

# AAMAS 2010 TORONTO



The 9th International Conference on  
Autonomous Agents and Multiagent Systems  
May 10-14, 2010  
Toronto, Canada

## Workshop 12

### Agent-based Technologies and applications for enterprise interOPERability

### ATOP 2010

Editors:

Wiebe van der Hoek

Gal A. Kaminka

Yves Lespérance

Michael Luck

Sandip Sen





# Proceedings of the 4th ATOP Workshop

---

## Agent-based Technologies and Applications for Enterprise Interoperability

---

**AAMAS Workshop, 10 May 2010**

**Editors:**

*Jörg P. Müller*

*Klaus Fischer*

*Renato Levy*



# Preface

Today's enterprises must adapt their business processes to work in open settings, such as online marketplaces and, more generally, the Web, where business relationships exhibit a high degree of dynamism. Moreover, open settings are characterized by the autonomy and heterogeneity of the enterprises. In such settings, interoperability of / between the information systems that support and automate business processes and applications is a key concern: how do we ensure that diverse enterprises can work together towards a mutually desirable end? Interoperability problems occur at different levels: at the business level (how organizations do business together, what needs to be described and how?), at the knowledge level (different formats, schemas, and ontologies), and at the infrastructure level (the underlying information and communication technologies and systems). Agents, Model-Driven Architecture (MDA), and Service-Oriented Architecture (SOA) are complementary approaches to addressing the enterprise interoperability problem.

Agent technologies provide a cross-cutting approach promising to enable intelligent and proactive automation, adaptive planning and execution, decentralized coordination, and semantic interoperability. Agents enable dynamic collaboration and orchestration in changing and unpredictable situations. MDA supports interoperability due to its promise of providing consistent models at different abstraction layers with well-defined mappings in between these layers and provides mechanisms that generate artifacts for different platforms. SOA tries to reach interoperability, focusing upon, but not restricted to, the information and communication technology (ICT) level. It provides late-binding interoperability between business process requirements and providers of service implementations which results in loose coupling among software entities representing business objects (processes, organizational units, etc.).

The workshop focuses on technologies that support interoperability in networked organizations, on successful applications of these technologies, and on lessons learned. The main goal is to stimulate a discussion on in how far agent technologies can support interoperability in this context and to compare current trends in the development of agent technologies with recent developments in service-oriented and model-driven system design with respect to their ability to solve interoperability problems. Regarding model-driven system design the presentation and discussion of metamodels of the underlying technologies like for example agent technologies and service-oriented architectures is especially of interest.

Jörg P. Müller  
Klaus Fischer  
Renato Levy

Toronto, May 2010

## Workshop Chairs

Jörg P. Müller	TU Clausthal, Germany
Klaus Fischer	DFKI, Germany
Renato Levy	Intelligent Automation Inc., USA

## Program Committee

Sahin Albayrak	TU Berlin, Germany
Bernhard Bauer	University Augsburg, Germany
Amit Chopra	North Carolina State University, USA
Maksims Fiosins	TU Clausthal, Germany
Dominic Greenwood	Whitestein Technologies, Switzerland
Axel Hahn	University Oldenburg, Germany
Christian Hahn	DFKI, Germany
Øystein Haugen	SINTEF, Norway
Sebastian Kämper	IWi, Germany
Stefan Kirn	Hohenheim University, Germany
Margaret Lyell	IAI, USA
Saber Mansour	Oslo Software, France
Eugenio Oliveira	University of Porto, Portugal
Herve Panetto	University Nancy, France
Omer Rana	Cardiff University, UK
Ralph Ronnquist	Intendico Pty. Ltd., Australia
Omar Shafiq	University of Calgary, Canada
Carles Sierra	IIIA, Spain
Ingo Timm	Goethe-University Frankfurt/Main, Germany
Jörg Ziemann	IWi, Germany
Ingo Zinnikus	DFKI, Germany

## Table of Contents

Agent-Supported Collaboration and Interoperability for Networked Enterprises . . . . .	1
<i>Klaus Fischer and Ingo Zinnikus</i>	
A Model-Driven Approach to Close the Gap between Business and Agent-Based Execution . . . . .	13
<i>Christian Hahn, Dmytro Panfilenko, and Klaus Fischer</i>	
Inter-organizational Interoperability through integration of Multiagent, Web Service, and Semantic Web Technologies . . . . .	25
<i>Paul Karaenke, Michael Schuele, András Micsik, Alexander Kipp</i>	
Ontology Matching across Domains . . . . .	37
<i>Renato Levy, Jakob Henriksson, Margaret Lyell, Xiong Liu, and Michael J. Mayhew</i>	
Agent Metamodel and Profile: Current Status and Perspectives . . . . .	53
<i>James Odell</i>	
Using ontologies to support decentral product development processes . . . .	55
<i>Patrick D. Stiefel, Christian Hausknecht and Jörg P. Müller</i>	
Decentralized Semantic Service Discovery in Preferential Attachment Networks . . . . .	71
<i>E. del Val, M. Rebollo, and V. Botti</i>	
Goal-Directed Approach for Process Specification and Service Composition in Customer Life Cycle Management . . . . .	83
<i>Kumari Wickramasinghe, Ian Thomas, Michael Georgeff, and Christian Guttman</i>	





# Agent-Supported Collaboration and Interoperability for Networked Enterprises

Klaus Fischer<sup>1</sup> and Ingo Zinnikus<sup>1</sup>

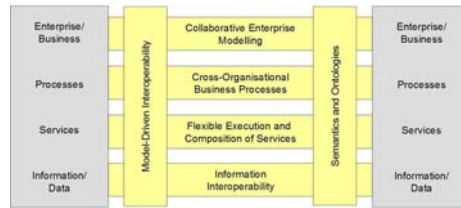
German Research Center for Artificial Intelligence (DFKI) GmbH, Saarbrücken,  
Germany

**Abstract.** The paper presents results from the COIN project which deals with improving solutions for enterprise interoperability and enterprise collaboration. We present the context of COIN in the European research area and explain the basic approach and system architecture COIN is aiming at. Special emphasis is put on how agents can support enterprise interoperability as well as enterprise collaboration services.

## 1 Background

In European research Enterprise Collaboration (EC) and Enterprise Interoperability (EI) have been the two major research catalysts for DG INFSO D4 *Networked Enterprise & Radio Frequency Identification (RFID)* and aggregated tens of projects and hundreds of researchers in their project cluster initiatives. Further information on EU projects and the cluster activities can be found at [1]. The COIN project [2] is one of the major current activities in this context. It is the conviction of the COIN project promoters that EC and EI are different concepts which cannot be merged and should not be confused. The former comes from a business perspective and identifies the process of enterprises to set-up and manage cross-enterprise win-win business relations in response to business opportunities. The latter originated by the ICT world and identifying a capability of enterprise software and applications to be integrated at the level of data, applications, processes, and models. EC and EI are, however, so interdependent, interconnected, and simultaneously present in every networked enterprise, that they can be considered as the two sides of the same coin.

EC was mainly fed by projects like ECOLEAD and DBE (both were so-called integrated projects (IP)) and by several smaller projects (small or medium-scale focused research projects (STREP)) like myTreasury, E4, Fluid-Win, Pabadis-Promise and Smart, aims at finding new paradigms for European enterprises (mainly SMEs) aggregation, synchronization, and co-operation in response to the more and more demanding and complex business opportunities coming from the global market. The key phases for achieving EC is the identification of an SME's core competencies, the focus of the SMEs internal resources with respect to these core competencies, and the valorization of these core competences by exposing them in the marketplace searching for other SMEs to collaborate with. Several collaboration models, ICT infrastructures, and support services have been



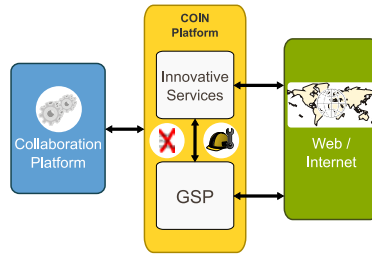
**Fig. 1.** ATHENA Interoperability Reference Architecture

so far successfully developed in such projects with the utmost concern to meet SMEs expectations and requirements. An important collaboration is the *closed* Breeding Environment more similar to a club or a gymnasium where SMEs are getting prepared to collaborate or the *open* Business Ecosystem where constellations of providers dynamically offer their services. Regarding ICT infrastructure the Enterprise Service Bus (ESB) for Enterprise Applications Integration (EAI), the Semantically Enabled Service Oriented Architectures (SESA), or the Digital Ecosystem peer-to-peer architecture can be mentioned.

However, in order to fully exploit its potentials mainly for European SMEs, EC research does not only aim at achieving important and relevant results from the scientific, organizational, business standpoint, but tries to also magnify the investments on resources in the ICT implementation of the key collaboration processes and cross-enterprise applications needed to make collaboration easier and profitable. For instance, the two projects ECOLEAD and DBE have achieved significant results in the field of IT infrastructure and IT support to collaboration management and performance management, but they could not address properly the problem of EAI and operational support to collaborative processes in different industries and application domains.

EI research was so far mainly fed by the ATHENA and TRUSTCOM (both EU IP projects), by the INTEROP NoE and by several STREPs like Abilities, Genesis, Fusion and Satine, started from an IT perspective of Enterprise Application and Software interoperability (inherited by the IDEAS road-mapping effort) and focused on enterprise modeling, architecture and platforms, ontologies and semantics as the basic pillars for EI. According to the ATHENA railroad for interoperability (see Fig. 1) this research stream proceeded very well in an analytical way to deeply investigate the various interoperability problems which affect European enterprises (and SMEs) and came out with a set of innovative ICT solutions (for Enterprise Models Interoperability, Cross-organizational Business Processes, Semantic Business Document Reconciliation, IT Service selection and composition and, finally, a model-driven approach for SOA).

EI solutions have already proved to be efficient and effective in the ICT and research community (e.g. with respect to software and service engineering, model-driven architectures, semantic interoperability) but have in their innovation potential privileged the big enterprises and regarding ICT more developed sectors and domains, while there is a tremendous need for EI efficient and effec-



**Fig. 2.** COIN Architecture

tive solutions in the SMEs environment and in some less ICT-developed sectors and domains like textile, food, or tourism. Moreover, it seems that EI solutions so far lack flexibility and adaptation to different business scenarios and collaboration forms like Supply Chains, Collaborative Networks and Business Ecosystems.

## 2 The COIN Project

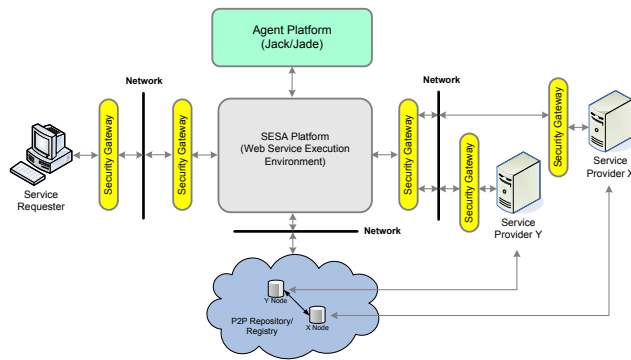
In its general approach to support EC and EI services in networked enterprises, the main objectives of COIN is to design and develop a pervasive, adaptive service platform to host baseline and innovative COIN services for EI and EC and make them available under innovative on-demand, utility-oriented business models to European enterprises (and SMEs in particular) for running their business in a secure, reliable and efficient way. Such a service platform, including business and knowledge interoperability models and tools, represents the innovative glue to fully exploit pre-existing and new services in the overall COIN mission.

Figure 2 displays the basic architecture of the COIN platform. On top of the basic infrastructure of the Internet and the Web, COIN develops a generic service platform (GSP). The WSMO/WSMX framework forms the core of the GSP. Therefore the main part of the services in the GSP are semantically described. Additionally, the COIN platform provides innovative services to support EC and EI. These innovative services can access Web services registered in the GSP or can be built directly with traditional Internet technology. Proof of concept tools to support end users in the execution of collaborative processes in networked enterprises are developed on top of the COIN platform.

The COIN Platform represents a service provisioning platform capable of supporting the SaaS-U<sup>1</sup> paradigm for EC/EI services. The overall idea is to provide a set of services through the platform that enable enterprises to create virtual organizations leveraging on collaborative services and exchanging information through integration services. All collaborative and interoperability services, according to the service agreements subscribed by the different stake

<sup>1</sup> SaaS-U: Software as a utility service where utility means that the service is so widely available that it becomes a commodity.

holders of the platform, are provided within a pay-per-use fashion or long term subscription to the platform. As presented in the more detailed view to the platform in Figure 3, the platform is composed of the Web Service Execution Environment (WSMX see [3]) as the core of the platform, the TrustCoM<sup>2</sup> security gateways which provide trust and security in a SOA implementation, a peer-to-peer repository/registry (coming from Digital Business Ecosystems project<sup>3</sup>) which provides fail-safe storage facilities, and an agent platform component for intelligent service compositions and negotiations.



**Fig. 3.** The COIN Service Provisioning Platform

## 2.1 Agents and Semantic SOAs

In COIN the agent subsystem is an important part of the platform supporting EC and EI services. A crucial feature of agents is the use of an explicit—and, in most cases publicly available—representation of entities in a collaboration (services, contracts, protocols) for reasoning and decision-making. WSDL (see [4]) is for example an explicit, publicly available (if stored in a UDDI registry) representation about technical aspects of a service, but it can hardly be used for reasoning or decision-making. Its main purpose is to support the (in most cases manual) integration of a service. The initial Web service technology stack allows exchange of messages between the parties by leveraging SOAP (see [5]). Furthermore, it allows for description of the technical interface for Web service consumption in the form of WSDL. These technologies form the foundation for an implementation of the Service Oriented Architecture (SOA) paradigm that represents the dominant approach in employing service orientation in delivery of business functions. However, these technologies only support Web service usage by manual inspection and integration, i.e. existing SOA solutions are proving

<sup>2</sup> <http://www.eu-trustcom.com>

<sup>3</sup> <http://www.digital-ecosystem.org>

difficult to scale without a proper degree of automation [6]. Tasks such as service discovery, selection and ranking, composition, mediation, negotiation, and execution of Web services require that involved services are completely characterized by their interfaces. However, in traditional Web Service implementations, the lack of information to express the meaning of data and of the vocabulary referenced by the interface as well as the lack of the formalization of the behavior is as a matter of fact prohibiting or at least hindering the automation of the envisioned tasks.

The Semantic Web initiative took up this challenge and developed a set of formalisms for semantic markup of services and other entities. In the vision of Semantic Web services (SWS) [7], creating semantic markup of Web services makes them machine understandable and use-apparent. Based on that, agent technologies can be developed that exploit this semantic markup to support automated Web service composition and interoperability. Semantic markup of Web services enables the automation of service discovery, execution, composition, and interoperation which drives the research and development of appropriate markup languages and agent technologies. SWS utilize ontologies as the underlying data model in order to support semantic interoperability between Web services and its clients and apply semantically enabled automated mechanisms that span the whole SWS life cycle. More generally, the introduction of semantics as an extension of SOA and creation of Semantically Enabled Service Oriented Architectures [8], provides for next generation of service-oriented computing based on machine processable semantics.

In order to provide support for SWS, two main approaches are envisioned. The former approach (bottom-up) relies on changing and extending existing models of Web services with the support for explicit semantics. This approach is supported by W3C and some researcher groups through the effort of SAWSDL (see [9]). SAWSDL extends the de-facto standard for service description WSDL by annotating elements in a service description and providing schema mappings for the transformation of data. The annotation of elements can be used for service discovery whereas the mappings can be used for invocation of a service. It is this latter feature which makes SAWSDL an interesting candidate for service integration, because, based on the well-established standard WSDL, a service provider can supply its partners not only with a syntactical description of her service interface via a WSDL file, but additionally with the information required for ad-hoc invocation of this service. The latter approach (top-down) utilizes existing Web service technologies as foundational system, layering the semantics support on top of it. This approach is taken by several groups in academia where the most distinguishing representatives are Web Services Modeling Ontology (WSMO) (see [10, 11]) and OWL-S (see [12]).

The Web Service Execution Environment (WSMX), presented in Figure 4, is an execution environment, which intends to realize the SWS life cycle. It is a platform characterized by strong component decoupling, goal-driven Web service usage and direct support for mediation facilities. WSMX is a reference implementation of WSMO. WSMO provides ontological specifications for the

core elements of SWS and acts as the comprehensive conceptual model which describes various aspects of SWS consisting of formal descriptions of ontologies, web services, goals and mediators.

The COIN platform, as an extension of WSMX, is embracing WSMO and the corresponding set of languages and specifications for describing ontologies, services, goals and mediators.

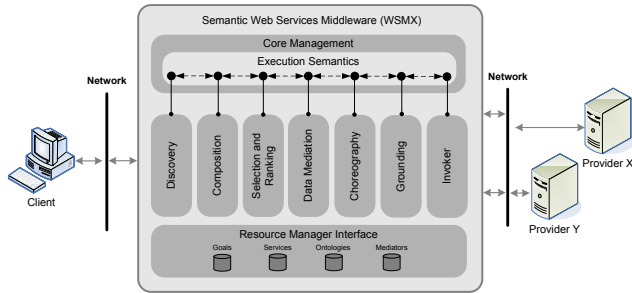


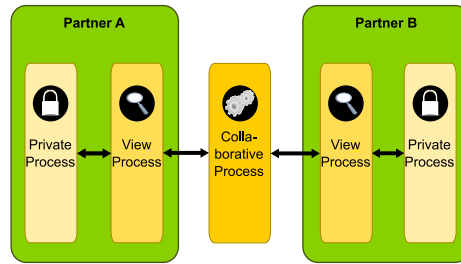
Fig. 4. Web Service Execution Environment

Agent technologies are used in the GSP for intelligent service matchmaking and for runtime negotiation of service selection and usage. Additionally agent technologies are used for service composition to build more complex services. In this manner the agents bridge the gap between innovative services and the GSP. In this sense agents can themselves become providers of innovative services. Negotiation services are one class of services that are investigated in this context.

### 3 Collaborative Processes

According to the main objectives of COIN as described in the last section, the design and execution of collaborative business processes is in the main focus of COIN. Figure 5 presents COIN's approach to collaborative business process design. The situation in the figure is simplified to the collaboration of two partners but of course COIN aims at the collaboration of any number of partners. In such a setting each partner has its private process from which so-called view processes are derived. On the other hand, a collaborative process is defined where the collaborative process can take advantage of the processes and services specified in the view processes. The approach seems to be centralized, however, this is only a conceptual description. It is actually not prescribed how the collaborative process is developed nor is it prescribed how the resulting collaborative process is eventually executed. However, to assume that the collaborative process is executed by any of the participating parties or by a third party is a straightforward solution once the collaborative process has been defined.

The view processes mediate between the private processes that are already available in the local environment of the individual partners and the collaborative



**Fig. 5.** COIN's Approach to Collaborative Process Design and Execution

process. In the design of collaborative processes already existing view processes might be used or new view processes might be derived by the requirements that are specified in the collaborative process. Two major concerns can be identified when it comes to the design and implementation of collaborative processes:

1. The integration of the collaborative workflow with EI services which support the interoperability between the collaborative process and the view processes.
2. The support of complex workflow patterns in the description and in the execution of collaborative workflows.

Both aspects can be understood as offline and runtime activities. In the offline case the user (most probably a system architect or a business process engineer) wants to have support in the design of the collaborative process. A major task is to find already existing services and process definitions. In the online case the decision on which concrete service endpoint should be called is in the focus. Agent technologies can provide innovative solutions for both areas in the online and the offline situation.

**Regarding 1.** the main advantage of agent technologies is their ability to deal with semantic descriptions and to reason about such descriptions. With this agents introduce more flexibility in binding services to the collaborative workflow (see [13] and [14] for detailed accounts on this topic).

**Regarding 2.** again two aspects are important:

1. Agent technologies are available that allow graphical modeling of complex interaction and service composition patterns.
2. Agents can be used to wrap complex workflow patterns into services which makes the overall collaborative process description less complex. Figure 6 shows how a complex workflow pattern (in this case parallel activities where the number of activities is known at design time) can be wrapped in a service that is called from a workflow which is described at a higher level of abstraction.

Complex interactions between the partners in collaborative environments imply a number of problems which have to be solved:

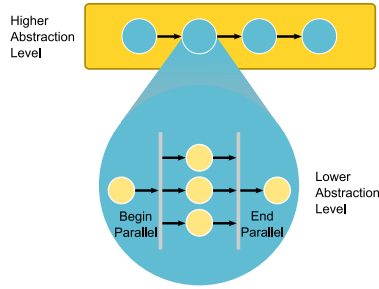
- changing the protocol and integration of a new partner should be possible in a rapid manner (scalability)
- the execution of the message exchange should be flexible, i.e. in case a partner is unavailable or busy, the protocol should nevertheless proceed
- the different partners (may) expect different atomic protocol steps (service granularity)
- the partners expect and/or provide differing data structures

These are typical interoperability problems occurring in cross-organizational scenarios. Agent technologies in COIN bring together different approaches and to combine them into a new framework: a modeling approach for designing collaborative processes, a model-driven development framework for semantic SOAs and an agent-based approach for flexible execution.

The collaborative processes are specified on a technology-independent level (e.g. using BPMN) and transformed to a platform-independent level so that refinements and modifications in the interaction of the partners can be made on the respective levels and code generated automatically. Flexibility is achieved by applying a BDI agent-based approach. BDI agents provide flexible behavior for exception-handling in a natural way (compared to e.g. BPEL4WS where specifying code for faults often leads to complicated, nested code). The problem of different service granularities is envisaged by specifying a collaborative protocol which allows adaptation to different service granularities. Finally, the mediation of the data is tackled with using transformations which are specified at design-time and executed at run-time by transforming the exchanged messages based on the design-time transformations. Two cases can be distinguished:

- (i) **Integrating consortial partners:** Partners define the shared process together. The common information/data model may also be defined together. Roughly speaking, two alternatives are possible, analogous to the local-as-view (LAV) vs. global-as-view (GAV) distinction in the fields of data integration (cf. [15]) and Enterprise Information Integration (cf. [16]).
- In a LAV approach, the common data structure is defined independently from the local data model of each partner. Each partner then defines a (local) mapping from the common information model to the local model. The mapping in turn can be executed (at run-time) either by the consumer of the service or the partner service itself. The first solution is the one preferred by SWS descriptions, e.g. OWL-S where the service provider describes the grounding to e.g. WSDL. The grounding is used by a service consumer who invokes the service. The second solution means that the service consumer always sends the same message (e.g. a SOAP message) to a partner service and does not care about the local data model. This is reasonable if specifying as well as testing the mapping is tedious and the mapping underlies many changes.
- In the GAV approach, each element in the global model is defined in terms of local sources. Adding a new source (i.e. partner) affects the global model. The LAV approach is preferable for cases where the global model is stable and new partners are frequently joining (or leaving) the collaboration space.





**Fig. 6.** Reducing Complexity in Collaborative Processes

(ii) **Integrating partners external to the collaboration:** For integrating external partners service discovery as well as process and data mediation have to be realized. Service discovery can be based on the service requirements which are specified implicitly or explicitly for a service invocation task. Integrating the discovered services requires data transformations which are either provided by the service descriptions or based on a mapping to the common information model.

In the top-down approach a business process engineer starts of with the definition of a high-level process. When she is more or less satisfied with the high-level definition she starts to work on the grounding of the collaborative process. The tedious task of grounding the collaborative process can be supported by agents. Depending on the intelligence of these agents the support can be automated or interactive. The main task to do is to fill the gap between the description of the collaborative process and the services that are available in the service environment. The most extreme case would be that the business process engineer just pushes a button and everything is done automatically. However, it is rather unlikely that this works in the situation that we have right now. So it is likely that the user will be required to do additional manual integration steps.

On the one hand this might mean that already existing services must be composed to provide more abstract services that can be deployed in the collaborative process or EI services must be integrated with the collaborative process that help to bridge the gap in interoperability among the involved partners.

Basic agent technologies are already available to support these system engineering tasks. The agent framework PIM4Agents developed at DFKI directly supports complex workflow patterns like the parallel execution shown in Figure 1 (for a more complete discussion of supported workflow and service interaction patterns see [17]). One should keep in mind that complex workflow and interaction patterns should be introduced at the lowest level of abstraction possible which keeps higher levels of abstraction simpler and easier to understand.

The second case described above where new partners are integrated on the fly (this means at run time because otherwise it comes down to the case that

was just described) would mean that the service discovery and deployment must be done completely automated which is only possible in restricted scenarios.

A task that needs to be additionally solved is the integration of human users. For this, support for the generation of service front ends is needed. Unfortunately, the WSMO framework does not support a generic way to define and deploy service front ends. It is always possible to use agents to provide the service front ends. However, in many cases it will be more desirable to use tools the user is already familiar with as service front ends and let the agents do their work behind the scenes. In any case this setting leaves space for innovative and creative solutions.

## 4 Related Work

Information on related activities in the European research context was already presented in Section 1. Besides the large amount of literature on business process modeling, enterprise application integration and SOAs, the relation between agents and SOAs has already been investigated. [18] cover several important aspects, [19] propose the application of agents for workflows in general. [20] provide an overview of agent-based modeling approaches for enterprises. [21] describe the TROPOS methodology for a model-driven design of agent-based software systems. However, the problems related to integration of agent platforms and service-oriented architectures are out of scope for their approach. [22] map BPMN models to BDI agents but do not consider an integration of agents and Web services. [23] and [24] present a technical and conceptual integration of an agent platform and Web services. [25] integrate Web services into agent-based workflows, [26] integrate BDI agents and Web services. However, the model-driven approach and the strong consideration of problems related to cross-organizational settings have not been investigated in this context. Furthermore, our focus on a tight and lightweight integration of BDI-style agents fits much better to a model-driven, process-centric setting than the Web service gateway to a JADE agent platform considered by e.g. [23]. A good starting point for details on the semantic web is [27] and [28].

## 5 Conclusion

The paper presented results from the COIN project which deal with improving solutions for enterprise interoperability and enterprise collaboration. We presented the context of COIN in the European research area and explained the basic approach and system architecture COIN is aiming at. Special emphasis was put on how agents can support enterprise interoperability as well as enterprise collaboration services.

Future work will include research on how the reasoning capabilities of the agents could be used to help a system engineer at design time to solve interoperability problems. An example of such support is the situation where the information models of the collaborating partners differ. When a set of transformation

services is available in the GSP the agents can search for composed services that transform documents represented in the format of one partner into the format that is understood by the partner who receives the document. Composed transformation services are necessary in case a direct transformation service between the two information models is not available.

