Managing Interoperability Knowledge in Open Service Ecosystems

Toni Ruokolainen and Lea Kutvonen
Department of Computer Science
University of Helsinki
Helsinki, FINLAND
{Toni.Ruokolainen,Lea.Kutvonen}@cs.Helsinki.FI

***Abstract*—Establishing loosely coupled collaborations between services provided by autonomous enterprises poses several requirements and challenges for the surrounding service ecosystem. In such context emphasis must be laid especially on the correctness of available metainformation and its usage. Towards this purpose, we characterize service ecosystems by a set of metamodels. The metamodels include domain ontology metamodels defining the vocabulary and knowledge elements needed for collaboration establishment and management, and metamodels relating such domain ontologies with infrastructure services and service production facilities. In this paper we describe the metamodels and discuss their application for interoperability knowledge management.**

I. Introduction

Establishing loosely coupled collaborations between services provided by autonomous enterprises poses several requirements and challenges for the surrounding service ecosystem. Openness and agility are required from the collaboration management infrastructure and service-oriented software engineering facilities for enabling efficient networked business. These requirements and stressed by the current business models based on highly specialised core competencies and outsourcing of secondary functionality.

Loosely coupled nature and openness of a service ecosystem, and autonomy of its participants induce interoperability problems. *Interoperability* means the capability of (enterprise information) systems to collaborate in such a fashion that eventually either their mutual goals become fulfilled or their co-operation is dissolved in a controllable manner in case of a fault. Interoperability problems can range from simple technological incompatibilities to conflicts between business strategies.

We characterize interoperability as having technical, semantic and pragmatic concerns [1]. *Technical interoperability* means that the technological facilities underlying the services are compatible, such that communication paths can be established, for example. *Semantic interoperability* deals with the meaning of exchanged information and message exchange patterns. *Pragmatic interoperability* is achieved if the intentions, business rules, and organizational policies of collaborating parties are compatible with each other.

We categorize the means for achieving interoperation to three approaches, namely integration, unification and federation [1]. Since integration and unification approaches do not preserve all relevant information about interoperation [2], features related to interoperation can not in these approaches be considered dynamically. In the federated approach interoperability is achieved by utilizing a shared metamodel and making the relevant information about features affecting interoperation explicit. This information may comprise different kinds of artifacts, such as prescriptive models, ontologies or service interface descriptions.

We call the set of artifacts conforming to the shared metamodel as *interoperability knowledge*, as its primary purpose is to provide sufficient grounds for establishing interoperable collaborations by declaring consistency and conformance criteria for knowledge elements. Each collaboration participant may have their own representations of the artifacts describing their business domain and services. The infrastructure services provided by the operational environment are used for maintaining the consistency of interoperability knowledge, and for instrumenting collaboration establishment processes based on this knowledge.

A service ecosystem comprises three fundamental components: *1)* a service-oriented collaborative computing environment, *2)* service and business process development facilities, and *3)* a conceptual framework conjoining the collaborative computing environment, development tools and processes, and all relevant actors within the ecosystem. The collaborative computing environment provides infrastructure services for establishing loosely coupled collaborations. Openness of the ecosystem is largely determined by the properties of the infrastructure services and collaboration establishment process provided by the computing environment. Development facilities provide for agile production of service-based solutions and utilize the infrastructure services of the computing environment for enabling global software engineering practices, for example. The conceptual framework must provide concepts for ensuring correctness of service and business process development activities, and runtime use of metainformation. The conceptual framework and the features of the collaborative computing environment prescribe concurrently what kinds of interoperability concerns can be addressed and managed.

In our previous work (see [1], [3], for example) we have described the Pilarcos platform for interoperability management. The consistency of the interoperability knowl-

edge has been maintained in this prototype platform by manually implemented repository functionality, collaboration establishment protocols and best practices shared by the system users. Based on the experiences gathered from the previous work, implementation of the prototype services, and extensive analysis on collaborative computing and interoperability, we are now making this knowledge about open service ecosystems explicit and formal with metamodels and domain ontologies.

This paper presents an approach for establishing service ecosystems where the elements of the ecosystems are characterized by a set of formal models. The models include domain ontologies defining the vocabulary and knowledge elements needed for collaboration establishment and management, and metamodels relating the domain ontologies with infrastructure services and service production facilities. From these models we can derive a reference architecture comprising of knowledge repositories needed for maintaining the corresponding service ecosystem metainformation. We describe the models, and discuss their role and usability in defining and implementing service ecosystems. The approach utilizes principles from model-driven engineering [4] and service-oriented computing [5], and contributes to the development of service ecosystems and instrumentation of service-oriented software engineering [6].

The structure of this paper is as follows. Section II first identifies the fundamental elements and characteristics of collaborative computing environments, and introduces the Pilarcos platform [1], [3]. In Section III we provide a metamodel defining the conceptual framework we use for establishing a service ecosystem based on the Pilarcos platform. More over, linguistic and ontological metamodelling [7], [8] principles are applied in the construction of the metamodel for providing foundations for knowledge management. To further fulfill the requirements and overcoming the challenges of open and agile service ecosystems, emphasis must be laid on the correctness of available metainformation and its usage. In Section IV we describe the foundations for managing interoperability knowledge in model repositories, and conclude in Section V with a discussion of related research.

