Ontology for Federated Management of Business Networks

Toni Ruokolainen, Lea Kutvonen, and Janne Metso

Department of Computer Science, University of Helsinki, Finland
{Toni.Ruokolainen|Lea.Kutvonen|Janne.Metso}@cs.Helsinki.FI

Abstract. The present trend towards inter-enterprise computing relies on model-driven engineering solutions, but further progress calls for development of theoretically solid management infrastructure with a federated architecture approach. For this purpose, federated architectures and their target concepts have to be elaborated and formalized. This paper introduces the ontologies that are fundamental for the Pilarcos middleware and express the semantic background of the federated architecture and the issues to be addressed. The presented ontologies define the metamodels needed for representing the target concepts and semantic restrictions between them in the proposed middleware.

1 Introduction

The present trend towards inter-enterprise computing relies on model-driven engineering solutions, but further progress calls for development of theoretically solid management infrastructure with a federated architecture approach. Examples of this kind of trend can be found in the new FP7 work program [5] and for example NESSI consortia [16] goals. The Pilarcos middleware [10, 12] provides federated support for inter-enterprise collaboration management by utilising a commonly available knowledge base that includes information about available services and collaboration structures understandable for potential peers. As the knowledge base is formed by multiple autonomous knowledge providers, there is a definite need for addressing issues of interoperability, evolution, trust, and availability.

The interoperability issues focus around the semantics of the services and collaborations, and therefore, our goal is to provide a middleware that provides service type safety, i.e., only services with technical, semantic, and pragmatic interoperability are joined into collaborations. To provide this, we need to have well-defined semantics for service declarations, verification of the fulfillment of these, and verified relationships between service definitions. The evolution issues address the need of enterprises to develop their services and of the business to develop the collaboration forms commonly accepted. Therefore, we provide a way to introduce new models to the system in a consistency-preserving way. The trust issues and the availability issues are not discussed in this paper.

In the Pilarcos framework, metainformation is used to manage the collaborations (using contracts) and their members (represented by service offers). The

semantics of these metainformation items is restricted by further models, such as business network models [12] and service types [22]. As this metainformation and model knowledge is used both a) at the service production time and at the service use time, and b) by the service providers, by the service users, and by the collaboration management infrastructure services, it is necessary to draw that knowledge into the common middleware. Therefore, the Pilarcos middleware provides three kinds of repositories: business network model repositories, service type repositories, and service offer repositories. The role of these repositories has been described elsewhere [9].

In addition, we have envisioned a tool-chain to support model-oriented development [12, 22] of service-oriented systems, where models and other metainformation are published in the repositories as part of a software-engineering process. First, service type descriptions are required: service types can be used as models from which a MDA-based service implementation can be generated. More over, the service type allows services actually provided to be mapped to roles in business networks. Second, the business network models (BNM) can be verified and published: the BNMs express the structure for collaborations in terms of roles, interactions, and restricting policies. The BNMs provide the structure for the contracts used for operational time management of the collaboration. Third, the service implementations developed can be provided as business services by some enterprise: the enterprise publishes the service offer to a corresponding repository as it wishes to become a member in some collaborations. Finally, the communities are formed semi-automatically as a partner initiates the establishment: a BNM is selected, populated by service offers using a populator service, the potential contract is tested for interoperability between member services, and finally pushed through a refining negotiation cycle between the suggested members.

The metainformation in the repositories is structured and semantically restricted according to an ontology that is the main contribution of this paper. This ontology addresses two issues. First, it expresses the semantic background of the federated architecture: how collaborations are structured and knowledge about interoperability obtained. Second, the ontology presented defines the target concepts required for capturing the collaborations, their contracts, behaviour and progress. The paper is structured as follows. Section 2 introduces the four-level modelling hierarchy for defining federated service communities. Sections 3 and 4 go further and describe the metamodels that restrict the form and relationships between the ontology elements. Section 5 discusses the consistency criteria that need to be maintained in the dynamic ontology that is built. We conclude by a short summary of the potential created by this contribution.

2 Ontological approach to knowledge

In the Pilarcos framework [11, 12] collaborative systems and especially federated service communities are described using metamodels. The resulting Pilarcos ontology for federated service communities comprises four layers of metamodels in

a hierarchy that is illustrated in Figure 1. The highest layer consists of the MOF constructs [17] applicable for defining actual metamodels.