## References

1. EI Cluster: Ict for enterprise networking. [http://cordis.europa.eu/fp7/ict/enet/ei\\_en.html](http://cordis.europa.eu/fp7/ict/enet/ei_en.html)
2. (COIN), E.C.I. <http://www.coin-ip.eu/>
3. Bussler, C., Cimpian, E., Fensel, D., Gomez, J.M., Hallerand, A., Haselwarterand, T., Kerriganand, M., Mocanand, A., Moranand, M., Orenand, E., Sapkotaand, B., Tomaand, I., Viskovaand, J., Zaremba, T.V.M.: Web service execution environment (wsmx). W3C Member Submission, available from <http://www.w3.org/Submission/WSMX> (3. June 2005)
4. Christensen, E., Curbera, F., Greg Meredith, M., Weerawarana, S.: Web services description language (wsdl) 1.1. W3C Note, available from <http://www.w3.org/TR/wsdl> (15. March 2001)
5. Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H.F., Thatte, S., Winer, D.: Simple object access protocol (soap) 1.1. W3C Note, Available from <http://www.w3.org/TR/2000/NOTE-SOAP-20000508> (8. May 2000)
6. Vitvar, T., Zaremba, M., Moran, M., Zaremba, M., Fensel, D.: SESA: Emerging Technology for Service-Centric Environments. *IEEE Software* **24**(6) (November 2007) 56–67
7. Studer, R., Grimm, S., Abecker, A., eds.: *Semantic Web Services, Concepts, Technologies, and Applications*. Springer-Verlag, Berlin, Heidelberg (2007)
8. Fensel, D., Kerrigan, M., Zaremba, M., eds.: *Implementing Semantic Web Services: The SESA Framework*. Springer-Verlag (2008)
9. Farrell, J., Lausen, H.: Semantic annotations for wsdl and xml schema. W3C Proposed Recommendation, Available from <http://www.w3.org/TR/sawsdl/> (5. July 2007)
10. Lausen, H., Polleres, A., Roman, D.: Web service modeling ontology (wsmo). W3C Member Submission, Available from <http://www.w3.org/Submission/WSMO/> (3. June 2005)
11. Fensel, D., Lausen, H., Polleres, A., de Bruijn, J., Stollberg, M., Roman, D., Domingue, J., eds.: *Enabling Semantic Web Services: The Web Service Modeling Ontology*. Springer-Verlag, Berlin (2007)
12. Martin, D., M.B., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K.: Owl-s: Semantic markup for web services. W3C Member Submission, Available from <http://www.w3.org/Submission/OWL-S/> (22. November 2004)
13. Zinnikus, I., Hahn, C., Fischer, K.: A model-driven, agent-based approach for the integration of services into a collaborative business process. In Padgham, Parkes, Müller, Parsons, eds.: *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Estoril, Portugal (May 2008) 241–248
14. Zinnikus, I., Hahn, C., Fischer, K.: Agent-driven Semantic Interoperability for Cross-organisational Business Processes. In: *Semantic Enterprise Application Integration for Business Processes: Service-Oriented Frameworks*. IGI Global (September 2009) 61–89

15. Lenzerini, M.: Data integration: a theoretical perspective. In: PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, New York, NY, USA, ACM Press (2002) 233–246
16. Halevy, A.Y., Ashish, N., Bitton, D., Carey, M., Draper, D., Pollock, J., Rosenthal, A., Sikka, V.: Enterprise information integration: successes, challenges and controversies. In: SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, New York, NY, USA, ACM (2005) 778–787
17. Hahn, C., Zinnikus, I.: Modeling and executing service interactions using an agent-oriented modeling language. In: Proceedings of the Forum at the CAiSE'08 Conference, Montpellier, France (June 2008) 37–40
18. Singh, M.P., Huhns, M.N.: Service-oriented Computing — Semantic, Processes, Agents. John Wiley & Sons, Ltd. (2005)
19. Vidal, J.M., Buhler, P., Stahl, C.: Multiagent systems with workflows. *Internet Computing, IEEE* **8**(1) (2004) 76–82
20. Cabri, G., Leonardi, L., Puviani, M.: Service-oriented agent methodologies. In: 5th IEEE International Workshop on Agent-Based Computing for Enterprise Collaboration (ACEC-07). (2007)
21. Penserini, L., Perini, A., Susi, A., Mylopoulos, J.: From stakeholder intentions to software agent implementations. In: Proc. of the 18th Conference on Advanced Information Systems Engineering (CAiSE'06). (June 2006) 465–479
22. Endert, H., Kuster, T., Hirsch, B., Albayrak, S.: Mapping BPMN to agents: An analysis. In: First international Workshop on Agents, Web-Services, and Ontologies Integrated Methodologies (AWESOME07). (2007) 164
23. Greenwood, D., Calisti, M.: Engineering web service — agent integration. In: IEEE International Conference on Systems, Man and Cybernetics. Volume 2. (2004) 1918–1925
24. Dickinson, I., Wooldridge, M.: Agents are not (just) web services: Considering BDI agents and web services. In: AAMAS 2005 Workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE). (2005)
25. Savarimuthu, B.T.R., Purvis, M., Purvis, M., Cranefield, S.: Agent-based integration of web services with workflow management systems. In: Fourth international joint conference on Autonomous agents and multiagent systems (AAMAS 05). (2005) 1345–1346
26. Bozzo, L., Mascardi, V., Ancona, D., Busetta, P.: Coows: Adaptive BDI agents meet service-oriented computing extended abstract. In: Proceedings of the Third European Workshop on Multi-Agent Systems (EUMAS05). (2005)
27. Semantic Web. [http://semanticweb.org/index.php?title=Semantic\\_Web&oldid=44024](http://semanticweb.org/index.php?title=Semantic_Web&oldid=44024)
28. Lausen, H., Ding, Y., Stollberg, M., Fensel, D., Hernandez, R.L., Han, S.K.: Semantic web portals: state-of-the-art survey. *Journal of Knowledge Management* **9**(5) (2005) 40–49

# A Model-Driven Approach to Close the Gap between Business Requirements and Agent-Based Execution

Christian Hahn<sup>1</sup>, Dmytro Panfilenko<sup>2</sup>, Klaus Fischer<sup>1</sup>

<sup>1</sup> Agents and Simulated Reality (ASR)  
at the German Research Center for Artificial Intelligence (DFKI),  
Campus D3 2 Stuhlsatzenhausweg 3,  
66123 Saarbrücken, Germany  
Christian.Hahn@dfki.de

<sup>2</sup> Institute for Information Systems (IWi)  
at the German Research Center for Artificial Intelligence (DFKI),  
Campus D3 2 Stuhlsatzenhausweg 3,  
66123 Saarbrücken, Germany  
Dima.Panfilenko@iwi.dfki.de

**Abstract:** The current state-of-the-art in developing software applications is to design the systems based on visual design tools and take the resulting design artifact as a base to manually code the software application. The process of transferring the (business) requirements to executable code involves several different parties which makes the whole process again very error prone. In this paper, we present a model-driven approach to overcome the gap between business requirements on the one side and multiagent systems on the other, as we consider the use of agents for implementation beneficial in contrast to more traditional approaches like the WS-BPEL engine.

**Keywords:** Model-driven Development, Model-driven Architecture, BPMN, CIMFlex, SOA, SoaML, Multiagent Systems, PIM4Agents

## 1 Introduction

Service-oriented architectures (SOAs) have emerged as a direct consequence of specific business and technology drivers that have materialized over the past decade. From the business side, major trends such as the outsourcing of non-core operations and the importance of business process re-engineering have been key influences driving the surfacing of SOA as an important architectural approach to business information technology (IT) today. From the SOA side, adequate mechanisms need to be explored to combine business requirements and the underlying execution engines. Therefore, often the principles of model-driven development (MDD) [3] and metamodeling are applied.

The aim of this paper is to present a model-driven development approach for managing and implementing interoperable business processes through SOAs and multiagent systems (MASs). We aim at filling the gap between business requirements made on a strategic level and the execution models on the implementation level. Though, it is possible to execute business models with traditional means, e.g. communicating workflow engines for E-business protocols like RoseetaNet, the use of MASs for implementation seems to offer various advantages.

The model-driven approach to close the gap between business requirements and executable agent systems has been developed in the SHAPE<sup>1</sup> (**S**emantically-enabled **H**eterogeneous **S**ervice **A**rchitecture and **P**latforms **E**ngineering, <http://www.shape-project.eu/>) project. SHAPE provides an integrated development environment that brings together MDD with the SOA paradigm and integrates other technologies like MAS, Semantic Web, Grid, and P2P. The technology developed in the project is centered on SoaML [8], a metamodel for describing service-oriented landscapes that is standardized in the Object Management Group (OMG). SoaML is extended with metamodels for the other technology platforms and advanced service engineering techniques

As a consequence of the development in the area of the service modeling the Model Driven Architecture (MDA), which is one instance of MDD, increases the level of abstraction of this and related concepts to a new state. MDA's goal is the faster system development through the model transformations from one level into another. These models are classified by the MDA concept into three levels of abstraction, namely the Computation Independent Model (CIM) level, the Platform Independent Model (PIM) level and the Platform Specific Model (PSM) level ([4], [5]).

As the most of the existing MDA-approaches [6] focus on PIM- and PSM-level and the model-to-model transformation between them, the more conceptual CIM-level is often as assumed to be present and is not investigated further. Hence, a real life software system development project comes to a problem during development in the starting phases where the conceptual modeling on the CIM level and even more unstructured verbal or media information about the application's domain is in play. As there is an existing standard for modeling services on the PIM level – SoaML – that is connected to the agent modeling on the PSM level, we introduce the link between the CIM level modeling with the aid of CIMFlex, a language for comprehensive representation of the requirements through different views on the considered model, and services modeling on the PIM level with SoaML through conceptual mapping rules for the model-to-model transformation. CIMFlex combines the expressiveness of Business Process Modeling Notation (BPMN) and the Architecture of Integrated Information Systems (ARIS) notation through providing an aggregated process view, where the information of the other views (data, organization and business rules) is represented with a higher degree of abstraction.

<sup>1</sup> SHAPE is a European Research Project under the 7<sup>th</sup> Framework Program, detailed information can be found at <http://www.shape-project.eu/>

The remainder of this paper is structured as follows: In Section 2, the model-driven service engineering approach is presented used in our approach. Afterwards, Section 3 illustrates the main concepts of the Service-Oriented Architecture Modeling Language (SoaML), followed by Section 4 detailing the mappings between the business level and SoaML. Section 5 then presents the agent-modeling approach called PIM4Agents and defines the mappings between SoaML and PIM4Agents. Section 6 represents related work, followed by Section 7 naming future extensions. Section 8 then concludes this paper.

## 2 Model-Driven Service Engineering

A central part of the SHAPE technology is model transformations that define the basis for (semi-)automated transformation among several model types, and in particular enable the MDE-based approach for integrated top-down modelling from the CIM level to the PSM level.

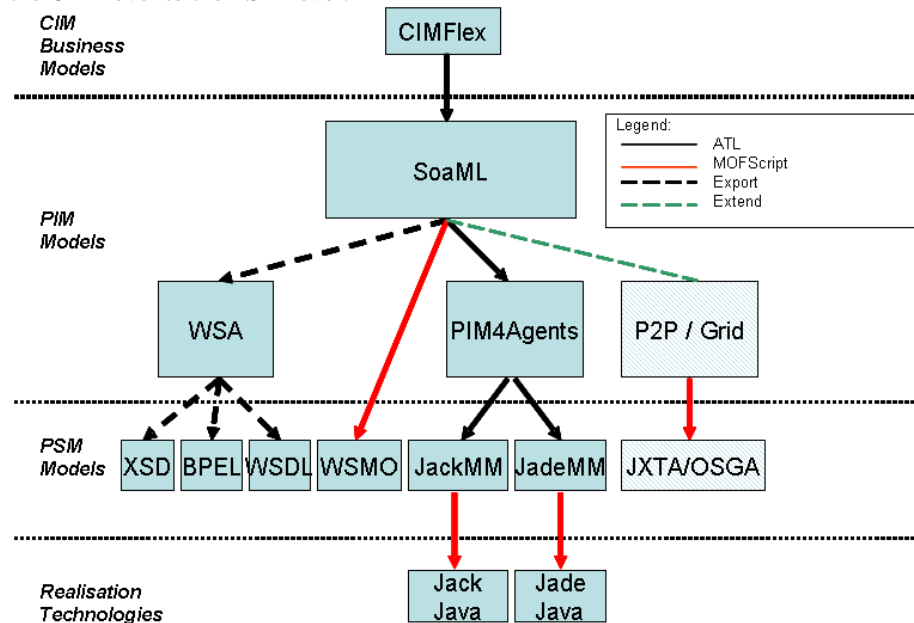


Figure 1: SHAPE Model Transformation Architecture

The model transformation architecture (cf. Figure 1) of SHAPE illustrates the core language used within the project, their relationship to the abstraction levels CIM, PIM and PSM as well as their relationship to other languages defines through model transformations, either model-to-model or model-to-text.

On the highest level, business models encompass business rules, processes, services and other issues such as contracts involving humans and organizations to

achieve business goals. These conform to the metamodel of CIMFlex, which supports the user to create and refine a semi-formal model of a business process, an organisational structure, a data structure or business rules based on the input coming from the domain users. The editor is able to create, change and store these types of models in EPC (Event-Driven Process Chain) or BPMN notation.

The middle layer contains the results of the proposal as transformation engines, extended SOA models, the standardized SoaML and extensions for semantically-enabled heterogeneous architectures (ShaML) that includes concepts for partial modeling of the different platforms (e.g. agents, Web services, etc.). This architecture allows the realization of one of the main goals of SHAPE namely to provide a transformation engine that maps business models to SOA models which are then transferred to the various execution platforms.

The model transformation architecture will in particular support the following model transformations:

#### **CIM to PIM transformation**

- Model transformation between CIMFlex, specifically BPMN, and SoaML: The challenge in transforming CIMFlex models to SoaML is to generate the appropriate system relevant constructs for SoaML according to the generic business context on CIM level. CIMFlex supports in its initial version the model-to-model transformation by making use of the Atlas Transformation Language (ATL) [9].

#### **PIM to PIM transformation**

- Model transformation between SoaML and Web Services: The transformation between SoaML and Web Services is done through a model-to-model transformation. The transformation will produce three kind of models from a single SoaML models. It will produce XML Schemas for information description, WSDL (Web Service Description Language) files for interface description and finally BPEL files for behavioural specification.
- Model transformation between SoaML and PIM4Agents: The transformation between SoaML and PIM4Agents is done through a model-to-model transformation using ATL.

#### **PIM to PSM transformation**

- Model transformations between PIM4Agents and the metamodel of JACK [12] and JADE [13], which are both agent execution platforms. The model transformations between the PIM4Agents metamodel and the metamodels of JACK/JADE (JackMM/JadeMM) are specified through a model-to-model



transformation using ATL. The final code is produced through a model-to-text transformation realized with MOFScript<sup>2</sup>.

- Model transformation between SoaML and WSMO (Web Service Modeling Ontology): The model transformation between the metamodel of SoaML and WSMO is specified through a model-to-text transformation.

In the remainder of this paper, we focus on the model transformation path from CIMFlex to SoaML and from SoaML to PIM4Agents.

### 3 Service-Oriented Architecture Modeling Language

The Service-Oriented Architecture Modeling Language (SoaML) [8] is standardized in OMG. It describes a UML profile and metamodel for designing services.

The goals of SoaML are to support the activities of service modelling and design and to fit into an overall model-driven development approach. The SoaML profile supports the range of modelling requirements for service-oriented architectures, including the specification of systems of services, the specification of individual service interfaces, and the specification of service implementations.

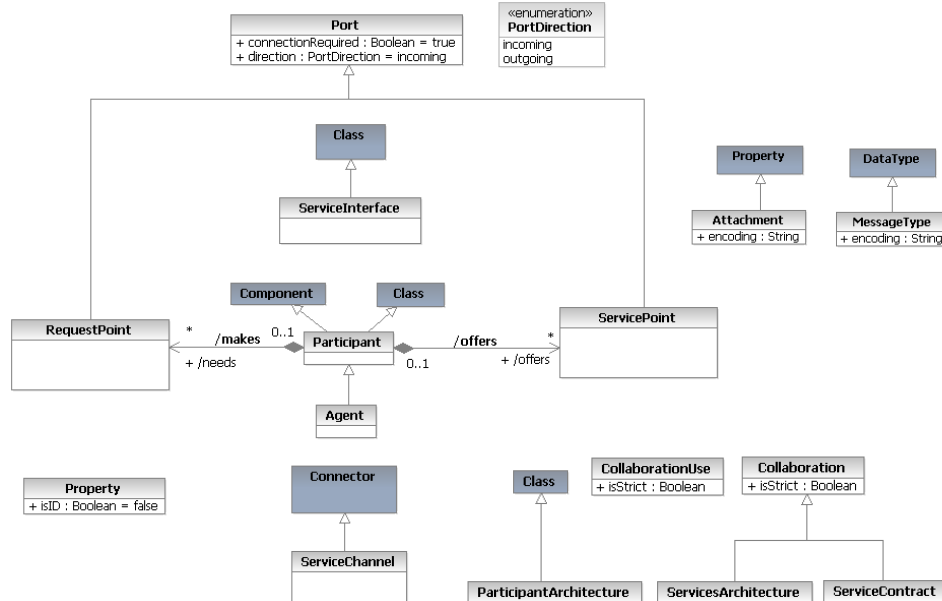


Figure 2: The SoaML profile

<sup>2</sup> <http://www.eclipse.org/gmt/mofscript/>

The SoaML profile (see Figure 2) extends the UML2 metamodel to support an explicit service modelling in distributed environments. This extension aims to support different service modelling scenarios such as single service description, service-oriented architecture modelling, or service contract definition. The main extension areas are:

- **Participants** to define the service providers and consumers in a system. A Participant may play the role of service provider, consumer or both. When a participant acts as a provider it contains ServicePoints, and when a participant acts as a consumer it contains RequestPoints.
- **Service interfaces** to describe the operation provided and required to complete the functionality of a service. A ServiceInterface can be used as the protocol for a ServicePoint or a RequestPoint.
- **Service contracts** to describe interaction patterns between service entities. A ServicesContract is used to model an agreement between two or more parties. Each service role in a ServiceContract has a ServiceInterface type that usually represents a provider or consumer.
- **Service data** to describe service messages and message attachments. The MessageType is used to specify the information exchanged between services, attached to rather than contained in the message.
- **Services and participant architectures** to define how a set of participants works together for some purpose by providing and using services. A ServicesArchitecture or a ParticipantArchitecture describes how participants work together by providing and using services expressed as ServiceContracts.

## 4 From CIMFlex to SoaML


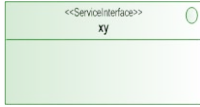
In this section we provide some aspects of the high CIM-level service modeling with the aid of the BPMN. This notation is well-known and established since the beginning of the 21<sup>st</sup> century, moreover it has been standardized and there are more than 50 products, both commercial and open-source, providing the implementation of this standard [11]. The particular considerations with respect to modelling services by the business users are that there is a little awareness of the services by CIM-level users, on the one hand, and even if there would be any knowledge about it, there are no direct constructs describing the services on the CIM-level in the BPMN notation anyway. Of course, the upcoming BPMN 2.0 standard includes the services modeling and the according constructs for it, but it only rules out the second, more technical problem, and not the first one – understanding.

For solving this problem, we propose a semi-automated approach in this section based on a model-to-model transformation from CIM-level BPMN models to PIM-level SoaML-based models. Those models on the higher abstraction level in BPMN would be analyzed through a set of mapping rules and would result in a service model

representing according constructs and architectures needed for the comprehensive PIM-level model as a basis for the further transformation to the PSM-level. The concrete mappings are as follows:



*Task to ServiceInterface* – as a task describes an activity that is possibly providing a useful output that could be consumed by the participants of the process, it can be then assigned to Service Interface construct in this mapping, as it gives the abstract interface for the job done and at the same time doesn't give further specification of the workflow implementing this task.

Table 1 Task to ServiceInterface

BPMN Description	Symbol	SoaML Description	Symbol
Task		ServiceInterface	

*Pool to Participant / ServicesArchitecture* – a pool in BPMN stands for a business entity or a participant of a process, on the one hand. It also can be structured with respect to further participants of the process, thus creating a participants' hierarchy. These two points together put the pool on a role of a candidate for a Participant or Service Architecture, depending on the modeller's intention.

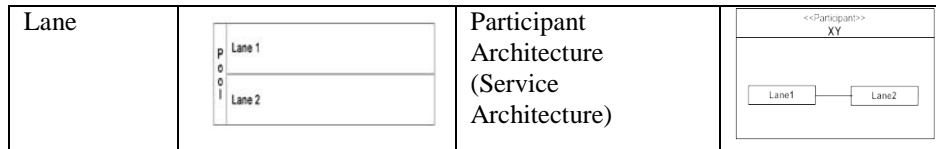
Table 2 Pool to Participant / Service Architecture

BPMN Description	Symbol	SoaML Description	Symbol
Pool		Participant Architecture (Service Architecture)	

*Lane to Participant* – a lane represents a participant or a department in BPMN and is situated in a pool, thus showing the two-tier hierarchy. In order to show the possibility for further subdivision (which is also ongoing in the current BPMN2 proposals), the lane is first mapped to a Participant and next tiers of this hierarchy are constructed using the role constructs described below.

Table 3 Lane to Participant / Service Architecture

BPMN Description	Symbol	SoaML Description	Symbol



*Lanes to ServiceContract* – this transformation also reflects the service contract from the CIM level model into the SoaML and later into PIM for agent interaction specification. There is also a task sequence connected by a sequence flow, but the participants are represented through different lanes in the same pool, thus showing another possibility for a participant architecture modelling. The two tasks that belong to a service contract also share a data object.

## 5 From SoaML to MAS

In this section, a rough overview on the platform independent metamodel for MASs (PIM4Agents) is given. Afterwards, the model transformations between SoaML and PIM4Agent are discussed.

### 5.1 Platform-Independent Metamodel for Multiagent Systems

For modelling MAS and in particular the protocol and the common data model, we developed a platform independent modeling language for MAS called domain-specific language for multiagent systems. The abstract syntax of this language is defined by a platform independent metamodel for MAS called PIM4Agents

The PIM4Agents metamodel [14] that defines the abstract syntax of the modelling language for MASs is a visual platform-independent model that specifies MASs in a technology independent manner. It represents an integrated view on agents in which different components can be deployed on different execution platforms. The PIM4Agents metamodel defines modelling concepts that can be used to model six different aspects or views of an agent system that are listed below:

- **Agent view** describes single autonomous entities, the capabilities they have to solve tasks and the roles they play within the MAS. An Agent has access to a set of Resources from its surrounding Environment. These Resources may include information or ontologies the Agent has access to. Furthermore, the Agent can perform particular DomainRoles that define in which specific context the Agent is acting and Behaviours that define how particular tasks are achieved.
- **Organization view** describes how autonomous entities cooperate within the MAS and how complex organizational structures can be defined. The Organization is a special kind of Agent and can therefore perform

DomainRoles and have Capabilities which can be performed by its members. In addition to the Agent properties, an Organization may have its own internal Protocols specifying (i) how the Organization communicates with other Agents be them atomic Agents or complex Organizations and (ii) how organizational members are coordinated.

- **Role view** covers feasible specializations of the Role concept (i.e. DomainRoles used to partition the organizational space and Actors used to define the message exchange within Protocols) and how they could be related to each other.
- **Interaction view** describes how the interaction between autonomous entities or organizations takes place. Each interaction specification includes the Actors involved and in which order ACLMessages are exchanged between these Actors in a protocol-like manner.
- **Behavioural view** describes how Plans are composed by complex control structures and simple atomic tasks like sending or receiving a Message and how information flows between those constructs. A Plan specifies the agents' internal processes.
- **Environment view** contains any kind of Resource (i.e. Service, Object) that is dynamically created, shared, or used by the Agents.
- **Deployment view** describes the run-time AgentInstances that are involved in the system and how these are assigned to the organization's roles.

To close the gap between design and implementation, we provide generic model transformations from PIM4Agents on the platform independent level to two underlying execution platforms (i.e. JACK or JADE on the platform specific level).

## 5.2 Model Transformations: From SoaML to PIM4Agents

*ServicesArchitecture to Organization:* The concept of a ServicesArchitecture nicely corresponds to the concept of an Organization PIM4Agents as both refer to roles that interact in accordance to some predefined processes. However, and this is the main differences between both constructs, a ServicesArchitecture does not perform any role to the outside. Hence, the generated Organization is more or less utilizes as a social structure providing the space for interaction. Hence, the Organization does neither own any Plans nor perform any DomainRole to the outside and hence should not be considered as an autonomous entity in the MAS, but rather as a form of grouping the necessary autonomous entities to fulfill the service. Likewise, the resulting Organization does not own any kind of knowledge, capability, or resource.

*ParticipantArchitecture to Organization:* A ParticipantArchitecture illustrates, in contrast to a ServicesArchitecture, a concrete entity in the system described. Thus, the target Organization may perform a DomainRole which is either required inside ServicesArchitectures/ServiceContracts or even in other ParticipantArchitectures. Moreover, the Organization may own certain knowledge which is used by the source ParticipantArchitecture.

*ServiceContract to Interaction:* The main purpose of a ServiceContract is to define the roles that agreed on the contract and how these interact with each other which is expressed through any kind of UML behavior. Hence, for representing a ServiceContract in PIM4Agents, the right choice is an Interaction, which defines how the exchange of messages between Actors is specified.

*ServiceInterface to Collaboration:* A ServiceInterface defines a bi-directional service which includes the two roles provider and requester as well as the choreography which specifies the global interaction. In PIM4Agents, the concept of Collaboration is the best match, as it binds AgentInstances to (i) DomainRoles of the Organization and (ii) Actors of the Interaction.

*Participant to AgentInstance:* In the same manner as ParticipantArchitutures are transformed to Agents, a Participant is mapped to an AgentInstance. The agent type of the AgentInstance is deduced from the ParticipantArchitecture that specifies the Participant.

*UML Behavior to Plan:* Any kind of UML Behavior is transformed to a Plan in PIM4Agents. This Plan then specifies the internal processes of the Organization.

*Interface to Capability:* A UML Interface defines a collection of operations and/or attributes that ideally define a set of processes. In order to represent this in an adequate manner in PIM4Agents, the concept of a Capability depicts the perfect match, as both, operations as well as attributes can be included into one of its Plans.

## **6 Related Work**

Only few works exist aiming to bridge the gap between business-oriented approaches and MASs. Taveter [15] presented an agent-based approach for business modeling where Agent-Object Relationship Modeling Language (AORML) [16] is used as underlying agent modeling language. However, normally, agent languages are not the preferred paradigm of business analysts when it comes to designing the particular business requirements. Particular tailored languages are normally used for this purpose. Consequently, Endert et al. [17] presented a transformation between BPMN and JIAC IV (Java-based Intelligent Agent Componentware) [18] to bridge the gap between business process languages on the one hand and agent-based systems on the other hand. However, beside the fact that only a single platform is involved in their model transformation architecture, the even more problematic issue is that in most cases the gap between business languages like BPMN and agent platform cannot be automatically bridged. An intermediate level like defined by SOAs is often considered as more beneficial.

## 7 Future Work

The future work on connecting the high-level service modeling with the systems based on services comprises the alignment of the current BPMN to SoaML mapping with the upcoming new version of the BPMN. As there should be service constructs directly in the BPMN CIM-level methodology, it would be much easier to put this high-level notation with more technical description of a system on a PIM-level with the aid of SoaML in one line together in order to prepare it for the transformation to the PSM-level eventually resulting in an initial snap of the working system's code.

Another point of the future work concerning the high-level service modeling will be the study of the acceptance and evaluation of the comprehensiveness grade of the new version of the upcoming BPMN standard. That is, the question arises how much of the information can be put on the CIM-level that is through different transformations "pushed down" towards the system code. The real problem is not that the information can or cannot be modeled at the highest level, but the question of the understandability of the modeling constructs by the business level users.

A third area of future work is the direct combination of BPMN 2.0 with agent-based systems defined by PIM4Agents. The question that have to be answered in this respect is how much information has to be manually added on the PIM4Agents level after applying BPMN to PIM4Agents model transformation to generate executable agent code.

## 8 Conclusions

This paper represents a model-driven approach to close the existing gap between business requirements specified using existing business modeling languages and agent technologies. To realize this, we defined two model-to-model transformations: The first transformation is specified between CIMFlex and SoaML, the second transformation maps SoaML models to PIM4Agents models. By utilizing the code generators of PIM4Agents, the generated models can be mapped to executable code based on the agent platforms JACK and JADE.

This approach allows the specification of business requirements using BPMN and the mapping to agent code. On each level of the SHAPE model transformation architecture, details with respect to the underlying language and technology can be added that normally requires different roles to be involved, from the business analyst to the agent programmer.

