of the system's safety properties.

In BPMN 2.0 the intermediate event (*timer*) admits the definition of a delay period, but the minimum and maximum execution time allowed for any activity cannot be specified as a task element. Consequently, it will now include temporal attributes for specifying temporal constraints for *task* and other modeling elements. An intermediate event (*timer*) and a sub–process (*timeout*) represent a temporal constraint on the task flow. Notice that in BPMN 2.0 we can use the attribute of the *timer* element to define the delay period of the task. In the sequel we present a formalization of the main modeling elements that we judge necessary to include in BPMN for the specification of temporal constraints.

Start Event. The schema labelled *Start event* represents the necessary instantiation of an activity prior to the start of its execution. To allow this in CSP+T, the specification of this event is represented by means of the ★ *instantiation event* according to the following pattern,

$$P(start) = \bigstar \bowtie v_{\bigstar} \rightarrow SKIP \fatsemi (P(S1) \Box \epsilon_{end} \rightarrow SKIP)$$

Minimum and Maximum Duration Time of an Activity. The invocation of activities that make up a BP must be performed timely. Let *bpmn* $S1$ be the activity that comes before activity *bpmn* $S2$ (Fig. 6). Thus, it should be guaranteed that the execution time of *bpmn* $S1$ activity does not overrun its maximum range of duration ($S1.ran.max$) and it does not occur before its minimum starting time ($S1.ran.min$) elapses. If the above times are not controlled, the activity fails and thus we cannot certify the temporal properties of the business process.

Thus, the occurrence of ϵ_{S2} must satisfy the following condition,

$$v_{S1} + S1.ran.min \leq s(\epsilon_{S2}) \leq v_{S1} + S1.ran.max.$$

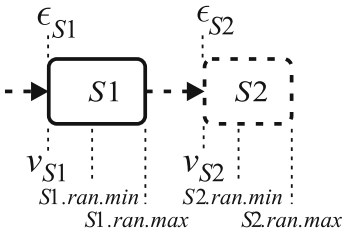


Fig. 6. Temporal annotations of the activity

With CSP+T we are allowed to precisely specify the time frame for the execution of an activity. The CSP+T pattern corresponding to two consecutive activities ($S1$ and $S2$) is as it follows,

$$P(S1) = \epsilon_{S1} \bowtie v_{S1} \rightarrow SKIP \fatsemi (\mathrm{I}(Time, v_{S1} + S1.ran.min).\epsilon_{S2} \rightarrow SKIP \fatsemi P(S2) \\ \Box \epsilon_{end} \rightarrow SKIP) \\ Time = S1.ran.max - S1.ran.min$$

The measured range values $S_x.ran.min$ and $S_x.ran.max$ depend on the occurrence of event ϵ_{Sx}.

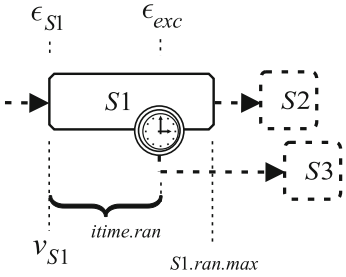


Fig. 7. Timed exception flow

Timed Exception Flow. BPMN proposes two versions of type boundary event to represent timeouts. *Timer boundary* represents the occurrence of an event ϵ_{exc} (see Fig. 7) that either triggers a parallel thread or interrupts the execution of activity *bpmn* $S1$ at *itime.ran* time units after the inception of S_1. Hence, there are two instances of the modeling entity *timer boundary event*, depending on whether the activity started at ϵ_{S1} is interrupted or not when this event occurs.