II. Collaborative computing environments

Collaboration is a process of shared creation among a group of entities which share information, resources, responsibilities and rewards to achieve a common goal [9]. A *collaborative computing environment* is a distributed computing system providing facilities for (electronic) collaboration. Collaborative computing environment comprises autonomous *collaboration agents* and a set of infrastructure services provided by the *operational environment* [10].

A collaboration agent provides a technical representation of an entity, such as an individual or an organization, willing to collaborate and mediates their activities and decisions towards other entities. Collaboration agents act during collaboration establishment and actual operation of the collaboration. The agents are loosely coupled and interoperate with each other to meet the collaboration objective. The collaboration is facilitated by a set of infrastructure services provided by the operational environment for establishing and controlling collaborations. The set of infrastructure services are exposed to collaboration agents and other actors in the collaborative computing environment typically in form of a middleware platform.

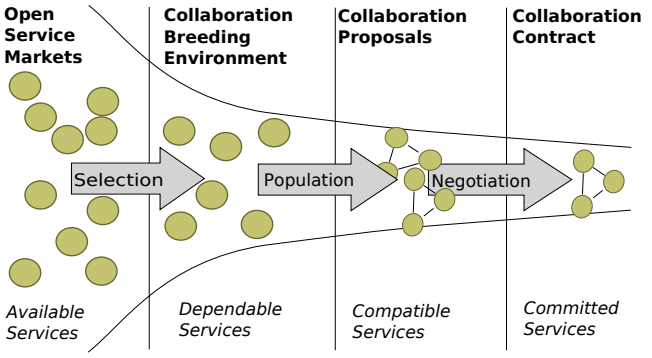

Figure 1. Illustrating the collaboration establishment process.

Collaborations are formed using a *collaboration establishment process* which involves service selection, and collaboration population and negotiation phases [1], as illustrated in Figure 1.

During the service selection phase appropriate services are located and selected from the ones available in open service markets. The selection is based on criteria set by the form of the collaboration and the requirements set by its initiator. The services selected to the breeding environment are dependable in sense that they fulfill the necessary technical and semantic interoperability requirements of the corresponding kind of collaboration and they are provided by trusted partners. Infrastructure services providing discovery and selection of services, as well as trust and reputation management mechanisms [3], are utilized for this purpose.

The collaboration breeding environment acts as a catalysis platform for potential collaborations. The population phase produces a set of collaboration proposals from the dependable services and a model characterizing the structure and requirements of the collaboration [1], [3]. Semantic interoperability is addressed by the population phase: compatibility between non-functional property metrics and service usage policies are addressed at this stage, for example.

The collaboration proposals are further refined using negotiations taken between the collaboration agents. Especially the pragmatic aspects of interoperability, such as expression of the agents' willingness to collaborate, are considered during the negotiation phase. The negotiations result in formulation of a collaboration contract which states the

responsibilities for each participating entity, the structure of the collaboration, and non-functional features expected from the corresponding cooperation facilities, such as communication channels. The collaboration contract is then used for managing the operation of the collaboration [11].

A. Characteristics of environments

Collaborative computing environments can be classified and compared with respect to their openness and interoperability management mechanisms. These are the primary properties that determine their usability for enabling loosely coupled service ecosystems. Properties such as collaboration agent activity, sociality and level of participant autonomy supported can also be used as classifying features of collaborative computing environments [10].

Environment openness measures the level of dynamism with respect to the selection of services available for collaboration during the establishment process. The degree of openness can be characterized as *1)* closed, *2)* semi-open, or *3)* open [10]. In the closed case, the set of compatible services for collaboration proposals is static. That is, the phases of service selection and collaboration population are predetermined. Consequently, the collaboration establishment process involves only the negotiation phase, typically also neglected in systems involving a closed environment.

In the semi-open case, the set of dependable services located in the collaboration breeding environment is static (closed world), but services can be selected dynamically for collaborations from this pool (open world with respect to the actual collaboration). In this scenario, the service selection phase is pre-determined, but collaboration population can be done dynamically.

An open collaboration environment provides most flexibility as the set of services available for collaborations can be introduced within open service markets. The collaboration breeding environment matching the requirements and properties of a specific collaboration is established dynamically from the available services. However, this freedom needs to be regulated with feasible infrastructure mechanisms that provide trusted and interoperable computing support.

Different collaborative computing environments have distinctive properties and maturity with respect to the degree of opennenness (closed, semi-open, open) they support and on the balance between dependence (integrated, unified, federated) and autonomy of collaborating systems. The traditional ERP, EAI [12], [13] and other integration systems belong closed integrated solutions: collaboration is achieved through tightly coupled integration between pre-determined subsystems.