## 9 References

- [1] Erl, T.: SOA - Entwurfsprinzipien für serviceorientierte Architektur. Addison-Wesley, München et al. 2008
- [2] Mathas, C.: SOA intern. Hanser, München 2008
- [3] Stahl, T.; Völter, M.: Modellgetriebene Softwareentwicklung. Techniken, Engineering, Management. dpunkt, Heidelberg 2005
- [4] Frankel, D. S.: Model Driven Architecture. Applying MDA to Enterprise Computing. Wiley, Indianapolis (2003)
- [5] Object Management Group: MDA Guide Version 1.0.1, [www.omg.org/docs/omg/03-06-01.pdf](http://www.omg.org/docs/omg/03-06-01.pdf)
- [6] Mellor, S.J.; Scott, K.; Uhl, A.; Weise, D.: MDA Distilled: Principles of Model-Driven Architecture. Addison-Wesley, (2004)
- [7] Object Management Group: Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. <http://www.omg.org/cgi-bin/apps/doc?ptc/05-11-01.pdf>
- [8] Object Management Group: Service oriented architecture Modeling Language (SoaML) – Specification for the UML Profile and Metamodel for Services (UPMS). <http://www.omg.org/docs/ad/08-08-04.pdf>
- [9] INRIA & LINA: ATLAS Transformation Language (ATL) project documentation. <http://www.eclipse.org/gmt/am3/doc/>
- [10] Hahn, Christian (2009-01-14) SHAPE – Deliverable 5.1: Model transformation and deployment architecture description. [http://www.shape-project.eu/wp-content/uploads/2009/01/shape\\_d51.pdf](http://www.shape-project.eu/wp-content/uploads/2009/01/shape_d51.pdf)
- [11] Object Management Group / Business Process Management Initiative: BPMN implementations. [http://www.bpmn.org/BPMN\\_Supporters.htm](http://www.bpmn.org/BPMN_Supporters.htm)
- [12] Papisimeon, M., Heinze, C.: Extending the UML for designing JACK agents. In: Proceedings of the Australian Software Engineering Conference (ASWEC 01). (2001)
- [13] Bellifemine, F., Bergenti, F., Caire, G., Poggi, A.: 5. In: JADE - a Java agent development framework. Volume 15 of Multiagent Systems, Artificial Societies, and Simulated Organizations. Springer-Verlag, Berlin et al. (2005) 125–147
- [14] Hahn, C.: A platform independent agent-based modeling language. In: Proceedings of the Seventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS). (2008) 233–240
- [15] Taveter, K. A. (2004). A Multi-Perspective Methodology for Agent-. Oriented Business Modelling and Simulation, PhD thesis
- [16] Wagner, G. (2003). The Agent-Object-Relationship meta-model: Towards a unified view of state and behavior, Information Systems 28(5): 475–504
- [17] Endert, H., Hirsch, B., Küster, T. and Albayrak, S. (2007). Towards a mapping from BPMN to agents, in J. Huang, R. Kowalczyk, Z.Maamar, D. L.Martin, I.Müller, S. Stoutenburg and K. P. Sycara (eds), Proceedings on the Internal Workshop on Service-Oriented Computing: Agents, Semantics, and Engineering (SOCASE 2007), AAMAS 2007 Honolulu, HI, USA, May 14, 2007, Vol. 4504 of Lecture Notes in Computer Science, Springer Verlag, Berlin et al., pp. 92–106
- [18] Albayrak, S. and Wieczcorek, D. (1999). Jiac - a toolkit for telecommunication applications, Proceedings of the Third International Workshop on Intelligent Agents for Telecommunication Applications (IATA '99), Springer-Verlag, London, UK, pp. 1–18



# Inter-organizational Interoperability through integration of Multiagent, Web Service, and Semantic Web Technologies

Paul Karaenke<sup>1</sup>, Michael Schuele<sup>1</sup>, András Micsik<sup>2</sup>, Alexander Kipp<sup>3</sup>

<sup>1</sup>University of Hohenheim, Information Systems 2, Schwerzstr. 35, 70599 Stuttgart, Germany, {karaenke, mschuele}@uni-hohenheim.de

<sup>2</sup>MTA SZTAKI, Budapest Lágymányosi u. 11. H-1111, Hungary, micsik@sztaki.hu

<sup>3</sup>High Performance Computing Center Stuttgart, Nobelstr. 19, 70569 Stuttgart, Germany, kipp@hlrs.de

**Abstract.** This paper presents a software architecture for inter-organizational multiagent systems. The architecture integrates multiagent technology with Web service technology to overcome the technical interoperability problem of current multiagent systems in the fast growing service-oriented environments. We integrate Semantic Web technology to overcome the semantic interoperability problem in cross-organizational service provisioning. We address the problem of interoperability regarding interfaces, messaging protocol, data exchanged, and security whilst considering a dynamic e-business environment. The proposed architecture enables service virtualization, secure service access across organizational boundaries, service-to-agent communication, and OWL reasoning within agents.

**Keywords:** multiagent systems, web services, semantic web

## 1 Introduction

In recent years, Service-oriented computing (SOC) has led to significant transformations of industrial software architectures. SOC is in particular aiming at more flexible and open architectures which improve cross-organizational applications. It has originated a remarkable technology stack for Web services (WS) and respective standards which all contribute to systems interoperability. The same can not be said for multiagent systems. However, it has been argued that both technologies “need each other” [5]. Both are concerned with problem solving by distributed systems, but focus on different approaches and offer divergent capabilities: WS-\* standards facilitate the building of secure, robust and reliable virtual organizations (VOs) to solve problems with distributed resources, but lack the capability to react or adapt to undesired conditions, changing requirements and in dynamic environments.

Multiagent technology, in contrast, offers the capability for flexible and adaptive problem solving behavior both on single agent and multiagent level, but lacks reliability, security, and robustness [5]. Thus, combining WS and multiagent

technology could make use of the advantages of both technologies while avoiding their respective drawbacks.

WS specifications foster interoperability on the *technical* level regarding interfaces (e.g., WSDL) and messaging protocol (e.g., SOAP). However, specifications of the data exchanged (i.e., message contents) is beyond the scope of WS specifications. The Semantic Web (SW) approach, in contrast, focuses on *semantic* interoperability. Thus, we propose an approach combining multiagent, WS, and SW technologies and contribute a software architecture for inter-organizational multiagent systems. We address the problem of interoperability regarding interfaces, messaging protocol, data exchanged, and security. Our proposed architecture enables service virtualization, secure service access across organizational boundaries, service-to-agent communication and OWL reasoning within agents.

The remainder of this paper is as follows: section 2 defines the architectural requirements. In section 3, we describe our approach for integrating multiagent and Web service technologies. Section 4 describes the integration of multiagent and Semantic Web technologies. We provide a preliminary evaluation of parts of the architecture in section 5. Section 6 discusses related work. Section 7 summarizes the result and gives an outlook to future work.

## 2 Requirements Analysis

For the targeted interoperability of agent-based systems, agents must have adapters to WS systems; i.e., agents need capabilities for interactions with systems following the Service-oriented architecture (SOA) paradigm. In addition, the inter-agent communication has to be conceptually based on standardized agent communication mechanisms (e.g., [3]). Communication means of agents have to be integrated with Web service technologies to (i) ensure the preservation of the SOA properties interoperability, reliability, security, and robustness for the integration and to (ii) enable the utilization of existing SOA infrastructures.

Communicating components need to have a common understanding of exchanged messages. This means at least, that exchanged messages can be enriched by annotations which refer to a shared conceptualization. Semantic technologies support the agents having behaviors with reasoning capabilities about the current agents' environment, the internal status and especially messages received from other agents. Table 1 summarizes these requirements.

**Table 1.** Requirements.

Requirement	Description
1	The inter-organizational communication interfaces and messaging protocol are based on Web service standards.
2	Agents have capabilities for calling Web services and for providing Web service interfaces.
3	The data exchanged between organizations is semantically annotated based on Semantic Web standards.
4	The agents' internal reasoning is linked to semantically annotated data utilized in inter-organizational communication.

### 3 Integrating Multiagent and Web Service Technologies

#### 3.1 Encapsulation of Web Service Resources

We propose an approach for coupling of WS and agent technology in a head body architecture [7]. The head body paradigm implies a conceptual separation of a software agent into two parts – head and body. The agent's head is used for interactions with other agents. This includes reasoning about interactions such as participating in cooperative processes for problem solving. The body is encapsulating any other (domain) functionality of an agent [7]. The head body paradigm is used in the approach shown in Fig. 1. WS resources that are represented by agents are part of the body. The agent's core capabilities are implemented in the head; i.e., interactions and especially coordination with other agents in the agent society. The agent communicates with the encapsulated WS via WS sensor and WS effector.

On the conceptual level, agent-to-agent communication is based on FIPA communication standards (e.g., [3]). On the technical level, agent-to-agent communication is based on WS technologies and standards. Therefore, the agents can be used in existing WS infrastructures and systems. The presented approach of coupling WS and agents allows the utilization of multiagent coordination protocols for the coordination of existing WSs in existing infrastructures. A WS which is represented by an agent can transparently be invoked by WS clients. The agent can evaluate the invocation requests and can reason if an invocation of the encapsulated WS is in accordance to its own goals. If the invocation request is opposed to the goals, the agent can intercept the invocation and the encapsulated WS is not invoked. Further, the agents can pro-actively work towards the goals; e.g., maximizing revenue for encapsulated resources by establishing Service Level Agreements (SLAs).

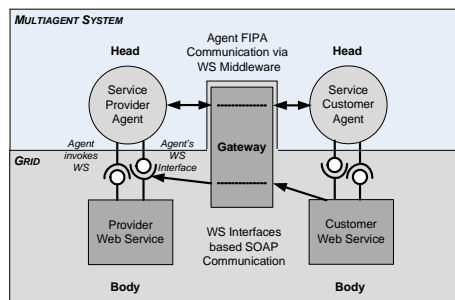


Fig. 1. Head Body Architecture

#### 3.2 Secure Inter-organizational Agent Communication

For multiagent systems, FIPA provides specifications in the area of Agent Communication and Agent Message Transport. The Agent Message Transport group defines ACL (Agent Communication Language), envelope representations, and the

transport protocols that can be used to transfer agent messages between hosts. Currently IIOP and HTTP protocols are supported for agent message transport (WAP support is experimental). The problem concerning multiagent and WS integrations is that the use of IIOP and WAP is declining, while the HTTP based MTP (Message Transport Protocol) is incompatible with SOAP, which is the most commonly used WS messaging protocol.

Furthermore, the communication between multiagent groups, containers or platforms also raises issues when this communication crosses organizational boundaries. Organizations often aim at providing a unified messaging architecture, which can be administered and monitored easily and centrally. The aim of such messaging architectures is to ensure the reliability, flexibility, and security of message transfers. Since we propose a mixed environment of WSs and agents, a natural solution is to transfer messages using SOAP and WS-\* standards. Therefore, we utilize SOAP as message transport between multiagent platforms.

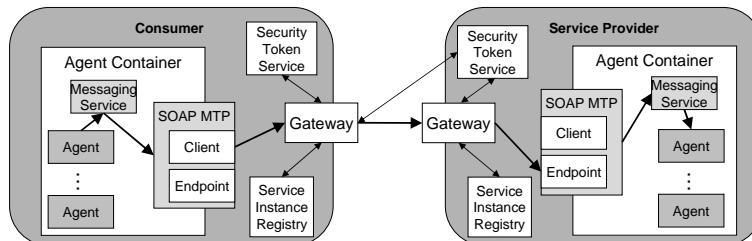
Uniform transportation of agent and WS messages simplifies system administration and enables common mechanisms to be introduced in routing and delivery. This is achieved by adding support for a new MTP to agent platforms. The SOAP MTP add-on [12] is a pluggable driver for sending and receiving SOAP messages and translating them to/from internal agent message format. Each agent platform uses the SOAP MTP add-on configured with a virtual endpoint address, which is mapped to the agent platform address in the Gateway component. The virtual endpoint address is also advertised in registries and directories outside the organizational domain, so that external entities will use the virtual address to reach the agent platform.

Agent platforms can be operated in separate organizational domains. Inside each platform the communication between agents is usually not supervised and not restricted. Similarly, agents can access WSs freely inside the domain. However, the communication between agent platforms has to be supervised, according to current policies of the embedding domains. In order to allow for a seamless integration of components, a corresponding flexible and adaptive messaging infrastructure, commonly titled as “Enterprise Message Bus”, has to be provided. The Gateway Toolkit provides such a messaging infrastructure by allowing for a “double-blind” virtualization approach. On customer side, the Gateway Toolkit allows to hide the corresponding service provider (SP) which allows to describe corresponding workflows in a more abstract manner. Additionally, the customer can easily change service providers by adapting the routing information of the Gateway Toolkit infrastructure whilst not affecting the corresponding workflows. The SP can easily hide the underlying service infrastructure by providing virtual, callable service endpoints to potential customers. The major benefit of the service provider hereby is that he is now able to adapt the underlying service infrastructure whilst not affecting the corresponding service customers. The SP is additionally enabled to involve third party SPs for particular sub-tasks without affecting the customer.

Through using “double-blind” virtualization mechanisms, i.e., by deploying the gateway on both consumer *and* SP side, it is possible to alter resources *and* providers without affecting the calling applications. This deployment of the gateway allows a transparent and secure interaction channel for the involved agents. In particular, the gateway allows the provision of virtual endpoints via which the corresponding agents

are able to interact with each other in a secure way without the need for the agent to explicitly consider the corresponding security as well as the according WS standards. The interaction is done completely transparent for the agent whilst considering dynamic e-business needs; e.g., the possibility to change service provider during runtime, transparent usage of resources whilst considering accounting and secure and reliable communication. Other approaches like the AgentWeb Gateway [16] or the WSIG [6] approach also provide basic support to enable agents to communicate via WS technologies. However, these approaches lack in facing these essential e-business requirements thus enforcing the agent developer to adapt the setup of the corresponding framework every time an evolution step has been processed (cf. §6). To this end the interaction via virtual endpoints allows the adaption of the communication infrastructure during runtime; e.g., in the case an agent has to be replaced by another without affecting the remaining involved agents at any time, which is an essential need for dynamic e-business environments.

Messages between administrative domains are sent and received by the Gateway of each domain (Fig. 2). In our example the Jade agent platform [8] is used. The Gateway mediates the communication between the front end WSs of the two domains. Each front end authenticates itself to their respective Gateway. The Gateway allows the invocation of virtual service endpoints by resolving these virtual to concrete endpoints via the service instance registry (SIR). The SIR also provides additional metadata such as the gateway endpoint that the message has to travel through, as well as the endpoint of the security token service (STS) where tokens can be requested. Virtual addresses used in SOAP messages can be translated dynamically to appropriate real services.



**Fig. 2.** Inter-organizational Agent Communication

The STS issues claim-based tokens to authenticated users, respectively agents, and is also involved in the process of establishing federation with the STS in the SP domain. The consumer-side role of the STS issues tokens that are necessary to pass the security check on the service side. The tokens are generated based on the information that is extracted from the service call message. The service-side role of the security token service acts not as token issuer but as verification instance for security tokens that are attached to the incoming message. It hence has the role of a policy decision point (PDP). In the example, the consumer requests a service from his own SIR by providing an URN (uniform resource name) and the SIR returns the virtual address along with the endpoints of the gateway and the STS. The provider side SIR will convert the server side virtual endpoint to an actual endpoint where the client request can be satisfied.

The following steps are executed when sending a message to a remote agent platform: (1) The consumer agent addresses the message using the virtual endpoint address of the remote agent on the SP side. (2) The Messaging Service detects that this address belongs to the SOAP MTP, and forwards the message to the SOAP MTP add-on for delivery. (3) The SOAP MTP client prepares the SOAP message, and delivers it to the virtual address of the remote agent, but the outgoing message is actually caught by the local Gateway. (4) The local Gateway identifies the recipient SP using the SIR, and arranges for a security token with the STS of both sides. (5) The message is sent to the Gateway at SP side. (6) The SP Gateway checks the access rights for the service, decrypts the message, then finds the real endpoint service using the SIR, and calls the endpoint of the Jade platform. (7) The SOAP MTP of the SP's Jade platform reconstructs the original agent message and passes it to the internal Messaging Service, which finally delivers it to the recipient agent.

#### **4 Integrating Multiagent and Semantic Web Technologies**

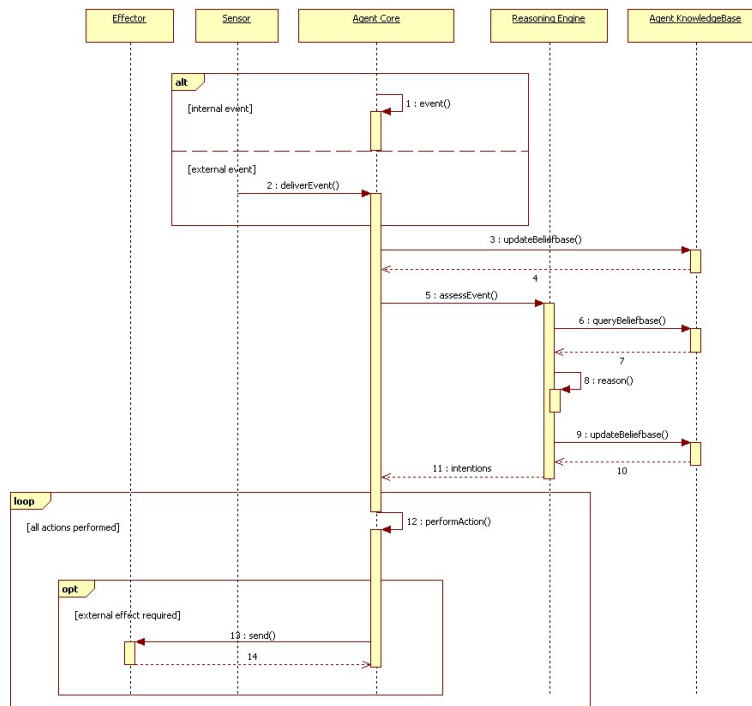
The belief-desire-intention (BDI) approach [1] is the most common architecture for deliberative agents, agents who deliberate over symbolic knowledge to reach given goals [18]. That is, the BDI architecture facilitates goal-driven system behavior. The model consists of the following concepts: beliefs capture informational attitudes realized as a data structure containing current facts about the world. Desires capture the motivational attitudes realized as goals that represent the concrete motivation; i.e., desires capture a set of goals to be realized. Intentions capture the deliberative attitudes realized by reasoning to select appropriate actions to achieve given goals or to react to particular situations.

A BDI agent is equipped with sensors to assist it on its environmental awareness, and effectors to impact the environment by actions. A reasoning mechanism between the sensors' input and the effectors' output deduces the necessary actions for achieving the agent's goals. The agent acquires new beliefs in response to changes in the environment and through the actions that it performs as it carries out its intentions [1]. Thus, the BDI agents allow reasoning regarding decisions to determine which, possibly conflicting, business goals can be achieved and how the agent is going to achieve these goals. For example, for an agent representing a resource of our case, beliefs correspond to the state, capabilities, and SLAs of the resource; desires represent the business goals of the resource provider, while intentions result from a collection of possible decision mechanisms to select and execute requests to use the resource.

In addition, the BDI concept has been integrated with semantics: the agent's beliefs, stored in the agent's beliefbase, are completely based on semantic data. Further, semantic reasoning is applied to derive new knowledge – especially required actions to reach goals – based on the semantic beliefs. Conceptual definitions of SLA parameters, metrics, and economic values as well as resource characteristics are given in an OWL DL ontology [17]. New data arriving to the agents are inserted into the knowledge base, which is automatically enriched using DL reasoning. Agents can then retrieve the results of reasoning via the beliefs. This provides essential support

towards the targeted technical interoperability over organizational boundaries, representing real-world business relationships.

Fig. 3 shows the interactions of the Semantic BDI Agent's internal components for semantic BDI reasoning: based on an internal (step 1) or external (step 2) event, the agent first stores new facts into its beliefbase (step 3). The agent utilizes semantic reasoning to assess the event, deriving new knowledge (step 5-10) and especially appropriate intentions to achieve the agent's goals (step 11). These intentions lead to actions (step 12) which potentially include interactions with external components via the agent's effector (step 13).



**Fig. 3.** Semantic BDI Reasoning Sequence Diagram

The semantic core, an embedded OWL engine is connected to the BDI agent via the beliefbase. The implemented BDI agent plans add or modify facts in the semantic database, which activate any OWL DL reasoning, SWRL or other rules inside the semantic core. The BDI agent core (Jadex [10] in this case) polls dedicated beliefs, which are actually stored in the semantic database. Thus, when reasoning changes the semantic representation of the agent beliefs inside the semantic core, it can trigger a goal via the BDI beliefs. Finally, when goals activate selected plans, the semantic core is updated and the loop starts again. We experimented with prototype implementations for the embedded lightweight semantic core using the Jena SW toolkit, Jena built-in rules and Pellet OWL reasoner (applicable as SWRL rule engine as well). Both solutions provided a small and effective extension to our BDI agents.

In order to exchange semantic data, the components can apply two methods. RDF can be exchanged as plain text (the N3 notation is more convenient during development). Further, semantic annotations can be used with existing XML message formats. An example for the latter is the Semantically Annotated SLA (SA-SLA) format, used for the negotiation of Service Level Agreements (SLAs) between agents [14]. SA-SLA takes an existing XML representation of SLA and connects XML elements to corresponding ontology concepts. Therefore, the common interpretation of loosely defined XML elements is ensured. In contrast to existing approaches (e.g., [11]), our approach allows utilization of OWL not only as a content language for agent messages but also for data exchange with other software components (e.g., WSs). In addition, it enables to determine appropriate actions to reach the agents' goals based on semantic reasoning.

## 5 Evaluation

This section provides an initial evaluation of the inter-organizational agent communication. The goal of this evaluation is to provide evidence that the proposed approach does not result in inappropriate temporal overhead. The experiments evaluate the proposed SOAP MTP and Gateway infrastructure in comparison to the established agent message transport via RMI and HTTP regarding performance in terms of communication time. In the following experiments we compare the agent message transport (1) locally via RMI, agent-to-agent communication on one platform, (2) distributed via HTTP, agent-to-agent communication on different platforms/machines via HTTP, (3) distributed via SOAP MTP, agent-to-agent communication on different platforms/machines via SOAP MTP, and (4) distributed using Gateway infrastructure, agent-to-agent communication on different platforms/machines via SOAP MTP using the Gateway infrastructure.

The setup of experiment1 contains two agents which interact accordant with the FIPA Request Interaction Protocol [4]. Agent1 constitutes the initiator; it sends a *request* to Agent2, the participant. Agent2 replies to the request with an *inform* message to Agent1. The experiment repeats this process 1,000 times for each type of agent message transport. The duration of the protocol execution is measured for every iteration. Fig. 4 shows the results of experiment1 in one chart for each type of agent message transport. Table 2 shows the average values of the protocol execution duration for every type of agent message transport.

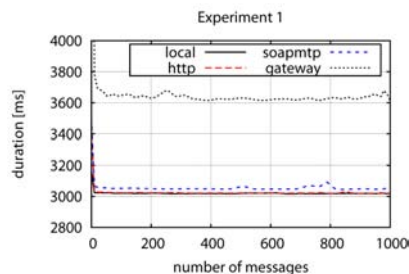


Fig. 4. Experiment 1



**Table 2.** Average duration of the protocol execution in experiment1

	Local	HTTP	SOAP MTP	Gateway
Average execution duration [ms]	3,019	3,023	3,053	3,641

In experiment2, the setup of experiment1 is extended to 20 agents. Ten agents act as initiators and ten as participants. Each of the ten initiators sends 100 requests to each of the ten participants. The duration of the protocol execution is measured for the 10,000 interaction iterations. Fig. 5 shows the protocol execution duration during experiment2 in one chart for every initiator agent (Agent1 – Agent10) and one chart for the average values over all initiator agents. Table 3 shows the average values of the protocol execution duration of every initiator agent for every type of agent message transport and the average values over all initiator agents.

The results of experiment1 shows that the local RMI transport, the distributed HTTP, and SOAP MTP transport differ in a small range of about 35ms. The gateway approach differs from the other approaches with about 600ms (20%).

In the experiment with 20 agents, the average execution duration of the local RMI, distributed HTTP, and SOAP MTP transport approach raises to approximately 4,400ms. However, the average value of the gateway transport approach differs from the other approaches in experiment2 with only about 500ms (11%). An analysis of the charts of the single agents shows that the divergences from the average values are more explicit in this experiment. These divergences are caused by the mechanisms of the BDI framework and relativize the duration differences regarding the type of agent message transport which is also visible in the chart with the consolidated average values over all initiator agents.

**Table 3.** Average duration of the protocol execution in experiment2

	Local	HTTP	SOAP MTP	Gateway
Agent1 [ms]	4,376	4,458	4,433	4,806
Agent2 [ms]	4,365	4,511	4,409	4,842
Agent3 [ms]	4,304	4,476	4,412	4,828
Agent4 [ms]	4,373	4,435	4,447	4,825
Agent5 [ms]	4,368	4,478	4,376	4,914
Agent6 [ms]	4,366	4,488	4,420	4,956
Agent7 [ms]	4,325	4,472	4,434	4,922
Agent8 [ms]	4,368	4,477	4,414	4,946
Agent9 [ms]	4,375	4,511	4,420	4,951
Agent10 [ms]	4,338	4,472	4,409	4,916
Average values Agent1 – Agent10 [ms]	4,356	4,478	4,417	4,891

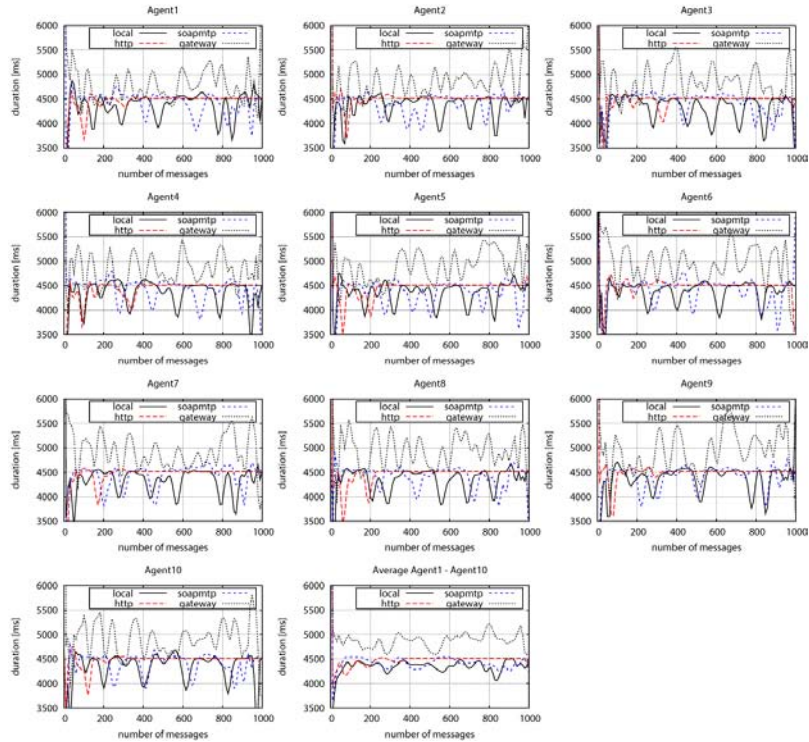


Fig. 5. Experiment 2

## 6 Related Work

The coupling of agents and WS resources in a similar approach is investigated in [15], though the authors remain on a very high level of abstraction and do not consider agent-to-agent communication based on WS-technology. The Web Service Integration Gateway (WSIG) [6][9] is an official Jade plug-in, which provides bidirectional invocation facility, by which Jade agents can call WSs and WS clients can call Jade agent services. The connection of agents and WSs is implemented using elaborate on-the-fly translation between agent messages and SOAP messages. However, WSIG cannot connect agent platforms via SOAP and does not allow transparent wrapping of agent message content into SOAP, thus it fails to support the above mentioned e-business scenarios (dynamically changing business partners, secure communication, etc.). The AgentWeb Gateway [16] is a middleware between agent platforms and WS platforms, which provides protocol transformations similar to WSIG. This approach also lacks the flexibility we missed for WSIG, and furthermore the code seems to be unavailable. We evaluated several other approaches for this aspect of our architecture and found that most of them are not available for re-use. Details of our evaluation can be found in [12].

Moreira et al. presented the AgentSpeak-DL dialect, which extends AgentSpeak towards Description Logic (DL) [13]. However, AgentSpeak-DL is a specific language, and its connection or interoperability with existing OWL ontologies is not straightforward. Therefore, it is not an ideal support for the use of shared OWL in a distributed scenario. AgentOWL is a Jade plug-in, enabling agents to exchange SPARQL and OWL in ACL messages [11]. It also contains an embedded inference engine and connection possibility with remote knowledge bases. It was still not feasible to use it in our scenario as AgentOWL was not integrated with the Jadex BDI implementation, and furthermore its interoperability with external ontologies was unsuitable for our purposes. Nuin [2] is an agent framework designed for practical development of agents in SW applications based on BDI principles. The agents running in Nuin may have access to different knowledge sources, including RDF and OWL ones. Despite that Nuin does not support RDF knowledge sources fully, our approach establishes a direct connection between the agent belief base and the underlying OWL knowledge base, which supports both accessing and modifying OWL facts.