The *itime.ran* time period must be therefore constrained by the maximum time limit that we have associated to any activity.

$$(itime.ran < S1.ran.max) \wedge (s(\epsilon_{exc}) = v_{S1} + itime.ran) \wedge (s(\epsilon_{exc}) \in \\ [v_{S1}, S1.ran.max))$$

To specify the behavior denoted by a *interrupting timed exception flow*, we use the *interrupt* operator ($\triangle$) in CSP+T, according to the following pattern,

$$
\begin{aligned}
P(S1) = & (\epsilon_{S1} \bowtie v_{S1} \rightarrow SKIP \fatsemi \\
& (\mathrm{I}(Time, v_{S1} + S1.ran.min).\epsilon_{S2} \rightarrow \\
& (SKIP \fatsemi P(S2) \bigtriangleup \mathrm{I}(itime.ran, v_{S1}) \rightarrow SKIP \fatsemi \epsilon_{exc} \rightarrow \\
& SKIP \fatsemi P(S3)) \\
& \Box \epsilon_{end} \rightarrow SKIP)) \\
& Time = S1.ran.max - S1.ran.min
\end{aligned}
$$

To represent a *non–interrupting exception flow*, we only need to substitute the *interrupt* operator ($\triangle$) for the parallel operator $\|$ or the interleaving $|||$ of activities in CSP+T.

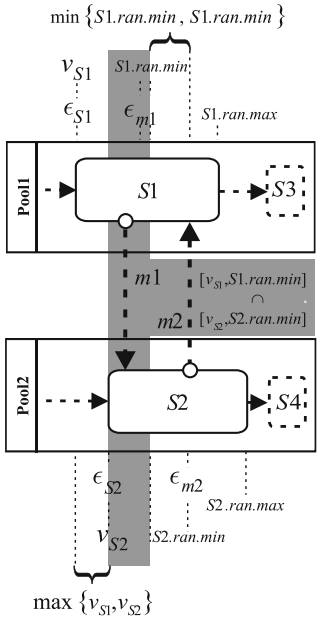


Fig. 8. Service task message flow

Service Tasks. This modeling element is intended to represent automated actions in the BPMN 2.0 metamodel and assumes synchronous communication. The *service task* is used for making synchronous service invocations performed by some external system, with receipt of that system's response.

The extended BPMN proposal affects *service tasks* in that message sending must be carried out within the duration time limits of the activity that sends, and message reception must be within the duration limits of the activity that receives. This guarantees the activities involved in communication does not enter in a

deadlock state; i.e., in an indefinite waiting for message. According to the schema, named *Message Flow*[1] for service tasks, the above conditions are specified by the following expressions,

$$(1)\quad (v_{S1} < s(\epsilon_{m1}) \leq (v_{S1} + S1.ran.min)) \wedge (v_{S2} < s(\epsilon_{m1}) < (v_{S2} + S2.ran.max)) \quad (2)$$

$$(2)\quad (v_{S2} < s(\epsilon_{m2}) \leq (v_{S2} + S2.ran.min)) \wedge (v_{S1} < s(\epsilon_{m2}) < (v_{S1} + S1.ran.max)) \quad (3)$$

or $s(\epsilon_{mx}) \in [v_{S1}, v_{S1} + S1.ran.min] \cap [v_{S2}, v_{S2} + S2.ran.min] \neq \emptyset$ and $s(\epsilon_{mx}) \in [v_{S1}, v_{S2} + S1.ran.max] \cap [v_{S1}, v_{S1} + S2.ran.max] \neq \emptyset$

We can see in Fig. 8 that communication must take place within the enabling intervals of activities $S1$ and $S2$. Or equivalently, message communications, denoted by occurrence of messages ϵ_{m1} and ϵ_{m2}, must satisfy:

$$s(\epsilon_{m1}), s(\epsilon_{m2}) \in [\max\{v_{S1}, v_{S2}\}, \min\{v_{S1} + S1.ran.max, v_{S2} + S2.ran.max\}]$$

The schemas of process terms that include communication between the activities *bpmn* $S1$ and *bpmn* $S2$ must therefore guarantee that rules (2) and (3) are always satisfied, according to the following patterns,

$$\begin{aligned} P(S1) = P(\epsilon_{S1}) \fatsemi (\mathrm{I}(MI, MA).P(!(m_1, s(\epsilon_{S1}))) \fatsemi \mathrm{I}(MI, MA).P(?(y, v_{S2})) \fatsemi \\ \mathrm{I}(Time, v_{S1} + S1.ran.min).P(\epsilon_{S3}) \fatsemi P(S3) \\ \Box P(\epsilon_{end.1})) \end{aligned}$$

