Model-Based Relationship Management for Service Networks

Aneta Kabzeva, University of Kaiserslautern, Kaiserslautern, Germany

Joachim Götze, University of Kaiserslautern, Kaiserslautern, Germany

Paul Müller, University of Kaiserslautern, Kaiserslautern, Germany

ABSTRACT

With the broad adoption of service-orientation for the realization of business applications and their provisioning and usage over cloud infrastructures, the topology of the resulting service networks is becoming extremely complex. Due to the composition of services for value-added business capabilities and the reusability of a service in multiple compositions, the execution of one service often depends on other services and changes in its provisioning can affect the health of large parts of the service network. The lack of insight on the relationships between the network components makes the management of the service network's health hard and error prone tasks. This article introduces a service network modeling approach for capturing the topology of a service network at design time. The approach considers the complete modeling process from representation, through collection, to analysis of the relationship information. The major contributions are a generic and adaptable modeling structure, a classification of service network entities and relationships, and a modular management framework automating the modeling process.

Keywords: *Adaptability, Evolution Support, Model Analysis, Model Extraction, Model Representation, Service Network Model, Socio-technical Model, Uniformity*

INTRODUCTION

While service orientation is broadly adopted for the realization of intra-organizational business applications, the provisioning and usage of services over cloud infrastructures provides access to new markets and introduces new possibilities for application design. With the composition of services for the provisioning of cross-organizational value-added capabilities and the reusability of a service in multiple compositions, complex interconnections are introduced between the loosely coupled services.

In event-driven SOA, the occurrence of an event serves as a trigger to the invocation of one or more business processes and associated services. These services are especially loosely coupled; hence, the need to manage relationships between services as well as between business

DOI: 10.4018/IJSSOE.2015100105

processes and services becomes predominant. Only an explicit and detailed relationship management can satisfy the demand for governance and infrastructure knowledge as well as the specific requirements of service health analysis.

Explicit relationship modeling offers an instrument for tracing and understanding the complex interactions between the actors and the resources involved in the emergence of a service network through the combination of services in the context of applications. It has been recognized as an important factor for analyzing the quality of the network and the applications it comprises. As a result, a number of models have been defined on different abstraction levels to support the various service selection needs of service consumers, the impact analysis for service providers, or the value flow within the network for service network operators. Usually, a single actor can interact with different roles within the network. The lack of standardized vocabulary and modeling notation as well as the rigidity of the modeling concepts make the integration of information for novel analysis purposes impossible.

This article introduces a concept of an explicit relationship model that serves as foundation for understanding the interconnections in service networks and for the purpose of health analysis support. Data originating from different abstraction levels is integrated into an exhaustive service network model to provide a uniform representation of the network topology. The DAME (Dependency Analysis and Management for Evolution) modeling approach is generic enough to provide independence from the modeling notation used for the specification of the service network components. The scope of knowledge captured in the model includes information of interest for the different stakeholder perspectives of service provider, consumer, and operator relevant in the context of service networks. Model view extraction reduces the model content's complexity and provides detailed data based on the specific analysis needs.

A uniform approach for capturing and representation of big amounts of relationships data is required to achieve standardization for describing service network models. It should be expressive enough to enable dynamic adjustment of the sets of entities and relationships between them according to changing needs of the network and its stakeholders. A metamodel-based modeling concept is described that provides a generic language schema to allow the integration of relationship information between the social, business, and technical abstraction levels of a service network. The service network model considers equivalence, associative, and hierarchical relationships between resources and actors contributing to the network.

In detail, the article will discuss three main questions. First, what different types of entities and relationships should be considered for service network modeling? Second, how to present these types of entities and relationships in a common model understandable for stakeholders from different domains of expertise? Third, how to provide automated support for model capturing, validation and representation?

BACKGROUND

As a basis for the metamodel-based modeling approach for service networks, a thorough understanding of terms and related work is required. Therefore, a short overview of the basic structuring approaches of services as well as service networks in an enterprise context will be introduced. Additionally, related work will be discussed and compared to the service network modelling approach presented in this article.

Service-Oriented Architecture

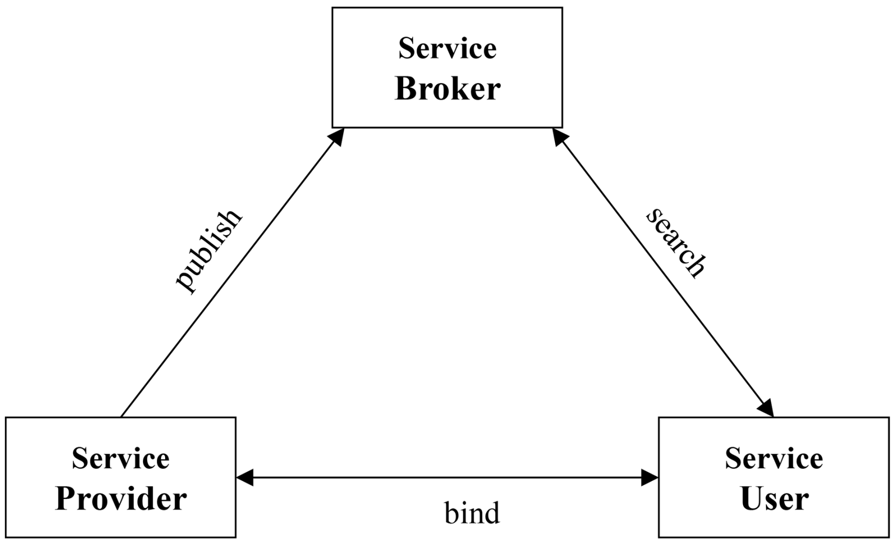
Services are at the core of a service-oriented architecture. According to (Erl, 2009) a service is a unit of solution logic, which has been created by applying the service-oriented design paradigms. In order to offer and use services, three roles are of importance: service user, service provider, and service broker (cf. Figure 1). In the lead-up, the service provider registers the service with the service broker. The service broker accepts search queries of the service user for suitable services. The service user then makes use of such a service and requests the offered service.

Designing a service broker, two approaches can be distinguished. First, a service broker can be designed as a registry. Such a registry manages references of services based on categories with respect to the service provided, e.g., Universal Description, Discovery, and Integration (UDDI). Second, a service broker really offers a brokering functionality. Therefore, properties and metadata of the services are collected such that the service broker can provide a suitable service based on the required characteristics of the service user.

Beyond accessing a single service, the question arises how a service-oriented architecture should be structured and what is a reasonable functionality a service should comprise. Usually, an enterprise architecture consists of the business process layer describing abstract business processes and the application layer focusing on implementing these business processes (cf. Figure 2). Multiple SOA reference architectures (Erl, 2007) (Krafzig, Banke, & Slama, 2005) (Josuttis, 2007) introduce the service layer in order to bridge the gap between business logic and process implementation (cf. Figure 3). The service layer consists of services providing the base elements of business processes as well as abstractions from the underlying technologies. In order to guide the design of these services, either a top-down approach primarily concerned with business processes or a bottom-up approach, i.e., a technology-oriented approach, can be used.

In the top-down approach, the service layer is structured further from top to bottom by the orchestration service layer, the business service layer, and the application service layer. The orchestration layer focuses on the implementation of the abstract business processes described

Figure 1. SOA Triangle (adopted from (Erl, 2005))



in the business process layer. The business service layer consists of services containing business logic utilized by tasks of the business processes. The application service layer comprises solution-agnostic services making technology-specific functionality available offered by the application layer.

In the bottom-up approach, the service layer is structured further from bottom to top by the basic service layer, the composed service layer, and the process service layer. The basic service layer provides basic inseparable business functionalities. Typically, these services are short-term running and conceptually stateless. Composed services represent services that are composed of other services (either basic or other composed services). These services operate at a higher level than basic services, but still these services are short-term running and conceptually stateless. In workflow terms, a composed service represents a micro flow, which is a short-running flow of activities (services) inside one business process. Services of the process service layer represent long-term workflows or business processes. As such, a process service represents a macro flow, which is a long-running flow of activities (services) that is even interruptible by human intervention. Unlike basic and composite services, a process service usually has a state that remains stable across multiple calls.

Figure 2. Pre-SOA system abstraction layers

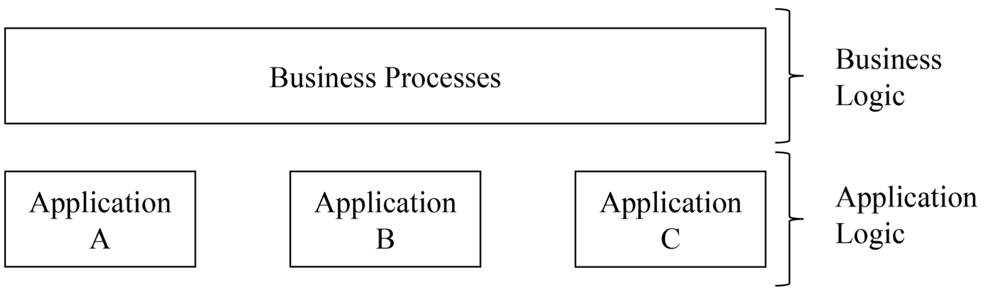
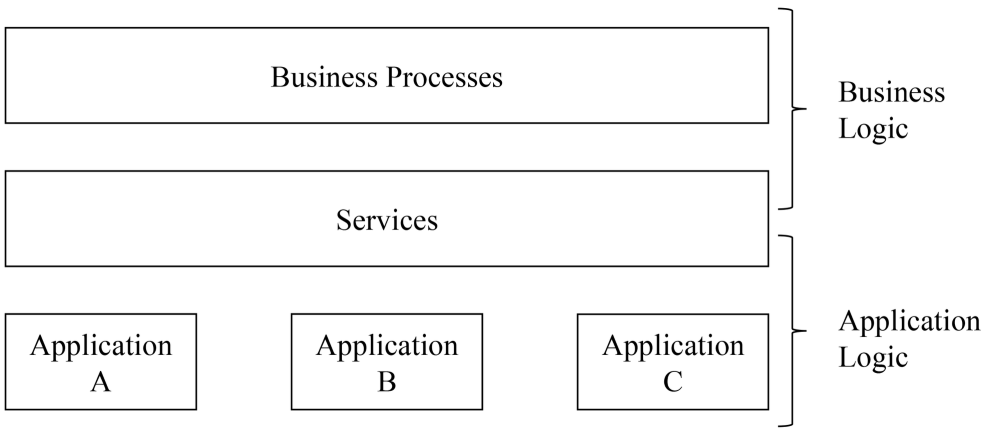


Figure 3. SOA system abstraction layers



Service Networks

Introducing services as a core component is a conceptual and technical transition in the software delivery model allowing service providers to concentrate on their core competencies. Additional functionality needed for the realization of a complete business process is integrated through the cooperation with third-party services (Jansen, Brinkkemper, & Finkelstein, 2009). With the on-demand delivery of flexible service-based components, service consumers are given the opportunity to select the most appropriate service offer for their needs, without binding to a single vendor. Moreover, consumers can compose services (Erl, 2007) from different providers and construct a customized solution for their specific requirements. The service-centric webs emerging through the composition of cross-organizational applications are referred to as service networks (SNs).