Inter-enterprise application integration (I-EAI) refers to integration of information systems residing in distinct enterprises for enabling supply-chain management or subcontracting relationships, for example. Standards such as ebXML [14] and RosettaNet [15] that define domain-specific business vocabulary and processes are also used for achieving limited kind of semantic interoperability between enterprises by means of unification. These kinds of systems can be classified as closed unified solutions.

Collaborative Networks (CN) [9], such as virtual organizations, are loosely coupled collaborations between entities bound together by an explicit collaboration contract for achieving common or compatible goals. They can be characterized as semi-open, unified solutions comprising a static breeding environments (VBE) of dependable services set for dynamic formation of virtual enterprises or organizations [16].

The challenges differ significantly from each other in each class of systems in the above classification based on the environment openness and interoperability management approach. It is relevant to make it clear that an interoperability knowledge base is relevant in federated approaches, although it might become handy in others as well. The Pilarcos approach described below belongs to open federated solutions.

B. Pilarcos framework

The Pilarcos framework [1], [3] for inter-enterprise collaboration management comprises a collaborative computing environment and concepts for managing loosely coupled B2B collaborations. The framework uses federation for establishing interoperability between autonomous ecosystem members, and repositories providing consistency and conformance of interoperability knowledge.

The Pilarcos B2B middleware provides an operational environment designed for lowering the cost and effort of collaboration establishment and to facilitate the management and maintenance of electronic business networks. The infrastructure services include [1]:

- services for establishing, modifying, monitoring, and terminating collaborations, or looking from the business service point of view, operations for joining and leaving a collaboration either voluntarily or by community decision and leaving a trace in the global business world about the success of the collaboration; and
- a set of repositories for storage of collaboration models, and ontologies of service types and services, for example, to support interoperability validation.

The Pilarcos framework proposes a model of inter-enterprise collaborations as *business networks* consisting of independently developed business services. A *business service* denotes a set of functionalities provided by an enterprise to its clientele and partners. It is governed by the enterprise's own business rules and policies, as well as by business contracts and regulatory systems controlling the business area. A business network is established dynamically to serve a certain business scenario or opportunity that is made commonly known by publishing a *business network*

model (BNM) [1]. The business network model captures the roles and business processes that are relevant for the business scenario, for example.

The collaboration agents in the Pilarcos framework are known as *network management agents* [11]. A network management agent (NMA) represents a collaboration member in the business network. It handles negotiations with potential new members and re-negotiations if members are changed, it up-keeps status information for the collaboration, and determines the suitable reaction to collaboration events such as contract termination. Every member of the service ecosystem has its own network management agent, and they are considered to be fully trusted local agents [3].

The collaboration establishment process in the Pilarcos framework is explicit with service selection, population and negotiation phases. When collaboration is needed, one of the partners initiates the collaboration establishment process via its local NMA. This NMA first calls *populator* [3], an infrastructure service of the Pilarcos B2B middleware responsible for executing collaboration establishment processes.

In the service selection phase *service types* [17] are used for discovering service offers compatible with the corresponding business network roles. A service type defines the characteristic features for a kind of business services by expressing its behavioural semantics and non-functional features. Service types are used to constrain behaviour of business services, to validate consistency of cooperation abstractions, and for verifying behavioural substitutability and compatibility between kinds of services [10].

After service discovery, the populator chooses the most suitable service offers for each role from the set of dependable services. Conformance with the constraints and requirements defined by the business network model and the collaboration initiator are used for selecting the set of compatible services. Based on the collaboration proposals given by the populator, the initiating NMA runs a negotiation with the NMAs of the other proposed partners [3].

The interoperability management approach is a federated one: all business services are developed independently, and the provided B2B middleware services are used to ensure that technical, semantic, and pragmatic interoperability is maintained in the business network. A shared metamodel formalizing the concepts of the Pilarcos framework provides consistency and conformance criteria to be maintained by the knowledge repositories.

III. A METAMODEL FOR SERVICE ECOSYSTEMS

The Pilarcos framework uses federation to establish interoperability by utilizing a metamodel that formalizes the concepts of the service ecosystem. Such a metamodel must address especially the openness of the collaboration environment and provide means for service discovery and selection, collaboration population, and negotiation. To support the openness of the environment, the metamodel must not restrict the kinds of services that can be declared. On the other hand, for enabling efficient service discovery and selection, some means for service categorization must be provided. That is, the service ecosystem metamodel must involve concepts that allow construction of dynamic and evolvable ontologies of service categories.

An open and agile service ecosystem necessitates two different kinds of models: prescriptive *system models* specifying artifacts usable for service-oriented software engineering purposes and configuration of collaborations, and descriptive *domain ontologies* providing the vocabulary for expressing the concepts of the ecosystem. The service ecosystem metamodel must address usage of both system models and domain ontologies.

These requirements laid for the Pilarcos service ecosystem metamodel are met by formalizing the elements of the service ecosystem as a set of models, and providing foundations for managing interoperability knowledge consisting of domain ontologies and system models [10]. These principles are discussed in the following subsections.