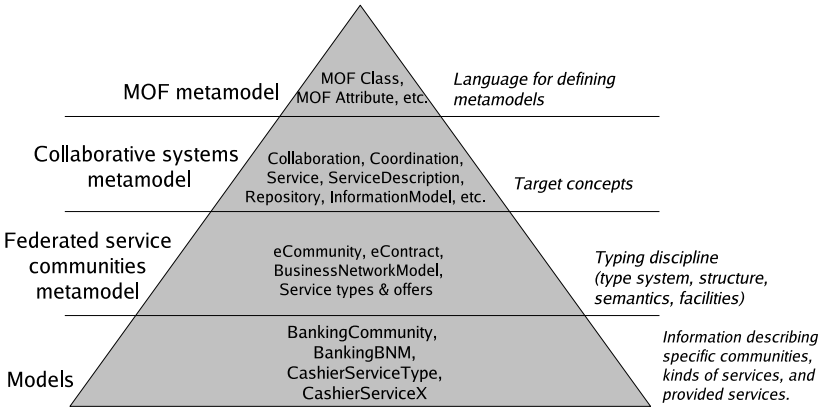

Fig. 1. A four-level modelling hierarchy for describing federated service communities.

The *collaborative systems metamodel* provides the target concepts for specific kinds of collaborative systems. The set of concepts include notions for collaboration, interaction, service-oriented computing etc. In addition, the metamodel provides a set of generic consistency rules that must be respected, and given domain specific interpretations by the specializations of this metamodel.

The concepts defined at the *federated service communities metamodel* describe such notions as service types [22], business network models, eCommunities [12], and electronic contracts [14]. In addition for defining modelling constructs for creating the knowledge required, the corresponding metamodel defines an abstract platform. The abstract platform definition characterizes the essential infrastructure services for establishing federated service communities. The metamodel has associated consistency rules and correspondences to be maintained between knowledge elements, such as service types and services offers, in form of a service typing discipline [22].

Finally, the domain models are represented at the lowest level of abstraction. The set of knowledge elements at this level include information describing for example business communities, definitions for different kinds of business services, service descriptions adhering to the service definitions, and electronic contracts.

3 Describing collaborative systems

Collaboration is an act of joint operation between a group of entities (such as individuals, organizations, or enterprises) that share a common interest or objective. Collaboration transcends the notion of co-operation [3] which is found

typically in scenarios that contain subcontracting or resource sharing, such as in the context of supply-chains or Grid-computing. The family of collaborative systems include for example multi-agent systems and federated service communities. In collaboration, the work of an individual constitutes an indispensable effort towards the objective. Sometimes the objective cannot even be achieved without the corresponding form of collaboration. Consequently, collaboration may be necessary for the existence of the entities. The Pilarcos framework takes the concept of collaboration as the basic tenet for achieving loosely coupled business networking in inter-enterprise environments. The corresponding metamodel for collaborative software systems comprises characterizations for the concepts of collaboration, interaction and service-oriented computing. These metamodels are elaborated below.

3.1 Collaboration metamodel

As illustrated in Figure 2, a *Collaboration* comprises a set of roles and coordination facilities to achieve some shared interest or objective which is represented as a set of more specific goals. A *Role* identifies the set of responsibilities and duties related to the realization of the collaboration goals that can be assigned to an entity taking part in the collaboration. Each *Role* has a unique identifier within a collaboration such that entities can be referred to indirectly through the role names. The concept of *Role* is also used in other metamodels for describing types of actors, their responsibilities and expected contributions towards the collaboration.

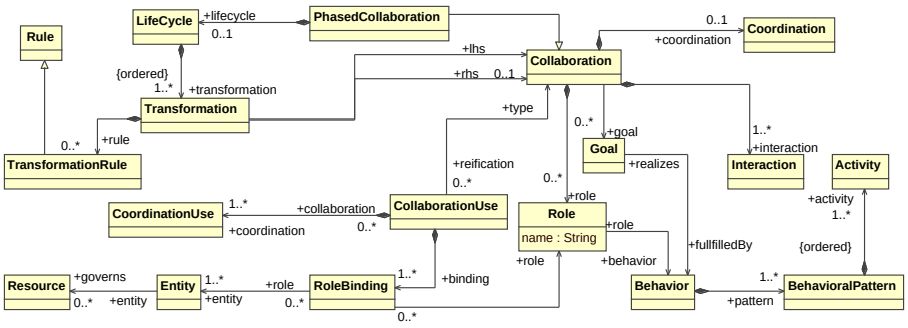

Fig. 2. Metamodel for collaboration.