Due to the reusability of loosely coupled services in the transparent compositions within a service network and their distributed administrative control across the different providers, the evolution and maintenance of such systems are still an open research issue (Papazoglou, 2008). Understanding the interactions within SNs has been recognized as an important factor for analyzing the impact of changes on the quality of the network and the applications it comprises. As a result, a number of new models (Basole & Rouse, 2008) have been defined to support the analysis of a service network from the different perspectives of service providers, consumers, or network operators, the interactions between them, and the resulting knowledge creation and exchange. However, a standardized definition for the term service network, its scope with respect to business and technical aspects, and its form of documentation (Basole & Rouse, 2008) is still missing. Therefore, the different modeling approaches apply different concepts for the specification of various relationships within a service-oriented application landscape.

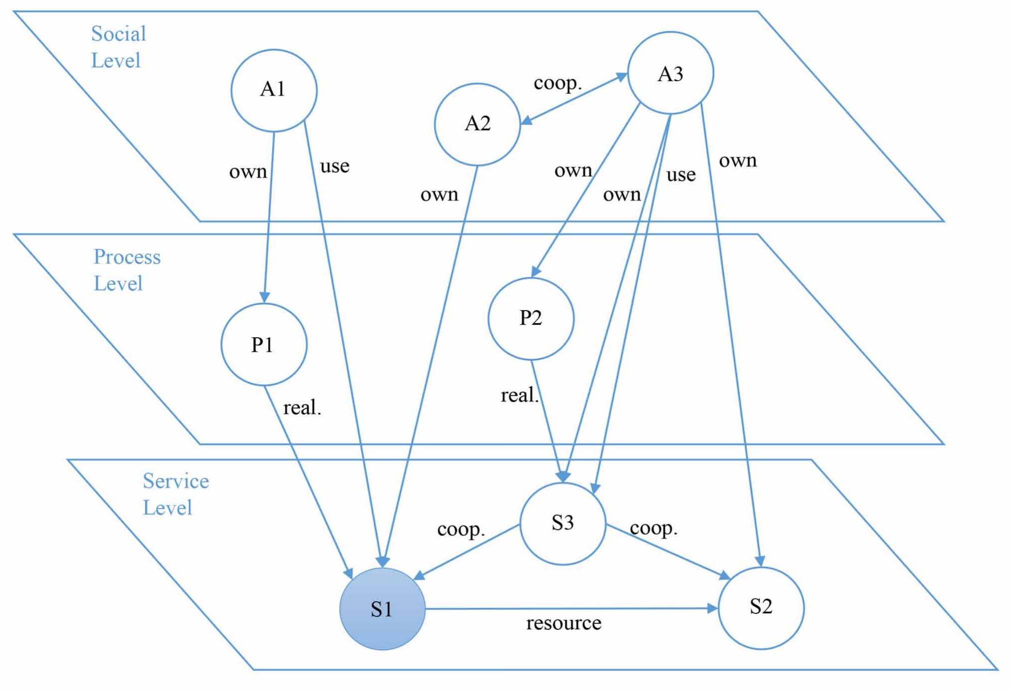
In order to be able to model a service network it is important to extend the view of the SOA reference models. Therefore, the concept based on the three layers business process layer, service layer, and application layer needs to be extend to incorporate the different stakeholders and their perspectives. This extension is called Integrated Service Network Model (cf. Figure 4) and contains the new stakeholder layer on top. Hence, the stakeholder perspectives introduce a social level to the modeling approach and allow to model not only relations between business entities and technical entities, but amongst social entities as well.

Related Work

Numerous research works are focusing on the definition of service network models covering a wide scope of application scenarios. Some of them apply the models for value flow calculation (Allee, 2000) (Gordijn & Akkermans, 2001), others for the derivation of possible service compositions (Blau, Kramer, Conte, & van Dinther, 2009), or providing an overview on business relationships (Danylevych, Karastoyanova, & Leymann, 2010). Cause-effect correlations analysis and service market share (Cardoso, Pedrinaci, & De Leenheer, 2013), service performance tracing (Wang, Taher, & van den Heuvel, 2012), and documentation of collaborations for risk-aware management (Liu, Fan, & Huang, 2013) (Schulz, et al., 2013) are also considered as motivation for the design of service network models. Depending on the application domain for a model, the scope of the considered model components and the nature of the relationships between them vary greatly.

The value-based model in (Allee, 2000) remains completely on the business entities level of service networks, modeling services, knowledge, and intangible value exchange between organizations. Business tasks are added to the business entities level in the models in (Gordijn & Akkermans, 2001) and (Danylevych, Karastoyanova, & Leymann, 2010). The relationships in (Gordijn & Akkermans, 2001) remain on the value flow level, while (Danylevych, Karastoyanova,

Figure 4. Integrated service network model (example) (Kabzeva, Götze, & Müller, 2014)



& Leymann, 2010) focuses on the ownership of offerings and requests, and the provisioning relations between them.

The model in (Cardoso, Pedrinaci, & De Leenheer, 2013) is based on rich relationships with multiple properties referred to as multi-layer relationships. A multi-layer relationship connects only services and is defined along six dimensions: role, level, involvement, comparison, association, and causality. Dependencies between services are also the focus of the model in (Schulz, et al., 2013). Here however, the service level agreements of a service are separately modeled and build their own dependency topology.

The approach in (Blau, Kramer, Conte, & van Dinther, 2009) proposes a formalism for modeling service networks as demand-driven, ad-hoc interplay of providers and their services. Each model targets the satisfaction of one complex service request and describes all feasible combinations of services that may satisfy the request. The best alternative can be selected by calculating the minimal costs noted on the relationship edges. The scope of our work is different, in that we support the modeling of existing relationships rather than possible ones. Existing collaboration relationships and competition relationships between services are considered in (Liu, Fan, & Huang, 2013). Provisioning and consumption relationships to the business entities are also defined as part of the approach. However, compared to our approach no possibility for adaptation or weighting of the relationships is considered. The modeling concept in (Wang, Taher, & van den Heuvel, 2012) is the only one considering all three abstraction levels of service networks including the business entities, the process, and the service level. However, the approach does not specify concrete relationships between the layer participants. The information on customer involvement, participant interaction, granularity, human operation involvement, software service involvement, and dependency are modeled as part of a service-centered assessment.

Motivation Scenario

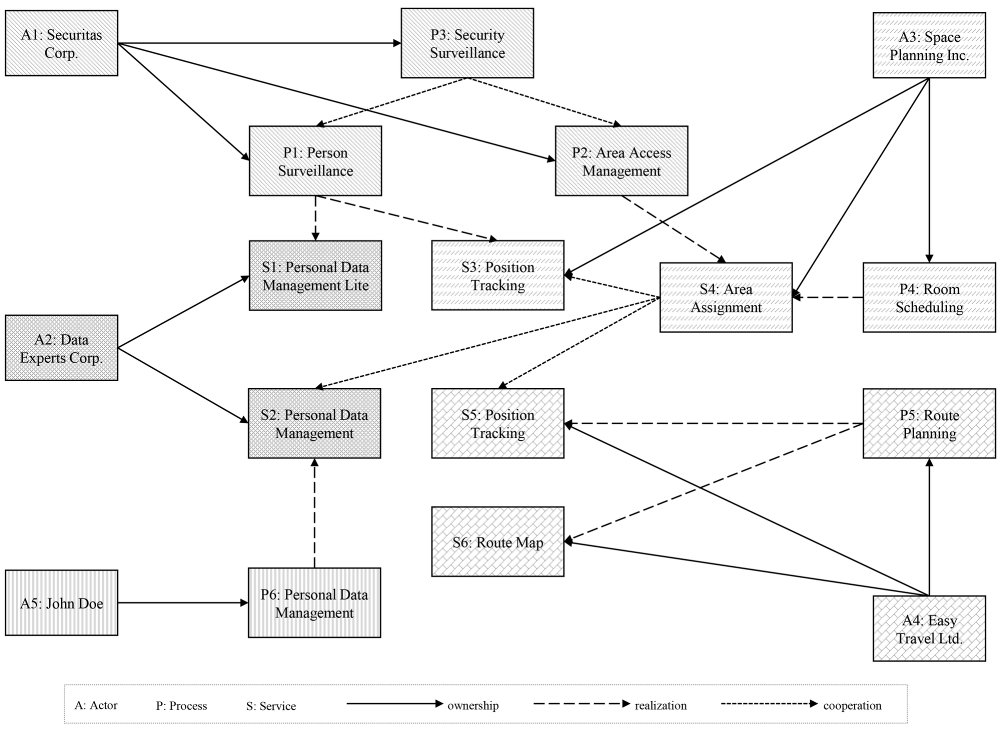
A simplified view of the service network is introduced to serve as a motivation scenario for the need for relationship management in service networks and as a guiding example for the relationship modeling approach. The scenario presumes a Cloud infrastructure allowing actors interested in a collaboration to publish or consume services via the platform. A security surveillance organization (*Securitas Corp.*) interacts with the platform to access the services of a data management expert (*Data Experts Corp.*) and a space-planning specialist (*Space Planning Inc.*) for the realization of its access management and surveillance processes. While *Data Experts Corp.* uses the platform only for the on demand provisioning of its services, *Space Planning Inc.* runs its own process for room scheduling also on the platform. It combines the functionality of a position tracking service with personal data information to provide an up-to-date room plan showing room occupancy and vacancy. For performance reasons, the room scheduling capability integrates a third-party position tracking service from *Easy Travel Ltd*, which uses the platform also for service provisioning and the process execution of its route planning capability. The last stakeholder, *John Doe*, is a representative of a physical entity using the platform to find a service for a personal data management request.

This example helps to understand the challenges for relationship management for service networks by showing the variety of entities and possible relationships between them. Furthermore, it is used to illustrate the concepts of the modeling approach defined in the next section.

The example service network is represented in Figure 5. It shows the collaboration of five actors, six processes expressing business capabilities of interest for the actors, and six services providing functionality for the technical realization of the capabilities. The interconnections between all these network participants are expressed via ownership, realization, and cooperation relationships, demonstrating a small set of possible relations, which could be of interest for the management of the network. The shading should illustrate the administrative domains of the five actors specifying the boundary of their responsibility and administrative control.

In this scenario, a number of the characteristics of service networks can be observed. These include the variety of stakeholder roles, the granularity of services and processes, as well as some service properties. Actors can be either private persons or organizations. They can interact within the network only as a provider offering services on the platform, e.g., *Data Experts Corp.*, or as a consumer using services accessible via the platform for the execution of its business capabilities, e.g., *Securitas Corp.*, or as both, e.g., *Space Planning Inc.* Some of the services within the network could be available in different versions, like the *Personal Data Management* service of *Data Experts Corp.*, where the lite version (S1) offers a limited functionality missing an identity check option compared to the extended version (S2) of the service. Other services could offer the same functionality provided by different actors, like the *Position Tracking* services S3 and S5 in the example. While most of the depicted services provide loosely coupled functionality, independent from other services, the *Area Assignment* service of *Space Planning Inc.* combines the functionality of three other services to provide a value-added capability, but two of which are provided by the third-party actors *Data Experts Corp.* and *Easy Travel Ltd*. This different granularity of entities can be observed also on the process level, combining business capabilities of the same actor to define a new process, like the *Security Surveillance* process of *Securitas Corp*.

