Ryszard Kowalczyk **Michael Huhns**
Zakaria Maamar **Quoc Bao Vo (Eds.)**

Service-Oriented Computing: Agents, Semantics, and Engineering

AAMAS 2009 International Workshop, SOCASE 2009
Budapest, Hungary, May 2009
Proceedings

Preface

The areas of service-oriented computing and semantic technology offer much of interest to the multiagent system community, including similarities in system architectures and provisioning processes, and powerful tools and standardizations to enable more flexible and dynamic business process integration and automation. Similarly, techniques developed in the multiagent systems and semantic technology areas are having a strong impact on the fast-growing service-oriented computing area. Other issues, such as quality of service, security, privacy, and reliability are common problems to both multiagent systems and service-oriented computing.

Service-oriented computing has emerged as an established paradigm for distributed computing and e-business processing. It utilizes services as fundamental building blocks to enable the development of agile networks of collaborating business applications distributed within and across organizational boundaries. Services are self-contained, platform-independent software components that can be described, published, discovered, orchestrated, and deployed for the purpose of developing distributed applications across large heterogeneous networks such as the Internet.

Multiagent systems, on the other hand, also aim at the development of distributed applications, however, from a different but complementary perspective. Service-oriented paradigms are mainly focused on syntactical and declarative definitions of software components, their interfaces, communication channels, and capabilities with the aim of creating interoperable and reliable infrastructures. In contrast, multiagent systems are focused on the development of reasoning and planning capabilities of autonomous problem solvers that actively apply behavioral concepts such as interaction, collaboration, and negotiation in order to create flexible and fault-tolerant distributed systems for dynamic and uncertain environments.

Semantic technology offers a semantic foundation for interactions among agents and services, forming the basis upon which machine-understandable service descriptions can be obtained, and as a result, autonomic coordination among agents is made possible. On the other hand, ontology-related technologies, ontology matching, learning, and automatic generation, etc., not only gain in potential power when used by agents, but also are meaningful only when adopted in real applications in areas such as service-oriented computing.

This volume consists of the proceedings of the Service-Oriented Computing: Agents, Semantics, and Engineering (SOCASE 2009) workshop held at the International Joint Conferences on Autonomous Agents and Multiagent Systems (AAMAS 2009). The papers in this volume cover a range of topics at the intersection of service-oriented computing, semantic technology, and intelligent multiagent systems, such as: service description and discovery; planning, composition and negotiation; semantic processes and service agents; and applications.

The workshop organizers would like to thank all members of the Program Committee for their excellent work, effort, and support in ensuring the high-quality program and successful outcome of the SOCASE 2009 workshop.

May 2009
Ryszard Kowalczyk
Michael Huhns
Zakaria Maamar
Quoc Bao Vo

Organization

SOCASE 2009 was held in conjunction with the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009) on May 11, 2009 in Budapest, Hungary.

Organizing Committee

Ryszard Kowalczyk, Swinburne University of Technology, Australia
Michael Huhns, University of South Carolina, USA
Zakaria Maamar, Zayed University Dubai, United Arab Emirates
Quoc Bao Vo, Swinburne University of Technology, Australia

Program Committee

Jamal Bentahar, Concordia University, Canada
M. Brian Blake, Georgetown University, USA
Athman Bouguettaya, CSIRO, Australia
Jakub Brzostowski, Silesian University of Technology, Poland
Paul Buhler, College of Charleston, USA
Mauro Gaspari, Universita' di Bologna, Italy
Christian Guttmann, Monash University, Australia
Slimane Hammoudi, ESEO, France
Jingshan Huang, Benedict College, USA
Clement Jonquet, Stanford University, USA
Luis Llana, Universidad Complutense de Madrid, Spain
Xuan Thang Nguyen, TIBRA, Australia
Manuel Nunez, Universidad Complutense de Madrid, Spain
Julian Padget, University of Bath, UK
Huaglory Tianfield, Glasgow Caledonian University, UK
Rainer Unland, University of Duisburg-Essen, Germany
Kunal Verma, Accenture, USA
Leandro Krug Wives, Federal University of Rio Grande do Sul, Brazil

Table of Contents

Contract Observation in Web Services Environments

Jiří Bíba, Jiří Hodík, Michal Jakob and Michal Pěchouček
Agent Technology Center, Dept. of Cybernetics, FEE, Czech Technical University
Technická 2, 16627 Prague 6, Czech Republic
{biba, hodik, jakob, pechoucek}@agents.felk.cvut.cz

ABSTRACT

Electronic contracting, based on explicit representation of different parties' commitments, is a promising way to specifying and regulating behaviour in distributed business applications. A key part of contract-based system is a process through which the actual behaviour of individual parties is checked for conformance with contracts set to govern such behaviour. Such checking requires that relevant information on the behaviour of the parties, both with respect to the application processes they execute and to managing their contractual relationships, is captured. The process of collecting all such information, termed *contract observation*, is the subject of this paper. First, we describe general properties and requirements of such an observation process; afterwards, we discuss specifics of realising contract observation in web services environments. Finally, we show how contract observation has been implemented as part of the IST-CONTRACT web services framework for contract-based systems.

1. INTRODUCTION

Of the ways in which agent behaviour can be regulated in a multi-agent system, electronic contracting – based on explicit representation of different parties' commitments and the agreement of all parties to them – has significant potential for modern distributed applications. In part, this is because it explicates different parties' responsibilities, and the agreement of all parties to them, allowing businesses to operate with expectations of the behaviour of others, but providing flexibility in how they fulfil their own obligations. Additionally, it mirrors existing (non-electronic) practise, aiding adoption.

A key element of contract-based systems is *contract monitoring*, i.e. a process by which the behaviour of individual parties and their participation in respective business processes is checked for compliance with contracts set to regulate such behaviour. The term monitoring has been traditionally used to refer both to the process through which contract-relevant events and states from a running contract-based distributed system are gathered, and the reasoning applied over the gathered information in order to determine the compliance with the respective contracts. Although such tight coupling can have certain advantages, it limits the ways in which both processes can be configured and combined to meet distinct requirements of different application domains.