## 7 Conclusion

The contribution of this research is a software architecture for inter-organizational multiagent systems. A key advantage of the presented architecture is a simple, yet powerful communication using a single message bus for both agents and WSs, gateways for the protection of organizational boundaries, and exchange of semantic content based on shared ontologies. An example for the successful application of all these benefits is SLA negotiation, where the SLA requests and offers can be exchanged using common semantics among several service providers. Furthermore, SLAs can be interpreted and reasoned about inside agents, enabling the use of agent cooperation mechanisms for SLA negotiation.

Using the proposed architecture based on WS technology, agent messages can be transferred in a secured way, agent messages can be routed through gateways, and agent addressing can be virtualized; i.e., the agent platform can be dynamically relocated to a different address. The accessibility of agent platforms can be enhanced, as SOAP-based transport is more tolerant with firewalls and other security restrictions. Heterogeneous WS and agent environments may use a homogeneous message transport layer which reduces the complexity of system administration. It also enables secure inter-organizational transfer of agent messages between agent platforms, thus facilitating the advantages of both, multiagent and WS, technologies in a single environment. The preliminary evaluation has shown that the relative temporal overhead of the gateway communication decreases with the number of communicating agents.

The utilization of explicit semantics further facilitates the semantic interoperability by incorporating domain knowledge in all phases of the service life cycle. Future research has to investigate how updates of the shared ontologies can be synchronized among the participants, including authorization aspects. Simulations of advanced scenarios have to further underpin the applicability and utility of the architecture.

**Acknowledgments.** This work has been supported by the BREIN project (<http://www.gridsforbusiness.eu>) and has been partly funded by the European Commission under contract FP6-034556.

## References

1. Bratman, M. E.; Israel, J., Pollack, M. E.: Plans and resource-bounded practical reasoning. In: *Computational Intelligence* 4, 349-355 (1988)
2. Dickinson, I.; Wooldridge, M.: Towards Practical Reasoning Agents for the Semantic Web. In: *Proceedings of the 2nd Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS-03)*, Melbourne, Australia (2003)
3. FIPA Communicative Act Library Specification, <http://www.fipa.org/specs/fipa00037/>
4. FIPA Request Interaction Protocol Specification, <http://www.fipa.org/specs/fipa00026/>
5. Foster, I.; Jennings, N. R.; Kesselman, C.: Brain meets brawn: why Grid and agents need each other. In: *Proceedings of the 3rd International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004)*, New York, USA. pp. 8-15 (2004)
6. Greenwood D.; Calisti, M.: Engineering Web Service - Agent Integration. In: *Proceedings of the IEEE International Conference on Systems, Man & Cybernetics*. The Hague, Netherlands, IEEE (2004)
7. Haugeneder, H.; Steiner, D.: Ein Mehragentenansatz zur Unterstützung kooperativer Arbeit. In: Hasenkamp, U.; Kirn, St.; Syring, M. (ed.): *CSCW – Computer Supported Cooperative Work*. Addison Wesley, Bonn et al., pp. 203–229 (1994)
8. Jade - Java Agent DEvelopment Framework, <http://jade.tilab.com/>
9. Jade Web Services Integration Gateway (WSIG) Guide, [http://jade.tilab.com/doc/tutorials/WSIG\\_Guide.pdf](http://jade.tilab.com/doc/tutorials/WSIG_Guide.pdf)
10. Jadex BDI agent system, <http://jadex.informatik.uni-hamburg.de>
11. Laclavík, M.; Balogh, Z.; Babík, M.: AgentOWL: Semantic Knowledge Model and Agent Architecture. *Computing and Informatics*. Vol. 25, no. 5, p. 419-437 (2006)
12. Micsik, A.; Pallinger, P.; Klein, A.: Soap based message transport for the jade multiagent platform. *The 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009) Industry track*, Budapest, Hungary, May 10-15 (2009)
13. Moreira, Á.F.; Vieira, R.; Bordini, R.H.; Hübner, J.: Agent-oriented programming with underlying ontological reasoning, In: Baldoni, M., Endriss, U., Omicini, A., & Torroni, P. (Eds.), *Proceedings of the Third International Workshop on Declarative Agent Languages and Technologies (DALT-05)*, held with AAMAS-05, 25th of July (2005)
14. Munoz, H.; Kotsiopoulos, I.; Vaquero, L. M.; Rodero, L.: Enhancing Service Selection by Semantic QoS. In *Proceedings of the 6th European Semantic Web Conference on the Semantic Web*, Crete, Greece (2009)
15. Negri, A.; Poggi, A.; Tomaiuolo, M.: Intelligent Task Composition and Allocation through Agents. In: *Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE'05)*, pp. 255-260 (2005)
16. Shafiq, O.M.; Ali, A.; Ahmad, H.F.; Suguri, H.: AgentWeb Gateway - a middleware for dynamic integration of Multi Agent System and Web Services Framework, *14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE'05)*, pp. 267-270 (2005)
17. W3C: Web Ontology Language (OWL), <http://www.w3.org/2004/OWL/>
18. Wooldridge, M.: *Reasoning about rational agents*. MIT Press (2000)

# Ontology Matching across Domains

Renato Levy<sup>1</sup>, Jakob Henriksson<sup>1</sup>, Margaret Lyell<sup>1</sup>, Xiong Liu<sup>1</sup>,  
Michael J. Mayhew<sup>2</sup>

<sup>1</sup> Intelligent Automation, Inc., 15400 Calhoun Drive, Suite 400  
Rockville, MD, USA, 20855, +1-301-294-5200  
{ rlevy | jhenriksson | mlyell | xliu }@i-a-i.com

<sup>2</sup> Air Force Research Laboratory, Information Directorate  
525 Brooks Road, Rome, NY, USA 13441, +1-315-330-7380  
michael.mayhew@rl.af.mil

**Abstract.** Ontologies are often used to annotate information (metadata) that is passed between domains during negotiation. In that sense, Ontology matching is critical for the receiving domain to gather the correct meaning of the data, and hence critical for interoperability. Many Ontology matching algorithms have been proposed in the literature but in general they all assume that there is a considerable amount of knowledge about both ontologies (sender and recipient). This assumption is not true in many cases. In this paper, we present an approach that does not require such assumption, allowing the parts to keep a considerable amount of secrecy on their Ontology while still providing the required matching functionality.

**Keywords:** Ontology, matching algorithm, metadata, cross-domain, multi-language interoperability

## 1 Introduction

Ontologies are often used to annotate information (metadata) that is passed between domains during negotiation. In that sense, Ontology matching is critical for the receiving domain to gather the correct meaning of the data, and hence critical for interoperability. Many Ontology matching algorithms have been proposed in the literature but in general they all assume that there is a considerable amount of knowledge about both ontologies (sender and recipient).

The manner in which an ontology is organized can give valuable insights on the organization's knowledge representation and the importance, complexity or amount of data expressed in this knowledge base. This information in itself is very valuable and the assumption that negotiations can happen across domain boundaries with full disclosure of the domain's ontologies are naïve at best.

In this paper, we present an approach that does not require such assumption, allowing the parts to keep a considerable amount of secrecy on their Ontology while still providing the required matching functionality. In fact, we want, as much as possible, to keep the upkeep of the sending and receiving ontologies separated. This creates an extra layer of complexity for Ontology matching problem, since the metadata associated with the information must be converted to the receiving ontology at the domain boundary.

The ontology matching process across domain boundaries has some extra requirements from the traditional academic problem that makes it unique. Some of these key issues are:

- Multilingual/multicultural → One important issue in the cross-domain arena is that the Ontologies to be matched maybe in different languages (multi-national negotiations), hence syntax proximity is not relevant.
- Independent management/runtime matching → Another important issue to be observed is the ability to handle the matches quickly at runtime, without an extensive preparatory effort, thus allowing the Ontologies to be managed independently.
- Limited information exchange → In the case of cross-domain, the participants of the Ontology matching may not want to disclose their full ontology, but only the necessary information for a correct matching to be performed. One must remember that the need for the ontology matching is often not a translation, but only the adjustment of the metadata and the coherence and continuity of its properties.

Although the existing literature does not directly apply to this practical extended problem, we were able to find relevant work that we believe can be adapted/enhanced to work in our domain.

## 1.1 Related work

Ontology matching is the process of finding semantic correspondence between similar entities of different ontologies. A lot of work has addressed the problem of ontology matching. Here we describe five major matching methods that have been reported in the literature, including graph-based matching, linguistic matching, hybrid matching, learning based matching and probabilistic matching.

1. Graph-based matching or structural matching uses graphs to represent ontologies and computes structural similarities of graphs. Examples of graph-based matching include GMO [1], Anchor-Prompt [2], and Similarity Flooding [3]. GMO is an iterative structural matcher, which uses RDF bipartite graphs to represent ontologies and computes structural similarities between entities by recursively propagating their similarities in the bipartite graphs. This is an approach that we possibly can exploit and hence take a closer look at it in section “Adjacency Matrix-Based Matching Algorithm” below. Anchor-Prompt is an ontology merging and mapping tool, which treats ontologies as directed labeled graphs. The basic idea is that if two pairs of entities are similar and there are paths connecting them, then the entities in these paths are often similar as well. Similarity Flooding is a graph matcher which uses fixpoint computation to determine corresponding nodes in the graphs. The basic idea is that the similarity between two nodes depends on the similarity between their adjacent nodes, or similarities of nodes can propagate to their respective neighbors.

2. Linguistic matching lies in the construction of virtual documents. The virtual document of an entity in an ontology contains the local descriptions as well as neighboring information that contains the meaning of the entity. Then calculating the similarities of entities translates to the problem of calculating document similarities

using traditional vector space techniques. V-Doc [4] is an example of linguistic matcher. It exploits the RDF graph to extract the description information from three sorts of neighboring entities, including subject neighbors, predicate neighbors and object neighbors.

3. Hybrid matching uses linguistic information (e.g., name, label, and description) and structural information (e.g., key properties, taxonomic structure) to find correspondences between entities. For example, PROMPT [5] is a hybrid matching tool for user oriented ontology merging. To make the initial suggestions, it uses a measure of linguistic similarity among concept names and mixes it with the structure of the ontology and user's actions. For each operation, it finds conflicts that the operation may introduce and presents new suggestions to the user.

4. Learning based matching is efficient when instances are available in ontologies. GLUE [6] is an example of learning based matching system. It first applies statistical analysis to the available data and uses multiple learners to exploit information in concept instances and taxonomic structure of ontologies. It then uses a probabilistic model to combine results of different learners. Finally it adopts relaxation labeling approach to search for the mapping that best satisfies the domain constraints and the common knowledge.

5. Probabilistic matching is also used on instance level in ontology matching. For example, OMEN [7] is a tool which describes mappings using probabilities and infers new mappings by means of Bayesian Network inference.

The rest of this paper is organized as follows: In section 2, we present the overall approach for the extended cross-domain Ontology matching problem, and in section 3 present an example on the matching methodology proposed. Section 4 gives more details on how to successfully implement such methodology. Finally, section 5 discusses our conclusions and the future work in this area.

## 2. Overview of Approach

In a collaborative environment, every participant has their own priorities and perspectives of their reality—they each have their own domain model. The domain model highlights what is considered important and formally structures the domain. Such a domain model is the base of the ontology used to describe the concepts on the domain. It is clear, however, that the ontology of one collaborative participant does not always match the ontology of another participant. In fact, it is highly unlikely that this is the case, while at the same time it is likely that the ontologies overlap to some degree. After all, the participants have a desire to communicate so it's likely they have some overlapping terms in their ontologies. We will later refer to these overlapping terms as anchors.

In a cross domain environment that intends to share information between domains, security plays an important role. Hence, not only do domains have an ontology that describes their world, but they also have limitations on how much of this ontology can be shared. These policies refer to the ontology since the policies are specified over the ontology terms (the resources in the domain). We will not focus on the policies here,

but it is important to understand that there are properties associated with the ontology terms and policies that limit the amount of information that can be shared.

So, each domain has a domain ontology and security restrictions specified by policies. This means that any data in the domain is classified according to the ontology, and hence has access restrictions in place.

The premise of interoperability is that information (data) can flow between domains and have their key properties recognized, while still being able to ensure the policy restrictions. In short, the premise of interoperability that data requires metadata to be understood, and the realization that there are limitations on how to translate the metadata context greatly increases the overall complexity of the principle. An illustration of the setup is given in Figure 1.

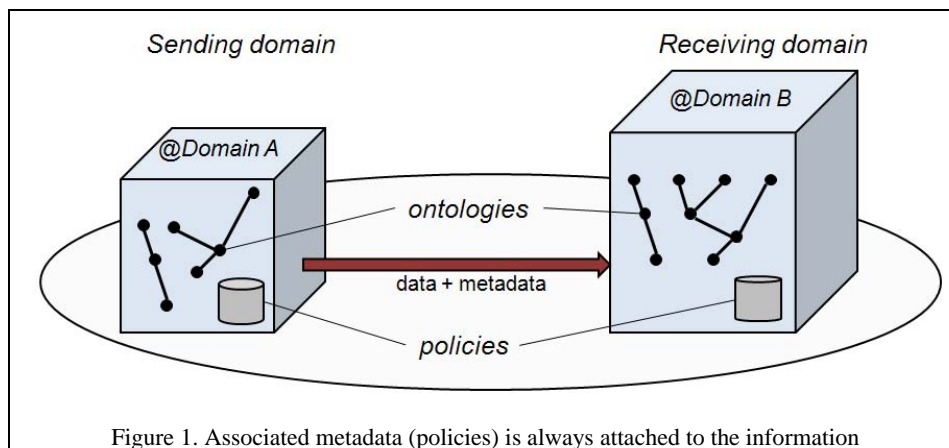


Figure 1. Associated metadata (policies) is always attached to the information

Since the ontologies differ, the security insurances (policies) also do not fit exactly. None the less, the data sent across domain boundaries eventually need to be stored according to the ontology of the receiving domain, and secured by its policies. Hence, to overcome these problems, some issues must be addressed:

- Ontologies cannot be assumed to be fully shared or disclosed between domains, since each domain wants to protect the details of its domain understanding
- Even if we can identify an appropriate concept in the receiving domain ontology that can be used to classify a data item, we cannot assume that the sending domain fully accepts the policy restrictions that are placed on this resource concept.

Even if domains do not want to fully disclose their ontologies, they can agree on certain concepts that they share and can disclose. These concepts would manually be determined by human representatives of the domains. In order to be consistent with nomenclature used in previous literature [8], we call these concepts anchors.

By having a guaranteed partial ontology overlap, it is possible to match a concept in the sending domain ontology with a concept in the receiving domain ontology to a sufficient degree of accuracy. Even if the match is not exact, the sending domain might agree with the security assurances and overall properties provided by the receiving domain and may send the data confident. An initial overview of the process is shown in Figure 2.



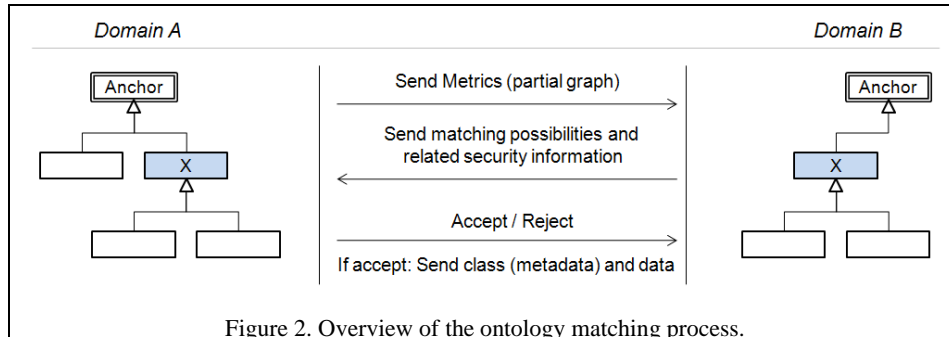


Figure 2. Overview of the ontology matching process.

In more detail, the steps are the following:

1. **Describe matching metrics.** The first step is for the sending domain to get assurances that the data that eventually can be sent with a sufficient matching concept in the receiving domain. Hence, before any data is actually sent, the domains need to negotiate. The data to be sent from the sending domain has some metadata associated with it. In particular, it is declared to be an instance of a particular concept C in the domain ontology. The sending domain, averse to disclosing too much of its ontology, effectively decides on an appropriate subset of the ontology to send over to the receiving domain. This subset is however made anonymous. By this is meant that all entity names, except the anchors (which are already shared), are removed (or changed to meaningless names).

The aim of the sending domain is to provide enough information to the receiving domain such that a similar concept can be found in its ontology by which the received information can eventually be categorized and secured by policies. It should be noted that more than a simple topology of the selected ontology subset is communicated. Rather, what is sent is a set of metrics that can be used for matching against the receiving domain ontology. These metrics are descriptions of how the concept C relates to the anchors. This is important since this is the only way for the receiving domain to be able to find a matching concept (since they share anchors).

An example of a metrics for concept C could be:

(IS-A, 1, "Anchor Concept 1").

This metric says that the concept C is a subclass of the anchor "Anchor Concept 1". The metrics:

(IS-A, 2, "Anchor Concept 1")

for concept C means that C is related to "Anchor Concept 1" by two IS-A (or subclass-of) relationships. That is, there is some concept X that is a subclass of "Anchor Concept 1" and C is a subclass of X. Another metrics that partially describe the sending domain ontology can also be given as we will see in examples below.

2. **Match ontologies.** Once a set of metrics  $\langle M_1, \dots, M_n \rangle$  has arrived at the receiving domain, it tries to determine which concept in its ontology, if any, might be a good match for the data that will arrive. This is done by applying graph search algorithms based on the received metrics. Each entity in the domain ontology is given

a score (value between 0 and 1) for each metric. The set of scores for each entity is then combined and normalized into a final value that represents the confidence of it being a good match (again, between 0 and 1). The best  $k$  matches are selected and each associated with a key. The reason for using keys is to avoid having to disclose anything of the domain ontology to the sending domain. Then  $k$  triples:

<key, relevant properties, matching score>

are sent to the sending domain. This gives the sending domain a chance to pick a desired match.

3. **Metadata and selection.** The sending domain can now make its decision based on the received response triples: the property set for a particular match and its likeliness of being a good match. The most likely scenario is for the sending domain to prioritize on a given property, but this must not be the case. In some cases a good match might be preferred despite detail degradation, while in other cases a lesser ontology match might be preferred when a given property has a higher priority. Nonetheless, the choice lies with the sending domain that is responsible for the data leaving its domain. Without acceptable guarantees given by the receiving domain, the response can also be “reject”, in which case the data is not sent at all. This means that the sending domain does not want to send the data to that particular receiving domain. If the choice instead is “accept”, one of the keys is picked and the data is sent together with the key. The key is here a representative of the metadata in the sense of “data and metadata are inseparable.”

4. **Data storage.** Once the response from the sending domain is received, the receiving domain can classify the newly received data by using the correspondent concept represented by the key.

### 3. Example

In the following we provide an example that demonstrates the intuition behind the steps described above.

Describe matching metrics. The domain ontology for the sending domain is shown in Figure 3. It describes concepts such as “Weather Reading”, “Wind Reading” and “Location”. Notice that we do not only have IS-A relationships, we also have disjoints (e.g. “Wind Reading” and “Hourly Temperature Reading” are disjoint), properties (e.g. location), domain and range restrictions (the domain and range of property “location” is “Weather Reading” and “Location”, respectively), and property restrictions (e.g. “= 1 location”, which means that an instance of “Weather Reading” is related to exactly one instance of “Location” via the property “location”). Hence, we make use of several different kinds of ontology constructs to model our domain.

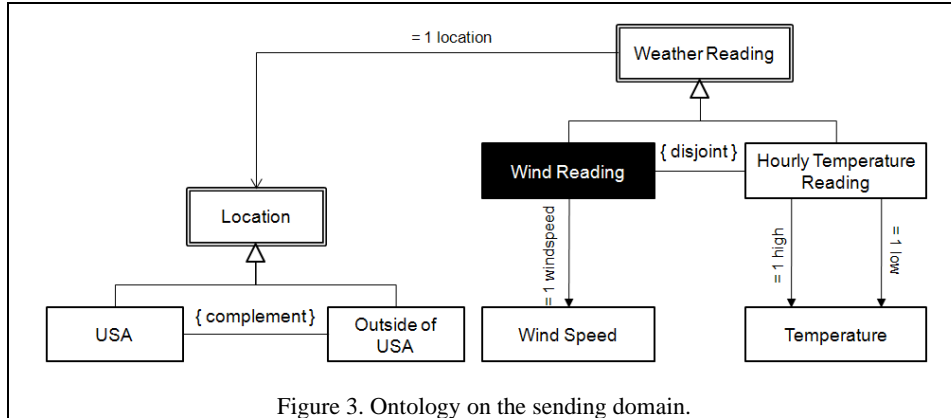


Figure 3. Ontology on the sending domain.

Concepts with a double border are anchors (pre-agreed and shared concepts). In this example the concept “Wind Reading” represents the metadata that is to be sent along with the. Since we know that the receiving domain ontology has the concepts “Weather Reading” and “Location” (they are the anchors), our goal is to describe enough about the concept “Wind Reading” in relation to the anchors such that the receiving domain can do a reasonable match onto its concepts and ontology structure. This description, which we call our metrics, could for example be the ones given in Table 1.

Table 1. Metrics describing a partial ontology

Metric	Explanation
(IS-A, 1, “Weather Reading”)	The metadata concept is one step removed from anchor “Weather Reading” via a IS-A relationship. That is, the metadata concept is a subclass of “Weather Reading”
(disjoint, sibling)	The metadata concept has a disjoint sibling via the IS-A relationship. <i>Notice that thanks to the first metric (above), we already know that this is in relation to the “Weather Reading” anchor.</i>
(domain, 1, = 1 restriction, range: “Location”)	The metadata concept is the domain of an “exactly one value” restricted property that has as range the anchor concept “Location”
(domain, 1, = 1 restriction, range: <i>unknown</i> )	The metadata concept is the domain of an “exactly one value” restricted property that has an unknown range concept. <i>Again, we know that this description is in relation to the “Weather Reading” anchor, and we know that the range is not an anchor, because otherwise it could be given.</i>

It should be noted that all the descriptions of the metadata concept given in Table 1

are in relation to an anchor. This is important since the anchors are the only agreed upon concepts between different domain ontologies. Intuitively, the descriptions in Table 1 correspond to the partial ontology structure highlighted in Figure 4.

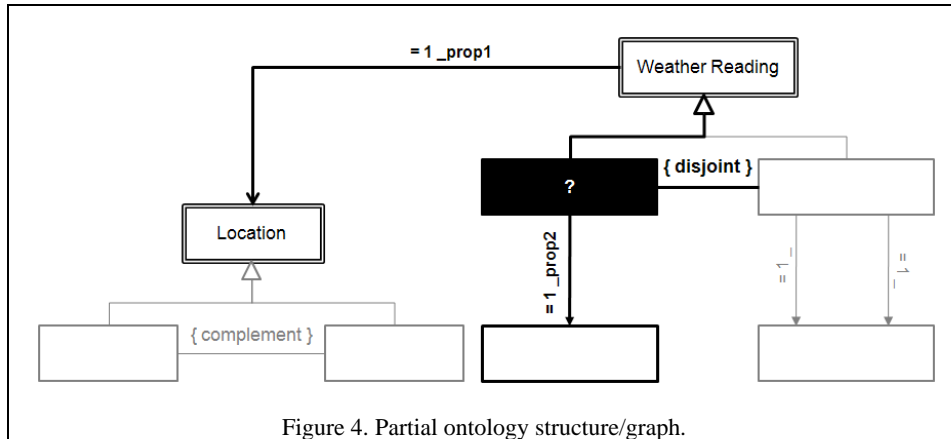


Figure 4. Partial ontology structure/graph.

It should be noted that it is not always desirable to only match the topological structure of ontology graphs. It can also be highly desirable to give larger weight, or preference, to certain kinds of relationships. For example, one could say that a matching IS-A relationship is more important than whether or not a concept is the domain for some specific property. Hence, the relationships between the “nodes” (entities) in the ontology structure can play an important role.

Match ontologies. At the receiving domain, we now assume that the metrics, the partial ontology description, from Table 1 has arrived. The task is now to determine which concept in the local ontology is a likely match for the metadata (concept) that will be sent from the sending domain. The local ontology is illustrated in Figure 5. The ontology describes similar, but different, terms compared to the ontology in Figure 3. That the ontology partly overlap should be clear since they already have agreed on some shared concepts (the anchors).

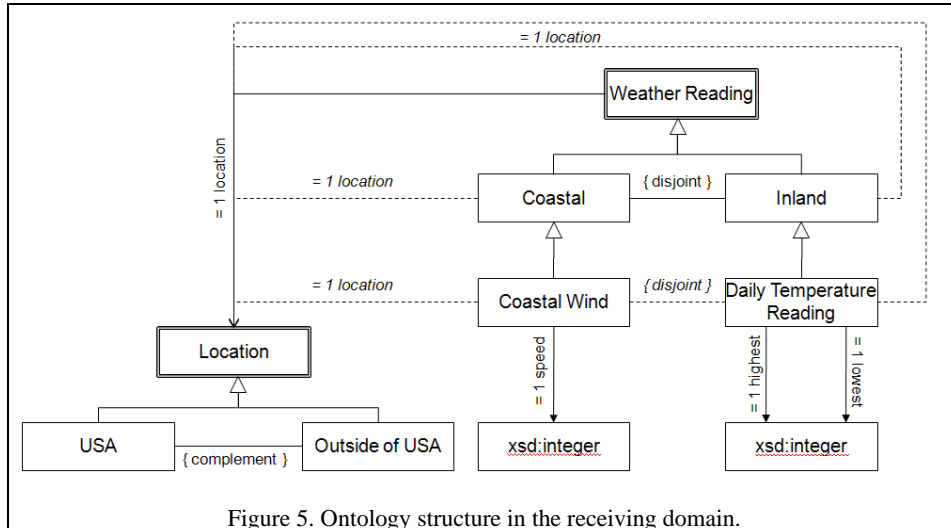


Figure 5. Ontology structure in the receiving domain.

The matching is done by searching the local ontology graph structure and assigning scores to nodes for each of the metrics. An example of scores given to only some of the concepts in the ontology is shown in Figure 6.

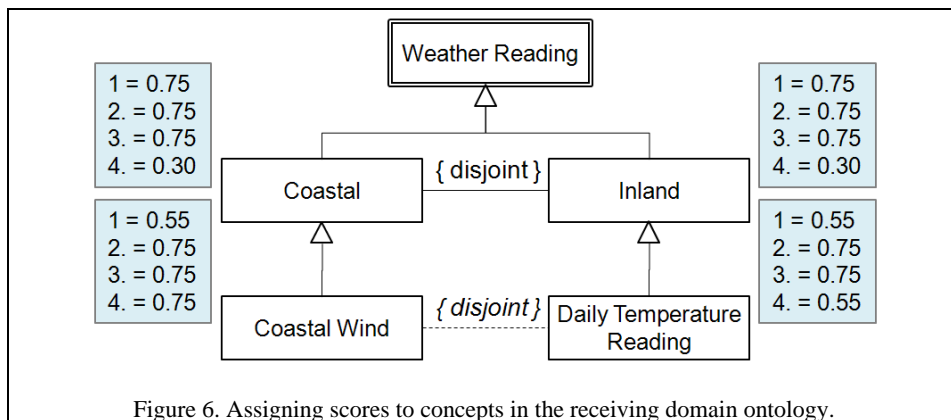


Figure 6. Assigning scores to concepts in the receiving domain ontology.

The orders of the scores correspond to the descriptions in Table 1. For example, the first metric in Table 1 states that the metadata concept is a direct subclass of the anchor “Weather Reading”. Both the concepts “Coastal” and “Inland” match this description and get a high score. The concepts “Coastal Wind” and “Daily Temperature Reading” are not exact matches, but close, and receive a lower score. These calculations are done for each of the metrics that are sent from the sending domain. When we add the scores together we get something like what is shown in Table 2. These scores can then be normalized, but this is left out here.