A role is primarily a declaration of the behaviour expected from an entity. *Behaviour* in this context refers to the externally observable manifestation of the internal activities or processes of an entity or an individual. A role's behaviour consists of a set of *behavioural patterns* which are explicit descriptions of the activities belonging to the role's repertoire of behaviour. For example, a

"Cashier"-role's behaviour may comprise different behavioural patterns depending on the method of payment (e.g. cash or credit card) preferred by the client.

Collaboration is a form of coordinated co-operation where coordination is used for managing the dependencies among activities [13] of the collaborating entities. *Coordination* provides a fabric between the activities of entities to join the individual behavioural patterns forming a joint behaviour required by the collaboration. The metamodel for coordination is elaborated in Section 3.2. A *Goal* is an explicit declaration of some duty that contributes to the objective of the collaboration. The objective of a collaboration can then be represented as a set of goals. Goals can be allocated explicitly to role behaviours to monitor the fulfillment of goals during collaboration operation.

When a group of entities decide to collaborate, they form a loosely coupled community where each of the entities are given a certain role in the collaboration. This role binding is described in the collaboration metamodel using the *CollaborationUse* [1] concept, which comprises a set of role bindings. The *RoleBinding* concept associates the responsibilities and behaviour prescribed in a certain role with a given entity. An entity fulfills the duties of a role by utilizing the resources it governs.

We identify an important category of collaboration as a subconcept of the *Collaboration*, namely phased collaborations. The *PhasedCollaboration* concept is described using a life-cycle which is comprised of a sequence of distinct phases; each phase constitutes a collaboration in itself. For example, in the context of virtual enterprises the phases of a collaboration consist of virtual enterprise formation, operation and dissolution (see for example [15]), each phase having its specific roles and entities. In the collaboration metamodel illustrated in Figure 2, the life-cycle is represented as the concept *LifeCycle* which comprises an ordered set of transformations between collaboration phases. Each *Transformation* comprises an optional set of transformation rules which define the correspondences between roles in subsequent collaboration phases. Source and destination collaborations for the transformation are represented with *lhs* (for left-hand side) and *rhs* (for right-hand side) associations, respectively. The right-hand side of a collaboration can be empty; this is needed for representing single-phased collaboration. Transformations without explicit transformation rules represent collaborations where the life-cycles consist of distinct phases that are related implicitly. A typical example of this kind of collaboration is a supply-chain or supply network scenario where the phases of the collaboration are related by the physical material flow. In this case the transformations are conditioned by the completion of phases as well as relocation of goods.

3.2 Interaction metamodel

Interoperation between collaborating participants is established via interactions. As collaborative systems consist of autonomous entities, the interactions are

[1] The naming convention with pattern *XXX* and *XXXUse* is taken from the UML 2.0 standardization. It manifests a correspondence between a type and its instance.

considered as subjective acts of behaviour conducted by an entity towards a set of other entities. Interactions are defined by prescribing the *Roles* involved in the interaction and by declaring the subjective model of behaviour comprising the actual inter-activities. The corresponding metamodel for interaction is illustrated in Figure 3.

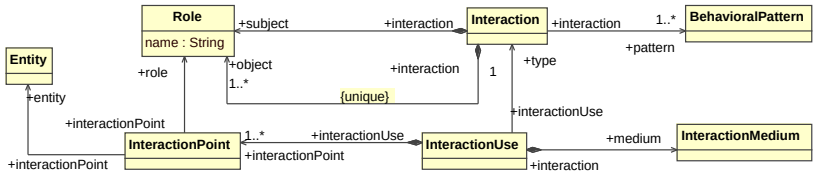

Fig. 3. Metamodel for interaction.

The *InteractionUse* concept is used to bind entities into the roles defined in an *Interaction* and defines the type of medium for realizing the interaction. The *InteractionPoint* defines and identifies an entity of interaction and binds it to a specific interaction role. The information provided by an interaction point enables directly or indirectly (via resolution mechanisms or mediators) the ability to perform activities corresponding to the behaviour defined in the provided interaction with respect to the entity identified by the interaction point.

It is important to notice that the set of *BehavioralPattern* elements referenced in an interaction must exists in the definition of the subject *Role*. However, the objects of interactions do not have to directly support the kinds of behaviour requested as long as they can be considered *compatible* in some way. This is regarded as an important aspect of interoperation in collaborative systems and necessitates the existence of validation procedures for behavioural compatibility. Compatibility of behaviours is elaborated in Section 5.