In our approach, we therefore separate the data gathering process, termed *contract observation* further on, and study it independently of other operations and process required for contract monitoring.

Specifically, in Section 2 we analyse the observation process in general, primarily from the perspective of the categories of information that need to be observed in order to determine fulfilment state of contracts. In Section 3, we continue by exploring different ways in which the general observation process can be realised in a web services environment. In Section 4 we further narrow down our scope and describe how contract observation has been implemented in the IST-CONTRACT middleware [1]. In Section 5 we discuss related work and we conclude in Section 6.

Cite as: Title, Author(s), *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

2. CONTRACT OBSERVATION PROCESS

In general, a *contract* is a set of restrictions on the behaviour of involved contract parties. Such restrictions may involve both achieve and maintain conditions, e.g. a prohibition to perform a particular operation or an obligation to keep a certain quantitative measure of agent's operation within prescribed limits, respectively. The behaviour of contract parties can be modelled as composed of elementary units of activity, termed *actions* further on. Contract behaviour restrictions can then be defined in terms of a set of contract clauses specifying which action is required, prohibited and/or allowed for contract parties under which conditions. These conditions usually refer to the state of the environment within which the parties operate (termed *contracting environment*) but can also refer to the state of other contracts. See [2] for a detailed discussion of contract structure and semantics.

In order to be able to determine fulfilment of a contract, it is necessary to observe information at two levels:

1. information about the actions carried out by parties involved in the contract and the state of the contracting environment (*domain-level information*)

2. information about the content and life-cycle status of the contract (*contract-level information*)

Whereas domain-level information is required to track agent behaviour (which is subject to contract regulation), contract-level information is neccessary to track which contracts are currently active and against which the behaviour of contract parties should thus be checked for compliance.

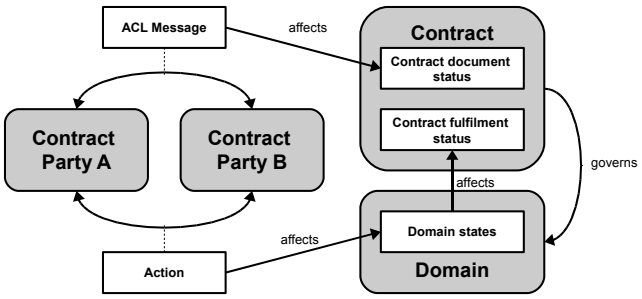

Figure 1: Observation in contract-based systems. Actions are observed at the domain level; contract-affecting communication (ACL messages) is observed at the contract level. The information is processed by the monitor to determine the contract lifecycle and fulfillment status.

Observing domain-level actions of a contract party agent requires gathering information on how the agent manipulates the environment in which the contract-regulated process executes. In the case of electronic systems, such an environment comprises a universe of electronic resources; observing agent's actions then correspond to the logging of operations on those resources (e.g. delivering a required file, transferring an agreed sum, launching a specified process etc). In order to allow to determine the compliance of a particular observed action, it is necessary that the observation process provides, at minimum, (i) the indentity of the agent performing the action, and (ii) detailed parameters of the action (e.g. the sum and the destination bank account in the case of the money transfer action).

In contrast, changes to the content (adding, removing or modifying contract clauses) and life-status of contracts (signature, termination, renewal etc) take place through direct interaction of the respective contract parties. Observing contract-level information thus requires monitoring the communication between the agents and logging any contract-affecting operations. The relation between the processes in a contract-based system and the different categories of information observed is depicted graphically in Figure 1.

Once the required information from the system is obtained, the actual decision on the contract compliance has to be made. This is generally a non-trivial operation requiring a reasoner capable of interpreting deontic concepts. Due to our focus on the observation, a module capable of such operation, termed *monitor*, is viewed as a blackbox and its implementation is not discussed in detail in this paper. We assume that the input to the monitor contains both domain- and contract-level information, specifically the state of the contracting environment, actions performed by the contract parties, contracts between the parties and the life-cycle status of the contracts (in particular whether any given contract is active). In its basic form, the monitor outputs, for each contract and/or each contract clause, the information whether the contract and/or the clause is being violated. We assume that the monitor operates sequentially and is causal in a sense that for determining the fulfilment state in a particular point in time it only needs information observed up to that point in time.

3. OBSERVATION IN WEB SERVICES ENVIRONMENTS

We now show how the abstract observation processes described in the previous section can be realized in web services (WS) environments. The basic assumption made is that exeuction of an action in WS environments corresponds to the invocation of a particular web service.

In general, any interaction between a contract party agent and other agents and/or resources deployed in a WS environment passes through several layers. It starts as an invocation of the agent platform API, continuing down the architecture through the agent platform messaging layer using a WS stack which, translating the invocation by a SOAP processor, creates an HTTP message sent to the other interacting party and/or resource using a TCP/IP connection – see Figure 2.

In theory, interactions can be intercepted on any of the above levels. In the following, we assume that the interception is performed by a special software module termed *sensor* which extracts the relevat data and sends them to a designated collection point. Specifically, we consider the following sensor insertion points (see Figure 2 for a graphcial overview):

application-level API [*both client and server side*]: the sensor is integrated into the contract party application code and explicitly called to report inter-agent messaging and/or action invocation

agent platform plug-in [*both client and server side*]: the sensor is integrated into the agent platform so that all communication and action invocation is automatically reported

proxy web service [*server side only*]: the target web service is encapsulated within a proxy web service mirroring the target web service operations; the proxy is invoked instead of the target web service and the sensor is integrated into the code of the proxy service (the proxy web service can be generated based on the WSDL of the target web service)