Table 2. Scores for metadata concepts.

Key	Metadata Concept	Score
key1	Coastal	$0.75 * 0.75 * 0.75 * .30 = 0.1265625$
key2	Inland	$0.75 * 0.75 * 0.75 * .30 = 0.1265625$
key3	Coastal Wind	$0.75 * 0.75 * 0.75 * .55 = 0.2320312$
key4	Daily Temperature Reading	$0.55 * 0.75 * 0.75 * .55 = 0.1701562$

A cut-off value can be selected to limit the number of choices sent back to the sending domain. For example, 0.16 might be chosen for the above example. Hence, the domain would send back the following choices:

<key3, “property1”, 0.23>

<key4, “property2”, 0.17>

Here we have assumed that there is some understanding of what the properties description mean, which could be more elaborate and must be pre-agreed between the domains along with the anchor concepts.

**Metadata and selection.** The sending domain looks at the options, decides that the “property1” is equivalent on its original concept for the data to be sent, and decides to go with what is considered the best match. Hence, the sending domain sends the data together with “key3.”

**Data storage.** The response is handled in the receiving domain by properly categorizing the data and hence protecting the data according to the policy. In this example, the received data would be tagged with the metadata “Coastal Wind”.

### Overview Summary

The example above has been used to demonstrate how ontology matching can be used to facilitate cross domain security communication. The approach is based on the knowledge that each domain has a domain ontology that acts as its data model. Policies are specified with respect to this data/domain model, which are used to guarantee the security of the underlying data.

For domains to share data (secure cross domain information sharing) they first need to negotiate the terms for sharing the data. This is done by matching their ontologies, but without the requirement to fully disclose their ontologies. To be able to do this, the domains have already agreed to certain common concepts. These fully disclosed and shared concepts are referred to as “anchors.”

The domain which is to receive the data, tries to find a good match in its ontology and sends some alternatives back to the sending domain, including information about the properties associated with those ontology matches. The sending domain then has the opportunity to decide what to do, and under which terms to send the data. Once the data is sent, the proper matching can be enforced in the receiving domain, as agreed to by the sending domain.

The crucial point here is to investigate good metrics for describing useful ontology structure with respect to agreed upon ontology anchors. Further to device a successful matching technique that properly can be evaluated and demonstrated to give good results. We outline our initial approach to this matching process below, but more work and evaluation is needed..

#### 4. Secure Ontology Matching Algorithm

In this section, we describe our approach to perform graph-based matching of ontologies. Unlike traditional ontology matching which matches the entities between different ontologies, our problem is to find the best match (i.e., an entity) in the receiving ontology for a given metadata in the sending ontology. Also, only graph-based matching (or structural matching) is allowed in our problem, because the descriptions of the entities in the ontology are not to be shared.

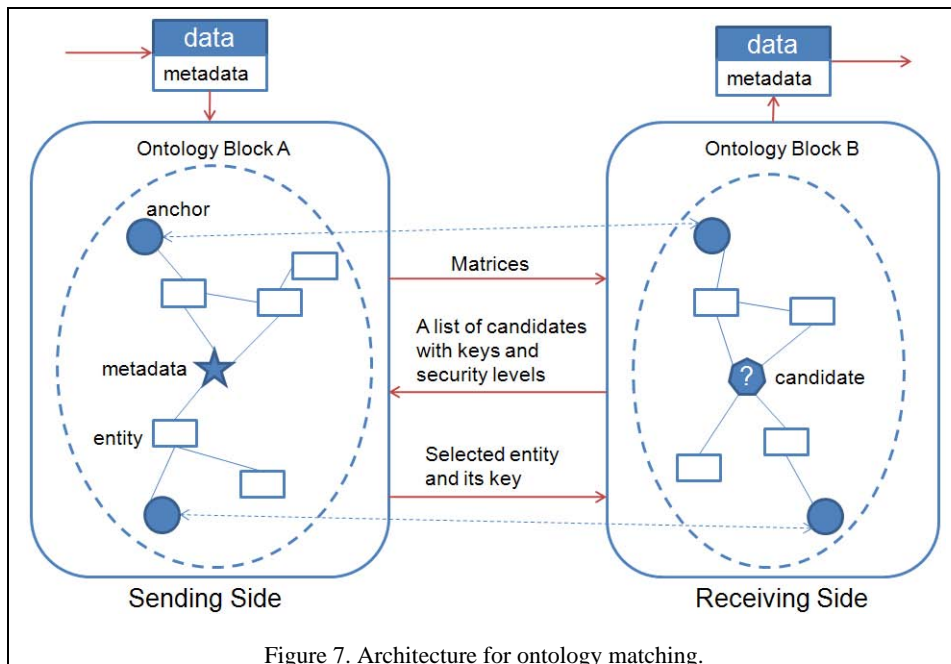


Figure 7. Architecture for ontology matching.

#### Architecture Overview

Figure 7 shows the architecture for ontology matching. On each side, there is a domain ontology (or ontology block). Also, there are anchors that are predefined by humans. Anchors are alignments with high similarities. They are necessary for finding matches of the metadata. The red arrows show the information flow. When the sending side receives a metadata, it will construct matrices or a set of descriptions describing the relationships between the metadata and anchors. The matrices are then sent to the receiving side, which is responsible for finding a list of candidates. Since the contents of the candidates are not allowed to be shared, the receiving side will only send a list of candidate keys with their properties to the sending side. Then the sending side will pick a candidate and send back its choice. Finally the receiving side will attach the selected entity (i.e., metadata) to the data and forward it.

The major processing steps on the sending side include:

1. If the ontology is a large ontology with more than 1000 entities, partition it

into blocks of RDF triples. The divide-and-conquer method described in [8] will be used for partitioning here.

2. Given a metadata M, retrieve the ontology block and construct a graph G for the block.
3. Apply a depth-first search algorithm to construct the matrices or a set of descriptions describing the position of the metadata with respect to the anchors in the block. (Note: refer to the algorithm design part for the details of the depth-first search algorithm)
4. Send the metrics to the receiving side.

The major processing steps on the receiving side include:

1. If the ontology is a large ontology with more than 1000 entities, partition it into blocks in the same way as on the sending side.
2. For each metric obtained from the sending side, extract the information about anchor. Retrieve the ontology blocks that contain the anchor, and construct graphs for the blocks.
3. For each metric, apply a width-first search algorithm to compute the scores of candidate entities that could potentially match the metadata. Note that this search is done in every retrieved ontology block. (Note: refer to the algorithm design part for the details of the width-first search algorithm)
4. Compute the weighted sums of scores for the candidate entities. Rank the candidates and attach information about key and security level.
5. Send the ranked list of candidate entities to the sending side.

### **Search-based Matching Algorithm**

In this section, we describe the algorithm design for the matching process. On the sending side, we will design a depth-first search algorithm to construct the metrics. On the receiving side, we will design a width-first search algorithm to generate a list of candidate entities.

Listing 1 below shows the pseudo code for constructing the matrices on the sending side. This depth-first search algorithm takes the ontology graph G and the metadata M (a vertex of G) as input. It assumes that there is a list of anchors in the ontology (Line 3). It initializes a tree T to the starting vertex, and a list L which stores the edges that are visited (Line 4, 5). In the search function Search(vertex v), the algorithm first marks the vertex as visited and checks if the vertex is an anchor or not. If the vertex is an anchor, the search stops (Line 21, 22, 23). If a vertex v has several unmarked neighbors, the edges between the vertex and the neighbors will be appended to the list L (Line 24, 25). Note that it would be equally correct to visit the neighbors in any order. The easiest method to implement would be to visit the neighbors in the order they are stored in the adjacency list for v. As a depth first search algorithm, it removes edges from end of list L so that the list acts as a stack rather than a queue (Line 10). Also, each vertex is clearly marked at most once, each edge is added to the list L at most once, and therefore removed from the list at most once. The spanning tree T constructed by the algorithm is a depth first search tree, where the leaf nodes contain the anchors. In T, a path from the root (i.e., metadata M) to a leaf node (i.e., an anchor) describes the position of the metadata with respect to the anchor. The easiest method to describe the position would be to count the steps from the metadata to the anchor.



```

1. DFS (G, M) /* G is the ontology graph, M is the
   metadata */
2. {
3.     List A = set of anchors in G
4.     List L = empty
5.     Tree T = empty
6.     Choose M as the starting vertex
7.     Search(M)
8.     While (L not empty)
9.     {
10.        Remove edge (v, w) from end of L
11.        If w not yet visited
12.        {
13.            Add edge(v, w) to T
14.            Search(w)
15.        }
16.    }
17. }
18.
19. Search(vertex v)
20. {
21.     Mark v as visited
22.     If v is an anchor in A
23.         return
24.     For each edge (v, w)
25.         Add edge (v, w) to end of L
26. }

```

Listing 1. Sending side matching

The depth-first search algorithm is mainly for describing the metadata's position with respect to the anchors. When we build metrics describing the relative position between the metadata and non-anchor entities, we will still apply this algorithm to reference the metadata's position with respect to anchors. This is because the reasoning on the receiving side requires the information about anchors.

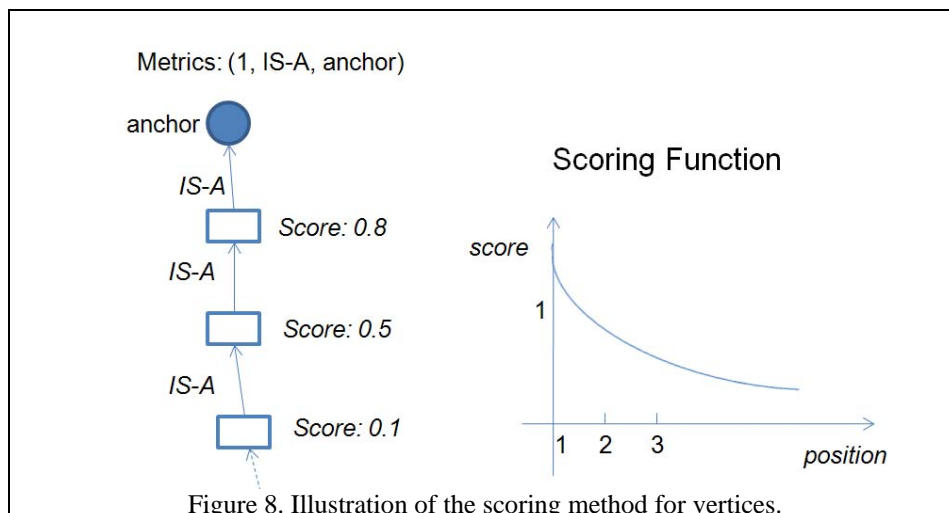
Listing 2 below shows the pseudo code for matching on the receiving side. This width-first search algorithm takes the ontology graph  $G$  and the anchor (vertex of  $G$ ) as input. It initializes a tree  $T$  to the starting vertex, and a list  $L$  which stores the edges that are visited (Line 3, 4). In the search function  $Search(vertex\ v)$ , the algorithm first marks the vertex as visited and assigns a score based on the metric description. If the score is smaller than a critical value, the search stops (Line 19-22). If a vertex  $v$  has several unmarked neighbors, the edges between the vertex and the neighbors will be appended to the list  $L$  (Line 23, 24). As a width first search algorithm, it removes edges from start of list  $L$  so that the list acts as a queue rather than a stack (Line 9). Also, each vertex is clearly marked at most once, each edge is added to the list  $L$  at most once, and therefore removed from the list at most once. The tree  $T$  constructed by the algorithm is a width first search tree of the vertices reached during the search. These vertices represent the candidates that could potentially match the metadata.

```

1. WFS (G, A) /* G is the ontology graph, A is the
   anchor */
2. {
3.     List L = empty
4.     Tree T = empty
5.     Choose A as the starting vertex
6.     Search(A)
7.     While (L not empty)
8.     {
9.         Remove edge (v, w) from start of L
10.        If w not yet visited
11.        {
12.            Add edge(v, w) to T
13.            Search(w)
14.        }
15.    }
16. }
17. Search(vertex v)
18. {
19.     Mark v as visited
20.     Assign a score S based on the metric
   description
21.     If the score S is smaller than a critical value
22.         return
23.     For each edge (v, w)
24.         Add edge (v, w) to end of L
25. }

```

Listing 2. Receiving side matching



We recognize that how to assign score to a vertex is a research issue that needs further investigation. Figure 8 illustrates a simple example to generate the score for a vertex. The basic idea is to define a scoring function based on the metric. According to the scoring function, each vertex will get a score based on its relative position to the anchor. The vertex that perfectly matches the metric gets a highest score; the next nearest vertex gets a smaller score, and so on. We will investigate various ways to define the scoring function.

## 5. Conclusions and Future Research

Our initial results indicate that very consistent matches can be achieved using the techniques describe in this paper. The quality of the matches between Ontologies is highly dependent on the choice of anchors, and the set of relationships (IS-A, etc...) and properties supported.

There are several points that require further research. Here we only list a few of them that we consider to be important:

- It is unclear how good anchor concepts are selected. There are certain requirements, such as them being sharable and general enough to facilitate successful ontology matching. What makes a good anchor is however unclear and will require evaluation of real world scenarios. A methodology to select good anchor points is needed.
- Once a data item has been successfully negotiated and transferred from domain A to domain B, it is important that the same data item can cross back, *without being incorrectly classified*. That is, a correct reverse operation must be guaranteed. The criteria for this operation must be defined, understood, and the ontology matching algorithms must guarantee this property.
- It is important that higher priority can be given to certain relationships. That is, in some cases it might be very important for the sending domain to ensure that a data item's metadata is matched against some concept in the receiving ontology that is closely related to some anchor A via the IS-A relationship. In contrast to, for example, the matching concept being the domain for a certain property. Hence, it should be possible for the sending domain to give *weight* to certain metrics that is sent to the receiving domain. The receiving domain must of course take this into consideration. How this weight is properly respected in the graph matching algorithm needs to be clarified.

## Acknowledgements

This research was partially funded by AFRL under contract# FA8750-09-C-0058. Paper authorized for public release, case #: 88ABW-2010-0886, approved by 88 ABW on 01 March 2010.

## References

- [1] Hu, W., Jian, N., Qu, Y., Wang, Y., “GMO: a graph matching for ontologies”, in: Proceedings of K-CAP Workshop on Integrating Ontologies, 2005, pp. 41–48.
- [2] Noy, N. and M. Musen, “Anchor-PROMPT: Using non-local context for semantic matching”, Proc. IJCAI 2001 workshop on ontology and information sharing, Seattle, WA, 2001
- [3] Melnik, S., H. Garcia-Molina, et al., “Similarity flooding: a versatile graph matching algorithm and its application to schema matching”, Proc. 18th International Conference on Data Engineering (ICDE), San Jose, CA, 2002.
- [4] Qu, Y., Hu, W., Cheng, G., “Constructing virtual documents for ontology matching”, in Proceedings of the 15<sup>th</sup> International World Wide Web Conference, ACM Press, 2006, pp. 23- 31, 2006
- [5] Noy, N. and M. Musen, “PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment”, in the Proc. of 17th AAAI, Austin, TX, 2000
- [6] Doan, A., J. Madhavan, et al. (2003). "Learning to Match Ontologies on the Semantic Web." VLDB Journal 12(4): 303-319
- [7] Mitra, P., N. Noy, et al., “OMEN: A Probabilistic Ontology Mapping Tool”, Proceedings of the Meaning Coordination and Negotiation workshop at the International Semantic Web Conference (ISWC), 2004
- [8] Hu, W., Qu, Y., Cheng, G., “Matching large ontologies: A divide-and-conquer approach”, in Data & Knowledge Engineering, **vol. 67**, 2008, pp. 140-160.

# Agent Metamodel and Profile: Current Status and Perspectives

James Odell

CSC, USA

**Abstract.** This presentation will discuss the agent-based standard currently in-progress at the (OMG) Object Management Group: the Agent Metamodel and Profile (AMP). The objective of AMP is to provide a metamodel and profile for extending UML with core capabilities to enable agents and agent-based software. At a minimum, these core capabilities include the notions of agent, role, and community and the structural and behavioral patterns defined by such constructs. The primary goals of this talk is to explore how the AMP submission will provide a foundation to enable the use of agent technology that can:

- Model agents and agent-enabled constructs that can aid in the design of agent-based systems and emphasize how they will interact and collaborate.
- Be used in conjunction with existing and upcoming OMG technologies, such as: UML, the UML Profile and Metamodel for Services (SoaML) and the Event Metamodel and Profile (EMP).
- Be completed in a timely manner (approximately two years). Multiple follow-on agent-related RFPs can be planned and issued over time. Here, each RFP needs to be tangible and deliverable in a timely manner and carefully coordinated with the other agent-related RFPs

This submission, then, is expected to be the first in a series of agent-related submission. As such, it seeks to address those basic foundational elements of agent technology that are both commonly used and can be defined in a reasonable amount of time.



# Using ontologies to support decentral product development processes

Patrick D. Stiefel<sup>1</sup>, Christian Hausknecht<sup>1</sup> and Jörg P. Müller<sup>1</sup>

<sup>1</sup> Clausthal University of Technology, Department of Informatics  
{patrick.stiefel, christian.hausknecht, joerg.mueller}@tu-clausthal.de

**Abstract.** Adaptive and open platforms for cross-organizational collaborative product development (CPD) need flexible architectures and network solutions as well as novel data integration concepts supporting distributed, decentralized collaboration. Previous approaches to solving this problem have largely ignored the requirement of providing interoperable formats for product model data that enable the support of collaborative product development activities. This paper proposes the use of ontology technology to address this problem; it presents the integration of ontology technology into existing model-driven approaches to product development, and evaluates the applicability of the approach by describing a use case with limited CPD platform functionality.

**Keywords:** Decentral and collaborative product development (CPD), Peer-To-Peer based collaboration, Ontologies in a distributed collaboration environment, Model-driven development of decentral organized information systems.

## 1 Introduction

Cross-organizational, collaborative Product Development (CPD) is the state-of-the-art approach to support knowledge sharing in multi-party cross-organizational engineering projects [1]. To support CPD processes, new collaboration platform technologies are required. They should provide an added value at product definition and execution stages by reducing collaboration complexity and obstacles.

There are many CPD platform design recommendations that mainly focus on data sharing and mapping. These recommendations address one of the key problems of product development: the integration of heterogeneous product model definitions [2]. What still remains is the question on how to accelerate the product development processes, especially in the early phases of the product lifecycle where the major challenge is to efficiently share rapidly changing product design approaches among development teams [3].

These requirements result in new design strategies for CPD platforms. Existing client-/server-based approaches are too inflexible to support loosely coupled, ad-hoc collaboration situations. Therefore, our research focuses on developing decentral information technologies to support CPD processes [4,5]. One of the results is the Product Collaboration Platform (PCP), an experimental peer-to-peer (P2P) software platform to support decentral product development processes. As presented in [1], we follow a model driven development (MDD) approach to design information systems

for decentral and collaborative product development (DeCPD). Based on Computation Independent Models (CIM) we employ an iterative process to develop different abstractions of IT models: starting from IT architecture models over platform specific models (PSM) to concrete software artifacts. We employ an agent-based approach to model the distributed collaboration logics, as is explained in Section 2.1.

When developing distributed systems for knowledge sharing there is a need to provide a common language. Product developers act in different languages and normally have unequal action courses and best-practices when doing their model design activities. A common global CIM process model constrains the user in his process sequence but it does not explain the objective of the collaboration. This results in a lack of interoperability. To deal with model-centric processes, we have to introduce a common formal language that defines the intent of each collaboration step. Ontologies give us the possibility to enhance decentral information knowledge sharing with semantics. Collaborations based upon a semantic data model provide the possibility to understand a collaborations partner design requirements (*specification*) and to answer in an understandable format (*proposal*) without changing local product data management strategies. Thus, ontologies are an important building block to achieve interoperability between distributed model repositories and DeCPD processes. Last but not least we are able to underpin the suitability of the model-driven approach [6].

The structure of this paper is the following: After introducing DeCPD and ontology concepts in Section 2, we will discuss some related work experiences in Section 3. Section 4 deals with DeCPD CIM level models to provide a technology independent need for concepts like ontologies. Section 5 with corresponding PSM models and introduces in our ontology concepts. In the last section we evaluate our approach by providing a reference scenario that we implemented in our Product Collaboration Platform (PCP). The scenario depicts on a simple LEGO™ building example.

## 2 Background

### 2.1 Model-driven decentral and collaborative product development

Each DeCPD process describes a distributed solution of a given product engineering problem (*specification*). Our approach is based on the Distributed Problem Solving developed for multi-agent systems [7, 8]. The DeCPD process provides a synthesis of the distributed partial solutions of the participants (*proposals*) to an overall solution satisfying the initial requirements set up by the initiator.

The engineering problem in our work can be represented as the search of a set of product model (PM) components matching the specification. Product engineers are normally confronted with this problem in the design phase of the product lifecycle. This view enables us to conceptualize a *PM Specification* by a query that describes features, which the target component has to fulfill; a corresponding *PM Proposal* represents one possible design solution.



As described in the introduction modern approaches to collaboration platforms are needed to support the engineering problem described above. In designing a DeCPD collaboration platform, we follow a model-driven top-down approach. To model the underlying collaboration process we start with **Computation Independent Models (CIM)** that describe the functionalities of the platform on the functional level. We use the Business Process Modeling Notation (BPMN) to express CIM models in a language suitable for the audience (i.e. engineers).

The result of performing this process are decentral architectures at **Platform Specific Model (PSM)** level. We design Business Process Execution Language (BPEL) workflows with especial architecture elements to cover the requirements given by decentral information systems subject to our work. Our BPEL workflows are model-centric; they cover event-driven trigger patterns (such as publish-subscribe) from the P2P overlay network.

The focus of this paper is not on the development of MDD models but on their application combined with the use of ontologies that provide the flexible and scalable approach required for collaborative product engineering platforms. In doing so, we aim at

- supporting ad-hoc interconnections between world-wide distributed partners that often did not collaborate in the past and
- effectively distributing product models among participating engineers for load balancing reasons, and a more efficient execution of product development processes for the reasons of task sharing.

## 2.1 Ontologies

We propose using ontologies to represent product model data and metadata. We will give a short introduction into the core concepts of ontologies and related technologies.

We use the web ontology language (OWL<sup>1</sup>) which is standardized by the W3C. OWL is based upon the resource description framework (RDF<sup>2</sup>), which is also a W3C standard. RDF enables the linking between objects, so called RDF Resources. This linking can be described by a directed graph, where nodes represent resources and the edges represent named links. This graph structure is also called *triple*. OWL itself defines some important concepts upon RDF:

- **classes** are similar to sets as they group together individuals having the same properties. For example, a class represents “Lego building bricks” or “plates”.
- **properties** describe the connection between individuals or the assignment of data values to them. For example, an individual of the class *brick* could have a property *hasWidth* which holds a value of “24”.
- **individuals** are instances of classes like in class-based programming languages (Java, C++, C# e.g.).

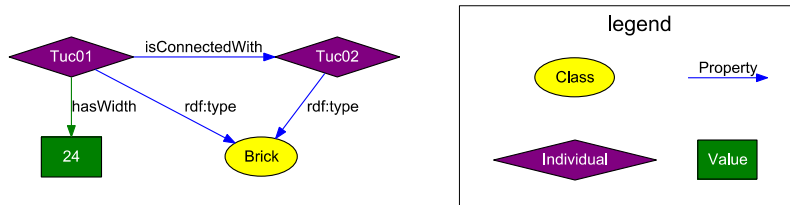
OWL provides a wealth of additional constructs, but these are the most important ones and sufficient for understanding this paper. Fig. 1 shows a simple example: Two individuals (*Tuc01* and *Tuc02*) belong to the class *Brick*. *Tuc01* has a property

---

<sup>1</sup> <http://www.w3.org/TR/owl-features/> [19<sup>th</sup> February 2010]

<sup>2</sup> <http://www.w3.org/RDF/> [19<sup>th</sup> February 2010]

*isConnectedWith*, which uses *Tuc02* as endpoint. Furthermore it has another property *hasWidth*, which assigns the data value “24” to it.



**Fig. 1.** OWL example

Real world ontologies are much more complex and much harder to understand and validate. For that reason, techniques are required to query triples. One query language to provide this functionality is SparQL<sup>3</sup>, which is also a recommendation by the W3C. SparQL has much similarity with SQL. SparQL enables to select arbitrary subsets (subgraphs) from a given RDF graph that fulfill the users constraints (= graph pattern). Furthermore, a SparQL query specifies which elements from the result should be returned. A SparQL query has the general form:

```

SELECT ?subject ?predicate ?object
WHERE {
    ?subject ?predicate ?object.
}

```

After the SELECT statement, elements that should be returned by the query are defined, no matter if these are individuals, classes, or properties. Within the WHERE clause, we define the graph pattern that the query engine has to match.

```

SELECT ?item
WHERE {
    ?item    rdf:type           Brick.
    ?item    isConnectedWith  ?other_item.
}

```

The following example (see Fig. 1) shows the selection of an individual (from the given ontology), the type of which is *Brick* and that has a relationship to another item, specified by the property *isConnectedWith*. The following SparQL query will return the name “Tuc01”.

### 3 Related Work

There are existing research approaches related to the topic of CPD that use semantic descriptions. We observe that there are mainly two ways how ontologies are used in

<sup>3</sup> <http://www.w3.org/TR/rdf-sparql-query/> [19<sup>th</sup> February 2010]

CPD: (1) as a formal description for product model content, or (2) as a structural base that enable a mapping between several product data formats.

### **3.1 Ontologies as formal product model description**

Kim [9] proposes the usage of ontologies for adding a semantic description to product model representations so that it possible to specify the meaning of design challenges. The author introduces an ontology-based specification of an assembly design, so called AsD. Based on this AsD, a product engineer can describe the topology of assemblies and their joining relations, mainly focusing on spatial, geometric characteristics. With computational reasoners it is possible to infer the meaning of an assembly connection. This helps developers understand the meaning of design decisions within a CPD process.

Similar to Kim's approach, Liang [10] studies the description of connections between LEGO™ objects due their given geometric structure. He also implements an ontology using OWL, that describes these connections (he calls them "ports") in order to use it as a tool to find possibilities for connecting assemblies or to prove their validity.

In [11], Mostefai describes a generic and extensible product ontology especially designed to use in the area of mechanics. This is mainly used to facilitate a common understanding among different people such as engineers with their CAD/CAM experience, production planners, IT technicians, etc. If all these people accept a "common ontology", they can contribute to a unified product model.

### **3.2 Ontologies as a support for product model data exchange**

Another use case for ontologies in CPD is the need to develop a platform-independent interchange format for product data. In [2], Patil, Dutta and Sriram propose an ontology-based framework to handle this task, called "Product Semantic Representation Language" (PSRL). They claim that the traditional solutions based upon industry standards for product data exchange (e.g. STEP, Standard for the Exchange of Product Model Data) are limited by the fact, that the semantics are ignored, which may lead to loss of information at transformation processes. They further claim that a better mapping solution can be achieved with PSRL that expresses semantics of a product model and not just plain parameters and values.

To conclude, what we did not find in existing ontology approaches yet is to use ontologies as semantic description in a decentrally organized product engineering process. That is what we need in our work: The cross-organizational process state has to be determined from the product model and that would be impossible when not using ontologies to define semantic information and process coherences upon flat product model data. As a result, in our approach; we can allocate IT modules the capability to interpret intermediate collaboration results and to make the right decisions until the final result is reached.

## 4 DeCPD CIM level models

In this section we give an overview of several model design aspects at CIM level. This should help understand the main motivation when designing a DeCDP platform.

### 4.1 CIM: Process aspect

Using BPMN we build generic, cross-enterprise business processes (CBPs) to describe global DeCPD processes. Each CBP defines participating roles, public processes, and a collaboration protocol, carefully matched to the requirements given in a product modeling environment.