A. Defining the elements of a service ecosystem

The Pilarcos service ecosystem is formalized by four inter-related metamodels [10]: a *domain ontology metamodel* defining the fundamental concepts in the ecosystem, a *methodology metamodel* providing means for defining service-oriented software engineering facilities, a *domain reference metamodel* used for describing the infrastructure services available, and a *knowledge management metamodel* enabling foundations for maintaining interoperability knowledge in the system,

The domain ontology metamodel is an ontological metamodel defining the vocabulary used in the service ecosystem. In the case of Pilarcos framework, this vocabulary includes business networks, business services, service types, etc. Collaboration is defined as a role-based cooperation taking place between entities. The entities encountered in cooperations are distinguished to two different kinds, namely functional and legal entities [10]. Functional entities provide the facilities for delivering cooperative activities while legal entities represent the cooperating parties that are bound by mutual agreements or contracts to deliver the required commitments. Entities have features that affect their suitability to a certain kind of cooperation. A *feature* is here considered as a distinguishable capability or property of an element of a cooperative community. A *functional feature* is a feature that can be directly associated with a functional entity [10]. A *non-functional feature* describes a feature of a legal entity, cooperation facility such as communication channels, or a service relationship [10]. More specific kinds of entities and features, such as service types and business protocols [17], are derived from these foundational concepts using ontological specialization.

The methodology metamodel is used for defining actors, artifacts, and processes used in service-oriented software engineering processes for producing work products. The metamodel is based on the OPEN Process Framework [18]. Especially, the methodology metamodel defines modelling artifacts that are used for representing concepts of the domain ontology.

The domain reference metamodel provides a language for describing the infrastructure services, collaboration agents and the collaboration establishment process of a collaborative computing environment. The domain reference metamodel is an extension [19] of the knowledge management metamodel; especially, it declares a knowledge repository as one sort of an infrastructure service.

B. Foundations for interoperability knowledge management

Foundations for interoperability knowledge management in the Pilarcos service ecosystem are given by metamodels that formalize a conceptual unification between ontological and linguistic modelling, concepts defined in domain ontologies, and knowledge repositories.

For addressing the system engineering and ontological role of models in a service ecosystem, we separate two orthogonal dimensions in the Pilarcos service ecosystem metamodel. A linguistic metamodel is used for defining modelling languages and their primitives on the metamodel level and so-called *linguistic instantiation* [7] is used for instantiating model elements from the types defined within a corresponding metamodel. Linguistic instantiation crosses modelling language abstraction levels and forms the basis for linguistic metalevels [7]. Typically the number of linguistic metalevels is restricted to three, which has been witnessed to be sufficient for providing facilities for development of domain specific languages. The MOF [20] standard comprises linguistic metalevels of metametamodels, metamodels and models, for example.

Domain concepts are declared by utilizing ontological metamodels where so-called *ontological instantiation* [7] is used for creating domain specific artifacts using the concepts defined at the upper-level ontology, or an ontology metamodel [7], [8]. Ontological instantiation takes place within a linguistic metalevel between two models representing concepts related by an ontological *"is-a"* relationship. Ontological instantiation provides the support for facilitating dynamic user extensions to modelling concepts, modelling notation and the models created from them [7]. The number of ontological metalevels depend on the domain of interest. A domain ontology describing biological species could contain 13 ontological metalevels for each of the levels in the Linnaean taxonomy [21], for example.

The Pilarcos service ecosystem metamodel uses three linguistic metalevels which follow the MOF [20] approach. We use two ontological metalevels of ontological types and instances [10]. In the Pilarcos service ecosystem concepts

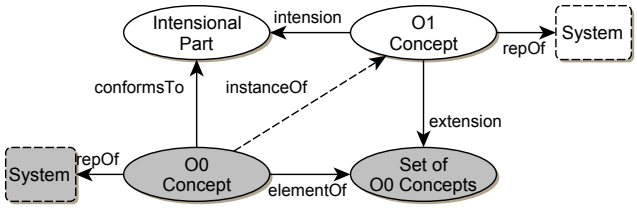

Figure 2. Ontological modelling relationships

are defined by domain ontologies conforming to the domain ontology metamodel. Ontological type level concepts are declared for defining new categories of concepts, such as new kinds of services or collaboration types, for example. Ontological instance level concepts are declared for representing member of the concept categories, such as business services or collaboration usages, for example.

Ontological relationships between type and instance level concepts are illustrated in Figure 2. While ontological instantiation is probably the primary means for a user to derive ontological hierarchies, this relationship is actually a derived relationship induced by *conformsTo* and *elementOf* relationships between an ontological instance level concept and the intensional and extensional meaning of its ontological type. An ontological instantiation relationship holds between two concepts if and only if a concept is conformant to the intentional part of its type and belongs to the extension of the ontological type [8]. Both at the type and instance level concepts are considered as representations of (*repOf*) of systems, following the principles of model-driven engineering [8], [22].