$$\begin{aligned} P(S2) = P(\epsilon_{S2}) \fatsemi (\mathrm{I}(MI, MA).P(?(x, v_{S1})) \fatsemi \mathrm{I}(MI, MA).P(!(m_2, s(\epsilon_{S2}))) \fatsemi \\ \mathrm{I}(Time, v_{S2} + S2.ran.min).P(\epsilon_{S4}) \fatsemi P(S4) \\ \Box P(\epsilon_{end.2})) \\ (MI = \min\{S1.ran.min, S2.ran.min\}, \text{ and} \\ MA = \max\{v_{S1}, v_{S2}\} Time = S1.ran.max - S1.ran.min) \end{aligned}$$

5 Choreography Model Checking

For each participant in the BPMN choreography diagram (BPMN-CD), we specify a parallel composition of parallel CSP+T processes, thereby we can define a bijection between processes and diagram states. The proposed formal specification abstracts the internal interaction between the individual peer states and only represents the sequence of task initializations and terminations that occur in the choreography model, and thus a compact model susceptible of being transformed into an LTS, and then verified by an automatic tool, is obtained.

[1] The activity *bpmn* $S1$ sends the message $m1$ that is received by the activity *bpmn* $S2$ then the activity *bpmn* $S2$ responds by sending the message $m2$.

5.1 CSP+T Transformation of the Choreography Model

The states of each two interacting peers in Fig. 3 are represented by the following CSP+T process pattern,

$$
\begin{aligned}
&bpmn\\
&name{=}\ (Y \mid \alpha(P_i) \mid X)\ hide\ \{connect.Customer\}\\
&\quad Y{=}\ \|_{i=1}^{i=m}\ Tasks \bullet \alpha(P_i) \circ P(i)\\
&name{=}\ (Customer \mid Ward \mid Pharmacy \mid DB);\ (peers)\\
&Tasks{=}\ (Search\ drug \mid Send\ internal\ order\\
&\quad \mid Prepare\ order) \mid Search\ drug \mid Deliver\ drug\\
&\quad \mid Purchase\ order \mid Prepare\ drug);
\end{aligned}
$$

The parallel composition of CSP+T {P(i)} processes is mechanically obtained by applying the transformation rules in Sect. 4.2.

$$
\begin{aligned}
&bpmnWard\\
&Y_{Ward} = (P(connect) \| P(search\_drug) \| P_{xgate.1}())\\
&P_{xgate.1}(2, P(send\_internal\_order) \| P_{send.1}(request),\\
&P(prepare\_drug) \| P_{receive}(received) \| P(end)))\\
&bpmnPharmacy\\
&Y_{Pharmacy} = (P_{receive}(request) \| P(prepare\_order)\\
&\| P(search\_drug) \| P_{xgate.2}())\\
&P_{xgate.2}(2, P(purchase\_order) \| P_{send.2}(place\_order) \triangle\\
&P(abort), P(deliver\_drug))
\end{aligned}
$$

The other peers (*DB* and *Customer*) have trivial CSP+T specifications.

5.2 Choreography Reification

The LTS reification of the choreography in Fig. 4 must have an equivalent behavior as the one shown by the peers–based model of the Ward-Pharmacy. The CSP+T specification ($P(System) = Ward \,\|\, Pharmacy, \,\| \, Customer \,\|\, DB$) should be checked against the choreography–LTS before starting the implementation of the distributed system. This verification can be carried out automatically with FDR2 [6] model–checker if the LST is transformed into a CSP+T process,