JAX-WS/JVM plug-in [*both client and server side*]: the SOAP processor in the JAX-WS library is extended to integrate the sensor or the sensor is plugged directly into the JVM; in either case, the container hosting the web services needs to use such a modified JAX-WS library or JVM instead of the standard distributions

web service/servlet container plug-in [*server side only*]: the target web service deployed in a web service container is observed using mechanisms offered by (some) servlet containers (e.g. Glassfish implements the necessary JSR monitoring and management specifications to intercept HTTP traffic between the container (server) and its clients) and a public API to attach the sensor to is available

http proxy [*client side only*]: all outbound traffic is filtered at the HTTP layer by a proxy application which filters SOAP calls and synthesise the required information

TCP/IP filter [*both client and server side*]: all traffic is filtered at the TCP/IP layer by means of a special application (usually integrated directly into the core of

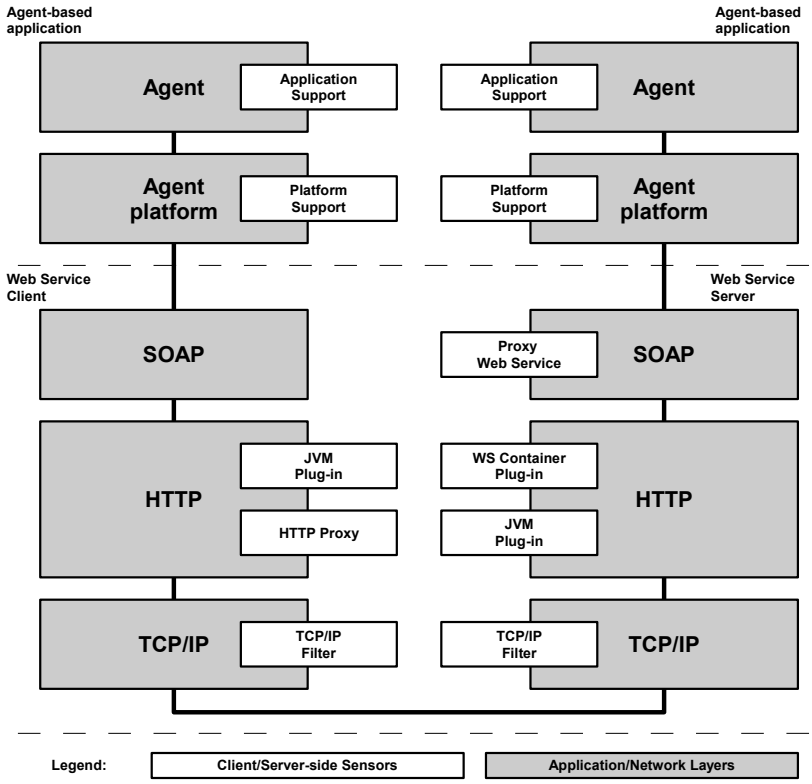

Figure 2: Possible implementations of observation sensors in the WS-based contract party interaction stack

the underlying operating system) and HTTP messages containing SOAP calls are identified and used to synthesise the information to be reported

Table 1 summarises the advantages and disadvantages of different sensor implementations.

4. OBSERVATION IN THE CONTRACT FRAMEWORK

In this section, we describe how contract observation has been implemented in the CONTRACT Framework developed within the IST-CONTRACT project[1]. We first overview the architecture of the whole framework and then describe the implemented observation process.

4.1 CONTRACT Framework Architecture

CONTRACT is a WS framework for developing, implementing and monitoring contract-based systems. The core of the framework is a JAX-WS compliant web-service-based agent platform[2], also developed within the IST-CONTRACT project. Agents are implemented as stateful web services which are accessed by means of a single stateless factory entry point returning a WS-Addressing compliant reference used for the invocation of individual agent operations. The web-service interface of each agent offers means for FIPA-ACL [3] compliant interactions between agents as well as operations for connecting the agent with external components such as a graphical front-end etc.

[1] `http://ist-contract.org`
[2] available from `http://ist-contract.sourceforge.net/`

4.2 Observation Pipeline

The observation process in the CONTRACT framework is referred to as the *observation pipeline* and is realized through a distributed collaboration of several different types of agents. After initial research and experimentation, the agent platform plug-in option has been chosen for implementing the observation gathering sensors (see Section 3 for details). The decision was made because of continuing difficulty to find a solution at a different level that would work reliably across a wide range of deployment scenarios supported. Agent communication and action selection modules have been therefore equipped with a sensor that reports any communication and action invocation performed by the agent.

Altogether, the following components are involved in the CONTRACT observation pipeline:

- **Sensor** – an interface implemented by the Contract Party to allow observation of its contract-related activities. The Sensor provides domain- and contract-level data to the Observer agent. The main functionality of Sensors, which are distributed in the contracting environment, is the pre-processing of the collected data into a form of observation reports described by XML schemas and the submission of these reports to the designated Observers. Three types of observation reports are used: *action reports* for notifications about actions performed by the contract parties, *domain predicate reports* for notifications about state changes of the observed contracting environment, and *ACL message reports* for communication between the contract party agents. The Sensor is implemented as a standalone

	Advantages (+)	Disadvantages (–)
application-level API	• all information needed for monitoring is instantly available	• reporting of selected actions and/or communication can be (intentionally) omitted
agent platform plug-in	• all information needed for monitoring is instantly available • reporting cannot be circumvented	• communication and action execution reporting has to be supported by the agent platform
proxy web service	• the proxy web service code can be extended by information needed for monitoring while the target web service remains intact and operational • platform/OS independent	• lack of stable open-source tools and frameworks implementing this functionality • may be a performance bottleneck
JAX-WS/JVM plug-in	• transparent for both client and server solutions (possibly using the same code) • does not require any modifications to existing services and/or clients	• the extension/modification may not be feasible • deployment on web service/servlet containers may be difficult
web service/servlet container plug-in	• usable transparently at the server side, uses container API • does not require any modifications to existing services and/or clients	• container dependence (not all containers implement the necessary JSR specifications)
http proxy	• usable transparently on the client side • minimal changes to existing web service clients (only configuration) • platform/OS independent	• more http proxies for more JVMs on the same computer may pose performance problems (stability, robustness, load)
TCP/IP filter	• transparent for both client and server solutions • does not require any modifications to existing services and/or clients	• platform/OS dependent • may rise security issues and conflicts with existing software operating at the TCP/IP level

Table 1: Comparison of different sensor implementations

Java library providing a reporting API.