Interoperability knowledge management is formalized by three metamodels: *1)* a systemic metamodel providing the foundations for linguistic and ontological metamodelling, *2)* a global model management metamodel describing the modelling artifacts and their inter-relationships, and *3)* a knowledge repository metamodel making explicit the relationships between ontological concepts, modelling artifacts and infrastructure services.

The systemic metamodel formalizes the metamodelling principles introduced in [8], [22], [23]. Three kinds of systems are identified and differentiated by the metamodel, following the (incomplete) classification presented in [23]: *PhysicalSystem*s, *DigitalSystem*s, and *AbstractSystem*s. An *AbstractSystem* is the only one associated explicitly with intensional and extensional descriptions, where the extensional semantics of a system is provided by the concept of *Set*. Subsequently, every ontology in this modelling framework is considered as an abstract system.

The global model management metamodel defines the meaning of different kinds of models, and especially declares relationships between concept intentions defined in

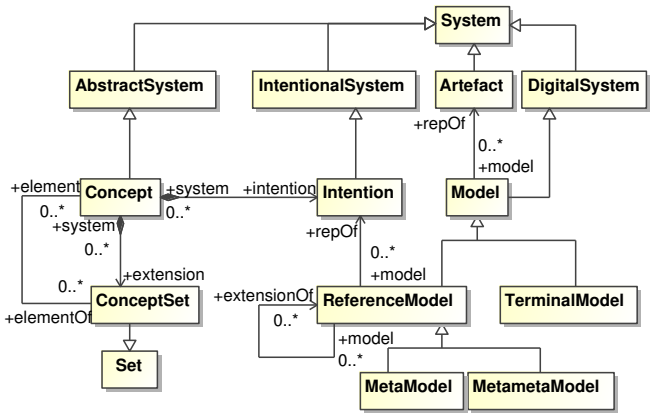

Figure 3. The global model management metamodel for the Pilarcos framework.

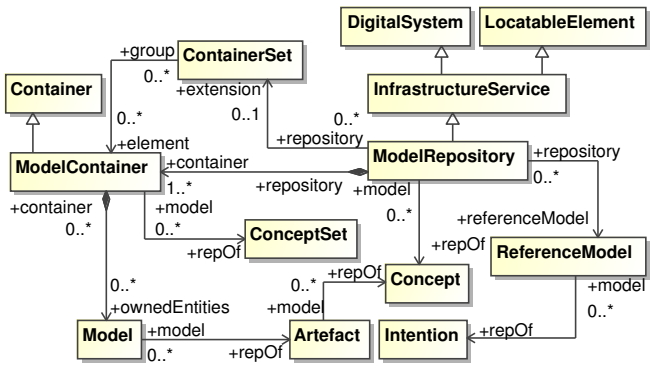

Figure 4. The model repository metamodel

domain ontologies and system models. A simplified illustration of the metamodel is given in Figure 3. The notion of a *Model* and the classification of different kinds of models, e.g. *ReferenceModels* and *TerminalModels*, is based on the global model management framework described in [22]; we however provide extensions to the notion of *Model* that are foundational from the interoperability knowledge management perspective.

In the global model management metamodel of the Pilarcos framework a *Model* is considered as a digital representation of an *Artefact*. Artifacts are defined as elements of methodologies by the methodology metamodel [10] and associated with a corresponding technical space.A technical space is *"a working context with a set of associated concepts, body of knowledge, tools, required skills and possibilites"* [24]. A technical space determines the kind of representation format, e.g. MOF model or OWL file, used for describing the Artefact, for example.

The unification between ontological and linguistic modelling is characterized by the relationship between reference models and concept intentions: linguistic type level models represented by the *ReferenceModel* are considered as representations of domain concept intentions. Domain ontologies are in this framework considered as collections of concepts and their inter-relationships. An ontological concept is considered as an abstract system. The intensional meaning of a concept is provided by the *Intension* element and extensional meaning by a set of concepts (*ConceptSet* element).

The model repository metamodel is part of the domain reference metamodel. It formalizes the connections between knowledge repositories, linguistic models and domain concept semantics. These connections provide the foundations for managing interoperability knowledge in model repositories and relate service-oriented software engineering facilities with collaborative computing environments. A simplified illustration of model repository metamodel is given in Figure 4.

A *ModelRepository* is considered as a locatable digital system that comprises model containers. A model repository is associated with a unique *ReferenceModel* which represents an intention of a domain ontology concept. The *ModelContainer* provided by a repository comprises a set of *Model* elements. As defined in the Pilarcos global model management metamodel, each *Model* is a representation of a methodology Artefact; this connection unifies service-oriented software engineering frameworks with a collaborative computing environment. Consequently, a *ModelContainer* is interpreted as a representation of a set of concepts (we assume transitivity of the *repOf* modelling relationship).

As the notion of an *Artefact* is associated with a technological space, a model repository may include several model containers corresponding to each technological space. A model repository thus may support different technological representations for the linguistic instances of the reference model. In this case, mappings between technical spaces should be provided.