To build CBPs that describe DeCPD processes we specified the following generic public processes:

- *GenerateSpecification* is the process to specify a query that describes the searched component characteristics in a *PM Specification*, while *GenerateProposal* is the counterpart of the process mentioned before, which is used by participants to describe a target component in a *PM Proposal*.
- *PublishProposal* makes a *PM Proposal* available to selected collaboration partners. At CIM layer, it is not specified how this is realized from a technical point of view; thus, e.g., on PSM layer the IT experts could provide a distributed database solution for storing *PM Proposal* information.
- By contrast, *PublishSpecification* distributes a *PM Specification* among the developers involved in the collaboration. Suppose that the product model handling was realized using databases; in that case the *PM Specification* would be a SQL-type query that should deliver answers to the collaboration initiator when executing it after several participants have provided valid proposals.
- The process *Search* describes the possibility to search for existing *PM Specifications / Proposals* in a distributed collaboration environment. The search mechanism has no bearing on the execution of a *PM Specification* query (perform a SQL query e.g.), but it means to find their physical location in the collaboration network. Beyond each search process, there is a complex request-acknowledge and routing method to assure that only trustworthy partners get requested *PM Specifications* and/or -Proposals. These methods are part of our research and will not be discussed in this paper.
- In some environments, it is useful to have a *Notify* process. In technical terms, this corresponds to event-triggered messaging. Independent from any technical realization, it means the participant to register on a particular event, like e.g. the event “newProposal”, whereby he gets informed about every public changes made in this collaboration instance.
- Last but not least a process *Analyze* is needed, that encapsulates the user’s decision on how to proceed a collaboration based on the incoming proposals.

Generic DeCPD processes differ in the following three characteristics. The **PM distribution** (specifies how physical *PM Specification* and/or -Proposal data resources are dedicated to collaboration participants), **hierarchies** (describes the

maximum levels of sub-collaborations, in the case of a 0-level collaboration there is no subdivision) and **iterations** (allows the compilation of versions and variants).

The DeCPD process sequences in CIM Business Process Diagrams (BPDs) depend on the exact values of these three parameters. To evaluate the necessity of ontologies in the field of DeCPD we reduce complexity by setting up a 0-level collaboration between one initiator and a set of participants. *PM Specifications* are distributed among all participants via broadcast and the *PM Proposals* are interchanged via point-to-point transfer between initiator and corresponding participant. To maintain relevance to real world scenarios at product development we do consider iterations. Fig. 2 shows an example BPD for only one participant.

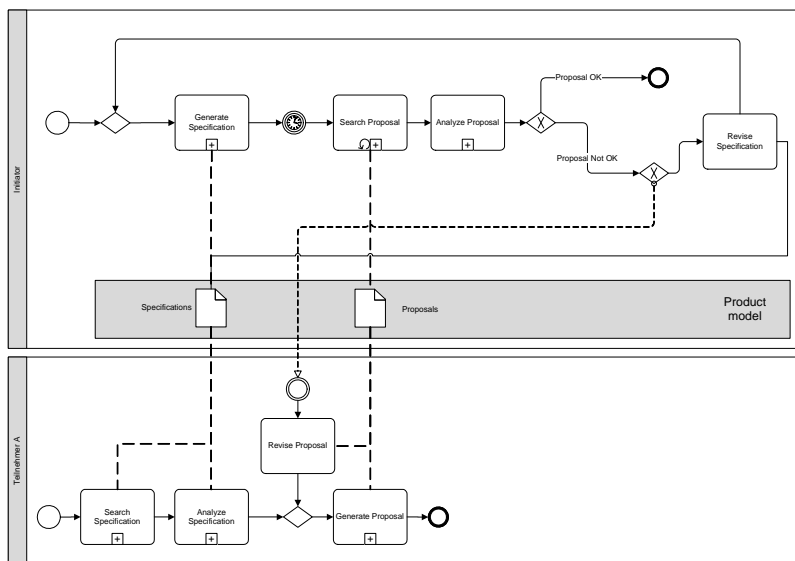


Fig. 2. BPD for o-level collaboration with iterations

## 4.2 CIM: Data aspect

As described in Section 3 we need a data model that fits on the process described above. We defined the meta-model as shown in Fig. 3.



Fig. 3. DeCPD data meta-model

A concrete instantiation of the meta-model and its resulting complexity depends on the DeCPD characteristics (hierarchies and iterations) of a DeCPD *project* and therefore mainly on the amount of sub-collaborations (*sub-projects*, *sub<sub>p</sub>*) and the maximum number of *PM Specification*- and *PM Proposal* variants (*vars/var<sub>p</sub>*), respectively versions (*vers/ver<sub>p</sub>*). As a *variant* we describe all *PM Specifications* /

Proposals that belong to one single component and that are published in parallel. Note that each variant has got only one valid *version*.

To conclude, due to the process limitation that no sub-projects are allowed (cp. Section 4.1), we can concentrate on the collaboration process itself and not on the difficulties that affect us when putting together the *sub-results* from *sub-projects*.

### 4.3 CIM: Service aspect

In describing DeCPD services and their behavior at CIM level, we distinguish between two dimensions: *Service Execution* and *Service Coordination*.

At execution dimensions we distinguish between local and distributed *service execution*. In local *service execution*, each collaboration partner hosts all needed DeCPD core services on its own; thus, the availability of needed services is ensured at any time (= local *service execution*). In the case that DeCPD core services are distributed between participants we have to provide additional service discovery methods at run time. Furthermore we distinguish between *central and decentral service execution*. In the central case, only selected partners provide a choice of core services, whereas in the decentralized case core services are distributed between participants.

Considering the coordination dimension we distinguish between *central and decentral coordination*. *Central coordination* needs a coordinator that controls the cross-enterprise service workflow; *decentral coordinated workflows* do not need a coordinator, but decentral, model-centric mechanisms that make a control flow possible.

To summarize, we decided to use a *decentralized coordinated collaboration* strategy with *local DeCPD core service execution*. Using local *service execution* we do not need to concentrate on strategies for a distributed service search that is as complex as decentral product model data exchange and provides nearly the same challenges. Rather we prefer not using a coordinator and control the workflows by querying the state of the distributed product model.

## 5 Ontologies for a concrete PSM level data model

The generic data model at CIM level has to be concretized at PSM level by specifying specific data elements needed to fulfill the requirements of a DeCPD. In this section we describe on how to use an ontology approach to specify the PSM data model.

### 5.1 Ontologies to describe components and connections

One of the core concepts of DeCPD is the decomposition of an overall product development problem (CIM: project) into several sub-problems (CIM: sub-projects). Central element of the CIM data meta-model is the sub-project related component.

The challenge at PSM level is to specify components in an interoperable way, that means to cover aspects of different business domains like design, functional

requirements, economic or of course geometric and topological parameters. To generate such a component description, all relevant parameters must be representable in a meaningful and feasible way. This is where we apply ontologies due to their flexibility and power of expression.

As known from many programming languages we provide a common domain specific base for the fundamental expressions. This ontology, that we call *BaseOntology* (*BaseOnt*), depends on the domain of the collaboration and must define appropriate *classes*, *properties* and even *individuals* if needed. For sure the scope of our *BaseOnt* is limited to the field of product development, although it contains essential concepts that could be use in other application domains. The *BaseOnt* must be available for all participants in a DeCPD process.

As we chose the LEGO™ system as domain for our example scenarios (cp. Section 6), our *BaseOnt* provides a special vocabulary for a collaboration in this product development area and we define the following five superclasses:

- **Component.** Holds the subclasses *Project*, *Assembly* and *Part* that are constructional superclasses themselves. Derived from *Part*, we define LEGO™ elements such as *Brick*, *Plate*, *Lego Technic Part*, and others. In the example, these special *Part* classes represent all the different kind of LEGO™ building bricks that can be purchased on the market.
- **Connection.** Within the *Connection* classes it is possible to describe the type and the implementation of connections between *parts*, assemblies and perhaps other important stuff. This is especially needed to ensure, that the different *PM Proposals* connects well during the synthesis phase of the collaboration.
- **Requirement.** This class helps to specify the various kinds of requirements concerning a component. We distinguish between functional (movability or flexibility) and non-functional requirements (price, weight, or material).
- **Resource.** Using this class, ontology elements can be linked to external data storages such as files or database tuples. Generally, an external data source contains more detailed information than provided by the ontology.
- **ValuePartitions.** This is a helper superclass to construct enumeration classes that are needed to restrict the ontology user when defining values.

Using these predefined elements, a participant can describe a component by generating needed *individuals* derived from the given classes. By using properties individuals can be tied together or atomic values can be assigned to them. Sometimes the problem may occur that the *BaseOnt*'s classes are not adequate to build up an individual. In this case the missing elements can easily be added to a custom *project ontology* (*PrOnt*), cp. Section 5.3.

Next to the simple description of components, connections between them have to be represented. We provide a solution for this requirement by defining special *connection classes* in our *BaseOnt*. So the connection modality (from physical point of view), the involved components and additional semantic information (why the connection is modeled the way it is) can be described in the same way as shown above.

### 5.3 Project and component queries

For describing the project and its components that should be developed during collaboration we use so called *project queries (PrQ)* that work upon a special *project ontology (PrOnt)* which extend the *BaseOnt* with project specific stuff.

In a *PrOnt* we define the main component that represents our project subject as a new class by deriving from the *Project* class using the *subClassOf* built-in property of OWL. Additionally components are defined by deriving from the *Assembly* class. In order to represent the topological structure, we create an individual for each new type we defined before and link them together via the *isChildOf* property of the *BaseOnt* like building up a *bill of material (BOM)*. To declare which components should really be developed in collaboration we mark each individual with a boolean value using the *BaseOnt's isExposed* property. In summary this results in an ontology structure that holds all the information about types and topological structure of the collaboration.

To extract these information from the *PrOnt* we use a project independent *PrQ*, which is of course defined in the W3C standardized SparQL query language. The *PrQ* basically searches for types of those individuals, which are children of another individual that represents a type being a subclass of the *Project* class. The query returns the name of the component, if it should be developed in the collaboration and the associated project type as shown below:

```
PREFIX lego: <http://tuc.de/ontologies/lego_base.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?components ?value ?root_type
WHERE {
  ?root      lego:hasChild    ?x;
             rdf:type        ?root_type.
  ?root_type rdfs:subClassOf lego:Project.
  ?x         lego:isExposed  ?value;
             rdf:type        ?components.
  ?components rdfs:subClassOf lego:Assembly.
}
```

In either case it might be the initiator that generates the first revision of the *PrQ* and the *PrOnt*. Dependent of the knowledge a collaboration initiator owns when specifying a new *PrQ/PrOnt* composition he might not be able to describe the components. This case might be suitable in real word scenarios when developing new products. So we must differentiate between a project partitioning made by the initiator and those made by the participants themselves.

In reality a one-step *PrQ* breakdown is unrealistic. According to the supply pyramid a given *PrQ* will be specified more precisely from tier to tier. So a *PrQ* distributed by a OEM will be republished in a further collaboration by a 1st tier supplier and so on. Regarding our data meta-model in Section 4.2 it means that projects are recursively subdivided into sub-projects that are connected together by the components interfaces in principal.

The distribution of *PrQ's* is in general only practicable between participants from maximally two supply pyramid layers due to their interest in collaborate. The question is also to allow and/ or how to handle supplier rivals.



After having distributed a project description the initiator hopes to find participants interested in developing selected components. Desired requirements of a *PM specification* are specified in a *component specification query (CSQ)*. In contrast to the *PrQ/PrOnt* the person that describes a *CSQ* (= the initiator in general) is not the one who will also provide the *CompOnt* (= one of the participants). The component is still to be defined (cp. Section 5.1.2).

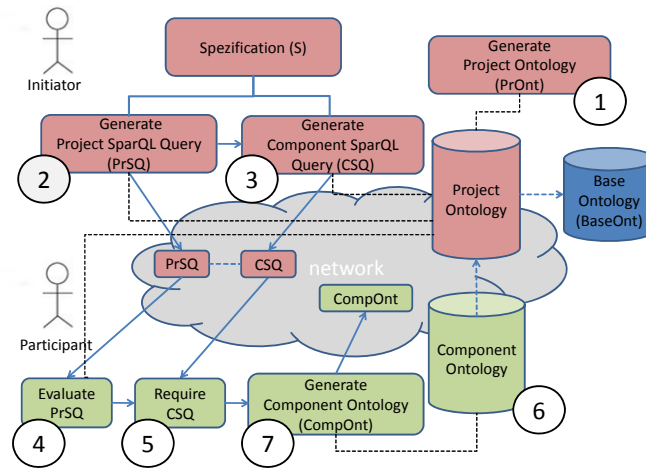


Fig. 4. DeCPD query routing

In the following we will take a look at the query routing that fulfills the requirements of the DeCPD process shown in Fig. 2. We consider a collaboration with only one initiator and an arbitrary amount of participants. Each sub-collaboration (again with one initiator and several participants) would adhere to the following base procedure:

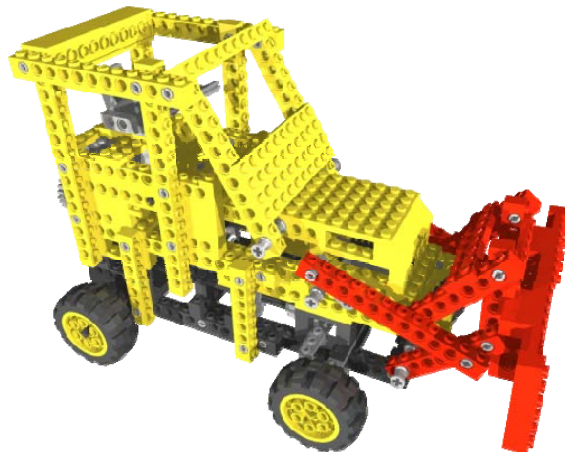
- **Step 1:** The *BaseOnt* is extended to the *PrOnt* by specifying additional components that are not yet contained.
- **Step 2:** The project describing *PrQ* is generated. Executed on the *PrOnt* each *PrQ* returns the list of components to construct during collaboration.
- **Step 3:** The initiator generates a separate *CSQ* (= *PM Specification*) in that the component requirements are specified.
- **Step 4:** A participant receives the *PrOnt* and the *PrQ*. Based on that he can decide whether to provide participate at collaboration or not.
- **Step 5:** Interested participants require a *CSQ* of the component they are interested in from the initiator.
- **Step 6:** The *PrOnt* is extended to the *CompOnt* (= *PM Proposal*) by again specifying additional components.
- **Step 7:** The *CompOnt* is submitted to the initiator and evaluated by executing the *CSQ*.

## 5.2 Component proposals

A *PM Proposal* at PSM level should describe a concrete implementation of a *PM Specification*. Therefore a *ComponentOntology (CompOnt)* is provided. Dependent on the used domain specific CAD models an individual mapping of relevant attributes and features into the *CompOnt* is needed. The mapping is again realized by generating and combining individuals using *PrOnt*'s classes and properties. Additionally the *CompOnt* should provide links to the ontology's source CAD files. During evaluation of a *PM Proposal* the original CAD file is still a useful and important representation.

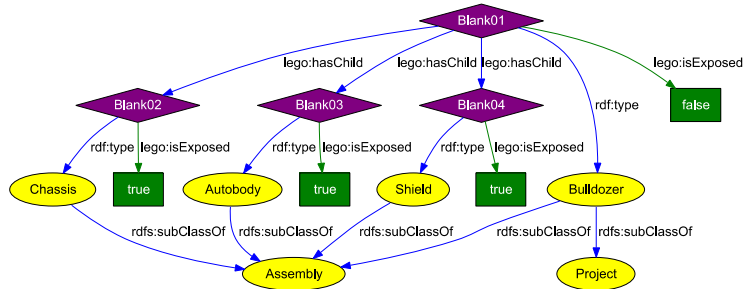
## 6 Evaluation: Sample use case

As proof of concept we implemented an example use case that combines the PSM service concepts with the ontology based data model approach. In the sample use case, we simulate the collaborative construction of a LEGO™ bulldozer (as illustrated in Fig. 5).



**Fig. 5.** LEGO™ bulldozer to evaluate ontologies at DeDCP

The bulldozer represents the data element project and consists of three components that are specified by the initiator: A chassis, a body and a shield.



**Fig. 6.** Project Ontology for the sample use case

As listed in Section 5.3, the **first step** is to use the *BaseOnt* and extend it to the *PrOnt* (cp. Fig. 4, Step 1) by adding project-specific components such as a class for the bulldozer, the chassis, the body and the shield and mark the bulldozer as the project class. For the topological structure we define individuals for each type using the *hasChild* property and mark them via the *isExposed* property to identify which component is a *PM Specification*. Mind that not each component of the underlying bill-of-material has to be developed within the collaboration.

With a special SparQL Query (= *PrQ*) that is developed in **step 2**, the exposed classes can be determined when running the query. Let us assume that in our reference scenario all three components should be developed by collaboration participants.

For each exposed component the initiator has to implement a specification as SparQL query in the **third step** (cp. Fig. 4, Step 3). In summary there are three requirements for this assembly, one restricting the dimension, a second demanding the movability, and the third limiting the costs. The schema of this query type is quite generic and can be used for arbitrary projects and not only for describing LEGO™ toys.

```

PREFIX spec: <http://tuc.de/ontologies/spec_bulldozer.owl#>
PREFIX lego: <http://tuc.de/ontologies/lego_base.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?item
WHERE {
  ?item      rdf:type          spec:Chassis;
             lego:hasRequirement ?req_dim, ?req_mov,
             ?req_costs.

  ?req_dim  rdf:type          lego:Dimension;
             lego:hasDepth      8;
             lego:hasWidth      ?width
             FILTER(?width >= 20 && ?width <= 30).

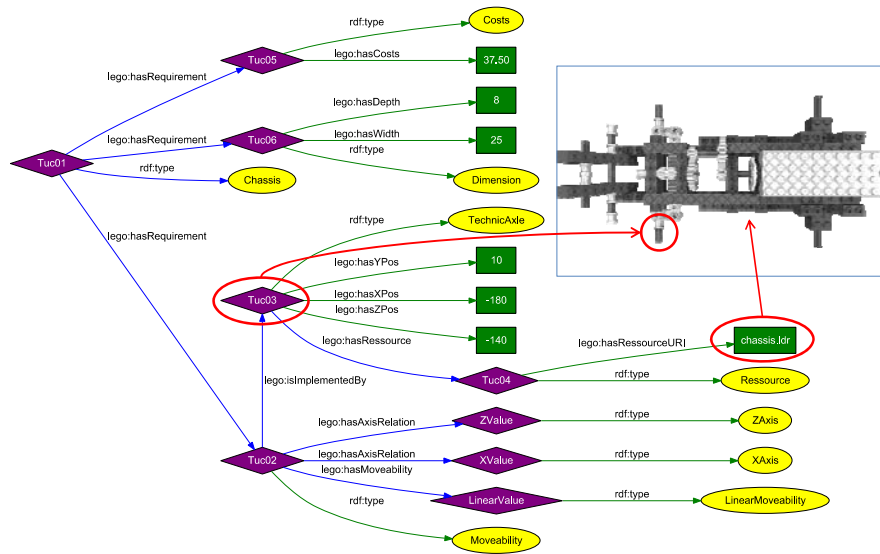
  ?req_mov  rdf:type          lego:Moveability;
             lego:hasAxisRelation lego:XValue,
             lego:ZValue;

  ?req_costs lego:hasMoveability lego:LinearValue.
             rdf:type          lego:Costs;
             lego:hasCosts     ?price FILTER(?price <= 45.99).
}

```

Based upon these requirements provided in a *CSQ*, a developer can now start to implement a *PM Proposal*. This is done by generating a *CompOnt* that follows the structure of the *CSQ*. Therefore a collaboration participant's work is to map his final parameters from his proposed product model into the *CompOnt*.

To check if the *CompOnt* is a valid proposal, both participants and the initiator need to execute the *CSQ* upon this *CompOnt*. If the result of a *CSQ* is an individual of a requested component (cp. individual *Tuc01* in Fig. 8 below) the proposal is valid, if not it is invalid and there may remain some unfulfilled requirements.



**Fig. 8.** Extract from component ontology for a CAD representation of a LEGO™ bulldozer chassis (LDraw).

When the initiator has received enough valid *PM Proposals* (*CompOnt*s) for each component, he can assemble them into a *result ontology* instance (*ResOnt*), which represents the package solution. In other words, it means that assembling all corresponding CAD model files will result in a final CAD model like the one shown in Fig. 5.

## 7 Conclusion and outlook

As mentioned in Section 3, there is a lack of using ontologies to support distributed collaboration processes like those addressed in this paper. The development and application of an ontology especially designed for decentral organized product development enables the full realization of DeCPD platforms. We are now able to complete our existing experimental PCP implementation, which was presented at the CeBIT trade fair in Hannover in 2009, by integrating the missing semantic data

description module. With it we have the possibility to interpret relevant information from the state of the distributed product model that allows us to follow a decentrally developed process model.

Beside the integration of the ontology concept into our PCP, we have to evaluate the applicability in real world engineering scenarios that is on CIM layer, i.e., independent from any IT realization. At the same time existing methods and protocols for query distribution and routing have to be evaluated and expanded to hierarchically designed DeCPD processes containing sub-collaborations. In that case we get confronted with the problem of describing interfaces between product models that could perhaps be realized in a similar way. This problem is yet unsolved. Finally, a more practical requirement which we shall address in future work is to provide automatic mappings from CAD files into the ontology approach.

## References

1. Li, W.D. and Qiu, Z.M. (2006): State-of-the-Art Technologies and Methodologies for Collaborative Product Development Systems. In: *International Journal of Production Research*, 44(13), pp. 2525-2559.
2. Patil, L., Dutta, D. and Sriram, R. (2005): Ontology-Based Exchange of Product Data Semantics. In: *IEEE Transactions on automation science and engineering*, Vol.2, No.3, pp. 213-225.
3. Li W.D., Ong S.K. and Nee A.Y.C. (2006): *Integrated and Collaborative Product Development Environment. Technologies and Implementations*. World Scientific Publishing Co. Pte. Ltd. Singapore.
4. Stiefel, P. D.; Müller, J. P. (2007): ICT interoperability challenges in decentral, cross-enterprise product engineering. In: *Gonçalves; Ricardo J.*, pp. 171–182.
5. Stiefel, P. D.; Müller J.P. (2008): Realizing dynamic product collaboration processes in a model-driven framework: Case study and lessons learnt. In: *K.-D. Thoben, K. S. Pawar, & R. Gonçalves, eds., 14th International Conference on Concurrent Enterprising, 23-25 June 2008, Lisbon, Portugal*.
6. Stiefel, P.D and Müller, J.P. (2009): A model-based software architecture to support decentral product development processes. In: *Proceedings of the Eighth Workshop on eBusiness, Web2009, Arizona; to be published*.
7. Müller, J. P. (1996): *The Design of Intelligent Agents – a Layered Approach*. Lecture Notes in Artificial Intelligence, Volume 1177, Springer-Verlag.
8. Smith, Reid G. (1981): Frameworks for Cooperation in Distributed Problem Solving. In: *IEEE Transactions on systems, man, and cybernetic*, Volume 11, Seiten 61-70.
9. Kim, K.-Y. et al. (2006): Ontology-based assembly design and information sharing for collaborative product development, In: *Computer-Aided Design*, Vol. 38, pp. 1233-1250.
10. Liang, Vei-Chung and Paredis, C.J.J. (2003): A port ontology for automated model composition. In: *Proceedings of the 2003 Winter Simulation Conference*.
11. Mostefai, S. et al. (2004): Effective Collaboration in Product Development via a Common Sharable Ontology. In: *Journal of Computational Intelligence*, Vol.2, No. 4, pp. 206-212.



# Decentralized Semantic Service Discovery in Preferential Attachment Networks

E. del Val, M. Rebollo, and V. Botti

Grupo de Tecnología Informática - Inteligencia Artificial  
Departamento de Sistemas Informáticos y Computación  
Universidad Politécnica de Valencia  
Camino de Vera S/N 46022 Valencia (Spain)  
{edelval,mrebollo,vbotti}@dsic.upv.es

**Abstract.** Service discovery plays an important role in large-scale and highly dynamic environments where the most valuable information is not widely available and may not be registered. In this paper, we present a distributed service discovery approach which makes use of decentralized search algorithms and social network models as underlying structure.

## 1 Introduction

In new paradigms for computing, such as peer-to-peer technologies, grid computing or autonomic computing, large systems can be seen in terms of service provider and consumer entities or agents [15]. The main feature of these domains is that they are open and dynamic, where new agents can enter to the system and existing ones leave. If we consider agents as service providers, the available services change dynamically and it is not an easy task to locate a suitable and available service in a crowded environment with services. In this context, one of the most challenge issue is service discovery. Conventional approaches in multiagent systems such as registries or matchmakers partially address this problem. However, in highly dynamic environments, the most valuable information is not widely available or it is registered in a centralized repository or may not be registered[28]. Much of this information may only be accessed by contacting the right agents. This fact is present in human society. There are scenarios, such as labor market, where the empirical evidences suggest that about half of all jobs are filled through contacts[7]. Recent literature stresses the role of contact networks in other economic phenomena such as buyer-sellers[12] or R&D (Research & Development)[6]. In the provider-consumer scenario, individuals seeking services read yellow-pages, browse in the web and mobilize their local networks of friends and relatives. Networks of personal contacts can mediate in provider-consumer location opportunities which flow through word-of-mouth and constitute a valid alternative source of service information to more traditional methods.

In this paper, we propose a distributed service discovery approach for Open Multi-Agent Systems (Open MAS) using social networks as underlying structure. When an agent asks for another service, a distributed search is made considering only local information associated to its neighbors: degree and service parameters contained in semantic service descriptions offered by each neighbor.

The paper is structured as follows. Section 2 gives an overview of several works in the area of service discovery in MAS. Section 3 describes the structure and the advantages of preferential attachment networks. In section 4, the proposal for distributed semantic service discovery is presented. Section 5 analyzes the performance of the proposal, comparing with other approaches used in distributed search. In section 6 conclusions and future work are presented.

## 2 Related Work

Open and dynamic environments where the scalability and the workload are low make use of middle-agents [26][11][17] to facilitate service discovery. The main advantage is that matchmakers could provide an optimal matching because they consider all the registered services in the system. These middle-agents usually make an efficient search and get a good throughput. Unfortunately, this kind of agents could be a bottleneck when the workload increases. Other drawbacks of middle-agents are their complexity, the huge amount of memory needed to keep service advertisements and the cost of service composition as the number of services grows significantly. Different approaches have been suggested to overcome the above mentioned problems related to the centralized paradigm in service discovery.

*Peer-to-peer* approach takes advantage of the fact that each agent already knows its own capabilities and those of a few peers, and uses peer-to-peer search (recursively) for locating agents with the needed capability [5][23]. An agent broadcasts a query using its local knowledge to its neighbors and the agent that receives such a request either offers its services to the original caller or broadcasts the request to its own neighbors. The drawback of this approach to service discovery is that the communication among agents is essential and the overall communication traffic overhead may be large.

Another distributed way to locate distributed services is to form *coalitions* or clusters[22][18]. Nevertheless, the choice of what coalitions are going to be formed is a difficult task. This entails recursively to calculate the values of the coalitions and later selecting the coalition with the best result. The calculation of the coalition values can be made in parallel, but this phase requires that each agent knows the rest of system agents (global knowledge). In addition to determine the best value, they have to use broadcast. Therefore, in some situations, the system could be overloaded.

A third way for agents to discover services in efficiently is the distribution of the *middle agents* or *facilitators*. Jha et al.[8] suggest to split the function of the service facilitator among a group of agents. The system designer assigns a local matchmaker to each host or segment of the system, which provides matchmaking services to agents in its vicinity (its segment). The local matchmaker can consult its peers or a central matchmaker whenever it cannot provide an answer to a local query. This type of solution reduces communication traffic and confines it to network segments (in which communication is fast). Moreover, it reduces message queue sizes, improving scalability and fault tolerance. Sigdel et al. [25] present an adaptive system. The framework suggested allows automatically adaptable matchmaking methods for service localization depending on the network structure and characteristics. This approach is based on two levels: *system adaptation level* and *node adaptation level*. These approaches are



applicable in systems that have a hierarchical topology, in which information sharing can be confined to local segments. In systems with very large segments the problems of scalability are only marginally relieved by this approach because the large segments become overloaded systems which have local bottlenecks. Another case in which this approach is not useful is in systems with many crosslinks between segments. In this case the overhead of coordinating tasks among local matchmakers might be greater than the benefit obtained from their distribution.