- **Observer** – the central component of the pipeline. Observer's main responsibility consists in collecting data reported by sensors and providing them further to the other pipeline components. The Observer may by extended by plug-ins for the pre-processing and processing of the stored information. Observer provides an ACL-compliant interface supporting the Query and the Subscribe agent communication protocols.

- **Monitor** – a plug-in module for the Observer processing the observed data to determine contract fulfilment. In the case of the CONTRACT project, the Monitor module has been implemented by a means of a reasoner based on augmented transition networks (see [4] for details). As its input, the Monitor receives the observation reports gathered by the encompassing Observer; the output of the Monitor (the fulfilment state of all monitored contracts and their clauses) is forwarded to the Observer from which it is made available to other components in the system.

- **Contract Storer** – the agent keeping track of contracts in the system, including their life-cycle status (in cooperation with the Observer). The contract storer acts as an authoritative source of information on contract content and life-cycle status, which is used by other components of the pipeline. Internally, the Contract Storer agent uses an eXist XML database to store contract documents.

In addition, there is the Analyser agent providing a user front-end to the pipeline, presenting the contract status and fulfilment data to the administrators of the monitored contract-based system. Interactions between the components within the CONTRACT Framework are depicted in Figure 3; their involvement in individual stages of the contract life-cycle is depicted in the Figure 4.

The concept of presented pipeline and the underlying WS framework has been validated on several use-cases, including modular certification testing, insurance claim handling and the maintenance and support of aircraft engine units (see [5] for the description of the use cases).

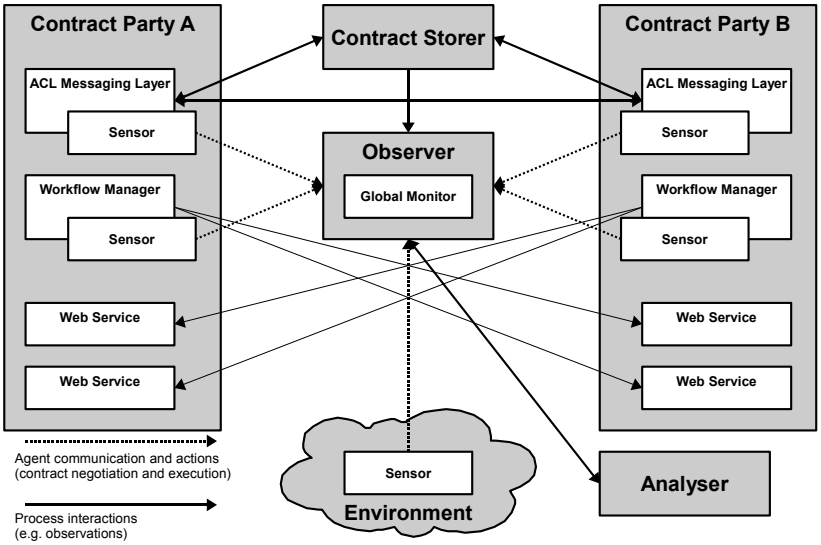

Figure 3: Architecture of the CONTRACT framework

5. RELATED WORK

This work focuses on the observation and monitoring of distributed systems whose operation can be modelled as a choreography of well-defined elementary actions. This contrasts with most of the work on monitoring service level agreements (e.g. [6, 7, 8]) which focus on continuous evaluation of a set of performance metrics and their comparison with agreed thresholds.

Approaches more relevant to our work range from theoretical concepts establishing service frameworks for contract descriptions and monitoring (e.g. [9, 10]) to implementations of tools and other means for run-time monitoring of workflow executions based on electronic contracts as declarative specifications of multi-party cooperative behaviours (such as [11, 12, 13, 14]).

Mahbub and Spanoudakis [11] use event calculus for defining monitoring requirements on top of a workflow described in BPEL4WS [15]. The behaviour requirements are automatically extracted from the workflow description and can be extended in order to describe an overall behaviour of a service composition in terms of temporal constraints and properties of the data processed during web service invocations.

Barbon et Al. [13] present a monitoring module extending the open-source Active BPEL workflow engine. The monitoring module intercepts events in the workflow life-cycle (creation and termination) and invocations of external services; based on the recorded data, the monitor checks for run-time errors, time-outs, and functional properties. The monitoring specification is expressed as logical formulas in the RunTime Monitor specification Language (RTML) and can be automatically translated into a Java code implementing the monitor functionality. The monitoring logic is kept separate from the BPEL process (no modification required) but cannot be deployed in other BPEL engines.

Another approach for monitoring workflows is presented by Baresi et Al. [12]. They provide a tool for annotating workflows with assertions (monitoring rules) described in a proprietary Web Service Constraint Language (WS-CoL) being inspired by the JML [16]. Their approach clearly separates the original business logic from the superimposed monitoring code and is independent of any specific workflow engine.

All of the above work, however, views observation as an inseparable part of the complete monitoring solution. Modular approach taken in this paper, viewing observation as an abstract process with multiple possible implementations, is to our best knowledge novel.

6. CONCLUSIONS

Observation, i.e. the process of obtaining the relevant information on the operation of a distributed business system, is a key pre-requisite for determining whether the system operation complies with contracts set to regulate it. The observation has to ensure that sufficient information on the action of contract parties in the system is recorded and provided in a suitable form. In this paper, we identified the type and form of information required and analysed possible ways in which the information can be obtained in web-services based environments. We then showed a particular implementation of the observation process, as implemented by the IST-CONTRACT project in a web services-based framework for contract-based systems.