The models that are stored within a model container conform to the reference model associated with the corresponding repository. Most importantly, a model repository can be considered as a technological representation of an ontological concept. The extensional semantics of a type level concept is declared by using the *ContainerSet* element, which includes all model containers for instance level repositories that contain instances of the ontological type. The mechanics of knowledge management, including the maintenance of the concept extensions, is further discussed in the next section.

IV. Controlling Metainformation Correctness in Knowledge Repositories

A *knowledge repository* is an infrastructure service that is responsible for maintaining conformance and consistency between domain concepts. From a knowledge management

perspective the Pilarcos service ecosystem metamodel involves several kinds of conformance and consistency relationships that have to be addressed. In this section, we describe how these relationships are maintained during the operation of a knowledge repository. During the discussion, we use the Pilarcos service type repository [1], [10], [17] as an example repository.

Service type is an abstract description of business service capabilities, behaviour and structure. It is a unit of service design and composition representing a bilateral, conversational service-interface. The notion of service type [10], [17] is provided with a behavioural type system based on the notion of session types [25], [26] that provides a formalization for behavioural compatibility and subtyping, and behavioural typing of processes. The ontological instances of service types in the Pilarcos service ecosystem metamodel are the business services. A knowledge repository maintaining service type models is known as a *service type repository*.

A knowledge repository maintains two kinds of conformance relationships between knowledge elements: linguistic conformance between a model and the repository reference model, and ontological conformance between a concept instance and its ontological type. The linguistic conformance relationship is defined as part of the technical space associated with the corresponding engineering Artefact, which is described by the methodology metamodel. Most typically, a linguistic conformance relation similar to the MOF framework [20] will be used. In the case of the service type repository, linguistic conformance is based on a metamodel representation of the service type concept intention. The service type metamodel comprises a set of service features and a set of business protocols [10], and is provided concretely as an Eclipse Modelling Framework model [27].

Ontological conformance is a relationship defined between two ontological metalevels and typically involves semantic interpretation. Formulation for the semantic correspondence between a concept at the ontological instance level and the intention of a concept at ontological type level must be provided. This semantic correspondence is defined as part of the domain ontology metamodel. In a modelling framework comprising two ontological metalevels ontological conformance relationship is not declared for ontological types; for ontological instances the corresponding type has to be declared. In the case of service types, the ontological conformance relationship between a business service and the corresponding service type becomes a behavioural typing relationship based on the session typing discipline [25], [26]. In addition, service features (e.g. "service price") declared in a service type must be provided with conforming properties (e.g. an amount in a specific currency) in a corresponding business service.

In addition to the conformance relationships, a knowledge repository has to maintain several consistency relationships between ontological concepts. The ontological modelling relationship of *elementOf* is represented in the model repository metamodel by the *ownedEntities* relationship between a repository model container and the models it contains. The model container is the owner of the models it contains, and is considered responsible for managing their life-cycle, and especially, naming. Names are fundamental elements in distributed computing systems as they can be used for referencing different entities in different contexts [28] and also because of their ontological relevance. Naming policies, e.g. use and construction of name spaces and naming conventions, are prescribed by the technical space used for creating the corresponding engineering artifacts, and by policies of the business domain and repository authority, for example. In the Pilarcos model repositories naming is URI-based. Each repository may declare their own naming policies; federated name management is not yet supported. For simplicity, we currently assume that the model URI is derived from the repository host name such that model name resolution is a simple process of contacting the corresponding host using the model URI prefix as a host name.

The extensional semantics of ontological type concepts is represented in model repositories as references to external model containers. When a new instance level concept is published in a repository (let us call it the "instance repository"), the repository owning the corresponding type level concept (the "type repository") is contacted. The type repository is used for validating the ontological conformance of the ontological instance with respect to the type level concept. If the conformance validation succeeds, the type repository adds the model container of the instance repository to the set of containers representing the extension of the type level concept. That is, a type repository is aware of the instance repositories that have instance level concepts conforming to the ontological types that are contained in the type repository. Consequently, ontological instances can be queried based on their type from the type repositories. From administrative perspective, it is a matter of choice between trust, efficiency, and maintainability related issues if a type repository and corresponding instance repositories should be distributed in different administrative domains.

Finally, ontological relationships can be added and removed between concepts by the actors in the ecosystem. The semantics of the relationships are defined as part of the domain ontologies. In the case of service types, a service type can be subtype of another, for example [10], [17]. The semantics of this relationship between two service type concepts is defined by the session subtyping discipline [25], [26] and is an intra-repository relationship. Ontological relationships can be declared between concepts maintained in different repositories as well. In this case, the owner of the relationship endpoint is considered responsible for validating the relationship between the concepts. For the purpose of validating the domain-specific semantics of onto-

logical relationships transformations are typically needed for appropriate representation of the concept models. In service type repositories Prolog is used for representing models when implementing service subtyping and compatibility validation.