Figure 5. Example of a service network



GENERIC MODELING FOR SERVICE NETWORKS

Different types of models use different vocabulary, their stakeholders have different domain expertise and understand only corresponding notations. In addition, the content differs, i.e., the set of entities and relationships between them, which are captured within the model. After defining the steps of an efficient service network modeling process to specify the requirements on a generic modeling approach for service networks, this section introduces such an approach able to extract, represent, and analyze the different types of service network models in a uniform form. The approach is further adaptable to existing specification languages for processes and services and the resulting model is applicable for various analysis purposes. It makes use of a metamodeling concept to define a schema with language elements for the description of service network models. A preset containing an initial set of entity and relationship types following the schema is defined to provide a foundation for uniform model documentation.

Modeling Process

The interdisciplinary significance of a service network makes it a complex concept. Thus, a service network can have a domain, a theory, or a method conceptualization (Wynn, 2007). Each one of them considers the network from a different perspective. From the domain point of view, the collection of elements and the relationships between them characterize the service network. The theory conceptualization is interested in the resource flow between the network participants for the purpose of change impact analysis. According to the method viewpoint, the main purpose of a service network is the investigation of the resulting network behavior itself.

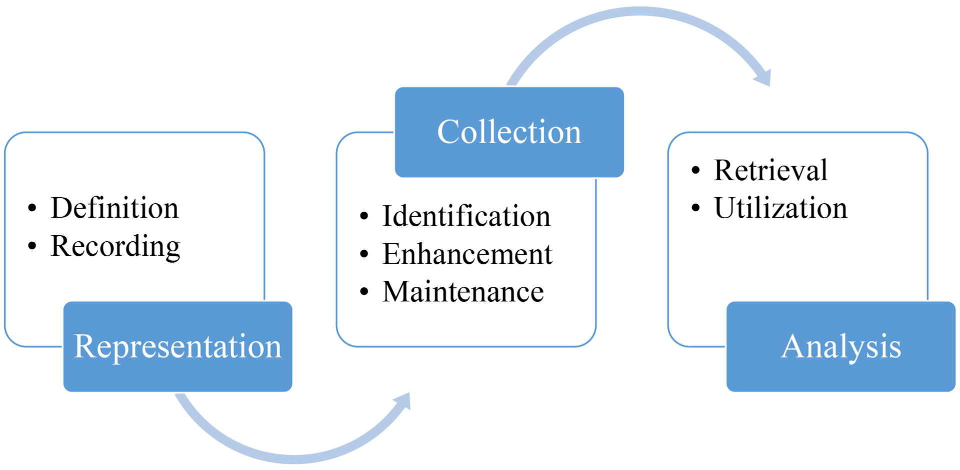
Taking into consideration the issues of all three service network conceptualizations, an efficient service network modeling process has to cover three main steps handling the information on network elements. Each step comprises a set of activities according to the terms of traceability management (Schwarz, Ebert, & Winter, 2010). The allocation of the separate activities to the three steps is shown in Figure 6 and explained in the following.

The *Representation* step determines the model content and an appropriate data structure to provide a formally structured output. Two activities address the resource flow traceability of the theory conceptualization. First, the types of entities and relationships to be represented are determined during the *Definition* activity. Afterwards, the *Recording* activity specifies an appointed data structure for recording the defined data.

Engaged with the collection of information on the elements in the service network and the relations between them according to the guidelines from the previous step, the *Collection* step of the service network modeling process addresses the prospects of the domain conceptualization. The scope of functions provided by this step splits in three activities. The discovery of information on previously unknown network entities or relations from available specification documentation is performed by the *Identification* activity. If the captured information is insufficient for the intended utilization purposes of the model, the *Enhancement* activity provides complementing possibility. The subsequent keeping of the extracted information up-to-date with the underlying service network is the field of responsibility of the *Maintenance* activity.

The last *Analysis* step uses the uniform representation of the collected relationship information as input for studying the interdependencies within the network from the point of view of particular analysis purposes in the sense of the method conceptualization. Each analysis initiative is performed in two steps. During the first *Retrieval* activity a selection and extraction of the relevant relationship records for the specific analysis purpose from the recorded extensive network model is performed. The following processing of the identified relevant information for the concrete analysis scenario is referred to as the *Utilization* activity.

Figure 6. Service network modeling process steps and their separate activities



After completing these steps, the documented and analyzed relationship information can be provided to responsible domain experts or decision automation tools, which can use it as basis for the identification of specific steps towards network optimization or correction, in case of compliance violations and failure detection.

Model Requirements

Regarding the steps of the modeling process and the specifics of service networks regarding the heterogeneity of stakeholder roles, specification concepts, and analysis needs respectively, the following requirements have to be considered by a generic modeling approach. These requirements are defined in relation to the process activities and the three main goals – uniformity, adaptability, and applicability – pursued by the modeling approach.

Various Stakeholder Perspectives

Service consumer, service provider, and service network operator are the three stakeholder roles of relevance in the context of service networks. A service network model can cover the scope of interest of one or more of these roles by providing enough information during *Definition*, which can serve during *Utilization* for role-specific analysis purposes. Considering the needs of all three roles in order to provide efficient decision support for all parties involved in the evolution of a service network, promotes the uniformity and the applicability of the modeling approach.

Heterogeneous Entities

A variety of entity types have to be considered during *Definition* in order to cover the information needs of the stakeholders from the business and the IT domain of expertise. The different granularity of these entities according to the maturity level of the modeled service network (Arsanjani & Holley, 2006) should also be possible to express. While business stakeholders are mainly interested in the value generation of business processes and tasks, the reusable and loosely coupled encapsulation of application logic into services and service compositions is the main area of interest in the IT domain. Depending on the type and purpose of an entity, different properties can be of interest for its definition. Providing the ability to specify entities from the business and technical abstraction levels of a service-based infrastructure fosters the uniformity of the modeling approach. The definition of entity-specific properties independent from the language of the entity specification supports the adaptability goal. Finally, with the heterogeneity of entities covered by the approach the variety of possible analysis questions supportable by the service network model grows and thus, contributes to the applicability of the modeling approach.

Heterogeneous Relationships

The relevant types and properties of relationships to be captured between service network entities depend on the background of the stakeholders interested in service network analysis. During *Definition*, the variety of functional and non-functional relations significant for the later *Utilization* of the model should be considered. Existing service alternatives for price reduction or performance optimization for the service consumer, flow dependencies for the service provider, or resource ownership ratios on the platform for the network operator are a subset of possible relationship interests. Similar with the scope of clear defined entities, the scope of well-defined relationships contributes to the uniformity of the modeling approach. The adaptability and applicability properties are also supported by a wide variety of relationships included in the service network model.

Extensibility

While providing enough guidance during *Definition* and *Recording* in order to achieve a uniform model representation for a service network, the modeling approach should offer the possibility to fit the set of considered entities and relationships to the needs of the network and its stakeholders. The solution should not be hardwired with a fixed set of entity types and relations. An easy integration of new types of components and collaborations evolving with the maturity of the environment, its participants, and the *Utilization* scope, exerts positive influence on the applicability scope of the approach.

Technology Independence

The input data available for the discovery of information on the entities interconnected within the network during *Identification* should be extracted from existing entity specifications. Since the providers of these specifications come either from the business or from the IT domain, it is not realistic to expect a single input notation. The storage of the collected information during the *Recording* in one of the existing specification notations for SOA would also lead to misunderstandings for non-domain experts. Therefore, both the input and the output form of the information should be independent from the existing specification technologies. The modeling approach should be able to process established specification languages as input and relatively easy integrate new specification notations evolving with the technology advances in service orientation to spare additional overhead for interested stakeholders and thus, support the adaptability of the approach to the specifications of the underlying network. Furthermore, the output form should provide unambiguous representation of the service network model for both business and IT domain experts and thus support the uniformity of the modeling approach.

Human and Machine Processing

The complementation of the model during *Enhancement* requires either manual interference or automated deduction from multiple information sources. In order to support both, the output form for the service network model should allow human and machine processing of the collected relationship information. Given the size of a service network and the extensive information in its model, the *Retrieval* and *Utilization* activities, will also profit from an automated model processing feature. With the automated integration of information from different sources for the purpose of model enhancement according to the requirements of a specific utilization purpose, this requirement fosters the adaptability and the applicability of the modeling approach.

Change Traceability

To provide efficient support for service network analysis, the model should be always up-to-date with the service network. In order to support the *Maintenance* of the model, the modeling approach should document all the changes within the network and allow displaying the differences between two points in time. This feature will support also the *Utilization* for time-based analysis purposes and thus contribute to the applicability goal of the modeling approach.

Model Customization

Change impact analysis (Basu, Casati, & Daniel, 2008) and network health analysis (Lucassen, van Rooij, & Jansen, 2013) are two application scenarios, which would profit from an explicit service network model. Yet, the set of data required as input for the analysis differs in both cases. Understanding which parts of the service network will be affected when an entity becomes unavailable requires a relationship model including only the subset of entities having functional relationship to the changing entity. The identification of health issues within the network requires general information on the activity of its entities, the recognition of most interconnected entities, which could become a critical factor, or the evolution of the interconnectivity between the network entities over time. Moreover, both of the analysis could be performed from the perspective of a different stakeholder role determining the scope of visibility on the network structure. Thus, during *Retrieval*, various parameters should be configurable, allowing the selection of customized model views according to the access rights of the stakeholder roles and the data requirements of the analysis purpose. A fitted model customization improves the applicability of the approach without increasing the complexity of the analysis by providing enormous irrelevant data.

Customid Conflict Detection

The processing of the model excerpt identified as relevant for a concrete analysis scenario during *Utilization* is strictly dependent on the analysis algorithm. In order to provide the possibility for stakeholder to apply their preferred algorithms, the modeling approach should allow the specification of critical structures and provide interfaces for their analysis and detection in case of changes affecting the relevant model excerpt. This customizable configuration of the analysis utilization fosters further the applicability of the modeling approach fitted to the needs of the interested parties.

Model Overview

In order to fulfill the abovementioned requirements for a generic service modeling approach, a conceptual service network model has to be defined as foundation. While a service network model contains the different entities interweaved in the network and the relation between each other, the purpose of the conceptual model is to define the types of entities and relationships that can be expressed by the model. Information collection and analysis processes can then make use of the formal description of the conceptual model to provide adaptable retrieval and validation of the network model.