7. ACKNOWLEDGEMENTS

The research described is part-funded by the EC FP6 projects CONTRACT (contract No. 034418), I*PROMS Network of Excellence, and also by the Ministry of Education, Youth and Sports of the Czech Republic grant No. MSM 6840770038. The opinions expressed herein are those of the named authors only and should not be taken as necessarily representative of the opinion of the European Commission or CONTRACT project partners.

8. REFERENCES

[1] Confalonieri, R., Álvarez Napagao, S., Panagiotidi, S., Vázquez-Salceda, J., Willmott, S.: A middleware architecture for building contract-aware agent-based services. In Kowalczyk, R., Huhns, M.N., Klusch, M.,

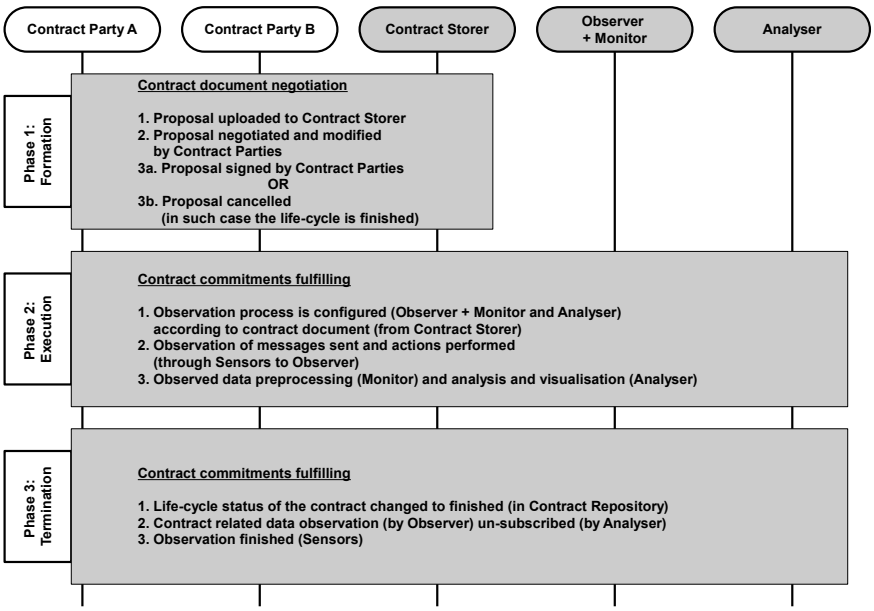

Figure 4: Interactions between the CONTRACT framework components during individual stages of the contract life-cycle

Maamar, Z., Vo, Q.B., eds.: SOCASE 2008: Proceedings of Intl. Workshop on Service-Oriented Computing: Agents, Semantics, and Engineering. Volume 5006 of Lecture Notes in Computer Science., Springer (2008) 1–14

[2] Oren, N., Panagiotidi, S., Vazquez-Salceda, J., Modgil, S., Luck, M., Miles, S.: Towards a formalisation of electronic contracting environments. In: COIN 2008: Proceedings of AAAI Workhop on Coordination, Organization, Institutions and Norms in Agent Systems. (2008)

[3] FIPA: Foundation for intelligent physical agents [online]. ⟨`http://www.fipa.org`⟩ (12 2003)

[4] Faci, N., Modgil, S., Oren, N., Meneguzzi, F., Miles, S., Luck, M.: Towards a monitoring framework for agent-based contract systems. In: CIA '08: Proceedings of the 12th international workshop on Cooperative Information Agents XII, Berlin, Heidelberg, Springer-Verlag (2008) 292–305

[5] Jakob, M., Miles, S., Luck, M., Oren, N., Kollingbaum, M., Holt, C., Vazquez, J., Storms, P., Dehn, M.: Case Studies for Contract-based Systems. In: Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems. (2008)

[6] Keller, A., Ludwig, H.: The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. Journal of Network and Systems Management **11**(1) (2003) 57–81

[7] Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web Services Agreement Specification (WS-Agreement). In: Global Grid Forum GRAAP-WG, Draft, August. (2004)

[8] Schopf, J., Raicu, I.: Pearlman Let al. Monitoring and discovery in a web services framework: Functionality and performance of Globus Toolkit MDS4. Technical report, Technical Report, Mathematics and Computer Science Division, Argonne National Laboratory, 2006

[9] Xu, L., Jeusfeld, M.: Pro-active Monitoring of Electronic Contracts. In: Advanced Information Systems Engineering, Springer (2003) 584–600

[10] Milosevic, Z., Gibson, S., Linington, P., Cole, J., Kulkarni, S.: On design and implementation of a contract monitoring facility. In: Electronic Contracting, 2004. Proceedings. First IEEE International Workshop on. (2004) 62–70

[11] Mahbub, K., Spanoudakis, G.: A framework for requirents monitoring of service based systems. In: Proceedings of the 2nd international conference on Service oriented computing, ACM New York, NY, USA (2004) 84–93

[12] Baresi, L., Guinea, S.: Towards Dynamic Monitoring of WS-BPEL Processes. In: Service-Oriented Computing - ICSOC 2005, Springer (2005) 369–282

[13] Barbon, F., Traverso, P., Pistore, M., Trainotti, M.: Run-time monitoring of instances and classes of web service compositions. In: ICWS. Volume 6. 63–71

[14] Radha Krishna, P., Karlapalem, K., Chiu, D.: An EREC framework for e-contract modeling, enactment and monitoring. Data & Knowledge Engineering **51**(1) (2004) 31–58

[15] OASIS: Oasis web services business process execution language (wsbpel) [online]. ⟨`http://www.oasis-open.org/committees/download.php/18714/wsbpel-specification-draft-May17.htm`⟩ (5 2006) Web Services Business Process Execution Language Version 2.0. Committee Draft May 2006.