V. Conclusion

For the establishment and runtime control of loosely-coupled federated collaborations, globally available interoperability knowledge is required. Corresponding knowledge repositories provide vocabulary for business network structures (aggregations of business processes for named business scenarios), service types declaring known interface types and properties relevant for the kind of services in question, and information about actual services provided by enterprises.

The interoperability knowledge is shared between the design and production environments for services, and the operational time environment for collaborations. For example, the service type knowledge can be created at service design time, and be used for model-driven production techniques. On the other hand, the same service type knowledge is used at operational time for determining if an actual service on the market can be plugged in to a collaboration or not, in terms of various levels of interoperability.

This twofold use of the same models brings in the necessity of including both linguistic and ontological considerations to the models. As far as we are aware, Pilarcos is one of the very few architectures tackling this issue. In distinction to such work as [29], which use transformations or annotations to unify ontological and linguistic technical spaces, our approach provides a more comprehensive unification between the descriptive ontologies and prescriptive system models.

Approaches for service interoperability based on shared metamodels have been taken for example in COSMO [30] and PIM4SOA [31]. These approaches concentrate on on providing a unifying metamodel for service modelling and for validating the consistency and conformance of service models during design. In comparison to COSMO and PIM4SOA, our metamodel is more suitable for acting as a metamodel for open service ecosystems, since in addition for defining a metamodel for service descriptions it also makes explicit the relationships between the different elements of service ecosystems, and puts emphasis also on the dynamic nature of service ecosystems.

In the future, a modelling methodology that addresses both the ontological and linguistic modelling viewpoints should be given for fully taking advantage of this approach. Until then, tools and methodologies can be used separately in their respective modelling domains (e.g. OWL for ontological modelling and UML for linguistic modelling) to provide the modelling framework with domain ontology concepts and their intentions.

The model repository concept described in this paper is related to research in model management [32], [33]. The AM3Core metamodel defined by the AMMA framework [32] has been used as a starting point for the global model management metamodel presented in this paper. More over, the Pilarcos model repositories are implemented using model transformations that generate repository interfaces conformant with the ModelBus [33] framework [10].

Acknowledgement

This work has been performed within the CINCO group (Collaborative and Interoperable Computing Group) at the University of Helsinki, lead by Lea Kutvonen. The group mission is to forward facilities of service interoperability and dynamic business networks and has worked through a number of national projects funded by the Academy of Finland, the Finnish Funding Agency for Technology and Innovation (TEKES) and Finnish companies, and the INTEROP NoE.

References

[1] L. Kutvonen, T. Ruokolainen, and J. Metso, "Interoperability middleware for federated business services in web-Pilarcos," *International Journal in Enterprise Information Systems, Special issue on INTEROP-ESA 2005*, vol. 3, no. 1, 2007.

[2] T. Ruokolainen and L. Kutvonen, "Interoperability in Service-Based Communities," in *Business Process Management Workshops: BPM 2005 International Workshops, BPI, BPD, ENEI, BPRM, WSCOBPM, BPS*, ser. Lecture Notes in Computer Science, C. Bussler and A. Haller, Eds., vol. 3812. Springer-Verlag, 2006, pp. 317–328. [Online]. Available: http://dx.doi.org/10.1007/11678564_28

[3] L. Kutvonen, J. Metso, and S. Ruohomaa, "From trading to eCommunity management: Responding to social and contractual challenges," *Information Systems Frontiers (ISF) - Special Issue on Enterprise Services Computing: Evolution and Challenges*, vol. 9, no. 2–3, pp. 181–194, Jul. 2007.

[4] D. C. Schmidt, "Model-Driven Engineering," *Computer*, vol. 39, no. 2, pp. 25–31, Feb. 2006.

[5] M. P. Papazoglou and D. Georgakopoulos, "Service-oriented computing," *Commun. ACM*, vol. 46, no. 10, pp. 24–28, 2003.

[6] Z. Stojanovic and A. Dahanayake, Eds., *Service-Oriented Software System Engineering: Challenges and Practices*. Idea Group Publishing, 2005.

[7] C. Atkinson and T. Kühne, "Model-driven development: A metamodeling foundation," *IEEE Softw.*, vol. 20, no. 5, pp. 36–41, 2003.

[8] D. Gasevic, N. Kaviani, and M. Hatala, "On Metamodeling in Megamodels," in *Model Driven Engineering Languages and Systems*, ser. Lecture Notes in Computer Science, vol. 4735. Springer, 2007, pp. 91–105.

[9] L. M. Camarinha-Matos and H. Afsarmanesh, "Collaborative networks: Value creation in knowledge society," in *PROLAMAT 2006, Knowledge Enterprise: Intelligent Strategies in Product Design, Manufacturing, and Management*, vol. 207, 2006, pp. 26–40.

[10] T. Ruokolainen, "Modelling framework for interoperability management in collaborative computing environments," Jun. 2009, Licentiate Thesis.