To provide a concise and at the same time dynamically extensible notation, the conceptual modeling language applies the principles of two-dimensional metamodeling, which distinguishes between linguistic and ontological instantiation (Atkinson & Kuhne, 2003). The linguistic instantiation is responsible for the clear form of representation. It makes use of the constructs of XML and XML Schema as a widespread machine-processable notation in the context of service-orientation. The self-describing property of the XML notation supports further the dynamic extension of the data model. The ontological instantiation describes the domain-specific concepts and their properties, which reside at the same abstraction level from the linguistic point of view. It structures the organization of service networks in ontological layers allowing the definition of extensible models according to the constraints of the domain. Following these principles, the modeling language defines three ontological layers, as introduced in (Kabzeva, Götze, & Müller, 2014):

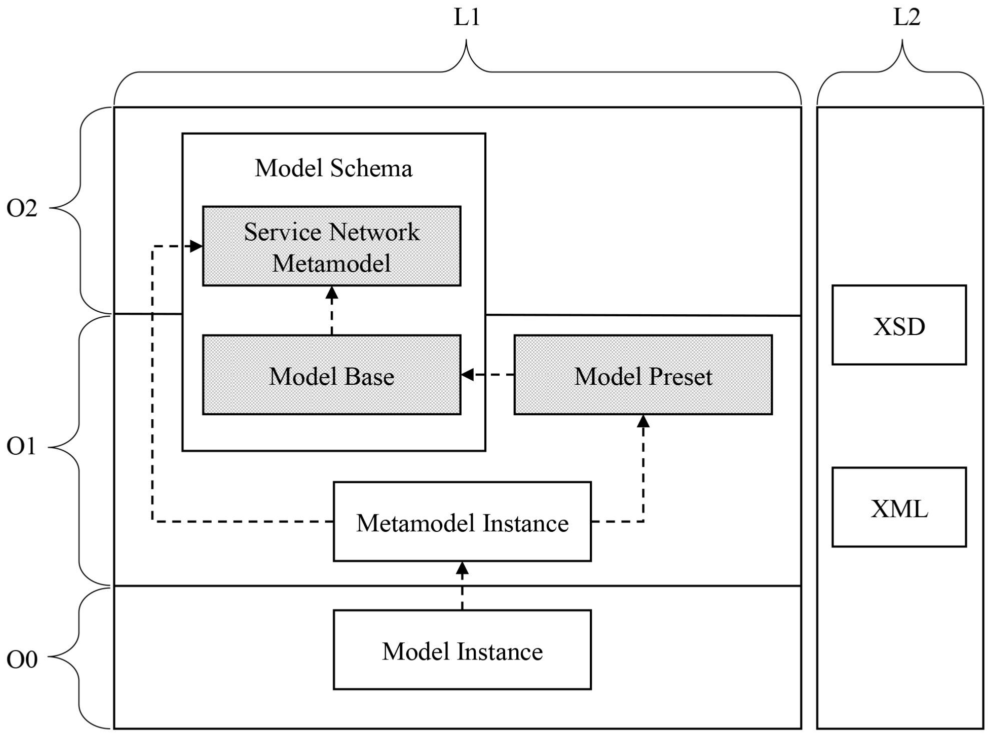
- **O2:** The metamodel layer defining the language for the specification of service network models.

- **O1:** The model layer defining the concepts for service network models conform to the language from the O2 layer.
- **O0:** The instance layer containing concrete service network models described with the concepts from the O1 layer.

The specification of the domain concepts and the ontological language structures required for their description distinguish between two modules of the conceptual model– the *Model Schema* and the *Model Preset*, as depicted in Figure 7. The *Model Schema* provides the structures for the description of the *Model Preset*. According to their ontological abstraction level, the language elements build the *Service Network Metamodel* part of the schema, while their instantiation defines the content of the *Model Base*. The *Model Preset* on its turn defines reusable, domain-specific concepts for the modeling of service networks. A *Metamodel Instance* reflects the needs of the service network to be modeled and its stakeholders can import any combination of the predefined concepts and serve then as basis for the retrieval of an actual *Model Instance*.

The modular structure of the approach fosters the uniformity and adaptability properties of the solution. With the generic metaconcepts for service network structures, the schema is independent from SOA-specific modeling nations and facilitates the dynamic addition of new types of entities and relationships, while the definition of reusable domain-specific concepts in the preset allows for uniform representation. The applicability property of the solution is addressed by the flexible instantiation of the metamodel.

Figure 7. Allocation of the conceptual model components to the ontological and linguistic layers



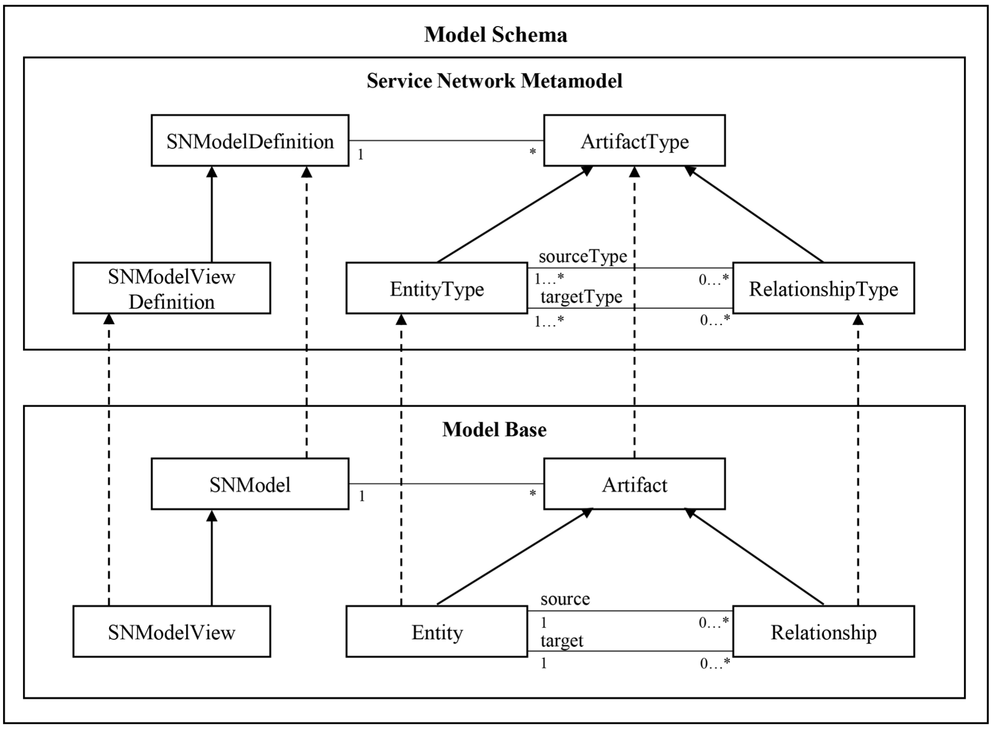
Model Schema

A model schema in general is a structured framework, guiding the description of characteristic constructs and properties of a system type. In the context of service networks in particular, a model schema should provide an organized way of specifying the wide range of artifacts designating the evolution of the network structure. In order to span the different levels of abstraction, the DAME (Dependency Analysis and Management for Evolution) model schema considers a service network as a collection of genetic artifacts, which can be either an entity or a relationship between entities. All artifacts are typed to specify particular properties and thus provide semantic refinement. In accordance with the principles of ontological metamodeling, the typing schema is defined in the *Service Network Metamodel*, the *Model Base* contains the general concepts defining the guidelines for the description of instances from a specific type.

The *Service Network Metamodel* provides the definition rules for a service network model (*SNModelDefinition*) by specifying the properties of three typing meta-concepts. A service network model definition comprises a set of *ArtifactType* definitions. This generalization of all typing concepts specifies the major feature relevant for each and every type definition – the uniqueness of their name. This name should be carefully selected to reflect the designated role of the type. The two specializations – *EntityType* and *RelationshipType* – introduce the distinction of two key typing concepts and their specific properties. While the wide span of entity types relevant for the context of service networks does not call for the appointment of specific properties for the definition of an entity type, three standard properties are distinguished as required for the definition of relationship types by the metamodel. Each relationship connects two entities. The specification of allowed types for the source and target entity connected by a relationship type restricts the occurrence scope of a relationship type. The automated or manual collection of the relationship data considered in the third property is directed to the trustworthiness and freshness of the information. In addition, the metamodel introduces the concept of model view definitions (*SNModelViewDefinition*). Being a restricted subset of a model optimized for the support of a concrete analysis objective (Danylevych, Leymann, & Nikolaou, 2011), the definition of a view can correspondingly restrict the relevant entity and relationship types.

The *Model Base* is an instantiation of a service network model definition, which states additional properties that must be specified for instantiations of entity and relationship types. The root element *SNModel* represents the service network model concept and defines the topology of a service network as an assembly of entities and relationships. These are generally referred to as artifacts. An *Artifact* is just an abstract element outlining the commonalities in the definitions of entities and relationships. Each of them has to be uniquely identifiable within the model and be tagged with a timestamp pointing out the date of its insertion in the service network. In addition, each of them specifies an extensible set of artifact-specific properties, depending of the referenced artifact type. The two specializations – *Entity* and *Relationship* – define the specification guidelines for concrete instances from a specific entity type or relationship type respectively. Additionally to the artifact properties, an entity has to provide a state information indicating the operability of the entity according to the validity of its interconnection within the service network. An instantiation of a relationship type specifies the unique identifiers of the source and target entities connected by the relationship. The entity types should be thereby consistent with the types permitted by the definition of the instantiated relationship type. Moreover, regarding two of the main properties of service networks, i.e., the on demand consumption of third-party resources and the reusability of these resources in various contexts, locality and weight of the relationship are specified as pointers for the crossing of domain borders and the frequency of a relationship between the same two entities in different contexts respectively.

Figure 8. The Model Schema and the relations between its components



Model Preset

The purpose of the *Model Preset* is to provide support for the collection, representation, and analysis of relationship information via the definition of concrete concepts. Existing entity specifications, which can serve as sources for the collection of relationship data, shape the *Collection* phase. The variety of current specification languages leads to the anticipated adaptability to new notations with reasonable effort. The approach of the preset for this modeling step is the provisioning of universal domain-specific mechanisms (Sendall & Kozaczynski, 2003) for each abstraction level. These should be applied for the automated identification and maintenance of entities and their relationships though model transformations. A new entity specification standard can be then integrated through the mapping of its internal information to the information concepts for the corresponding category.

The *Representation* phase specifies the information model providing a uniform representation of the information concepts for extraction. This model comprises a set of three entity types and eight relationship types. The content of the set of entity types reflects thereby the socio-technical nature of service networks with its business and IT characteristics derived from its ultra-large-scale systems roots (Northrop, et al., 2006). For the definition of the set of relationship types, the process of emergence of service networks via the composition of intra- and inter-organizational services as well as the structuring of thesauri relationships used for the representation of knowledge organization (National Information Standards Organization, 2005).