There are two important categories of interactions that require their own metamodels, namely coordination and communication. Coordination is used for regulating and directing the operation of a collaboration in such a way that the dependencies between activities become completed. Coordination functions can be either centralised or distributed (location of control), static or dynamic with respect to permanence of coordination medium and topology, endogenous (implicit coordination) or exogenous (explicit coordination), and based on a different technological mechanisms such as messaging, event notifications, or distributed tuple spaces. The metamodel that is applied for modeling coordination in the Pilarcos framework is given in Figure 4.

Communication is a kind of interaction which involves exchange of information through an explicit communication channel. Consequently, the behaviour applied for communication consists of activities for receiving and sending information. The corresponding metamodel is not illustrated in this paper; briefly, it

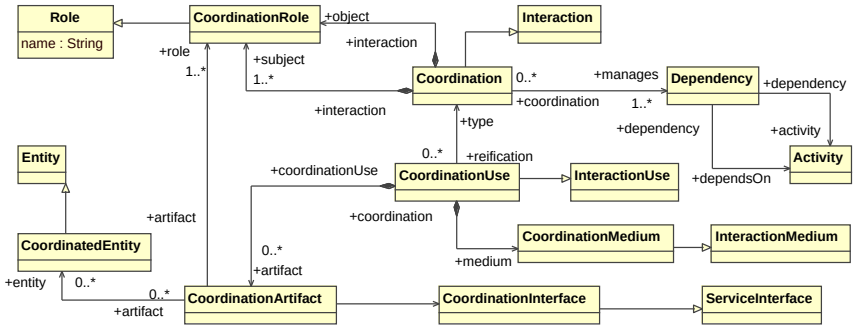

Fig. 4. Metamodel for coordination.

is essentially a specialization of the interaction concepts, such as roles, behaviour and interaction media for communication purposes.

3.3 Service-oriented computing metamodel

In the Pilarcos framework, service-oriented computing (SOC) [20] is considered as an essential mechanism for attaining loosely coupled collaborations. The service-oriented computing paradigm consists basically of four conceptual elements: services, service descriptions, service-oriented architectures (SOA), and service composition. Corresponding metamodels are needed for making the elements of service-oriented computing and their inter-relationships explicit. For this purpose, the Pilarcos framework defines a service-oriented computing metamodel that comprises metamodels for 1) service declarations, 2) service-oriented architectures, and 3) service collaborations. These metamodels are elaborated below.

The Pilarcos framework makes a distinction between two kinds of service declarations, namely service definitions and service descriptions [21] as defined by the the service declarations metamodel illustrated in Figure 5. A *ServiceDeclaration* has two sub-concepts, namely *ServiceDefinition* and *ServiceDescription*. Service definitions are formal specifications of services capabilities; their primary purpose is to introduce means for attaining service interoperability and to categorize available services. Service descriptions are more technical declarations used for advertising service properties and for establishing communication paths between service endpoints. For example in the OWL-S [19] the *ServiceProfile* and *ServiceModel* can be considered as service definitions whereas the *ServiceGrounding* provides a description for a concrete service.

A conformance relationship (*conformsTo*) between instances of *ServiceDescription* and *ServiceDefinition* concepts is assumed to hold. Furthermore, two additional roles are prescribed: service designers that provide service definitions, and service providers that advertise their services using the service descriptions.

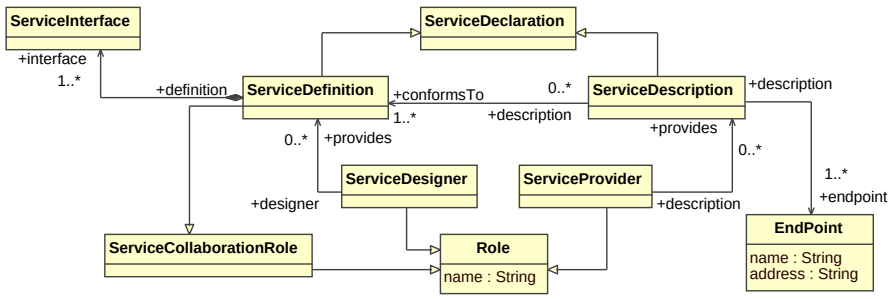

Fig. 5. Metamodel for service declarations.

Service-Oriented Architecture (SOA) is an architectural style for establishing loosely coupled distributed systems. The corresponding metamodel defines three roles, namely service provider, service consumer, and a service broker, and their inter-relationships. In addition, the metamodel relates services, corresponding metainformation elements (service declarations), and metainformation repositories. The resulting metamodel for service-oriented architectures is illustrated in Figure 6.