[11] J. Metso and L. Kutvonen, "Managing Virtual Organizations with Contracts," in *Workshop on Contract Architectures and Languages (CoALa2005)*, Enschede, The Netherlands, Sep. 2005.

[12] J. Lee, K. Siau, and S. Hong, "Enterprise integration with erp and eai," *Commun. ACM*, vol. 46, no. 2, pp. 54–60, 2003.

[13] N. Erasala, D. C. Yen, and T. M. Rajkumar, "Enterprise application integration in the electronic commerce world," *Comput. Stand. Interfaces*, vol. 25, no. 2, pp. 69–82, 2003.

[14] A. Tsalgatidou and T. Pilioura, "An overview of standards and related technology in web services," *Distrib. Parallel Databases*, vol. 12, no. 2-3, pp. 135–162, 2002.

[15] S. Damodaran, "B2B integration over the Internet with XML: RosettaNet successes and challenges," in *13th international World Wide Web conference on Alternate track papers & posters*. New York, NY, USA: ACM, 2004, pp. 188–195.

[16] L. M. Camarinha-Matos and H. Afsarmanesh, "Elements of a base VE infrastructure," *Comput. Ind.*, vol. 51, no. 2, pp. 139–163, 2003.

[17] T. Ruokolainen and L. Kutvonen, "Service Typing in Collaborative Systems," in *Enterprise Interoperability: New Challenges and Approaches*, G. Doumeingts, J. Müller, G. Morel, and B. Vallespir, Eds. Springer, Apr. 2007, pp. 343–354.

[18] B. Henderson-Sellers, "Method engineering for OO systems development," *Commun. ACM*, vol. 46, no. 10, pp. 73–78, 2003.

[19] M. Barbero, F. Jouault, J. Gray, and J. Bézivin, "A Practical Approach to Model Extension," in *ECMDA-FA 2007*, ser. Lecture Notes in Computer Science, vol. 4530. Springer-Verlag, 2007, pp. 32–42.

[20] *Meta Object Facility (MOF) Core Specification*, 2nd ed., Object Management Group, Jan. 2006, oMG Available Specification – formal/06-01-01.

[21] "Wikipedia: Linnaean taxonomy," http://en.wikipedia.org/wiki/Linnaean_taxonomy, May 2009.

[22] J.-M. Favre, "Foundations of Model (Driven) (Reverse) Engineering : Models - Episode I: Stories of The Fidus Papyrus and of The Solarus," in *Language Engineering for Model-Driven Software Development*, 2004.

[23] J. Favre, "Towards a Basic Theory to Model Model Driven Engineering," in *3rd Workshop in Software Model Engineering in conjunction with UML2004, WiSME*, 2004.

[24] I. Kurtev, J. Bézivin, and M. Aksit, "Technological Spaces: An Initial Appraisal," 2002, http://www.scientificcommons.org/27172017 (05.06.2008).

[25] K. Honda, V. T. Vasconcelos, and M. Kubo, "Language primitives and type discipline for structured communication-based programming," in *Proceedings of the 7th European Symposium on Programming*. Springer-Verlag, 1998, pp. 122–138.

[26] A. Vallecillo, V. T. Vasconcelos, and A. Ravara, "Typing the Behavior of Objects and Components using Session Types," *Electronic Notes in Theoretical Computer Science*, vol. 68, no. 3, 2003, presented at FOCLASA'02.

[27] "Eclipse Modeling Framework website," http://www.eclipse.org/modeling/emf/, 2008.

[28] *ISO/IEC 10746-1: Information technology – Open Distributed Processing – Reference model: Overview*, ISO/IEC JTC1/SC7, Dec. 1998.

[29] F. S. Parreiras, S. Staab, and A. Winter, "On marrying ontological and metamodeling technical spaces," in *ESEC-FSE '07: Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*. New York, NY, USA: ACM, 2007, pp. 439–448.

[30] D. A. Quartel, M. W. Steen, S. Pokraev, and M. J. Sinderen, "COSMO: A conceptual framework for service modelling and refinement," *Information Systems Frontiers*, vol. 9, no. 2-3, pp. 225–244, 2007.

[31] G. Benguaria, X. Larrucea, B. Elvesaeter, T. Neple, A. Beardsmore, and M. Friess, "A Platform Independent Model for Service Oriented Architectures," in *Enterprise Interoperability: New Challenges and Approaches*, G. Doumeingts, J. MÃijller, G. Morel, and B. Vallespir, Eds. Springer, Apr. 2007, pp. 23–32.

[32] H. Bruneliére, F. Allilaire, J. Bézivin, and F. Jouault, "Global model management in eclipse gmt/am3," in *Eclipse Technology eXchange workshop (eTX) at the ECOOP 2006 Conference*, 2006.

[33] X. Blanc, M.-P. Gervais, and P. Sriplakich, "Model Bus: Towards the Interoperability of Modelling Tools," in *Model Driven Architecture*, ser. Lecture Notes in Computer Science, vol. 3599. Springer, 2005, pp. 17–32.