With the erosion of the people/system boundary and the new paradigms for simultaneous acquisition and operation of services (Northrop, et al., 2006), the development, operation, and evolution of service network involves multiple actors. Thus, together with the services responsible for the automation of business capabilities, the set of entities shaping the network has to include also the human actors and the processes defining the need for services and their collaboration (Sommerville, et al., 2012). Following these influences, the preset defines an initial set $\forall p \in P, \exists a \in A : (a, p) \in R_o \wedge \nexists a_1, a_2 \in A : (a_1, p) \wedge (a_2, p) \in R_o$ of entity types including actors, processes, and services.

$$\forall s \in S, \exists a \in A : (a, s) \in R_o \wedge \nexists a_1, a_2 \in A : (a_1, s) \wedge (a_2, s) \in R_o$$

The set of actors $\boldsymbol{A}$ contains physical and legal entities fostering the evolution of the service network through the provisioning and/or consumption of resources. Additionally to its id, name, timestamp, and state properties defined in the model schema, an actor is characterized with its role within the network and its type. The role is a mandatory property, which states how the actor collaborates within the network and offers the possibility to choose between a provider, a consumer, or both. A type property is optional to indicate the nature of the actor, i.e., a private human or a business organization (Danylevych, Karastoyanova, & Leymann, 2010).

$$\forall \boldsymbol{a} \in \boldsymbol{A} : \{\boldsymbol{id}, \boldsymbol{name}, \boldsymbol{timestamp}, \boldsymbol{state}, \boldsymbol{role}, \boldsymbol{type}\}$$

Table 1 provides an overview of exemplary actor specifications for the example service network from the Motivation Scenario section. The definitions are provided in tabular form for better readability.

The set of processes $\boldsymbol{P}$ represents the non-executable descriptions of business capabilities, defined as service requests on the network, which execution should use network resources. The realization of a process by one or more services defines the need for consumption of services and their interlinking into a network. Additionally to its properties inherited from the model schema, a process entity is characterized with two more properties. Its version allows the definition of variations of the same capability. Type is used to specify the complexity of the process, i.e., atomic or composite, referring to the reusability of existing process definitions.

Table 1. Exemplary actor specifications for the example service network

Id	Name	Timestamp	State	Role	Type
A1	Securitas Corp.	2014-10-21	operable	consumer	legal
A2	Data Experts Corp.	2014-09-12	operable	provider	legal
A3	Space Planning Inc.	2014-10-01	operable	consumer provider	legal
A4	Easy Travel Ltd.	2014-09-20	operable	consumer provider	legal
A5	John Doe	2014-09-03	operable	consumer	physical

$$\forall \boldsymbol{p} \in \boldsymbol{P} : \left\{ \boldsymbol{id}, \boldsymbol{name}, \boldsymbol{timestamp}, \boldsymbol{state}, \boldsymbol{version}, \boldsymbol{type} \right\}$$

Table 2 provides an overview of exemplary process specifications for the example service network from the Motivation Scenario section. The definitions are provided again in tabular form for better readability.

The set of services $\boldsymbol{S}$ includes the technical entities enabling the access to one or more capabilities. According to the service definition in (OASIS, 2006), a service consists of a service description and an execution component provided by an actor. The service description is an interface specifying available operations and contract clauses relevant for the usage of a service. The execution component implements the service logic and remains a black box for the service network. Five type-specific properties are defined for a service specification – state, version, type, lifecycle stage, and access. The purpose of state, version, and type correspond thereby to the ones defined for a process. The *lifecycleStage* property contains the development stage of the service, e.g., design, testing, running, and thus allows the collection of service specifications at any point of their development. Last but not least, the access property defines the target audience of a service to differentiate between domain-internal and domain-external services with respect to their accessibility and allow for better estimation of the Quality of Service requirements and customization effort on a service (Josuttis, 2007).

$$\forall \boldsymbol{s} \in \boldsymbol{S} : \left\{ \boldsymbol{id}, \boldsymbol{name}, \boldsymbol{timestamp}, \boldsymbol{state}, \boldsymbol{version}, \boldsymbol{type}, \boldsymbol{lifecycleStage}, \boldsymbol{access} \right\}$$

Table 3 provides an overview of exemplary service specifications for the example service network from the Motivation Scenario section. The definitions are provided again in tabular form for better readability.

Regarding the adaptability to changing requirements, the reusability of services for shorter time-to-market and cost reduction, and the inter-organizational interoperability as the main drivers for the application of the service-orientation paradigm for the development of business applications, the need for service network modeling is comparable to the need for vocabulary control. Analogous to the latter, which arises from variety of words for a single concept and the various meaning of a word with the same spelling (National Information Standards Organization, 2005), the need for service network modeling results from the variety of services from different providers applicable for a single business capability and the various contexts (service compositions) integrating the same service. Considering this similarity, the information model in the

Table 2. Exemplary process specifications for the example service network

Id	Name	Timestamp	State	Version	Type
P1	Person Surveillance	2014-10-21	operable	1	atomic
P2	Area Access Management	2014-10-23	operable	1	atomic
P3	Security Surveillance	2014-10-23	operable	1	composite
P4	Room Scheduling	2014-10-01	operable	1	atomic
P5	Route Planning	2014-09-20	initiating	1	atomic
P6	Personal Data Management	2014-09-12	operable	1	atomic

Table 3. Exemplary service specifications for the example service network

Id	Name	Timestamp	State	Version	Type	Stage	Access
S1	Personal Data Management Lite	2014-10-21	operable	1	atomic	running	external
S2	Personal Data Management	2014-10-23	operable	1	atomic	running	external
S3	Position Tracking	2014-10-23	operable	1	atomic	running	external internal
S4	Area Assignment	2014-10-01	operable	1	composite	running	external internal
S5	Position Tracking	2014-09-20	operable	1	atomic	running	external internal
S6	Route Map	2014-09-12	initiating	1	atomic	design	internal

DAME preset adopts the relationship concepts from thesauri construction for the management of service network models. The semantic of the three general groups of relationships is defined thereby as follows:

- **Hierarchical Relationships:** connect entities from different types across the abstraction levels of the model.
- **Equivalence Relationships:** express non-functional similarity relations between entities of the same type, which can be interchanged with less effort.
- **Associative Relationships:** express functional dependency relations between entities of the same type, which result from the composition of resources for value-added capability provisioning.

The vertical hierarchical relationships are based on levels of subordination, where the superordinate entity type decides on the insertion of the subordinate entity types. Realization, ownership, and consumption are the three representatives of the hierarchical relationships group $\boldsymbol{HR}$, which are predefined in the model preset.

$$\boldsymbol{HR} = \boldsymbol{R}_r \bigcup \boldsymbol{R}_o \bigcup \boldsymbol{R}_u$$

Realization relationships $\boldsymbol{R}_r$ indicate the refinement of a source entity by the target entity. This type provides the basis for expressing the interconnection between the business level and IT level of the service network. A target service entity has to fulfill thereby at least one requirement of the source process entity.

$$\boldsymbol{R}_r \subseteq \boldsymbol{P} \times \boldsymbol{S} : \ \forall \boldsymbol{p} \in \boldsymbol{P}, \ \exists \boldsymbol{s} \in \boldsymbol{S} : \left(\boldsymbol{p}, \boldsymbol{s}\right) \in \boldsymbol{R}_r \wedge \boldsymbol{s} \sim \boldsymbol{p}$$

$$\boldsymbol{where} \ \ \boldsymbol{s} \sim \boldsymbol{p} := \boldsymbol{s} \ \ \boldsymbol{fulfills} \ \ \boldsymbol{requirement} \ \ \boldsymbol{of} \ \ \boldsymbol{p}$$

The example service network from Figure 5 illustrates the realization relationships between processes and services with dashed arrows, e.g., between process P1 and service S1. Each process has a realization relationship to at least one service providing an implementation for the business capability.

An ownership relationship $\boldsymbol{R_o}$ states the responsibility of a source entity for the target entity. This type of relationship connects the social level of the service network with the resources of business and IT level. Thus, the source of an ownership relation is always an actor. Permitted target entity types include processes and services. One actor can own multiple processes and services. Each process and service however, belongs exactly to one actor. This actor is responsible for their provisioning and quality.

$$\boldsymbol{R_o} \subseteq \left(\boldsymbol{A} \times \boldsymbol{P}\right) \bigcup \left(\boldsymbol{A} \times \boldsymbol{S}\right)$$
$$\forall \boldsymbol{p} \in \boldsymbol{P}, \exists \boldsymbol{a} \in \boldsymbol{A} : \left(\boldsymbol{a}, \boldsymbol{p}\right) \in \boldsymbol{R_o} \wedge \nexists \boldsymbol{a_1}, \boldsymbol{a_2} \in \boldsymbol{A} : \left(\boldsymbol{a_1}, \boldsymbol{p}\right) \wedge \left(\boldsymbol{a_2}, \boldsymbol{p}\right) \in \boldsymbol{R_o}$$
$$\forall \boldsymbol{s} \in \boldsymbol{S}, \exists \boldsymbol{a} \in \boldsymbol{A} : \left(\boldsymbol{a}, \boldsymbol{s}\right) \in \boldsymbol{R_o} \wedge \nexists \boldsymbol{a_1}, \boldsymbol{a_2} \in \boldsymbol{A} : \left(\boldsymbol{a_1}, \boldsymbol{s}\right) \wedge \left(\boldsymbol{a_2}, \boldsymbol{s}\right) \in \boldsymbol{R_o}$$

Examples for ownership relationships can be observed in Figure 5, connecting the actors via a solid arrow with processes and services, e.g., actor A3 and service S4. Processes and services falling into the responsibility of a specific actor are further marked by the same shading in the example service network.

A consumption relationship $\boldsymbol{R_u}$ models the usage of resources within the network. It does not specify how the source makes use of the target, only that the target is of interest for the source. This relationship connects the social network layer with the technical services layer. The information is automatically deduced from the realization relationships of the actor's processes. Each actor can consume multiple services. Moreover, a service can be consumed by the same actor in multiple processes. The more of the actor's processes are realized by a service, the higher the weight of the consumption relationship for the actor.

$$\boldsymbol{R_u} \subseteq \boldsymbol{A} \times \boldsymbol{S} : \forall \left(\boldsymbol{a}, \boldsymbol{s}\right) \in \boldsymbol{R_u}, \exists \boldsymbol{p} \in \boldsymbol{P} : \left(\boldsymbol{a}, \boldsymbol{p}\right) \in \boldsymbol{R_o} \wedge \left(\boldsymbol{p}, \boldsymbol{s}\right) \in \boldsymbol{R_r}$$

Using the example service network from Figure 5, Table 4 defines the set of consumption relationships for the network. Since services S1 and S3 are responsible for the realization of processes P1 and P3, the table captures the consumption relationships between A1 and the services with weight 2. The same applies to S4, which is in a realization relationship with P2 and P3.