[16] Leavens, G., Baker, A., Ruby, C.: Preliminary design of JML: a behavioral interface specification language for java. ACM SIGSOFT Software Engineering Notes **31**(3) (2006) 1–38

An Agent-Oriented Service Model for Personal Information Manager

Tarek Helmy
King Fahd University of Petroleum and Minerals, Dhahran 31261, Mail Box 413, Kingdom of Saudi Arabia
helmy@kfupm.edu.sa

Ali Bahrani
King Fahd University of Petroleum and Minerals, Dhahran 31261, Kingdom of Saudi Arabia
ali.bahrani.10@aramco.com

Jeffery M. Bradshaw
Florida Institute for Human and Machine Cognition (IHMC) Pensacola, FL 32502, U.S.A.
jbrashaw@ihmc.us

ABSTRACT
Building multi-agent-based systems requires great attention to all phases of the development life cycle in order to come up with a reusable model of a high quality. The main goal of this paper is the investigation and development of a methodology for describing and designing a service model for personal information manger based on the agent-oriented paradigm. Several agent-oriented software engineering methodologies are developed to tailor the special characteristics of multi-agent systems. In this paper, the Gaia methodology is used to guide us through the development of an agent-oriented model for a personal information manager. The proposed model is shown to be complete, scalable, independent of specific development frameworks, and supportive of a high degree of autonomous behavior. The extensibility of the model is shown by elaborating the original model to support speech recognition and calendar scheduling based on user's preferences and learning from history.

Categories and Subject Descriptors
I.2.11 [**Artificial Intelligence**]: *Artificial Intelligent, Intelligent agent.* D.2.7 **[Software Engineering]:** *Agent-oriented.*

General Terms
Design, Documentation, Reliability, Experimentation.

Keywords
Personal Information Manager, Gaia, Agent-Oriented Software Engineering Methodology.

1. INTRODUCTION
In their everyday lives, people often use a personal organizer record and track their personal information. While the first personal organizers were small books that usually contain a calendar, an address book, and notebook paper. Personal Information Manager (PIM) software is increasingly replacing paper-based approaches.

People spend a lot of time and effort on reading, filtering, searching and managing their to-do lists, contacts, emails, and appointments. Hence, scientists in the fields of computer science and time management have tried (and are still trying) to increase user productivity by inventing new practices and techniques that help users to better manage their personal information [11], [12]. Some of these practices can be automated or learned. Software applications can be developed to support such practices and to carry out some actions automatically on behalf of the users without their interactions. In addition, they may be developed so that they learn user's preferences and build a user's profile in order to accomplish personalized actions. Ideally, relationships among applications handling different personal information types are also modeled and leveraged. For example, a new appointment may be created as the result of email exchanges with someone who is already stored in the contacts database. The strong dependencies among components of such applications make it difficult to handle modifications in an incremental way. The model being developed in this paper represents a foundation for a sophisticated PIM that can support a high degree of autonomous behavior as well as the ability to learn. In addition, its extensibility allows people to elaborate the model to support specialized requirements while continuing to take advantage of existing features. Such extensibility is a key to reduce developer's time and effort.

In this paper, we define a development-framework-independent agent-oriented model for PIMs that integrates the main features of a generic PIM: Contacts, Tasks, Calendar and Email. Several artifacts are prepared in different phases of the development life cycle including: documents, schemas, tables and diagrams. The Gaia methodology is used to guide us through the development life cycle [8]. The model contains the agent types involved in the system and illustrates how they interact with each other. In addition, it specifies the responsibilities and permissions of each agent. The paper is organized as follows: Section 2 provides an overview of the related work. Section 3 provides details of Gaia as a well-known agent-oriented software engineering methodology. Section 4 shows the extensibility of the model through the addition of two intelligent agents. Section 5 presents an execution scenario for the model. Finally, Section 6 concludes the paper.

2. RELATED WORK
There are a lot of products and tools for managing users' Personal Information (PI). Some of these tools have been proposed in the research domain, whereas others are innovative commercial systems. They can be classified into the following categories, based on the level of integration they provide.

PI-Specific Tools: These tools provide technology aimed at a specific type of PI such as email messages or tasks on a to-do list. In addition, they do not consider integration between distinct PI

Cite as: An Agent-Oriented Service Model for Personal Information Manager, Tarek Helmy, Ali Bahrani, Jeffery M. Bradshaw, *Proc. of 8th Int. Conf. on Autonomous Agents and Multi-agent Systems workshops*, Decker, Sichman, Sierra, and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

tools as a primary design goal. EMMA, Towel and Data Mountain are examples of tools in this category. **EMMA** is an email system focusing on email management as a process. It manages the sorting, prioritizing, reading, replying, archiving, and deleting of email messages. It caters to a wide variety of users by adopting a knowledge acquisition technique known as "Ripple Down Rules" (RDR). RDR is an incremental technique in which the user starts with an empty knowledge base and adds rules while processing examples [3]. **Towel** is a task management application that manages users' to-do lists, and was developed by SRI International. It provides a unified environment that enables users to manage their tasks, delegate tasks to others, or collaborate with other users. It uses various AI technologies that result in saving user's time, reducing load, and improving task performance [10]. **Data Mountain** is a technique that allows users to place documents at arbitrary positions in a 3D desktop virtual environment using a simple 2D interaction technique. Its interface is designed specifically to take the advantage of human spatial memory in managing documents (i.e. the ability to remember where you put something). It is designed with a fixed viewpoint so that users need not to navigate around the space. Users can identify and distinguish between documents, both through their thumbnail representation and also through pop-up titles [7].

Systems providing integration between distinct PI-specific tools: Tools in this category offer limited integration based on some kinds of structured information. For example, a tool in this category may allow the user to access the contact manager when selecting an email address in a message. Stuff-I've-Seen (SIS) is an example of a tool in this category. It has been developed to make it easy for people to find information they have seen before. There are two main concepts in the design of SIS that help it to achieve information reuse. First, the system provides a unified index of information that a person has seen on his computer. Second, because a person has seen the information before, rich contextual cues such as time, author, thumbnails, and previews can be used to search for and present information [1, 2]. SIS indexes many types of information such as emails, web pages, documents, media files, calendar appointments, file system hierarchy, email folder hierarchies, favorites, and web pages history. All of these are integrated into a single index, regardless of the form or origin of the information.