In the SOA metamodel a concrete service represented by the *Service* concept is a kind of an *Entity*. This relationship is needed such that collaborations consisting of different services as co-operating entities can be declared. Each service is attached with an endpoint which identifies the point of interaction. An *EndPoint* has a name and an address; the former can be used for example to maintain the binding between a service and it's client during service migration while the latter provides the actual "physical" handle to be used for communicating with the service. A service endpoint conforms to an *Interface* given in a corresponding service definition.

A *Repository* is a shared database of information which is used to store common models and information contents of a certain (engineering) domain. In the SOA metamodel, a *Repository* constitutes a unique namespace which contains a collection of *RepositoryItem* elements (especially service declarations). A repository item may contain information for example about document structures, interface definitions or ontologies Each *RepositoryItem* has a unique name within its repository's namespace. A repository maintains an *information model* which specifies semantics and structure for the repository's functionality and information contents. The *InformationModel* is defined using invariant, static, and dynamic schemata similarly to ODP information viewpoint specification [7]. Invariant schemata are defined as a set of predicates which must hold for knowledge elements at any point of time during the whole life-time of the corresponding information [7]. The invariant schemata describe immutable structures and rules, such as ontologies and typing disciplines that the knowledge elements are subject to. Static schemata describe the state of knowledge maintained in repos-

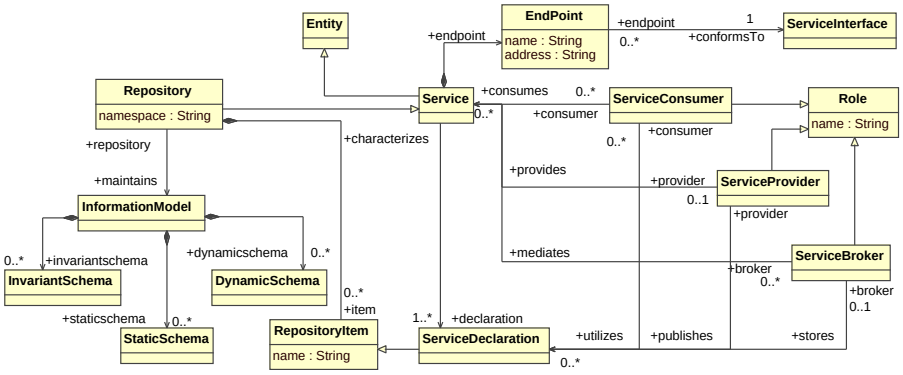

Fig. 6. Service-Oriented Architecture (SOA) metamodel.

itories at some point of time and are subject to constraints set by the invariant schemata [7]. Finally, dynamic schemata represent allowable knowledge transformations. Typically static schemata are used in the description of dynamic schemata, describing the state of knowledge before and after an transformation. Also dynamic schemata are subject to the constraints declared by the invariant schemata.

Finally, the SOA metamodel defines three roles. The *ServiceConsumer*-role represents an entity that acts as the "client" in a service provision scenario. A consumer utilises the service brokering infrastructure for finding appropriate services and service providers. The service brokering infrastructure is maintained by an entity in the *ServiceBroker* role. Such an entity stores service declarations in an appropriate repository which provides the functionality for querying, matching and locating the services. Consequently, a service broker mediates the services between providers and consumers. The *ServiceProvider* role states that corresponding entities provide services and publish their declarations. The inter-relationships between these roles can be elaborated using the collaboration metamodel.

In the context of service-oriented computing two different kinds of service collaborations can be identified: service composition and service choreographies. Service collaborations are defined between service definitions using the collaboration metamodel described in Section 3.1. To serve this purpose, each *ServiceDefinition* is also a subconcept of *ServiceCollaborationRole* as illustrated in Figure 5.

4 Describing federated service communities

In this Section we briefly introduce a specialization of the collaborative system metamodel, namely the one for federated service communities. A federated service community [12, 22] is a business collaboration between organizations that

export their business functionality as business services. The approach for establishing interoperability is a federated one and builds on foundations such as multilateral community population [10], interoperability validation based on shared metamodels [22], and eContracting [14].

Collaborations between business partners in the Pilarcos framework are called *eCommunities*. The corresponding metamodel is illustrated in Figure 7. The entities engaging in an eCommunity are called *Partners* and they represent organizations providing business services to the eCommunity as a legal entity. An *Organization* actually has a metamodel itself, but due to lack of space it cannot be described here. Nevertheless, the metamodel for organizations relates the business services provided by the *Partners* with the autonomic intentions of organizations, such as business rules and organizational policies, and also relate the business services to local resources.