The types of relationships predefined in the preset for the second group of relationships ER connect similar entities from the same type. Two types of similarity are recognized as relevant in the context of service networks – competition and evolution.

$$\boldsymbol{ER} = \boldsymbol{R_c} \bigcup \boldsymbol{R_e}$$

When the same business process can be realized by two or more services provided in the network, one of these services is selected as preferred. The relationship between the preferred and the non-preferred services is a competition relationship $\boldsymbol{R_c}$. The actors owning the compet-

Table 4. Set of consumption relationships for the example service network

Actor	Service	weight
A1	S1	2
	S3	2
	S4	2
A3	S4	1
	S2	1
	S3	1
	S5	1
A4	S5	1
	S6	1
A5	S2	1

ing services are respectively in a competition relationship on the social level of the network model. A service can have multiple competitors, but not every service has a competitor, e.g., services providing niche functionality. With the number of the competing service pairs, the weight of the competition relationship between the actors providing them grows. A competition relationship between services of the same provider, which is possible in case of intra-organizational redundancies, will not lead to a competition relationship on the social level of the model.

$$\boldsymbol{R_c} = \boldsymbol{R_{ca}} \bigcup \boldsymbol{R_{cs}} \textbf{ with } \boldsymbol{R_{ca}} \subseteq (\boldsymbol{A} \times \boldsymbol{A}) \textbf{ and } \boldsymbol{R_{cs}} \subseteq (\boldsymbol{S} \times \boldsymbol{S})$$
$$\forall \boldsymbol{r} \in \boldsymbol{R_{cs}}, \exists \boldsymbol{s}_1, \boldsymbol{s}_2 \in \boldsymbol{S} \wedge \boldsymbol{p} \in \boldsymbol{P} : \boldsymbol{r} = (\boldsymbol{s}_1, \boldsymbol{s}_2) \wedge \boldsymbol{s}_1 \sim \boldsymbol{p} \wedge \boldsymbol{s}_2 \sim \boldsymbol{p}$$
$$\forall \boldsymbol{r} \in \boldsymbol{R_{ca}}, \exists \boldsymbol{s}_1, \boldsymbol{s}_2 \in \boldsymbol{S} \wedge \boldsymbol{a}_1, \boldsymbol{a}_2 \in \boldsymbol{A} : \boldsymbol{r} = (\boldsymbol{a}_1, \boldsymbol{a}_2) \wedge (\boldsymbol{a}_1, \boldsymbol{s}_1) \in \boldsymbol{R_o} \wedge (\boldsymbol{a}_2, \boldsymbol{s}_2) \in \boldsymbol{R_o} \wedge (\boldsymbol{s}_1, \boldsymbol{s}_2) \in \boldsymbol{R_c}$$

For the example service network from Figure 5 a competition relationship should be modeled between services S3 and S5, both of which implement a position tracking interface. Their providers A3 and A4 are correspondingly also competitors. Since they have only one competing service pair, the competition relationship between A3 and A4 has weight 1.

An evolution relationship $\boldsymbol{R_e}$ indicates that a source entity is a descendant from the target entity. Thus, the second type of similarity relationships connects two versions of the same entity with a modified interface, existing simultaneously in the service network. This relationship type is relevant for the business and IT abstraction level. Multiple versions of a process or a service may be present within the network with slightly different scope of the provided capabilities or quality. Yet, an evolution relationship is allowed only between resources provided by the same actor.

$$\boldsymbol{R_e} = \boldsymbol{R_{ep}} \bigcup \boldsymbol{R_{es}} \textbf{ with } \boldsymbol{R_{ep}} \subseteq (\boldsymbol{P} \times \boldsymbol{P}) \textbf{ and } \boldsymbol{R_{es}} \subseteq (\boldsymbol{S} \times \boldsymbol{S})$$
$$\forall \boldsymbol{r} \in \boldsymbol{R_{ep}}, \exists \boldsymbol{p}_1, \boldsymbol{p}_2 \in \boldsymbol{P} \wedge \boldsymbol{a} \in \boldsymbol{A} : \boldsymbol{r} = (\boldsymbol{p}_1, \boldsymbol{p}_2) \wedge (\boldsymbol{a}, \boldsymbol{p}_1) \in \boldsymbol{R_o} \wedge (\boldsymbol{a}, \boldsymbol{p}_2) \in \boldsymbol{R_o} \wedge \boldsymbol{p}_1 \cong \boldsymbol{p}_2$$

$$\textbf{\textit{where }} \boldsymbol{p}_1 \cong \boldsymbol{p}_2 := \boldsymbol{p}_2 \textbf{\textit{ succeeding version of }} \boldsymbol{p}_1$$

$$\forall \boldsymbol{r} \in \boldsymbol{R}_{es}, \exists \boldsymbol{s}_1, \boldsymbol{s}_2 \in \boldsymbol{S} \wedge \boldsymbol{a} \in \boldsymbol{A} : \boldsymbol{r} = \left(\boldsymbol{s}_1, \boldsymbol{s}_2\right) \wedge \left(\boldsymbol{a}, \boldsymbol{s}_1\right) \in \boldsymbol{R}_o \wedge \left(\boldsymbol{a}, \boldsymbol{s}_2\right) \in \boldsymbol{R}_o \wedge \boldsymbol{s}_1 \cong \boldsymbol{s}_2$$

$$\textbf{\textit{where }} \boldsymbol{s}_1 \cong \boldsymbol{s}_2 := \boldsymbol{s}_2 \textbf{\textit{ succeeding version of }} \boldsymbol{s}_1$$

Within the example service network from Figure 5, S1 is a lite version of S2. Thus, the two services of provider A2 can be connected with an evolution relationship to indicate a possible interchange in case of deficiency or withdrawal.

The group of associative relationships $\boldsymbol{AR}$ includes also dependency types restricting the source and target entities to be from the same type. The preset differentiates between cooperation, flow, and resource relationships in this group.

$$\boldsymbol{AR} = \boldsymbol{R}_v \bigcup \boldsymbol{R}_f \bigcup \boldsymbol{R}_d$$

A cooperation relationship $\boldsymbol{R}_v$ models the collaboration of entities to provide a more complex capability. The result of a collaboration is a value-added entity combining the capabilities of existing entities. The composition of entities in a complex value-added entity can take place on the business and the IT level. Therefore, processes and services are the permitted entity types for a cooperation relationship. A composite entity can combine multiple existing entities. Thus, a cooperation relationship has to be introduced between the composite entity and each one of the composition members. Via deduction from the cooperation between entities from different providers, cooperation relationships can be defined also on the social level of the model between all actors participating in the provisioning of the value-added entity.

$$\boldsymbol{R}_v = \boldsymbol{R}_{va} \bigcup \boldsymbol{R}_{vp} \bigcup \boldsymbol{R}_{vs} \textbf{\textit{ with }} \boldsymbol{R}_{va} \subseteq \left(\boldsymbol{A} \times \boldsymbol{A}\right), \boldsymbol{R}_{vp} \subseteq \left(\boldsymbol{P} \times \boldsymbol{P}\right), \textbf{\textit{and }} \boldsymbol{R}_{vs} \subseteq \left(\boldsymbol{S} \times \boldsymbol{S}\right)$$

$$\forall \boldsymbol{r} \in \boldsymbol{R}_{va}, \exists \boldsymbol{s}_1, \boldsymbol{s}_2 \in \boldsymbol{S} \wedge \boldsymbol{a}_1, \boldsymbol{a}_2 \in \boldsymbol{A} : \boldsymbol{r} = \left(\boldsymbol{a}_1, \boldsymbol{a}_2\right) \wedge \left(\boldsymbol{a}_1, \boldsymbol{s}_1\right) \in \boldsymbol{R}_o \wedge \left(\boldsymbol{a}_2, \boldsymbol{s}_2\right) \in \boldsymbol{R}_o \wedge \left(\boldsymbol{s}_1, \boldsymbol{s}_2\right) \in \boldsymbol{AR}$$

$$\forall \boldsymbol{r} \in \boldsymbol{R}_{vp}, \exists \boldsymbol{p}_1, \boldsymbol{p}_2 \in \boldsymbol{P} \wedge \boldsymbol{a} \in \boldsymbol{A} : \boldsymbol{r} = \left(\boldsymbol{p}_1, \boldsymbol{p}_2\right) \wedge \left(\boldsymbol{a}, \boldsymbol{p}_1\right) \in \boldsymbol{R}_o \wedge \left(\boldsymbol{a}, \boldsymbol{p}_2\right) \in \boldsymbol{R}_o \wedge \boldsymbol{p}_2 \subset \boldsymbol{p}_1$$

$$\forall \boldsymbol{r} \in \boldsymbol{R}_{vs}, \exists \boldsymbol{s}_1, \boldsymbol{s}_2 \in \boldsymbol{S} : \boldsymbol{r} = \left(\boldsymbol{s}_1, \boldsymbol{s}_2\right) \wedge \boldsymbol{s}_2 \subset \boldsymbol{s}_1$$

The example service network from Figure 5 provides examples for cooperation relationships on all three levels of abstraction. The composite process $\boldsymbol{P}3$ introduces the cooperation relationships $\left(\boldsymbol{P}3, \boldsymbol{P}1\right)$ and $\left(\boldsymbol{P}3, \boldsymbol{P}2\right)$. On the service level, the composite service S4 leads to the three cooperation relationships $\left(\boldsymbol{S}4, \boldsymbol{S}2\right)$, $\left(\boldsymbol{S}4, \boldsymbol{S}2\right)$ and $\left(\boldsymbol{S}4, \boldsymbol{S}5\right)$. Since the services participating in the composition belong to different providers, the cooperation transfers also at the actor level with the relationships $\left(\boldsymbol{A}3, \boldsymbol{A}4\right)$, $\left(\boldsymbol{A}3, \boldsymbol{A}2\right)$ and $\left(\boldsymbol{A}2, \boldsymbol{A}4\right)$.

The remaining two associative relationships define a specialization of the cooperation relation for the members of a value-added composition. A flow relationship $\boldsymbol{R}_f$ represents a temporal dependency from the source entity regarding the provisioning of the target entity. Flow relationships occur between entities participating in the composition of a value-added entity and represent the control flow defined by the value-added entity. Thus, a flow relation connects either process or service entities on the member level of a composition. The weight of the relationship between two entities in a flow relation increases with the number of value-added entities defining the same control flow between both entities.

$$\boldsymbol{R_f = R_{fp} \bigcup R_{fs}}\ \textbf{\textit{with}}\ \boldsymbol{R_{fp} \subseteq \left(P \times P\right)}\ \textbf{\textit{and}}\ \boldsymbol{R_{fs} \subseteq \left(S \times S\right)}$$

$$\forall \boldsymbol{r} \in \boldsymbol{R_{fp}}, \exists \boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{q} \in \boldsymbol{P} : \boldsymbol{r} = \left(\boldsymbol{p}_1, \boldsymbol{p}_2\right) \wedge \left(\boldsymbol{q}, \boldsymbol{p}_1\right) \in \boldsymbol{R_v} \wedge \left(\boldsymbol{q}, \boldsymbol{p}_2\right) \in \boldsymbol{R_v} \wedge \boldsymbol{p}_2 \succ \boldsymbol{p}_1$$

$$\textbf{\textit{where}}\ \boldsymbol{p}_2 \succ \boldsymbol{p}_1 := \boldsymbol{p}_2\ \textbf{\textit{succeeds}}\ \boldsymbol{p}_1$$

$$\forall \boldsymbol{r} \in \boldsymbol{R_{fs}}, \exists \boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{t} \in \boldsymbol{S} : \boldsymbol{r} = \left(\boldsymbol{s}_1, \boldsymbol{s}_2\right) \wedge \left(\boldsymbol{t}, \boldsymbol{s}_1\right) \in \boldsymbol{R_v} \wedge \left(\boldsymbol{t}, \boldsymbol{s}_2\right) \in \boldsymbol{R_v} \wedge \boldsymbol{s}_2 \succ \boldsymbol{s}_1$$

$$\textbf{\textit{where}}\ \boldsymbol{s}_2 \succ \boldsymbol{s}_1 := \boldsymbol{s}_2\ \textbf{\textit{succeeds}}\ \boldsymbol{s}_1$$

In the example service network from Figure 5, a flow relationship would be expected between P1 and P2. On the service level the flow will be observed between the members of the S4 composition – S2 and S3, and S2 and S5, provided that the invocation of the two competing services S3 and S5 is defined as optional in the composition.