The main advantage of the presented proposals are fault tolerant and adaptable to changes in the environment. Besides that, they decrease communication time and spread the information among agents. The main drawbacks are that distributed approaches such as coalitions or peer-to-peer have performance problems (network traffic, slow response and congestion) and the coordination effort required is not appropriated for highly dynamic environments. Our proposal tries to overcome these drawbacks through a completely distributed approach based on social networks as underlying structure. In the next section the main features of these networks are presented.

### 3 Social Networks

As MAS continue to grow and migrate to heterogeneous environments, such as the Internet and the Semantic Web, the structure of societies in MAS and the interconnections among the agents in these societies will be fundamental to the effectiveness of service discovery. Social network models are an appropriate representation of agent interconnections, such as friendship, financial exchange, relationships of beliefs, knowledge or prestige. Recent studies using data on communication within organizations[1] and the friendships communities[14] have established the fact that human social networks closely match some mathematical models present in social networks. Another interesting feature of these networks is the property of being *searchable*: *'ordinary people are capable of directing messages through their acquaintance networks to reach a specific but distant target person in only a few steps'*[30]. This feature makes social networks not only suitable to model relationships between agents, but also to discover services offered by agents, situated in large networks whose topology is known only locally, in few steps.

To frame the underlying problem, we go back to one of the most well-known social network analysis: *'Six degrees of separation'*[27]. In this experiment Milgram discovered that individuals are connected via short paths, but also that the individuals in these networks, only considering local information about their own neighbors, are able to find these paths. Decentralized search can be classified considering if the network is structured or unstructured. This classification is presented in [31]: *'in structured networks the global position of the target node in the space can guide the search process to reach the target node more quickly. In unstructured networks, the global position of the node is unknown and it is difficult to know whether a step in the search process is towards the target node or away from the target node'*. One of these model sof unstructured networks is the *scale-free*[2].

*Scale-Free Networks.* The scale-free network model is defined as: *'a mathematic model extracted from the real world. The distribution of the number of network neighbors*

(degree distribution) is typically right-skewed with a heavy tail, meaning that a majority of nodes have less-than-average-degree and that a small fraction of hubs are many times better connected than average'[29]. This qualitative description can be satisfied by several mathematical functions, but the most common in the current literature is a power law [2]:

$$P[k] \sim k^{-\alpha} \quad (1)$$

in which,  $k$  is an integer denoting the node degree,  $P[k]$  is the probability that a node connects with  $k$  other nodes. The parameter  $\alpha$  is a scalar coefficient, which usually ranges in:

$$\alpha \in (2, \infty) \quad (2)$$

The power law distribution denotes that some nodes have high degree although most nodes have low degree. This property is called *preferential attachment*: new network members prefer to make a connection to the more popular members in the network.

These kind of networks have many predominant advantages which can be used to improve the cooperative performance in Open MAS, for instance in service discovery. This model has a robust topology which is immune to random errors such as random removal of links or nodes. Thus, for service location, the preferential attachment network offers a reliable topology to ensure that a service can be found under the condition that certain agents leave the system. The main disadvantage is that this kind of networks are very sensitive to 'sabotage' (attacks to highly connected nodes). Another feature of these networks is the path length between two nodes: '*with most disordered networks, such as the small world network model, the average distance between two vertices in the network is very small relative to a highly ordered network such as a lattice*[21]. More concretely, power-law graphs having  $2 < \alpha < 3$  have small diameter  $\log(n)$  where  $n$  is the number of nodes.

Due to all the described features of scale-free networks with preferential attachment, a service discovery system has been proposed based on this kind of complex networks. In the next section this system is described with detail.

## 4 Social Discovery System

We formulate the service discovery problem in an Open MAS as a probabilistic decision-making task in which the goal is to find an appropriated service minimizing the length of the path travelled by the request message. Our system is based on social networks, therefore agents are situated in a network with preferential attachment. We assume that each agent knows about its immediate neighbors including their identity, degree, and parameters related to the service they offer but it is unaware of the rest of the agents in the network. At the source agent, and at each agent along the path, the optimal decision rule is to send the message to the neighbor from which the message will reach the target agent which offers the desired service in the smallest number of steps, assuming that all future agents will make their decision using the same algorithm and only considering local information related to its neighbors.

If a decentralized search is to succeed, an important point to consider is that the underlying network possess some form of structure that can help to guide the search. There are two features that structure the preferential attachment network: *degree* (it is an intrinsic property of preferential attachment networks) and *homophily*. In the next subsections, the main concepts and components of the service discovery system are described.

#### 4.1 Modeling Agent Homophily

*Homophily* is a compact word that expresses the idea that a contact between similar people occurs at a higher rate than among dissimilar people[16]. This is often say with the expression '*Birds of a feather flock together*' - that you tend to be friend, talk to, work with and share ideas with people who share with you a common ethnic, religious and economic background. This word was used by Lazarsfeld and Merton in 1954 in an essay titled '*Friendship as a Social Process*'[13]. Empirical work related to *homophily* within social networks shows that is one of the most robust and pronounced characteristics of social networks. There are two types of homophily: status homophily, '*individuals are considered similar to one to another on the basis of informal, formal or ascribed status*', and value homophily, '*individuals are considered similar to one another on the basis of shared values, attitudes, and believes*'[3].

There is no global and application independent law on how *homophily* is measured. It is difficult to select an appropriate measure for a particular application area and to compare the existing homophily measures. Although homophily measurement is not restricted to solve a particular task, most homophily measures have been developed for a specific purpose. In our network, the homophily between two agents is based on semantic information contained in the service descriptions.

Given the agents  $a_1$  and  $a_2$ , the homophily between them is calculated as follows. If we consider  $s_1$  and  $s_2$  as the services offered by the agents  $a_1$  and  $a_2$  respectively,

$$s_1 = \langle I_{s_1}, O_{s_1} \rangle \quad s_2 = \langle I_{s_2}, O_{s_2} \rangle \quad (3)$$

the homophily between  $a_1$  and  $a_2$  can be computed as:

$$homophily(a_1, a_2) = \alpha sim(I_{s_1}, I_{s_2}) + \beta sim(O_{s_1}, O_{s_2}) \quad (4)$$

in which  $\alpha + \beta = 1$ ,  $0 \leq \alpha \leq 1$ , and the values of  $\alpha$  and  $\beta$  depends of the number of inputs or outputs of the services. If the number of inputs is higher than the outputs, the value of  $\alpha$  will be higher than the  $\beta$ . The similarity function  $sim(X, Y)$ , where  $X$  and  $Y$  represent the input or output parameters of two services, means the degree that  $Y$  satisfies  $X$  and is defined as a bipartite matching problem for service inputs and outputs.

A matching of a bipartite graph  $G=(V, E)$  is a subgraph  $G'=(V, E')$ ,  $E' \subseteq E$ , such that no two edges  $e_1, e_2 \in E'$  share the same vertex. Given a bipartite graph  $G=(V_1 \cup V_2, E)$  and its matching  $G'$ , the matching is complete if and only if all vertices in  $V_1$  are matched.

Let consider  $S_{1out}$  and  $S_{2out}$  the set of concepts in services  $s_1$  and  $s_2$  respectively. Consider the graph  $G=(V_1 \cup V_2, E)$  where  $V_1 = S_{1out}$  and  $V_2 = S_{2out}$ . Consider two concepts  $o_i \in V_1$  and  $o_j \in V_2$ . We differentiate among the four degrees of match proposed

by Paolucci et al. [24]. We calculate the degree of match of this two concepts using a semantic similarity measure. With the value of this measure we decide the degree of match  $R$ .  $R$  can be one of these values: *Exact*, *Plugin*, *Subsume* or *Fail*. If  $R$  is one of these degrees, an edge is defined between  $(o_i, o_j)$ ,  $o_i \in V_1$   $o_j \in V_2$  in the graph and label it with a weight  $(\omega_{ij})$ . Once we have the weighted bipartite graph, we have to compute a complete matching of the bipartite graph such that the sum of weights of the edges in the matching,  $\sum \omega_{ij}$ , is minimized. For this task we use the Hungarian algorithm [19] which computes it in a polynomial time bound.

## 4.2 Agent Social Network

In this proposal, we use a preferential attachment network  $G = (V, E)$  which consists of a set of nodes  $V$  and a set of edges  $E$  between them. The set of nodes represent agents which offer semantic services. The edges represent a relationship between agents which provide similar services. This network possess some form of structure that can guide the search. Basically, the preferential attachment network has two features that create such structure [4]. The first is *homophily*: agents tend to be linked with other agents that have services with similar category. The second feature is *degree*: some agents have more neighbors than others and may act as hubs that connect agents with different service categories. The consideration of *homophily* favors the neighbors that offer a service more similar to the target service. Consideration of *degree* favors the neighbor with the highest degree.

We create an undirected network with a power-law degree distribution. Each agent in the network offers a semantic service and has defined two vectors: one with the service inputs ( $I$ 's) and the other with the service outputs ( $O$ 's). Each  $I/O$  is a semantic concept defined in an ontology. The link between two agents is established considering the ratio preference between agents  $a_1$  and  $a_2$  to the sum of preferences from  $a_1$  to all the agents in the network. To approximate the preference from  $a_1$  to all the agents in the network, using only local information, we use the degree of the agent  $a_1$  ( $k_{a_1}$ ) and the preference between the agent  $a_1$  and its neighbors.

$$q_{a_1, a_2} = f_{a_1, a_2} / k_{a_1} * \max(f_{a_1, neighbour}), \quad (5)$$

The preference between two agents  $a_1$  and  $a_2$  with services  $s_1$  and  $s_2$  respectively  $f_{a_1, a_2}$  is defined as follows:

$$f_{a_1, a_2} = (\max\{homophily(s_1, s_2), 0.01\})^r \quad (6)$$

where  $homophily(s_1, s_2)$  is the homophily function between the service offered by agent  $a_1$  and the service offered by agent  $a_2$ . The return value of these function is a real number which ranges in the interval  $[0..1]$  (1 if the service provided by agent  $a_1$  is equal to the service provided by agent  $a_2$ ). The  $r$  parameter is a homophily regulator. When  $r$  is zero, the graph shows no homophily, agents are not grouped by similar service categories. As  $r$  grows, links connect agents with more similar services. Basically  $r$  makes the network to show groups of agents (communities) with similar services [9].

### 4.3 Semantic Distributed Searching of Services

Preferential attachment networks grow according to a simple self-organizing process. These networks need efficient search algorithms in order to function well. Algorithms should rely on local information in order to avoid a dependence on a unique point of failure and to avoid the effects of the changes in the network structure. There are several algorithms proposed for decentralized search in networks. Some methods do not consider the special features for the corresponding network models such as breadth-first searching methods based on limited flooding or random walks [32]. By making use of special features of the system topologies the algorithms can be classified in three groups:

- *degree*: the degree-based search methods typically make the assumptions defined in [32]: (i) 'each node knows its own neighborhood network topology'; and (ii) 'each node can locate the target if and only if the target is within a certain range of its neighborhood'. Generally speaking, the algorithm navigates through the network selecting in each step the neighbor agent with highest degree. In case that all the neighbor agents have been visited, the algorithm selects one randomly.
- *similarity*: The algorithm basically navigates the network selecting in each step the neighbor agent which has the service more similar to the target service. If all the neighbor agents have been visited, the algorithm selects one randomly [32]. In our case, it navigates using semantic similarities among service description (or parameters) using formula 4 as similarity measure.
- *mixed*: the algorithm navigates through the network selecting the neighbor agent whose service is more similar to the target service. In case that the neighbor agent do not offer the information needed to calculate the similarity between services, the algorithm selects the next agent between its neighborhood considering the degree. In the case that similarity and degree values are available, both parameters can be used to calculate the next neighbor. If all the neighbor agents have been visited, the algorithm selects one randomly.

In the context presented in this paper, the selected algorithm to search in preferential attachment networks is the Expected-Value Navigation(EVN) described in[4] which is a *mixed algorithm*. To apply this algorithm in the agent network, it is necessary to consider degree and the *homophily* between agents which is based on semantic similarity between the semantic services provided by the agents (see formula 4). In our scenario we assume that if the agents do not share the same ontology a previous step of ontology alignment is done. With this information, we can estimate the probability that a link exists from one agent to another. This probability is calculated assuming that each link is placed independently of the others. For a link from agent  $a_1$  to agent  $a_2$  the probability  $p_{a_1 a_2}$  can be calculated as the inverse of  $q_{a_1 a_2}$ :

$$p_{a_1 a_2} = 1 - (1 - q_{a_1 a_2})^k \quad (7)$$

where  $q_{a_1 a_2}$  is the probability that the first link for  $a_1$  ends at  $a_2$  (see formula 5), and  $k$  is the degree of node  $a_2$ .

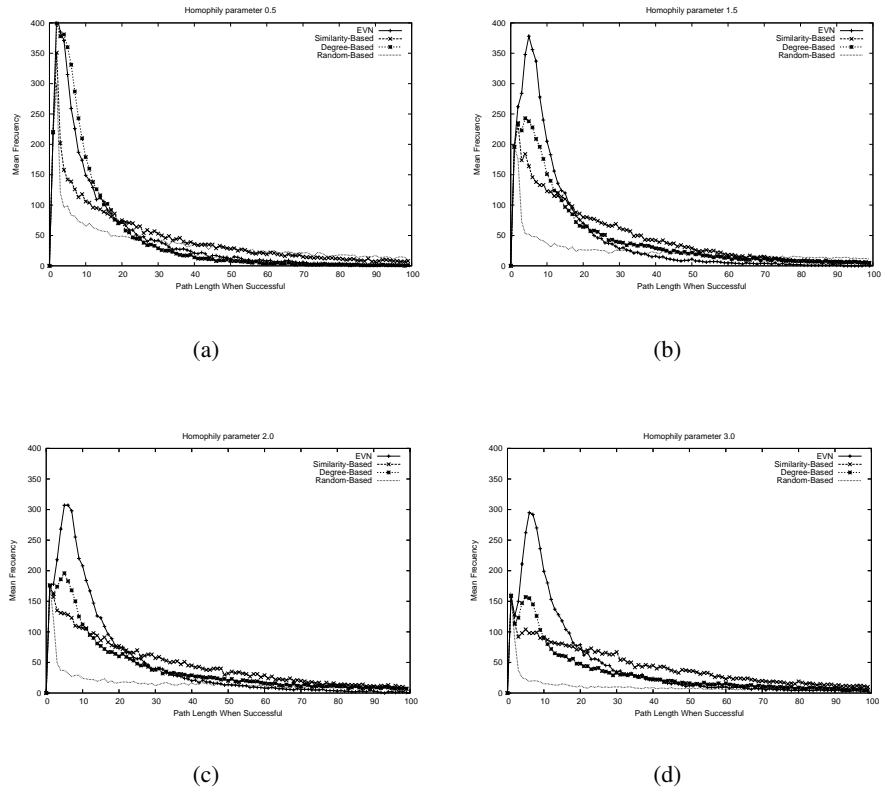


Fig. 1: Path Length When Successful

## 5 Experiments

### 5.1 Network Characterization

The experiments have been done in a set of synthetic networks. These networks are preferential attachment networks with the features explained in section 4.2. Each network is composed of 1000 agents with one semantic service each one. There are 100 service categories. The services have been assigned to the agents using a uniform distribution. We have created six sets of 10 random agent social networks. Each set of networks have been generated with different homophily parameter. This degree ranges from 0.5 to 3.0.

### 5.2 Experimental Results

In order to evaluate the proposed service discovery system in preferential attachment networks, we have analyze the behavior of the *EVN* algorithm with respect the other

distributed searching algorithms: *random*, *degree* and *similarity*. We have made 5000 searches in each of the previous networks.

In Figures 1 and 2 the data gathered from the previous described experiment is shown. In figure 1 we present the results obtained varying the homophily parameter from 0.5 to 3.0. Each figure indicates the frequency of path lengths for each distributed algorithm (EVN, degree, similarity and random). From these set of graphs we see that the EVN algorithm, in general has a better performance than the other algorithms independently of the homophily parameter. The EVN algorithm has the higher frequency of short paths (around 9 hops). In Figure 1a, the EVN have the same behavior than the degree-based algorithm. This is due to the homophily degree is too low, so the network does not show homophily and the EVN algorithm follows selects the neighbors only considering the degree. When the homophily parameter increases (Fig.1b,1c), the performance of the EVN becomes better than the other algorithms. This is because the EVN considers both parameters to guide the search: degree and homophily and can take more advantage of the network structure. The best performance of the EVN with respect the others is with the homophily parameter varying from 1.5 to 3.0 (Fig.1b - 1d).

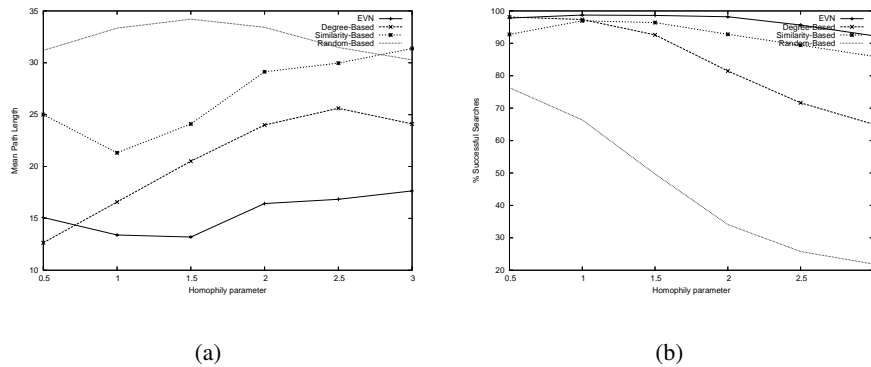


Fig. 2: Network with different homophily degrees

Figure 2a shows the mean path length obtained with each algorithm in networks with the homophily parameter varying from 0.5 to 3.0. In general, the EVN always return the shortest path except in graphs with a low value of the homophily parameter. This is because EVN takes more advantage of the network structure. In Figure 2b the success rate of each algorithm is shown. The EVN algorithm in the 90% of searches finds a path between the source agent to the agent that has the service that it was interested in.

The last and very important check is the behavior of the network under failures. The problem appears when a broken link splits the network into tow isolated parts, since some nodes will no longer be reachable. To analyze it, node failures have been

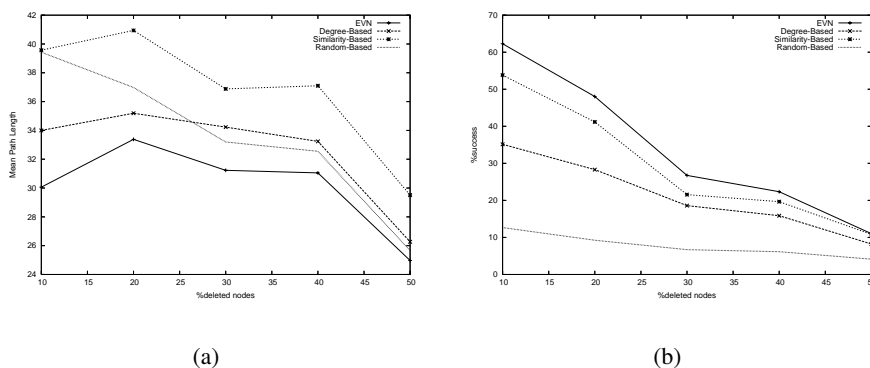


Fig. 3: Network with random failures

modelled as a failure of all its connexions. When some links are broken, an alternative path has to be found.

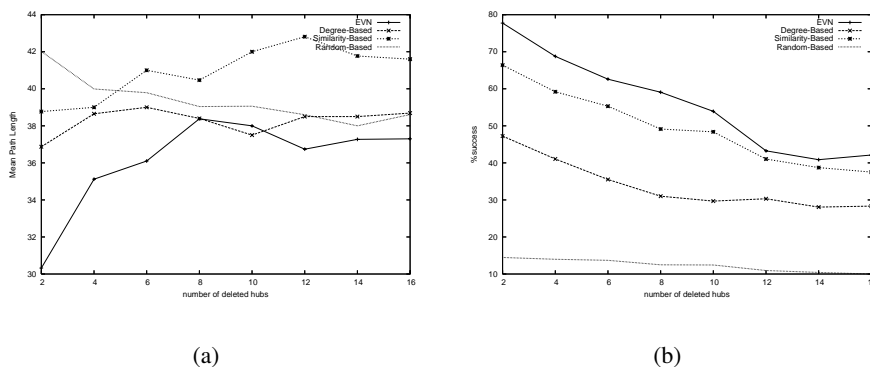


Fig. 4: Network under 'sabotage' conditions

For random failures (see Fig.3a and Fig.3b), it can be observed that the path length decreases when the number of deleted nodes increases. But this is a consequence of the failure in most of the searches when the percentage of deleted nodes approaches 30% because many searches cannot end successfully.

An interesting case is what happens when a deliberate failure is provoked. In the case of power-law networks, the worst case occurs when nodes with highest degree (hubs) are disconnected. Figure 4a and 4b shows how 'sabotage' affects the performance of the search process. In this case, the path length increases due to only a few



highly connected hubs have been deleted and an alternative path exists. The performance attending the number of successful searches decreases considerably as the number of deleted hub increases.

## 6 Conclusions and future work

The aim of this work is to investigate the use of social networks and distributed search algorithms to provide a fully distributed service discovery approach in Open MAS environment. Our proposal tries to overcome drawbacks present in other centralized (bottlenecks, complexity, huge amount of memory needed, global knowledge) and distributed (network traffic, congestion, coordination effort, data consistency between distributed registries, update data) discovery approaches. In our proposal, agents are situated in a social network with homophily factor. Each agent maintains the information about the current available services it offers. Agents in the network act as a 'matchmakers' and make use of a distributed search algorithm (EVN) that only makes use of local information to guide the search. The experimental results show that the EVN can be considered a good algorithm for service discovery domain.

As a future work we consider how does the problem of service discovery changes when the network evolves over time and what happen when agents do not follow a fixed algorithm. Furthermore, we will consider organizational information in the discovery process to guide the search.

## Acknowledgment

This work is supported by TIN2009-13839-C03-01 and TIN2008-04446 projects, CONSOLIDER-INGENIO 2010 under grant CSD2007-00022, FPU grant AP-2008-00601 awarded to E. del Val.

## References

1. L. A. Adamic and E. Adar. How to search a social network, 2004.
2. A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, October 1999.
3. B. Chaudhry, C. Marton, and H. Shepherd. Homophily and structure in multiplex networks.
4. O. Şimşek and Jensen. Navigating networks by using homophily and degree. *Proceedings of the National Academy of Sciences*, 2008.
5. J. Dang and M. Hungs. *Concurrent Multiple-Issue Negotiation for Internet-Based Services*. Number Vol.10 - 6. 2006.
6. S. Goyal and J. Moraga. R&d networks. Econometric Institute Report 202, Erasmus University Rotterdam, Econometric Institute, 2000.
7. H. Holzer. Search method used by unemployed youth. *Journal of Labor Economics*, 6:1–20, 1988.
8. S. Jha, P. Chalasani, O. Shehory, and K. Sycara. A formal treatment of distributed matchmaking. In *Proc. of the 2nd Int. Conference on Autonomous Agents*, number Vol.3, pages 457–458, 1998.

9. J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, pages 163–170, 2000.
10. J. M. Kleinberg. Navigation in a small world. *Nature*, 406(6798), 2000.
11. M. Klusch, B. Fries, and K. Sycara. Automated semantic web service discovery with owls-mx. In *Proceedings of 5th AAMAS*, Hakodate, Japan, 2006.
12. R. E. Kranton and D. F. Minehart. A theory of buyer-seller networks. *American Economic Review*, 91(3):485–508, 2001.
13. P. Lazarsfeld. Friendship as a social process: A substantive and methodological analysis. *Freedom and Control in Modern Society*, 1954.
14. D. Liben-Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins. Geographic routing in social networks. *Proceedings of the National Academy of Sciences of the USA*, 102(33):11623–11628, 2005.
15. M. Luck, P. McBurney, O. Shehory, and S. Willmott. *Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing)*. AgentLink, 2005.
16. M. Mcpherson, L. S. Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001.
17. I. Mecer. Agent-oriented semantic discovery and matchmaking of web services. In *8th International Conference on Telecommunications*, pages 603–607, 2005.
18. M. Moore and T. Suda. A decentralized and self-organizing discovery mechanism. In *Proc. Of the First Annual Symposium on Autonomous Intelligent Networks and Systems*, 2002.
19. K. Nedas. Implementation of munkres-kuhn (hungarian) algorithm, 2005.
20. M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.
21. <http://en.wikipedia.org/wiki/Scale-free-network>.
22. E. Ogston and S. Vassiliadis. Local distributed agent matchmaking. In *Proceedings of the 9th International Conference on Cooperative Information Systems*, 2001.
23. A. Ouksel, Y. Babad, and T. Tesch. Matchmaking software agents in b2b markets. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04)*, 2004.
24. M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara. Semantic matching of web services capabilities, 2002.
25. K. Sigdel, K. Bertels, B. Pourebrahimi, S. Vassiliadis, and L. Shuai. A framework for adaptive matchmaking in distributed computing. In *In proceeding of GRID Workshop*, 2005.
26. K. Sycara and M. Klusch. Brokering and matchmaking for coordination of agent societies: A survey. *Coordination of Internet Agents: Models, Technologies and Applications*, pages 197–224, 2001.
27. J. Travers and S. Milgram. An experimental study of the small world problem. *Sociometry*, 32(4):425–443, 1969.
28. E. Del Val and M. Rebollo. A survey on web service discovering and composition. In *Webist*, volume I, pages 135–142, 2008.
29. D. J. Watts. The new science of networks. *Annual Review of Sociology*, 30, 2004.
30. D. J. Watts, P. S. Dodds, and M. E. J. Newman. Identity and search in social networks, May 2002.
31. H. P. Thadakamalla, R. Albert, S. R. TKumara. Search in spatial scale-free networks *New Journal of Physics*, Volume 9, Issue 6, pp. 190 (2007).
32. S. Xiao and G. Xiao. On degree-based decentralized search in complex networks. *CoRR*, abs/cs/0610173, 2006.

# Goal-Directed Approach for Process Specification and Service Selection in Customer Life Cycle Management

Kumari Wickramasinghe<sup>1</sup>, Michael Georgeff<sup>1</sup>, Heinz Schmidt<sup>2</sup>, Ian Thomas<sup>1</sup>, and Christian Guttman<sup>1</sup>

<sup>1</sup> Department of General Practice

Faculty of Medicine, Nursing and Health Sciences

Monash University, Melbourne, Australia

{leelani.wickramasinghe|ian.thomas|michael.georgeff  
|christian.guttman}@med.monash.edu.au

<sup>2</sup> School of Computer Science and Information Technology

RMIT University, Melbourne, Australia

Heinz.Schmidt@rmit.edu.au

**Abstract.** Service selection is the first step in customer life cycle management where services are selected to meet a customer's goals or objectives, personalised to the circumstances of the customer. The aim of this paper is twofold: (1) to develop concepts and algorithms for goal-directed service selection; and (2) to compare and reconcile our goal-directed approach with a service-oriented approach. The proposed goal-directed service selection algorithm is based on a goal-directed domain description that represents the customer objectives and the business processes. We use service component architectures with formalised contractual service process definitions as a software engineering approach to architectural design and realisation of service-oriented architectures (SOA). The comparison aims to understand the relationship between and benefits of a goal-directed approach and a service oriented approach . We use case studies from two complex customer care management domains to demonstrate the concepts. The implemented algorithms are tested in a health care case study.

## 1 Introduction

This paper is concerned with the problem of selecting an appropriate set of services to meet a customer's goals or needs, personalised to the circumstances of that customer. For example, a customer of a telecommunications company may have the objective of getting a range of communication services. The services that best meet the customer's needs will depend on the location of the customer, with whom the customer needs to communicate, the communication media, and the size of the customer's business. Similarly, in a quite different setting, a patient with chronic disease needs a care team (provider services) to help manage the disease, and the composition of that team will depend on a range of conditions, such as the particular disease or diseases the patient has, the patient's age, health status, family history, and so on.