Systems embedding additional support for managing multiple types of information within one PI-specific tool: TaskMaster is an example of tools in this category. It allows the user to manage multiple types of PI with one PIM tool through the embedding of extra functionality. It is a client that provides a mechanism for labeling any item of information with to-do metadata. In addition, it manages multiple types of PI [1].

Systems consolidating the management of all PI in a single new interface: This is in contrast to systems that embed PI within an existing tool. Examples of tools in this category include MEMOIRS, ContactMap and UMEA, which unify PI management in a single interface based on time, contacts, and activities [1]. **A MEMOIR is** a prototype that has been developed based on the chronological organization of the PI. It is based on integrating a user's diary and filing the system with a chronological mechanism. The main design goals of this prototype are to enable PI retrieval based on temporal context, and to promote retrieving by recognition rather than by coloring items. **ContactMap** integrates the management of different PI based on the representation of a user's social network, which is derived from the user's address book. The interface maps between a social network, files, bookmarks and email messages. It also enables users to use the information retrieved from the address book for communication. Thus, **ContactMap** integrates both information management and communication functionality. The **UMEA** allows the user to organize multiple types of PI based on their projects. The design rationale is based on the observation that a user's activity often involves multiple PIM tools. Hence, the user is asked to provide the project name that he is currently working on. Thereafter, all the information details that is associated with the proposed model.

3. BUILDING THE MODEL

3.1 Rationale for Selecting Gaia

Formal guidelines on how to progress through different phases of development life cycle can be helpful. Software development methodologies are intended to save time and efforts since they are designed for usability, built according to best practices, and describe the important steps that the designer should follow [4]. In general, a software development methodology consists of process, heuristic rules, artifacts, notations and pattern [4, 5]. A process is a sequence of phases and activities that guide the developer to build a system. Heuristic rules are those supporting the developer in making relevant choices. Artifacts are diagrams, schemas or documents in the form of text or graphics. Notations are those which used in representing the artifacts. A pattern is what can be applied to solve common situations [4, 5]. Over the past two decades, complex systems have been engineered using powerful and natural, high-level abstractions. Examples of these abstractions include procedural abstractions, abstract data types, objects and components. Software agents can be seen as advanced abstractions that may be used by software developers to more naturally understand, develop, and model complex distributed systems. Since agents provide an advanced abstraction for complex distributed systems, it is necessary to use software engineering techniques that are specifically tailored for them. Existing software development techniques such as object-oriented analysis and design fail to represent agents' characteristics. They fail to adequately capture agents' flexible and autonomous behaviors, the richness of the agents' interactions and the complexity of the agents' system organization structure [9]. Gaia has become well-known as an agent-oriented software engineering methodology because it is tailored specifically to the analysis and design of agent-based systems and deals with macro-level (social) as well as micro-level (agent) design aspects [8]. In addition, it captures agents' characteristics of being proactive, reactive, and having social ability. The Gaia methodology is characterized by its:

1. Precision: the live-ness and safety properties in role definition make it accurate and prevent misunderstanding of the modeled functionality.
2. Accessibility: Gaia is easy to understand and use due to its simple models and clarity.
3. Expressiveness: Gaia can handle a large variety of systems due to its generic structure.
4. Modularity: Gaia is modular because of its building blocks such as roles, protocols and activities.

The Gaia methodology enables analysts to move from abstract concepts to increasingly concrete ones. Abstract concepts are those used during analysis to conceptualize the system like roles, permissions, responsibilities, protocols, activities, live-ness and safety properties. However, concrete concepts are those that are used within the design process and have direct counterparts in the runtime system like agent types and acquaintances. The following sections define the different phases we went through in order to build the model and the artifacts produced in each phase.

3.2 Requirements

Since Gaia methodology has two phases only, analysis and design, requirements capture should be done independently beforehand. The most popular way to capture the potential function requirements of the system is to use 'use cases' where each use case represents one or more scenarios that demonstrate how the system should interact with users or other systems. Use cases are not object-oriented in nature and hence they can be used to capture the functional requirements of multi-agent systems without modifications [4]. The first step in developing the use cases is to define the set of actors that are involved in the story where actors are the different people or devices that use the system [6]. The only actor in our PIM example is the user who is accessing the system and is willing to manage his/her personal information. The second step is identifying the system's major use cases. A use case represents a major piece of functionality that delivers some valuable functions to the user. In our PIM, the major use cases identified are: 'Maintain Tasks', 'Maintain Contacts', 'Maintain Calendar' and 'Maintain Emails'. In addition to these four main use cases, there is one use case 'Verify User' which needs to be executed as a prerequisite for each of the main use cases. Hence, this use case is related to the main use cases and the relationship is of type <<includes>>. See Figure 1 below for the main use cases diagram.

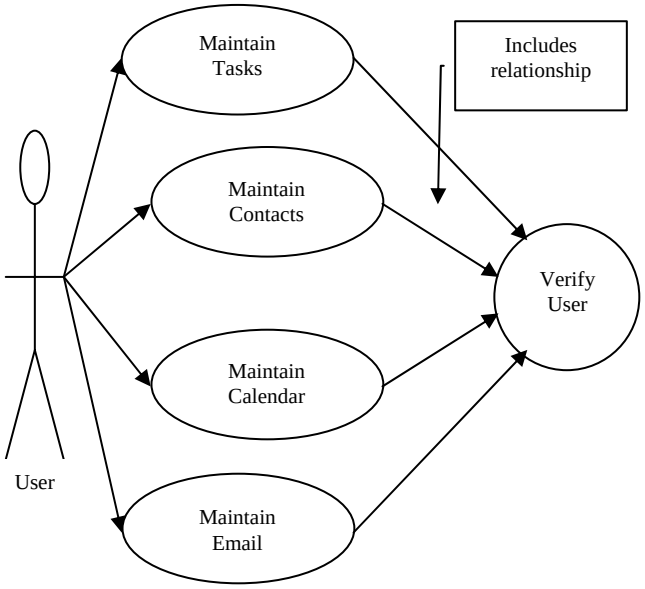

Figure 1: Main Use Cases Diagram

3.3 Analysis

The objective of the analysis phase in Gaia methodology is to develop an understanding of the system and its structure. It includes the identifications of the environment model, the role model and the interaction model of the system. The following sections discuss these models for our PIM.