A resource relationship $\boldsymbol{R_d}$ expresses a data dependency from the source entity regarding the execution of the target entity. Source and target should be participants in the same composition and exchange of data objects provided by the source for further processed by the target. The weight of the relationship relates to the number of objects exchanged between the connected entities.

$$\boldsymbol{R_d = R_{dp} \bigcup R_{ds}}\ \textbf{\textit{with}}\ \boldsymbol{R_{dp} \subseteq \left(P \times P\right)}\ \textbf{\textit{and}}\ \boldsymbol{R_{ds} \subseteq \left(S \times S\right)}$$

$$\forall \boldsymbol{r} \in \boldsymbol{R_{dp}}, \exists \boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{q} \in \boldsymbol{P} : \boldsymbol{r} = \left(\boldsymbol{p}_1, \boldsymbol{p}_2\right) \wedge \left(\boldsymbol{q}, \boldsymbol{p}_1\right) \in \boldsymbol{R_v} \wedge \left(\boldsymbol{q}, \boldsymbol{p}_2\right) \in \boldsymbol{R_v} \wedge \textbf{\textit{output}}(\boldsymbol{p}_1) \supseteq \textbf{\textit{input}}(\boldsymbol{p}_2)$$

$$\forall \boldsymbol{r} \in \boldsymbol{R_{ds}}, \exists \boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{t} \in \boldsymbol{S} : \boldsymbol{r} = \left(\boldsymbol{s}_1, \boldsymbol{s}_2\right) \wedge \left(\boldsymbol{t}, \boldsymbol{s}_1\right) \in \boldsymbol{R_v} \wedge \left(\boldsymbol{t}, \boldsymbol{s}_2\right) \in \boldsymbol{R_v} \wedge \textbf{\textit{output}}(\boldsymbol{s}_1) \supseteq \textbf{\textit{input}}(\boldsymbol{s}_2)$$

A resource relationship in the example service network from Figure 5 could exist between the same entities discussed for the flow relationships.

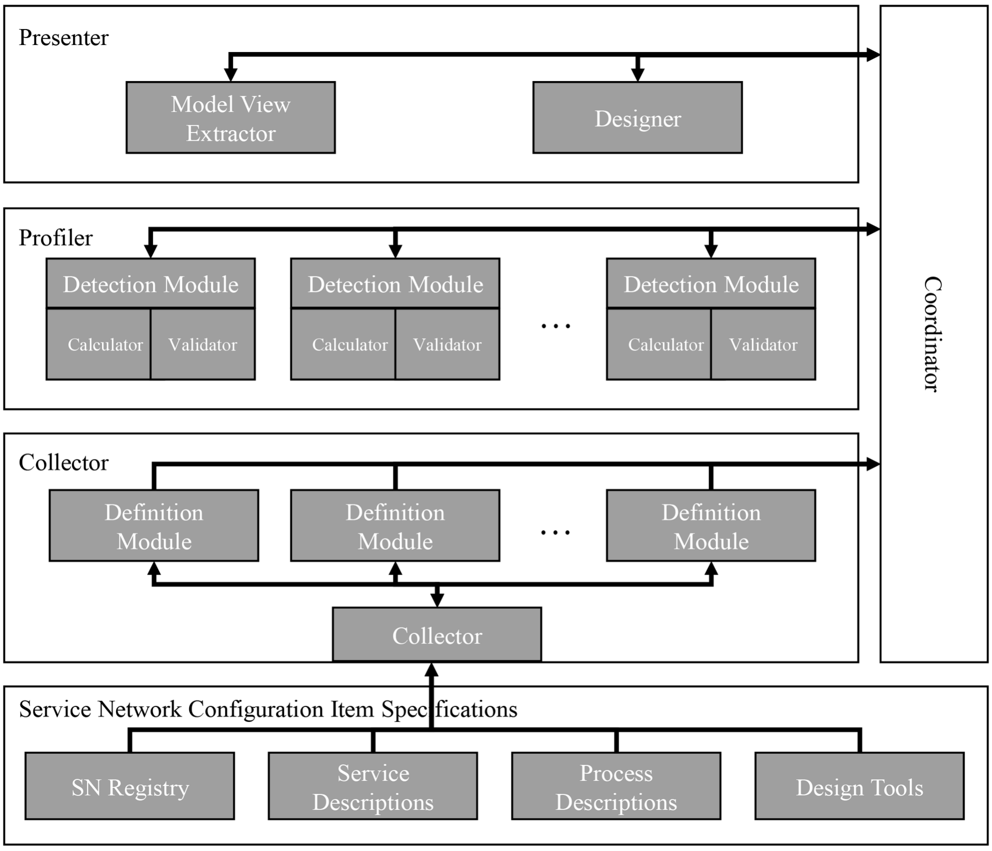
After the collection of the relationship information and its formal representation according to the defined information model, retrieval and validation processes can be defined. Generic rules can be defined, which can be filled with the desired retrieval patterns (Schwarz, Ebert, & Winter, 2010), for chosen entity types connected by a wanted relationship type. The existence or absence of the required relationships can then be applied for the definition of active analysis rules.

RELATIONSHIP DATA MANAGEMENT FRAMEWORK

This section describes the DAME framework, which defines a modular architecture for the realization of the complete modeling process – from the (semi-)automated capturing and representation of a service network model to the extensible analysis functionalities based on the model. The architecture of the framework was originally published in (Kabzeva, Götze, Lottermann, & Müller, 2013). It comprises three horizontal layers: a relationship collector layer responsible for extracting relationship information from configuration item descriptions, a relationship profiler, which calculates additional relationships based on aggregated inputs from the relationship collector, and a relationship presenter layer, which prepares the relationship information for stakeholder-specific extraction and visualization. The vertical coordination layer supports the three horizontal layer activities by the definition of patterns and constraints specific for the structure of service-based application landscapes and guides them through the steps of the modeling process. As illustrated in Figure 9, the framework works on top of the service-based application infrastructure to be modeled. It works on top of existing items' descriptions without requiring any specific input format or additional tagging in the specifications.

The collector layer takes as input raw data from existing configuration item descriptions originating from design tools or service network registries and repositories and transforms it into a uniform specification according to the information model and entity-specific relationship patterns provided from the vertical layer. To provide extensibility for various entity types, the collector layer has an extensible, modular structure. For each type of configuration item, a specialized definition module is provided that knows what type of information to search for in its documentation. To be able to understand different modeling notations, like BPMN (Business Process Model and Notation) (OMG, 2011) or EPC (Event-driven Process Chain) (Scheer, Thomas, & Adam, 2005) for business process descriptions, a set of patterns mapping the notation-specific structures

Figure 9. DAME framework architecture on top of a service-based application infrastructure



to the unified information model should be provided to the collector. The collector provides an entry point for new entity descriptions to the framework. It acts as a central definition hub, which forwards provided definitions or deletion requests to the responsible definition modules. The definition module translates the language-specific description of an entity into the uniform data format defined by the DAME model and marks it with a unique ID for the network model. When a deletion request arrives, the coordinator is triggered to mark the change in the model and throws an exception in case of modeling rules violations. The collected information in the uniform specification format is passed for further processing to the relationships profiler layer.

The objectives of the profiler layer are the calculation of indirect relationships and the validation for completeness and inconsistencies. While the collector layer processes one entity description at a time, the profiler layer combines the information from multiple sources. Again, following the extensible approach advanced by the previous layer, an analysis-specific definition module determines how to search for relevant description files for a calculation. Through a concept of pluggable detection modules, the recording of new relationship types is achieved by simply binding a new module to the coordinator. The completeness and consistency checks are done against analysis-specific rules defined in the vertical architectural layer based on possible relationships. The completeness of the collected data is dependent on the content provided in the underlying item specifications, e.g., all services invoked in a composed service are known to the network model. To allow for completion of mandatory information, the coordinator has to notify the responsible stakeholder via the presenter layer and request for missing inputs, e.g., initial task-to-service operation record. In case of inconsistency, the framework only notifies the responsible stakeholder. This is a necessity, because the goal of the framework is to capture the topology of service network and point out possible flaws. The correction process is a complex issue in itself and usually requires human decisions. To map the landscape architecture as it is presented by the stakeholder, inconsistent relationships have to be kept within the model until their correction through the modification of the related resources. The modification will result in a specification change, which will trigger a re-calculation of the relationships in the profiler, and thus will update the relationship model. The calculated relationship profiles are finally saved for stakeholder-based retrieval and reasoning.

The topmost presenter layer handles the preparation of the captured relationship model for representation, according to the analysis-specific content requirements of a stakeholder. An automated selection of the desired subset of entities and relationships needs predefined rules. Under consideration of their access rights on the resources of the service network, actors can choose what part of the overall network model should be visualized for their analysis purpose. Thereby, they should be able to restrict the visibility of configuration items as well as the type of relationships between them. The Model View Extractor provides a graphical representation of the desired view. The Designer communicates analysis issues to relevant actors and allows manual modifications for non-automatable resolutions.

A prototypical realization of the framework has been implemented with the purpose to validate the usability and the inconsistence prevention of the developed approach. The prototype provides definition modules for WSDL (Web Services Description Language), BPEL (Business Process Execution Language), BPMN, and EPC. Additionally, three detection modules have been implemented. A mapping detection module collects and validates realization relationships between process and service entities. An inter-service dependency detection module deduces resource relationships between services from process resource relationships and their realization relations to the service level. A service classification detection module is capable of classifying an executable service according to its technical complexity (Josuttis, 2007) and validating it automatically on model change.

The prototype was applied in two case studies. First, it was used to collect, validate, and represent the relationships in a service network emerging during the development of a composite stock trader application using BPMN for the business process definition and WSDL for the service specifications. The goal was to observe the usability of the framework as support for the composite service provider. During the top-down development process, each entity specification is parsed after creation by the corresponding definition module and the discovered entities added in a service network model. Consequently, depending on the entity type, the relevant detection modules are invoked. Resulting notifications on missing or incomplete documentation and service classification inconsistencies have supported the provisioning of a complete and consistent application design. Second, the prototype was used to evaluate the network indicators (Liu, Fan, & Huang, 2013) of the existing service network comprising the 604 SAP R/3 processes (Keller & Teufel, 1998) documented in EPC notation. Based on the discovered service composition and collaboration rates, the impact of services on the health of the overall service network could be estimated.