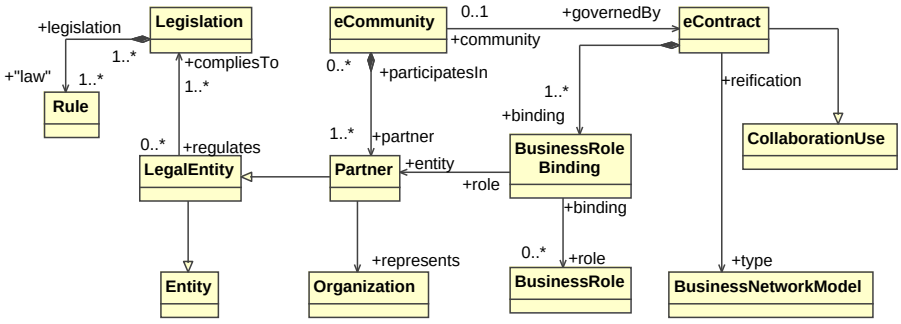

Fig. 7. Metamodel for eCommunities.

An *eCommunity* is not a direct subconcept of the collaboration concepts described in Section 3.1 but is instead regulated by a specialization of the *CollaborationUse*, namely an *eContract* which is the outcome of a successful population process [10]. The structure of an *eContract* is given by a *BusinessNetworkModel*. A simplified illustration of the Business Network Model (BNM) is given in Figure 8. A BNM defines a phased collaboration consisting of one or more *epochs* in the life-cycle. Each epoch is a phase of collaboration whose properties and structure are defined by a *BusinessNetworkArchitecture*. The architecture defines a configuration of *BusinessRoles* and *BusinessRoleConnectors*. Business roles are compositions of *BusinessRolePorts* which again are formalized by corresponding *ServiceType* concepts. Service type [22] provides the service definitions prescribed by the service declarations metamodel.

The Pilarcos approach aligns with the exogenous coordination model [1]. This is manifested in Figure 8 by the fact that it is actually the connectors that are responsible for coordinating the business network operation (they are

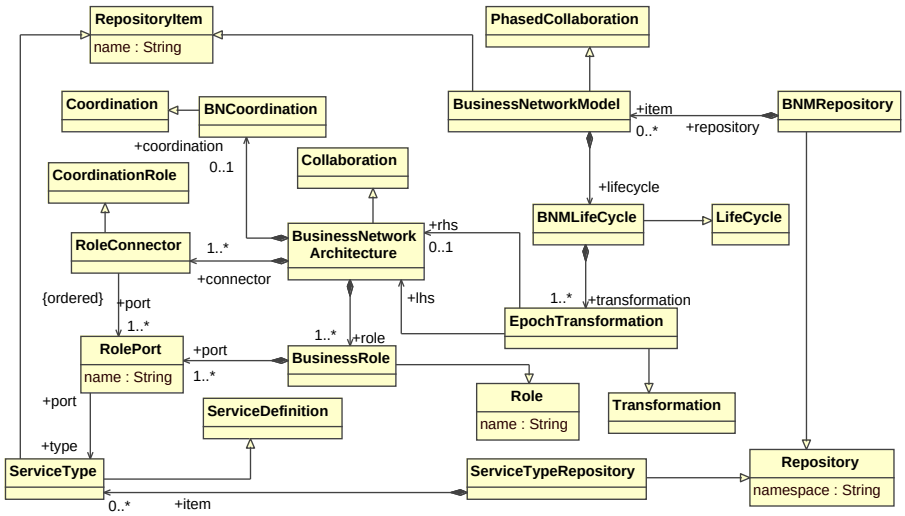

Fig. 8. A simplified top-level illustration of Business Network Models.

bound with *CoordinationRoles*), not the business roles. Furthermore, Figure 8 accentuates the existence of two Pilarcos metainformation repositories: a BNM repository [12] and a service type repository [22].

5 Introducing consistency criteria

Metamodels as represented in the previous sections are useful as such for describing and identifying the concepts and constructs needed for realizing collaborative systems. However, to establish a true ontology, the dependencies and consistency criteria between the different concepts have to be identified and more over, formalized. In the following, we briefly discuss some of the most obvious correspondences found in the metamodels.