$$
\begin{aligned}
&T(LTS) = t_0.\bigstar \rightarrow T(connect)\\
&\quad T(connect) = \quad \mathrm{I}((b-6)-a, a).init.Ward.search\_drug\\
&\qquad \rightarrow T(search\_drug)\\
&\quad T(search\_drug) = \quad \mathrm{I}((b-4)-(a+2), a+2).init.Ward.xgate.1\\
&\qquad \rightarrow T(xgate.1))\\
&\quad T(xgate.1) = \quad \mathrm{I}((b-3)-(a+3), a+3).init.Ward.send\_internal\_order\\
&\rightarrow T(send\_internal\_order)\\
&\quad T(send\_internal\_order) = \quad \mathrm{I}((b-2)-(a+4), a+4).init.\\
&Pharmacy.prepare\_order \rightarrow T(prepare\_order)\\
&\quad T(prepare\_order) = \quad \mathrm{I}((b-1)-(a+5), a+5).init.Ward.purchase\_order\\
&\qquad \Box purchased \rightarrow T(send\_found)\\
&\quad T(send\_found) = \quad \mathrm{I}(b-(a+6), a+6).init.Ward.delivered \rightarrow T(prepare\_drug)\\
&\quad T(prepare\_drug) = \quad \mathrm{I}(b-(a+6), a+6).init.Ward.End1 \rightarrow T(prepare\_drug)\\
&\quad T(End.1) = \quad SKIP
\end{aligned}
$$

5.3 Verification

The reification of the choreography is realizable if the set of interactions specified by the above process term $T(LTS)$ and those executed by the interacting peers in the target distributed system, specified by the process $P_{BPMN} = Customer \parallel Ward \| bpmn \parallel DB$ are the same. Thus, according to *traces and failures* semantics of CSP, it must be ascertained that the following refining assertion is true,

$$T(LTS) \sqsubseteq_F \ P_{BPMN} \tag{4}$$

However, the FDR2 returned *false* since the trace <connect,search drug, send order, prepare order, deliver, prepare drug > appears in both models, but the trace < connect, search drug, send order, prepare order, deliver, prepare drug, purchaseorder, abort > is present in the peers–based distributed system and not in the LTS of the choreography.

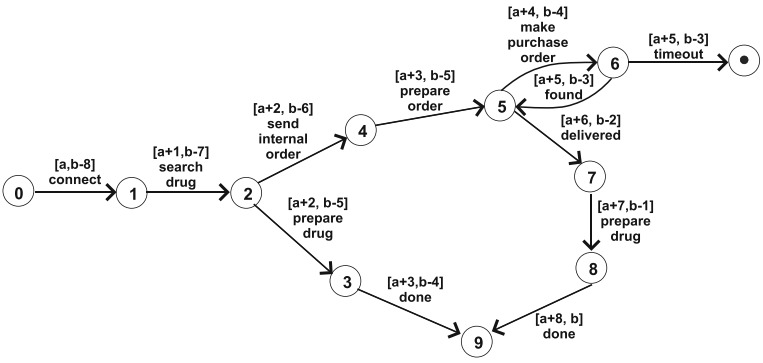


Fig. 9. Timed LTS model of the Ward-Pharmacy

The solution to this error in the LTS model is to make explicit an extra state in which the LTS is waiting for completing the purchase of the drug and add a timeout to this state If that time period expires then the LTS will reach an `abort` state signifying that the purchase is cancelled, since probably the distributor's stock has been exhausted. The new LTS can be seen in Fig. 9.

6 Conclusion

In order to enforce the realization of feasible choreographies within the realm of business proceses, we have presented a feasible formalization of BPMN 2.0 constructs. The complexity of model–checking the behavioral equivalence of an LTS, which specifies the possible behavior of a highly interactive system, against the actual behavior of a peer–based distributed application has been tackled out and a feasible, conceptually manageable, solution is proposed here.

Of great importance to develop predictable and safe applications, it is to solve the *choreography realization problem*, i.e., a choreography is realizable if all the interactions specified in BPMN 2.0 diagrams are equivalent to those that can be executed by the interacting peers when the BP model is implemented in a service–description language, such as WS–CDL, and the BP rules are satisfied as well.

We have solved that problem by transformation of the LTS and the peer–system model into a process calculus named CSP+T. Process algebras, likes LOTOS–NT [14], CCS [10] or CSP+T [21] are good candidates for being chosen as the interactive abstract notation. These formal notations give expressive operators for translating BPMN 2.0 constructs Both notations, BPMN-CD and a process algebra, are high level languages and have similar operators: *sequence*, *choice*,*interleaving*. Furthermore, process algebras are supported by verification tools (like model–checkers SPIN, PVS, CADP, FDR, ...) that can be used to check all the possible behaviors that take place in the model execution of any specified choreography. In this way, we can analyze and automatically verify the conformance of the defined choreography for services that communicate through messages in a general, distributed, and highly parallel system.