FUTURE RESEARCH DIRECTIONS

The presented service network modeling approach provides a flexible solution for a representation of uniform service network topologies for service providers, consumers, and network operators from a business and technical point of view. Compared to other existing modeling solutions, the approach considers the complete modeling process and ensures adaptability and applicability according to the characteristics of the underlying network and the needs of its stakeholders. However, some of the introduced conceptual ideas still need a formal specification. In the following, some indications for expansion and further development of the model and of the framework are presented.

While the approach introduces the idea of generic rules serving as template for the retrieval of model views and the definition of analysis criteria, their formal representation is still an open issue. Guidelines for reasonable querying patterns for view retrieval in the context of service network analysis have to be defined and the applicability of existing querying languages for their realization has to be researched. The same applies for the definition of analysis criteria. Focusing on the use of event-condition-action rules, as future task, the events causing a topology evolution in conjunction with possible disruptions and required notifications will be specified.

Although not in the scope of the modeling process, a useful extension of the proposed framework would be the integration of automated solution propositions in case of discovered network health issues. A list of possible reactions derived from the overall network topology could be offered as solution in case of functional or quality violations. The subsequent automated execution of a selected solution would however still require implementation changes on the network level. Another useful addition to the approach would be the integration of run-time information. Data on the frequency of actual invocations would serve for better quantification of the modeled relationship weights and their impact on the utilization of the service network.

CONCLUSION

The loose coupling of services applied for the realization of event-driven SOA-applications makes the management of an emerging service network topology a challenging task. The explicit availability of information on the different types of collaborations between the artifacts in a service network is an important factor for the analysis of the operability state of the service network and for estimating the impact of changes that can occur during the evolution of the network. As illustrated in this article, the topology of a service network is a complex construct, spanning multiple entities on different abstraction levels, which all play a role in the emergence of the network. Relationships between these entities, which are relevant for the understanding of the network, can be observed vertically, between the social, process, and services abstraction layers of the model, as well as horizontally, between entities of the same type. Moreover, multiple relationships with different semantics can be of interest for the analysis purposes of the network stakeholders and their different domains of expertise.

With the purpose to offer a definition of an extensible and uniform service model, this article introduces the complete modeling process and defines requirements on a relationship management solution for service networks. Addressing these requirements, a metamodel-based modeling approach is proposed. It provides a generic language schema together with a preset of predefined entities and relationships, which have been identified as important in the context of service network analysis. A modular relationship management framework introduces a supporting environment for the realization of the defined modeling process. This supporting environment has been implemented prototypically and used to model and analyze two case studies, which demonstrate the benefits of the approach for the development and maintenance of service networks.

KEY TERMS AND DEFINITIONS

Actor: A physical or legal entity fostering the evolution of the service network through the provisioning and/or consumption of services.

Adaptability: Independence from a specific modeling language applied for the specification of service-based applications at the different levels of abstraction.

Applicability: Unrestricted application via low effort adaptation for various analysis needs.

Associative Relationships: Functional dependency relations between entities of the same type, which result from the composition of resources for the purpose of value-added capability provisioning.

Equivalence Relationships: Non-functional similarity relations between entities of the same type.

Hierarchical Relationships: Relationships across the abstraction levels in the scope of a service network model, connecting entities from different types.

Process: A non-executable description of a business capability, defined as service requests on the network, which execution depends on services in the network.

Service: The interface description of a technical entity enabling the access to one or more business capabilities.

Service Network: A system of service systems interconnecting easy adaptable and exchangeable modular components (services) in the context of value-added business capabilities (processes) on demand of physical or legal entities (actors) interacting over a free to access dynamic Internet-based infrastructure.

Uniformity: Clear definition of concepts in a form understandable for all stakeholders from the business and IT domain.

REFERENCES

Allee, V. (2000). Reconfiguring the value network. *The Journal of Business Strategy*, *21*(4), 36–39. doi:10.1108/eb040103

Arsanjani, A., & Holley, K. (2006). The Service Integration Maturity Model: Achieving Flexibility in the Transformation to SOA. *Proceedings of the IEEE International Conference on Services Computing* (p. 515). IEEE Computer Society. doi:10.1109/SCC.2006.104

Atkinson, C., & Kuhne, T. (2003). Model-Driven Development: A Metamodeling Foundation. *Software, IEEE*, *20*(5), 36–41. doi:10.1109/MS.2003.1231149

Basole, R. C., & Rouse, W. B. (2008). Complexity of service value networks: Conceptualization and empirical investigation. *IBM Systems Journal*, *47*(1), 53–70. doi:10.1147/sj.471.0053

Basu, S., Casati, F., & Daniel, F. (2008). Toward Web Service Dependency. *IEEE International Conference on Services Computing (SCC 2008)* (pp. 422-429). Honolulu: IEEE. doi:10.1109/SCC.2008.45

Blau, B., Kramer, J., Conte, T., & van Dinther, C. (2009). Service value networks. *IEEE Conference on Commerce and Enterprise Computing (CEC'09)*, (pp. 194–201). doi:10.1109/CEC.2009.64

Cardoso, J., Pedrinaci, C., & De Leenheer, P. (2013). Open semantic service networks: modeling and analysis. *Proceedings of The 4th International Conference on Exploring Service Science* (pp. 141–154). Springer. doi:10.1007/978-3-642-36356-6_11

Danylevych, O., Karastoyanova, D., & Leymann, F. (2010, September). Service Networks Modelling: An SOA & BPM Standpoint. *Journal of Universal Computer Science (J.UCS), 16*(13), 1668-1693.

Danylevych, O., Leymann, F., & Nikolaou, C. (2011). *A Framework of Views on Service Networks Models. Enterprise and Organizational Modeling and Simulation* (pp. 21–34). London, UK: Springer. doi:10.1007/978-3-642-24175-8_2

Erl, T. (2005). *Service-oriented Architecture – Concept, Technology, and Design*. Prentice Hall.

Erl, T. (2007). *SOA Principles of Service Design*. Prentice Hall.

Erl, T. (2009). *SOA Design Patterns*. Prentice Hall.

Gordijn, J., & Akkermans, H. (2001). Designing and evaluating e-business models. *IEEE Intelligent Systems*, *16*(4), 11–17. doi:10.1109/5254.941353

Jansen, S., Brinkkemper, S., & Finkelstein, A. (2009). Business network management as a survival strategy: A tale of two software ecosystems. *Proceedings of 1st Workshop on Software Ecosystems*, (pp. 34-48).

Josuttis, N. (2007). *SOA in Practice*. O'reilly.

Kabzeva, A., Götze, J., Lottermann, T., & Müller, P. (2013). Service Relationships Management for Maintenance and Evolution of Service Networks. *ICSEA 2013*, *The Eighth International Conference on Software Engineering Advances* (pp. 201-207). Venice, Italy: IARIA.

Kabzeva, A., Götze, J., & Müller, P. (2014). Service Network Modeling: A Generic Approach. *4th International Workshop on Adaptive Services for the Future Internet (WAS4FI 2014).* Manchester, UK.

Keller, G., & Teufel, T. (1998). *SAP R/3 Process Oriented Implementation*. Boston, MA, USA: Addison-Wesley Longman Publishing Co.

Krafzig, D., Banke, K., & Slama, D. (2005). *Enterprise SOA: service-oriented architecture best practices*. Prentice Hall Professional.

Liu, Y., Fan, Y., & Huang, K. (2013). Service ecosystem evolution and controlling: A research framework for the effects of dynamic services. *Proceedings of the International Conference on Service Sciences*.

Lucassen, G., van Rooij, K., & Jansen, S. (2013). *Ecosystem Health of Cloud PaaS Providers. Software Business. From Physical Products to Software Services and Solutions* (pp. 183–194). Potsdam, Germany: Springer. doi:10.1007/978-3-642-39336-5_18

National Information Standards Organization. (2005). *Guidelines for the Construction, Format, and Management of Monolingual Controlled Vocabularies. ANSI/NISO Z39.19-2005 (R2010)*. Baltimore, MD, USA: NISO.

Northrop, L., Feiler, P., Gabriel, R., Goodenoug, J., Linger, R., Longstaff, T., & Wallnau, K. (2006). *Ultra-Large-Scale Systems - The Software Challenge of the Future. Software Engineering Institute*. Pittsburgh, PA: Carnegie Mellon University.

OASIS. (2006). *Reference Model for Service Oriented Architecture 1.0*. OASIS.

OMG. (2011). *Business Process Model and Notation (BPMN) Version 2.0. OMG Specification*. OMG.

Papazoglou, M. P. (2008). *The challenges of service evolution. Advanced Information Systems Engineering* (pp. 1–15). Springer. doi:10.1007/978-3-540-69534-9_1

Scheer, A.-W., Thomas, O., & Adam, O. (2005). Process modeling using event-driven process chains. *Process-Aware Information Systems*, 119-146.

Schulz, F., Caton, S., Michalk, W., Haas, C., Momm, C., Hedwig, M., . . . Rolli, D. (2013). Integrated modeling of technical and business aspects in service networks. In M. Fathi, Integration of Practice-Oriented Knowledge Technology: Trends and Prospectives (pp. 119–128). Springer. doi:10.1007/978-3-642-34471-8_10

Schwarz, H., Ebert, J., & Winter, A. (2010). Graph-based traceability: A comprehensive approach. *Software & Systems Modeling*, *9*(4), 473–492. doi:10.1007/s10270-009-0141-4

Sendall, S., & Kozaczynski, W. (2003). Model Transformation – the Heart and Soul of Model-Driven Software Development. *IEEE Software*, *4*(5), 42–45. doi:10.1109/MS.2003.1231150

Sommerville, I., Cliff, D., Calinescu, R., Keen, J., Kelly, T., Kwiatkowska, M., & Paige, R. et al. (2012, July). Large-scale complex IT systems. *Communications of the ACM*, *55*(7), 71–77. doi:10.1145/2209249.2209268

Wang, Y., Taher, Y., & van den Heuvel, W.-J. (2012). *Towards smart service networks: An interdisciplinary service assessment metrics. Enterprise Distributed Object Computing Conference Workshops (EDOCW)* (pp. 94–103). IEEE.

Wynn, D. E. (2007, August). Understanding the health of technological ecosystems: the case of professional open source software. Doctoral dissertation, University of Georgia.

Aneta Kabzeva received her diploma in computer science from the University of Kaiserslautern, Germany, in 2007. Currently, she is a researcher in the Department of Computer Science at the University of Kaiserslautern, where she is pursuing her PhD on service network modeling. Her main research interests lie in the area of service-orientation, cloud computing, and software evolution.

Joachim Götze is a researcher at the Regional Computing Center Kaiserslautern, Germany. He received his PhD in computer science in 2014 from the University of Kaiserslautern, after studying computer science until 2004. His main research interests are in the area of distributed computing and service-orientation.

Paul Müller is professor of computer science at the University of Kaiserslautern. He received his PhD from the Faculty of Mathematics at the University of Ulm in the field of statistics. Thereafter, he was responsible for various research projects and the development of a statewide computer network in Germany. His current research interests are mainly focused on distributed systems, future internet-working architectures and service-oriented architectures. His research group Integrated Communications Systems Lab (ICSY) is aiming at the development of services to implement integrated communication within heterogeneous environments especially in the context of the emerging discussion about Future Internet.