One of the most fundamental correspondence relationships to fulfill is between concepts related by naming patterns *XXX* and *XXXUse*, such as *Collaboration* and *CollaborationUse*. The naming convention manifests a correspondence between specifications and their instances. The *type*-association from concept *XXX* to concept *XXXUse* implies that the properties given in the specification *XXX* are transfered through some reification process to its instance *XXXUse*. Especially this means that the responsibilities and behaviours specified in roles are transfered to the actual entities bound to them. The reification and the semantics of the correspondence relationship are specific for a kind of a collaborative system. For example in the case of federated service communities, the reification comprises a process with population and negotiation phases [10,14] to

refine a *BusinessNetworkModel* into an *eContract* (see Section 4). The correctness of this reification is provided by the semantics declared for the concepts in the federated service communities metamodel, information models maintained by the repositories and runtime monitoring of the community operation.

A collaboration's *Goal* is fulfilled by some role *Behaviour*. This relationship binds the activities taken by the entities directly to the objectives of the corresponding collaboration, thus giving mechanisms for monitoring the advance of the collaboration above the operational level. However, this correspondence relationship is not formalizable in the general case, but requires domain specific knowledge about best practices of the corresponding domain. In some cases, the correspondence between goals and behaviour of entities is more evident and concrete. For example in multi-agent systems high-level declarative goals can be used to induce the corresponding behavioural patterns for the participating entities using e.g. goal-based planning [23]. In federated service communities, the goals are represented as business rules and policies. When these rules are formalized using for example deontic logic, the conformance relationship can be made concrete and validated with methods such as model checking (see e.g. [18]).

The service declarations have a conformance relationship between *ServiceDescription* and *ServiceDefinition* concepts. The conformance criterion given by a definition must be met by the service description used for advertising such a service. In the context of the Pilarcos framework, the correspondences between service types and service offers are formalized using the session typing discipline [6, 24]. Session typing also provides means for categorization of services and interoperability validation through the notions of behavioural subtyping and compatibility [24].

The interaction metamodel describes a subjective model of interaction where behaviour of an interaction is solely determined by its subject. To achieve an interoperable interaction between two entities, their subjective views on the interaction behaviour have to be compatible. The notion of behavioural compatibility defined by the session typing discipline [24] is used for this purpose in the Pilarcos framework.

The correspondences between coordination and the dependencies it manages are not in general formalizable. Instead *Coordination* specifications represent "best practises" of a certain domain which are known to complete the corresponding *Dependencies*. Validation of such correspondences are provided by human actors, either *a priori* or *a posteriori*. The coordination metamodel describes coordination as a kind of interaction that affects the overall behaviour of a collaboration. Consequently, the behaviour declared by coordination may conflict with the behaviour of the collaboration roles. While coordination and role behaviour may by themselves be consistent, their collective behaviour may lead to inconsistencies, such as deadlocks. For this reason, it is important to formalize the coordination model and its relationships to the coordinated entities; such work has been done for example in [25].

6 Conclusion

This paper has introduced the metamodels used in the Pilarcos framework [10,12] for the dynamic ontology management system where interoperability related knowledge about business services and possible collaboration types can be managed. The metamodels propose a constructive approach to service-oriented computing suitable for establishing open service markets and service-oriented software engineering. This constructiveness emerges from the utilization of service types as the elementary concept which provides the typing discipline required for interoperable service delivery, consistency criteria for service offers and implementations, and a modular design artifact to be used in collaboration designs (BNMs). Promoting the separation of communication and coordination concerns already at the conceptual level, the metamodels in effect advocate a development model which cleanly separates "in-the-small" from the "in-the-large" [4].

The Pilarcos approach is based on a strong idea of preserving autonomy among community participants. This necessitates a federated approach, where the exact model of collaboration is constructed on-demand and checked dynamically across partners. For this purpose, a very detailed ontology defining the target concepts and their inter-relationships is needed. To maintain the consistency of knowledge needed during operation of a federated system, repositories use the semantic consistency criteria attached to the metamodels to restrict the publication of new models and to validate their correctness.

In comparison to related work, such as PIM4SOA [2] or WS-CDL [8], the constructiveness and federated approach are the most evident differences. Both PIM4SOA and WS-CDL are top-down approaches based on semantic unification where the capabilities of individual services are predetermined by the collaboration models. In the approach represented in this paper, individual services can exist independently of any collaboration forms. However, this does not rule out generative MDA-like approaches where services and their declarations are derived from collaboration descriptions.