References

1. van der Aalst, W.M.P.: Challenges in business process analysis. In: Filipe, J., Cordeiro, J., Cardoso, J. (eds.) ICEIS 2007. LNBIP, vol. 12, pp. 27–42. Springer, Heidelberg (2008)
2. Arkin, A., Askary, S., Bloch, B., Curbera, F., Goland, Y., Kartha, N., Liu, C.K., Thatte, S., Yendluri, P., Yiu, A. (eds.) Web Services Business Process Execution Language Version 2.0. Committee Draft. WS-BPEL TC OASIS (2005)
3. Bezem, M., Klop, J.W., de Vrijer, R.: "Terese" Term Rewriting Systems. Cambridge University Press, Cambridge (2003)
4. Cerone, A.: From process algebra to visual language. In: Proceedings of the Conference on Application and Theory of Petri Nets: Formal Methods in Software Engineering and Defence Systems, vol. 12, Adelaide (2002)
5. van Dongen, B.F., van der Aalst, W.M.P.: Multi-phase process mining: building instance graphs. In: Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T.-W. (eds.) ER 2004. LNCS, vol. 3288, pp. 362–376. Springer, Heidelberg (2004)
6. Formal Systems Europe Ltd. Failures-Divergence Refinement - FDR2 User Manual. Formal Systems Europe Ltd., Oxford (2005)
7. Ma, S., Zhang, L., He, J.: Towards formalization and verification of unified business process model based on Pi calculus. In: Proceedings ACIS International Conference on Software Engineering Research, Management and Applications, pp. 93–101 (2008)
8. Mendoza, L.E.: Una Contribución a las Técnicas Avanzadas de Verificación de Procesos de Negocio (In Spanish). Ph.D. Dissertation book, University of Granada (ISBN:978-980-12-4957-3) (2011)
9. Mendoza, L.E., Capel, M.I., Pérez, M.A.: Conceptual framework for business processes compositional verification. Inf. Softw. Technol. **54**, 149–161 (2012)
10. Milner, R.: Communication and Concurrency. International Series in Computer Science. Prentice Hall, Englewood Cliffs (1989). ISBN 0-13-115007-3

11. Web Services Business Process Execution Language Version 2.0 (2007). http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf
12. OMG. Business Process Model and Notation (BPMN) -version 2.0
13. Peltz, C.: Web services orchetration and choreography. IEEE Comput. **36**(10), 46–52 (2003)
14. Poizat, P., Salaûn, G.: Checking the realizability of BPMN 2.0 choreographies. In: Proceedings 27th Simposium of Applied Computing, Riva del Garda (Italy), March 25–29, pp. 1927–1934. ACM (2012)
15. Puhlmann, F.: Soundness verification of business processes specified in the pi-calculus. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 6–23. Springer, Heidelberg (2007)
16. Qiu, Z., et al.: Towards the theoretical foundation of choreography. In: Proceedings of the 16th International Conference on World Wide Web (WWW'07), pp. 973–982 (2007)
17. Rozinat, A., van der Aalst, W.M.P.: Conformance testing: measuring the fit and appropriateness of event logs and process models. In: Bussler, C.J., Haller, A. (eds.) BPM 2005. LNCS, vol. 3812, pp. 163–176. Springer, Heidelberg (2006)
18. Schneider, S.A.: Concurrent and Real-Time Systems - The CSP Approach. Wiley, Chichester (2000)
19. Thatte, S.: XLANG: Web Services for Business Process Design. Microsoft Corporation (2001). http://www.gotdotnet.com/team/xml/wsspace/xlang-c
20. Wong, P.Y.H., Gibbons, J.: A process semantics for BPMN. In: Liu, S., Araki, K. (eds.) ICFEM 2008. LNCS, vol. 5256, pp. 355–374. Springer, Heidelberg (2008)
21. Zic, J.: Time-constrained buffer specifications in CSP+T and timed CSP. ACM TOPLAS **16**(6), 1661–1674 (1994)

Author Index