We are interested in developing techniques for automating the selection of such services, including both their representation and the process of service selection. Automation requires formalisation. The focus of this paper is to consider two approaches:

(1) goal-directed, and (2) service-oriented. The former is formalised using multi-agent systems (MAS), in particular, goal-directed agents [1]. For the latter we use service-oriented architectures, in particular service-component architectures [2] with formalised, contractually agreed service coordination [3].

There has been considerable literature about service composition (e.g., web service composition [4], flexible Business Process Management [5, 6, 3]) and goal-directed processing [7, 8], and there have been some attempts to explicate the relationship between services in an SOA and goals in an agent framework [9]. However, such research deals primarily with run-time selection and orchestration of services. In this paper, we are interested in the static (prior to run-time) generation of a set of interoperable services that are capable of meeting a customer's requirements. In this sense, the approach is similar to program synthesis, at the one extreme, and mash-ups, at the other.

The work presented in this paper is carried out as part of the Intelligent Collaborative Care Management (ICCM) Project<sup>3</sup>[10], which investigates a comprehensive architecture for managing the complete life cycle of customer care. The ICCM architectural components include: (1) automatic selection of a number of (possibly interrelated) services to meet a customer's goals or objectives; (2) allocation of service providers to deliver the selected services; (3) maintenance of contractual relationships among the service providers; and (4) the delivery of services by the service providers in an agreed manner over time and potentially the entire lifetime of the customer. Among these, an investigation of the Item (1) is the objective of this paper (refer [11] for the other components of ICCM).

The customer-centric automatic service selection has two key elements: (1) a domain representation mechanism; and (2) service selection algorithms. The domain representation includes the specification of the actors (e.g., customers and service providers), their properties, services (a unit of independent third-party deployment that encapsulates internal data and state [12]) and the business processes which coordinate such services. In a tightly coupled world, coordination is typically simple, monolithic, deterministic, well understood and with few control paths. Here it is possible to specify all the business processes that determine the flow between services considering all possible customer properties. However, this approach is impractical in widely distributed, loosely coupled systems using different technologies implemented in different programming languages on a wide range of platforms. The determination of the exact process (static flow) at the design time is challenged (1) in dynamic environments where customer objectives and business processes change; and (2) when different business units interact with one another to realise customer objectives. To address these challenges we propose:

1. A goal-directed domain representation called a coordination-level specification ;

---

<sup>3</sup> The work reported here was supported in part by British Telecom (CT1080050530), the Australian Research Council (LP0774944), the Australian Governments Clever Networks program and the Victorian Department of Innovation, Industry and Regional Development, Department of Health, and Multi Media Victoria. We also gratefully acknowledge the contributions and advice from Dr Simon Thompson and Dr Hamid Gharib of British Telecom and Professor Leon Piterman, Dr Kay Jones, Associate Professor Peter Schattner, and Mr Akuh Adaji of the Department of General Practice, Monash University.

2. A goal-directed service selection algorithm based on the Item 1 above;
3. A goal-directed extension to a component-based service selection technique (from main stream SE) based on the goal-directed specification in Item 1 above; and
4. A comparison between component-based and goal-directed service selection mechanisms highlighting the benefit each approach can obtain from the other.

The paper is organized as follows. Case studies of customer care management are presented in Section 2. Service selection is described in Sections 3 (goal-directed) and Section 4 (service-oriented). Section 5 relates our work to existing research. Concluding remarks and future work are discussed in Section 6.

## **2 Case Studies of Customer Care Management**

This section describes case studies of three complex customer care management domains, health care and telecommunication, which can benefit from the proposed service selection strategies. In the subsequent sections, examples are mainly obtained from health care case study.

### **2.1 Health Care: Chronic Disease Management**

The current practice for caring for patients with chronic diseases involves the development and management of personalised care plans for these patients. A care plan includes tests, medications and treatments (services) together with the details of health care providers who provide each service. In ICCM project terms, patients are the customers and the health care providers, such as general practitioners, podiatrists, optometrists and dietitians are the service providers.

Once a chronically ill patient is identified, a General Practitioner (GP) creates a care plan using chronic disease care plan templates. While related to service selection, these templates only provide a guideline, requiring manual personalization for patient properties that demands much of GP's or practice nurse's time. We expect to minimize GP's workload by generating personalized care plans using automated context-sensitive service selection and service compatibility checking.

### **2.2 Telecommunication Domain: Customer Life Cycle Management (CLCM)**

Telecommunication is a competitive domain with many telecommunication companies providing similar services. The aim of Customer Life Cycle Management (CLCM) is to move a potential customer through several stages until he/she eventually becomes loyal, without losing them on the way to abandonment, attrition, churn, and migration [13]. Cutler and Sterne state that a customer moves through five stages: reach, acquisition, conversion, retention and loyalty [13]. In each stage, the customer's objectives may change, which demands personalised service selection to retain their customer. In addition, in telecommunication companies, different business units such as Internet and Voice provide different services. Usually a customer objective crosses among these units, requiring collaboration among processes from different business units.

### 3 Service Selection: Goal-Directed Approach

This section investigates a goal-directed service selection strategy based on goal-directed MAS (Section 3.1). In this approach, business processes and the customer objectives are specified as goals in the coordination-level specification (Section 3.2).

#### 3.1 Goal-Directed Coordination-Level Specification

The proposed coordination-level specification contains four distinct specifications:

**Environment specification** defines elements, artifacts or constructs of the domain;  
**Goal specification** defines the goals or objectives of the customer;  
**Process specification** defines the business processes available in the domain; and  
**Constraint specification** defines how different services may interact with one another.

**Environment Specification:** The entities (which are applicable to customer care management) and their properties are defined in the *environment specification*. The compulsory entities include the customer and service providers. Minimally, the environment specification contains the properties of the customer and the service providers. Formally, the environment specification  $D$  consists of a set of attributes and function definitions to describe each entity. For example, the patient entity from health care case study contains the attributes: height, weight, history, isSmoker, numberOfSmokesPerDay and functional definitions:  $BMI(patient) = weight/height * height$  and  $IsOverweight(patient) \text{ iff } (BMI(patient) > 25)$ . It includes only the definitions of attributes and functional properties, not instances of the entities. The entities are instantiated in the selection stage (Section 3.2) by assigning values to the definitions. The environment specification can take the form of any knowledge representation technique: lists, trees, semantic networks, schema, rule-based or logic-based representations.

**Goal Specification:** Goal specification,  $G$  defines the business objective that the company is trying to achieve or the objective of the customer or the desired outcome at the end of service selection. Using the goal-directed terminology such an objective is termed a goal. For example, patient's goal can be to "Manage Diabetes" or "Manage Weight" while the business objective is to provide "Manage Diabetes Service" or "Manage Weight Service". Goals are defined as predicates  $G(x)$  or  $G(x, y)$  with subject  $x$  and optional object  $y$ . For example,  $ManageNutrition(?SomePatient)$  means  $SomePatient$  does  $ManagesNutrition$ ;  $ObtainObesityEducation(SomePatient, SomeEducator)$  means that the  $SomePatient$  obtains obesity education from an educator.

We categorize two types of goals: (1) *service goals*; and (2) *abstract goals*. Service goals can be directly delivered by service providers. Service goals also map generally into service operations in SOA, the provision of a service for another entity. The abstract goals may not have specific concrete realizations as service provider operations, but provide ordering constraints on the use of service goals. Hence abstract goals form a management or orchestration layer over the main service goals. (For clarity, examples of goals are provided after the process specification is described).

Formally,  $SG$  is a set of service goals  $SG = \{sg_1, \dots, sg_n\}$  and  $AG$  is a set of abstract goals  $AG = \{ag_1, \dots, ag_m\}$  and a set of goals:  $G = SG \cup AG$  with  $SG \cap AG = \emptyset$ . If  $SPT = \{spt_1, \dots, spt_l\}$  denote the service provider types, then

the mapping of service goals to provider types that can deliver the goals is a simple many-to-many relation:  $H \subseteq 2^G \times 2^{SPT}$ .

**Process Specification:** The process specification defines the manner of achieving the goals specified in the goal specification. The proposed process specification takes into account: (1) processes may come from different business units; and (2) different business units utilize such processes as part of their collaborations among services.

Analogous to business process design methodologies (e.g., Six Sigma [14]), process specification establishes goals and decomposes them into sub-goals. It defines the process in terms of the sub-goals that need to be achieved to ensure a successful outcome of a goal. This goal-directed process definition allows loose coupling to exist at the business process level opposed to the conventional approaches that use explicit links to specific sub-processes (tightly coupled business processes).

Service goals are not subdivided further into sub-goals (they are considered primitive). Abstract goals are refined recursively until service goals are reached. Moreover, conditional goal refinement and service selection is supported. For example, the goal Manage Life Style may have three sub-goals: Manage Nutrition, Manage Weight and Cease Smoking. However, the Manage Weight goal is applicable only to overweight patients and Cease Smoking goal is applicable only to smokers.

As a representation of this refinement, we use the notion of an *operator*:  $op \subseteq AG \times BoolExp \rightarrow 2^G$ , which associates an abstract goal  $g \in AG$  with one or more conditions  $C$  and for each of these with a unique refinement into a set of subgoals  $s = \{g_1, \dots, g_m\}$ . We represent an operator instance as a triple  $\langle g, C, s \rangle$ .  $C$  is a logical expression over the environment specification.  $s$  is also called the *body* of  $op$ . Note that the order of subgoals is not formalised here. Ordering constraints are modeled outside of operators.

The following example illustrates two operators with their goals (from the case study):

**Operator 0**

Goal: ManageDiabetes(?SomePatient)  
 Context: Patient(?SomePatient) and Disease(?SomePatient, Diabetes)  
 Body: ManageLifestyle(?SomePatient)  
       ReduceRiskOfComplications(?SomePatient)

**Operator 1**

Goal: ManageLifestyle(?SomePatient)  
 Context: Patient(?SomePatient) and History(?SomePatient, Diabetes)  
 Body: ManageNutrition(?SomePatient)  
       If Overweight(?SomePatient) then ManageWeight(?SomePatient)  
       If Smoker (?SomePatient) then CeaseSmoking(?SomePatient)

**goal:** ManageDiabetes has two **sub-goals:** (1) ManageLifestyle (2) ReduceRiskOfComplications

**goal:** ManageLifestyle has three **sub-goals:** (1) ManageNutrition (2) ManageWeight (3) CeaseSmoking

ManageWeight is selected if the patient is overweight and CeaseSmoking is selected if the patient is a smoker.

$AG = \{ManageLifestyle, ReduceRiskOfComplications, ManageNutrition, ManageWeight, CeaseSmoking, SelfManageWeight\}$

$SG = \{ObtainObesityEducation, MonitorWeight, AttendWeightManagementProgram\}$

**Constraint Specification:** Constraint specification includes constraints over goals, such as ordering constraints or dependencies associated with abstract goals. A constraint can specify which goals must appear together, or must not appear together, ordering or timing constraints or a restriction on goals that cannot be selected together. We consider two types of constraints:

1. Exclusion constraints: Defines two or more goals which cannot be selected together (mutually exclusive goals for service selection). For example, if goal  $g_i$  is chosen for selection, then goal  $g_j$  can not be executed, denoted by the property :  $P(g_i \cap g_j) = 0$ .

2. Binding constraints: Defines two or more goals that must be selected together (mutually exclusive goals for service selection). For example, if goal  $g_i$  is selected, then goal  $g_j$  must be selected, too: denoted by  $g_i \rightarrow g_j$ . Bindings may specify any ordering or concurrency or services (e.g., if two services are executed in parallel or sequential).

The use of coordination-level specification in goal-directed service selection is described next.

### 3.2 Goal-Directed Service Selection Algorithm

The ICCM service selection stage generates a care plan by selecting services from the process specification to realise a customer objective. A customer instance is created by assigning values to the customer attributes and the functional properties defined in the environment specification, called the environment specification. Similarly, the goal specification is assigned with the customer objective and termed a goal model.

Each operator in the process specification encapsulates a goal  $g_i : g_i \in G$  and a set of sub-goals to achieve  $g_i$ . With the notion of goals and sub-goals, operator specification depicts a tree structure, which we term a *goal tree*. A service goal  $sg_i : sg_i \in SG$  and  $SG \subset G$  corresponds to a leaf node in the goal tree. The service selection aims to identify all the service goals required to realise a customer as the service goals corresponds to services in SOA which are provided by service providers.

**Goal-directed Algorithm:** The service selection algorithm: (1) traverses the goal tree; (2) starts with the node that corresponds to the goal model; and (3) finishes when all the relevant service goals are reached. This traversal depicts a branching tree structure which selects an operators if its context matches with that map to the environment model. A basic version of the service selection algorithm is:

```

SelectServices(customerObjective)
  let careplan = {} and let subGoals = {}
  subGoals = GetSubGoals(customerObjective)
  while (subGoals ≠ {} )
  for each  $g_i \in subGoals$ 
    subGoals = subGoals -  $g_i$ 
    if  $x \in SG$ 
      careplan = careplan +  $g_i$ 
    otherwise
      subGoals = subGoals + GetSubGoals( $g_i$ )
  return careplan

```



The *SelectServices* function gets invoked with the goal model as the customerObjective parameter. The *GetSubGoals* function returns the immediate child node(s) of a given goal from the goal tree.

For example (from the Process Specification in Section 3),

let *customerObjective* = *ManageDiabetes*

In the first iteration:

*subGoals* = {*ManageLifestyle, ReduceRiskOfComplications*}

where  $g_1 = \text{ManageLifestyle}$  and  $g_2 = \text{ReduceRiskOfComplications}$

At the end of first iteration:

*careplan* = {} (Since  $(g_1, g_2) \notin SG$ )

if the patient is overweight and a smoker:

*subGoals* = {*ManageWeight, CeaseSmoking, ReduceRiskOfComplications*}

if the patient is only overweight:

*subGoals* = {*ManageWeight, ReduceRiskOfComplications*}

and so on

The second iteration starts with the subGoals resulted at the end of the first iteration. The iterations continue until the careplan variable is populated with all the relevant service goals.

The service selection algorithm depicts a recursive nature: (1) due to the definition of goals and sub-goals in the operator specification; and (2) to get a set of service goals as the output of the service selection stage. The statement  $subGoals := subGoals + GetSubGoals(g_i)$  handles the recursive behaviour.

If there are multiple operators with matching context (OR tree), each such operator produces a distinct care plan. An extended algorithm, *Generate* to generate multiple care plans is given below. The *Generate* function invokes the *Combine* function to allocate service goals to the appropriate care plan.

*Generate(goal)*

*matchingOperator* = *Match(goal)*

for each  $g_i \in matchingOperator$

*finalServices* = {}

*subgoal* = *GetSubgoal(g<sub>i</sub>)*

if *subgoal* = {}

let  $(g_i \subset finalServices)$

otherwise

for each  $sg_j \in subgoal$

$z = Generate(sg_j)$

*finalServices* = *Combine(z, finalServices)*

return *finalServices*

*Combine(z, finalServices)*

*newFinalServices* = {}

for each  $x_i \subset z$

let  $((x_j \cup finalServices) \subset newFinalServices)$

return *newFinalServices*

The *Match* function returns a set of operators that match the goal and the context. The *Generate* function returns a set of sets of service goals, each set corresponds to a distinct care plan.

Next we describe a service-oriented service selection technique.

## 4 Service selection: Service-Oriented Approach

Component-based architecture focuses on the architectural design of systems with reusable components, interfaces, ports and connections between them, and the building of composite services [15]. Architectural design has demonstrated success in reducing software development and maintenance costs through *reuse*, *product lines* and architecture-based *quality assurance* for *dependable and trustworthy systems* [16]. While components and services are not the same, SOA is a specific style of component-based architecture [17]. Architecture definition languages for component-based architecture such as AADL and the UML2 are very similar in notation and modeling to those of SOA, such as the open-source Apache Tuscany *Service Component Architecture (SCA)* [2] now widely used in practice. Both aim at the separation of frameworks and components, the independent incremental development of components/services in different languages, their deployment into live systems on different platforms without loss of interoperability and dependability.

The approach of contract-based design which underlies our work in this and other projects [18] carries across to SOA [3]. Service providers and service coordination are specified with precise contractual obligations, distinguishing service requirements and conditional guarantees in service (composition/coordination) constraints. These are modeled by process languages defining service invocation constraints, typically modelled by process expressions or automata models. This formalization of interface behaviour is used by us in user-guided plan design and automatic selection algorithms.

### 4.1 Service Model

The service model associates the goals from the goal specification with a set of services to achieve such goals. These services are formalisations of the real-world service providers such as podiatrists, GPs etc. or in the case of abstract goals represent service components serving coordination. In service architecture terms, we distinguish *service provider types (SPT)* from *service instances*. A SPT is associated to other SPTs through interface descriptions. Our approach captures dependencies and constraints between service types via so-called *gates*. Beside a port name and an interface definition, a gate includes a *contractual service specification* possibly including service quality contracts. Independent of the specific service providers (the instances) selected at run-time, our approach designs and checks the *typical* coordination and interaction between these service types before run-time. When a service instance is selected at run-time, this only necessitates type checking.

As usual in component-based architecture and service component architecture languages such as SCA we distinguish *provided and required* interfaces, and here, gates. Unlike ports, connections to gates are only possible if the contracts are met. This allows us to separate contractual assumptions from guarantees. A service provider entity

may have multiple service operations that it can publish at a provided or required gate. Via gates, services can be connected, either by binary connectors that are linking up a required gate with a conforming provided gate, or by special coordinators that include architectural coordination constraints over several gates. Coordinators are modelled themselves like (light-weight) components. We use the Rich Architectural Description Language (RADL) [19] to formally represent SPTs and gate constraints. For example, from the health care case study, the goal “ManageDiabetes” is now termed a SPT description and the sub-goals: “ManageLifestyle” and “ReduceRiskOfComplications” become the required services of the gate. Thus we ‘componentise’, i.e. define an architecture, a structure in-the-large, over the otherwise unstructured collection of goals and operators in the goal-oriented approach.

#### 4.2 Process Model

While RADL gate processes are described by (possibly concurrent) automata, RADL features so-called abstract machines associated with software or service components. Like many other architecture description languages[20], RADL uses gate processes for conformance checks when components are selected or composed in a context. The abstract machines expose an abstract translation from service abstractions invoked via a provided gate to services called out to via required gates. They capture dependencies between gates realised by component implementations without revealing the implementations and are used for refined dependency analysis, testing and performance prediction across networks of mutually dependent service components.

Finally, this architecture provides structure to the set of services available to be deployed to fulfill the customer objectives. This structure forms a manifest of all available services. A sample structure extract is shown in Figure 2.

#### 4.3 Service selection: Goal-Directed vs SOA

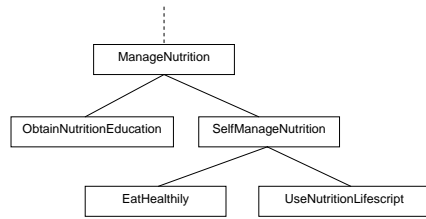
The service-oriented realisation of service selection, checking and composition mirrors the goal-directed approach described (Section 4.1) as follows:

1. each goal is translated to an SPT description;
2. each sub-goal is mapped to a port in the service-oriented architecture;
3. the sub-goal is then translated to a gate by enriching the port with the appropriate coordination process;
4. operators are identified in the SPT structure.

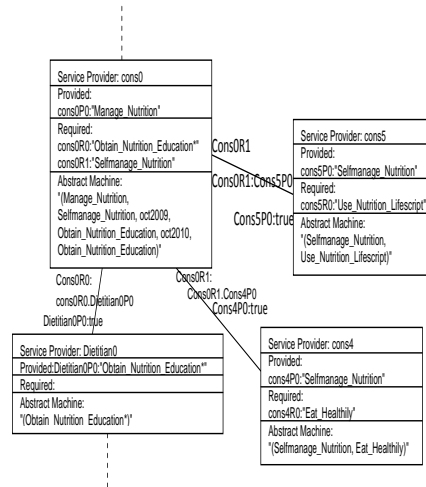
By construction we represent the same hierarchical structure for goals and service types (Figures 1 and 2).

The concept of a goal is a semantically rich concept. The goal-directed approach:

1. has inherent verification support to check the correctness of the goal tree;
2. has inbuilt goal tree maintenance support;
3. captures the knowledge-based semantics of goals in the operator specification;
4. in addition to the task-based service selection (e.g., manageWeight), it allows service selection based on:



**Fig. 1.** A sample goal tree



**Fig. 2.** SPT structure corresponds to the goal tree in Figure 1

- (a) the desired outcome (e.g., the patient wants to manage weight); and
  - (b) maintenance goals (e.g., maintain the patient’s weight gain less that 3% of the service selection-time body weight);
5. supports a distributed design environment through partial plans.
  6. supports the use of automatic goal planners.

On the other hand, formal approaches to SOA allow complex data and process types and contractual constraints to be specified. Future work is directed towards developing representation techniques for rich constraint specification in the goal-directed approach. This will require reconciling goal search with architecture-based process definition.

## 5 Related Research

A service, which is an implementation of a well-defined business functionality with a well-defined interface is the building block of SOA [12]. A web service describes a standard way to integrate web-based applications by programmatically providing business logic, data and processes. Web services are considered as the preferred way to realize SOA [21]. Web Service composition strategies are broadly categorized into 5 categories: static vs dynamic, model-driven service composition, declarative service composition, automated vs manual service composition, and Context-based service discovery and composition [4]. ICCM service selection strategies illustrate a combination of dynamic, model-driven, automated and context-based approach.

This paper investigates the applicability of the goal-directed approaches for service selection which occur at static-time. The application of agent-based goal-directed approaches for managing the run-time behaviour of systems is a research area in its own

right with already proven success [7, 8]. Already there are initiatives for context sensitive goal-directed service selection. For example, IBM has identified the importance of flexible business process management plans to obtain SOA objective to create new solutions by composing existing business services to handle dynamic business requirements [5]. To address such requirements IBM introduced the BPM Suite “WebSphere Business Services Fabric” which has shareable, context sensitive flexible modules called intelligent business services [6]. However, these intelligent business services are not formalized in MAS terms and, lack both the expression power of MAS and any formal basis.

The semantics of goal-directed BDI agents and formalisms to represent such semantics are well researched and mature [22]. The agent architectures (e.g., PRS [23], dMARS [24]) have mechanisms to represent such formalisms. There are programming languages (e.g., AgentSpeak [25]) based on BDI architectures and agent platforms (e.g., Jason [26]) to develop BDI agents. To our knowledge there is no research on goal-directed service selection using goal-directed agent semantics, formalisms and implementation tools, although it clearly relates to work in hierarchical planning [27].

## 6 Conclusions and Future Work

This paper investigated service selection strategies to meet a customer’s goals or objectives, personalised to the circumstances of that customer. It proposed a goal-directed domain representation technique, service selection algorithm, and extension to a service-oriented service selection algorithm. The comparison among the goal-directed and service-oriented approaches concludes that both the approaches can generate the same outputs and add value to the other. SOA can provide rich techniques to apply complex constraints over goals, while the goal-directed approach can provide knowledge-based semantics of goals and a distributed design environment.

Currently our goal-directed approach does not have a mechanism to represent rich constraints among goals. As future work we expect to enhance the goal-directed approach to represent process logical constraints, further integrate BDI agent-based goal-oriented selection with SOA, and implement the goal-directed approach in other domains, such as telecommunication.

## References

1. Rao, A.S., Georgeff, M.P.: Modeling rational agents within a BDI-architecture. In Allen, J., Fikes, R., Sandewall, E., eds.: Principles of Knowledge Representation and Reasoning(KR). Morgan Kaufmann, San Mateo, California, United States of America (USA) (1991) 473–484
2. Mahbod, H., Feng, R., Laws, S.: Building SOA with Tuscany SCA. Java Developer Journal (11 2007)
3. Ling, S., Poernomo, I., Schmidt, H.W.: Describing web service architectures through design-by-contract. In: Proc. 18<sup>th</sup> Intl. Symp. on Computer and Information Sciences (ISCIS’03). Volume 2869 of LNCS., Springer-Verlag, Berlin (2003) 1008–1018
4. Dustdar, S., Schreiner, W.: A survey on web services composition. International Journal of Web and Grid Services 1(1) (2005) 1–30

5. Fiammante, M.: *Dynamic SOA and BPM: Best Practices for Business Process Management and SOA Agility*. IBM Press (2009)
6. IBM: *Websphere business services fabric* (2009)
7. Burmeister, B., Arnold, M., Copaciu, F., Rimassa, G.: BDI-agents for agile goal-oriented business processes. In: Proc. 7<sup>th</sup> Intl. Joint Conf. on Autonomous Agents and Multiagent Systems: industrial track, Intl.Foundation for Autonomous Agents and Multiagent Systems (2008) 37–44
8. Ingrand, F., Georgeff, M., Rao, A.: An architecture for real-time reasoning and system control. *IEEE Expert* 7(6) (1992) 34–44
9. Georgeff, M.: *Service orchestration: The next big challenge*. DM Review Special Report (2006) 1056195–1
10. Wickramasinghe, K., Guttman, C., Georgeff, M., Gharib, H., Thomas, I., Thompson, S., Schmidt, H.: Agent-based intelligent collaborative care management. In: Proc. AAMAS-Volume 2, IFAAMS (2009) 1387–1388
11. Guttman, C., Thomas, I., Georgeff, M., Wickramasinghe, K., Gharib, H., Thompson, S., Schmidt, H.: Towards an intelligent agent framework to manage and coordinate collaborative care. In: Proc. First Workshop on Collaborative Agents – REsearch and development (CARE 2009). LNCS, Springer-Verlag, Berlin (accepted in 2009, to appear in 2010)
12. Erl, T.: *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall (2005)
13. Cutler, M., Sterne, J.: *E-Metrics: Business Metrics For The New Economy*, Cambridge (MA): NetGenesis Corp. 2000. Online im Internet 12 (2000)
14. Tennant, G.: *Six Sigma: SPC and TQM in manufacturing and services*. Gower (2001)
15. Szyperski, C.: *Components and architecture*. Software Development (2000)
16. Schmidt, H.W.: Trustworthy components: Compositionality and prediction. *Journal of Systems and Software: Component-Based Software Engineering* 65(3) (2003) 215–225
17. Krämer, B.: Component meets service: what does the mongrel look like? *Innovations in Systems and Software Engineering* 4(4) (2008) 385–394
18. Krämer, B.J., Reussner, R.H., Schmidt, H.W.: Predicting properties of component based software architectures through parameterised contracts. In Wirsing, M., ed.: *Radical Innovations of Software and Systems Engineering*. LNCS, Springer-Verlag, Berlin (October 2002)
19. Schmidt, H., Krämer, B., Poernomo, I., Reussner, R.: Predictable Component Architectures Using Dependent Finite State Machines. In: *Radical Innovations of Software and Systems Engineering in the Future*. Volume 2941 of LNCS. Springer-Verlag, Berlin (March 2004) 310–324
20. Medvidovic, N., Taylor, R.: A classification and comparison framework for software architecture description languages. *IEEE TSE* 26(1) (2000) 70–93
21. Mahmoud, Q.: *Service-oriented architecture (SOA) and web services: The road to Enterprise Application Integration (EAI)*. Sun Microsystems, April (2005)
22. Rao, A., Georgeff, M.: BDI agents: From theory to practice. In: Proc. First Intl.Conf. on multi-agent systems, San Francisco, CA (1995) 312–319
23. Georgeff, M., Lansky, A.: Reactive reasoning and planning. In: Proc. 6<sup>th</sup> Natl. Conf. on Artificial Intelligence (AAAI-87), Seattle, WA (1987) 677–682
24. d’Inverno, M., Kinny, D., Luck, M., Wooldridge, M.: A formal specification of dMARS. *Lecture notes in computer science* 155–176
25. Rao, A.: Agentspeak(L): BDI agents speak out in a logical computable language. In: Proc. of MAAMAW 96. Volume 1038 of LNAI., London, Springer-Verlag, Berlin (1996) 42–55
26. Bordini, R., Huebner, J., Wooldridge, M.: *Programming Multi-Agent Systems in AgentSpeak using Jason*. Wiley New York (2006)
27. Wilkins, D.: *Hierarchical planning: Definition and implementation* (1985)