References

1. F. Arbab. What do you mean, coordination? Appeared in the Bulletin of the Dutch Association for Theoretical Computer Science (NVTI), Mar. 1998.
2. G. Benguaria, X. Larrucea, B. Elvesaeter, T. Neple, A. Beardsmore, and M. Friess. A Platform Independent Model for Service Oriented Architectures. In G. Doumeingts, J. Müller, G. Morel, and B. Vallespir, editors, *Enterprise Interoperability: New Challenges and Approaches*. Springer, Apr. 2007.
3. L. M. Camarinha-Matos and H. Afsarmanesh. Collaborative networks: Value creation in knowledge society. In *Knowledge Enterprise: Intelligent Strategies in Product Design, Manufacturing, and Management*, volume 207, pages 26–40, 2006.
4. F. DeRemer and H. Kron. Programming-in-the large versus programming-in-the-small. In *Proceedings of the international conference on Reliable software*, pages 114–121, New York, NY, USA, 1975. ACM Press.
5. FP7 – European Commission 7th Framework Program. `http://ec.europa.eu/research/fp7`, Apr. 2006.

6. 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*, pages 122–138. Springer-Verlag, 1998.
7. ISO/IEC JTC1. *Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing. Reference Model of Open Distributed Processing. Part 1: Overview*, 1996. IS10746-1.
8. N. Kavantzas, D. Burdett, G. Ritzinger, T. Fletcher, Y. Lafon, and C. Barreto. *Web Services Choreography Description language*, Nov. 2005. W3C Candidate Recommendation.
9. L. Kutvonen. Building B2B interoperability middleware – knowledge management issues. In *I-ESA'07*, Mar. 2007.
10. L. Kutvonen, J. Metso, and S. Ruohomaa. From trading to eCommunity population: Responding to social and contractual challenges. In *EDOC 2006*. IEEE.
11. L. Kutvonen, J. Metso, and T. Ruokolainen. Inter-enterprise collaboration management in dynamic business networks. In *OTM Confederated International Conferences, CoopIS, DOA, and ODBASE*, volume 3760 of *Lecture Notes in Computer Science*, Nov. 2005.
12. L. Kutvonen, T. Ruokolainen, and J. Metso. Interoperability middleware for federated business services in web-Pilarcos. *International Journal in Enterprise Information Systems*, 3(1):1–21, Jan. 2007.
13. T. W. Malone and K. Crowston. The interdisciplinary study of coordination. *ACM Comput. Surv.*, 26(1):87–119, 1994.
14. J. Metso and L. Kutvonen. Managing Virtual Organizations with Contracts. In *Workshop on Contract Architectures and Languages (CoALa2005)*, Enschede, The Netherlands, Sept. 2005.
15. N. Nayak, K. Bhaskaran, and R. Das. Virtual enterprises: building blocks for dynamic e-business. In *Proceedings of the workshop on Information technology for virtual enterprises*, pages 80–87. IEEE Computer Society, 2001.
16. NESSI. Strategic Research Agenda, Feb. 2006. Public Draft 1.
17. Object Management Group. *Meta Object Facility (MOF) Core Specification*, 2.0 edition, Jan. 2006. OMG Available Specification – formal/06-01-01.
18. N. Osman, D. Robertson, and C. Walton. Run-time model checking of interaction and deontic models for multi-agent systems. In *5th International joint conference on Autonomous Agents and Multiagent Systems (AAMAS'06)*, pages 238–240, New York, NY, USA, 2006. ACM Press.
19. OWL-S Coalition. *OWL-S 1.1 Release*, Nov. 2004.
20. M. P. Papazoglou and D. Georgakopoulos. Special issue on Service-Oriented Computing. *Commun. ACM*, 46(10), 2003.
21. T. Ruokolainen and L. Kutvonen. Service Definitions and Binding Processes: Delivering the Promise of Service-Oriented Computing. Manuscript, Nov. 2006.
22. T. Ruokolainen and L. Kutvonen. Service Typing in Collaborative Systems. In G. Doumeingts, J. Müller, G. Morel, and B. Vallespir, editors, *Enterprise Interoperability: New Challenges and Approaches*. Springer, Apr. 2007.
23. S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 2nd edition, 2002.
24. A. Vallecillo, V. T. Vasconcelos, and A. Ravara. Typing the Behavior of Objects and Components using Session Types. *Electronic Notes in Computer Science*, 68, 2003.
25. M. Viroli, G. Moro, and A. Omicini. On observation as a coordination paradigm: an ontology and a formal framework. In *ACM Symposium on Applied computing*, pages 166–175, New York, NY, USA, 2001. ACM Press.