3.3.1 The Environment Model

The environment in Gaia is treated in terms of abstract computational resources such as variables that are made available to agents for sensing, affecting or consuming. Such variables are accessed by agents to be read or changed, or to extract their values. Hence, the environment model can be shown as a list of abstract computational resources associated with symbolic names and characterized by the types of actions agents are performing on them. In addition, it is also possible to add textual comments and descriptions to each resource.

Considering our system, we have used four resources; namely, Tasks Database, Contacts Database, Calendar Database and Emails Database. The user accessing the system is able to manage his personal information using these resources, which are categorized by the user such that, each user is able to access/manage his information only. For that reason, the user needs to be authenticated and verified by providing a user's name and password in the system's startup. Obviously, the main purpose of any PIM is to manage users' tasks, contacts, appointments and emails. Hence, the system needs read/write access to these resources.

Throughout the analysis, design and implementation phases of the system, the tasks, contacts, calendar items and emails resources are represented in two-dimensional arrays. The first index of the array represents the user identification, whereas the second index represents the resource's element identification. For example, the representation Task[i][j] means the j^{th} task for the user i. These discussions are summarized in Figure 2, which shows the environment model of the system.

- *changes Tasks[i][j] where i=1,…total_users and j=1,… total_tasks.*
- *changes Contacts[i][j] where i=1,…total_users and j=1,… total_contacts.*
- *changes Calendar Items [i][j] where i=1,… total_users and j=1, …. total_calendar_items.*
- *changes Emails[i][j] where i=1,…total_users and j=1,… total_Emails.*

Figure 2: The Environment Model of the System

3.3.2 The Role Model

Defining a multi-agent system using roles is quite a natural way of thinking [8]. For example, if we consider a typical company as a human organization, we can define several roles such as 'president', 'vice-president', 'manager' and so on. These roles will be instantiated with actual individuals such that an individual can take the role of a 'president' and another one can take the role of 'vice-president'. It is also possible in some small companies that an individual can take more than one role. For example, an individual can take the role of being a 'mail fetcher' and 'office cleaner'. In addition, it is possible that a role can be instantiated to more than one individual such as the 'salesman' role. The role model in Gaia methodology identifies the key roles in the system and their permissions and responsibilities attributes [9]. Permission attributes identify the resources that can be used to

carry out the role and resource's limits ('reads', 'changes' or 'generates'). In order to represent permissions, Gaia makes use of the same notation already used for representing the environment resources. However, the attributes associated with resources are no longer representing what can be done with such resources from the perspective of the environment. On the other hand, the attributes associated with resources are representing what the agent's role must, and must not, be allowed to do in order to accomplish the role's requirements. Conversely, the responsibilities' attributes are those that determine the expected behavior of a role. Responsibilities are divided into two types: live-ness responsibilities and safety responsibilities. Live-ness responsibilities are those that state "something good happens." They are so called because they tend to say that the agent carrying out the role is still alive. They are specified via a live-ness expression that defines the lifecycle of the role. The general form of live-ness expression is as follows: Role_Name = expression, where "Role_Name" is the name of the role whose live-ness responsibilities are being defined, and "expression" is the live-ness expression defining the live-ness properties of Role_Name. The atomic components of a live-ness expression are either activities or protocols. An activity is somewhat like a method in object-oriented terms or a procedure in procedural programming language. An activity is a unit of action that the agent may perform that does not involve interaction with any other agent. On the other hand, protocols are activities that require interaction with other agents. An activity is visually distinguished from a protocol through underlining. Sometimes, it is insufficient to specify the live-ness responsibilities of a role. This is because an agent carrying out a role will be required to maintain certain invariants while executing. For example, we might require that a particular agent taking part in an e-commerce application never spends more money than it has allocated. These invariants are called 'safety conditions', because they usually relate to the absence of some undesirable condition arising. Safety conditions in Gaia are specified by means of a list of predicates. The list is represented as a bulleted list such that each item in the list expresses individual safety responsibility.

Our PIM manages four types of PI namely: Tasks, Contacts, Calendar items and Emails. In order to satisfy the requirements, we have identified the required roles for each PI type, and these are listed in Table 1. In addition to the roles identified in the Table 1, we need these additional roles: Facilitator, Task, Contact, Calendar and Email. The facilitator is used to perform communication services such as forwarding messages and displaying interface screens. With this in mind, the task, contact, calendar and email roles act as a middle layer between the facilitator role and the other roles in the table. In other words, the roles are tiered into three levels. The first level contains only the facilitator role. The second level contains the task, contact, calendar and emails roles while all the other roles are in the third level. When the facilitator receives a command from the user, it forwards it to the 2nd level roles asking them whether they can satisfy the command. Then, they parse the command and reply with the name of the responsible 3rd level role that can execute the command. Then, the facilitator forwards the command to that specific responsible 3rd level agent asking for execution. Figure 3 shows the tiers of the model. After we have identified the role hierarchy, we have defined all roles mentioned in the hierarchy using the recommended Gaia's role schema. Figure 4 and 5 show the schemas for the 'Task' and 'Task Creator' roles as examples.

Table 1: System's Roles List

Personal Information Type	Roles
Task	- Task Creator, Task Modifier - Task Deleter, Task Reassigner
Contact	- Contact Creator, Contact Modifier - Contact Deleter, Contact Searcher - Contact Sender
Calendar Items	- Calendar Creator, Calendar Modifier - Calendar Deleter, Calendar Reminder - Calendar Inviter, Calendar Invitation Tracker
Emails	- Email Creator, Email Deleter - Email Sender, Email Receiver

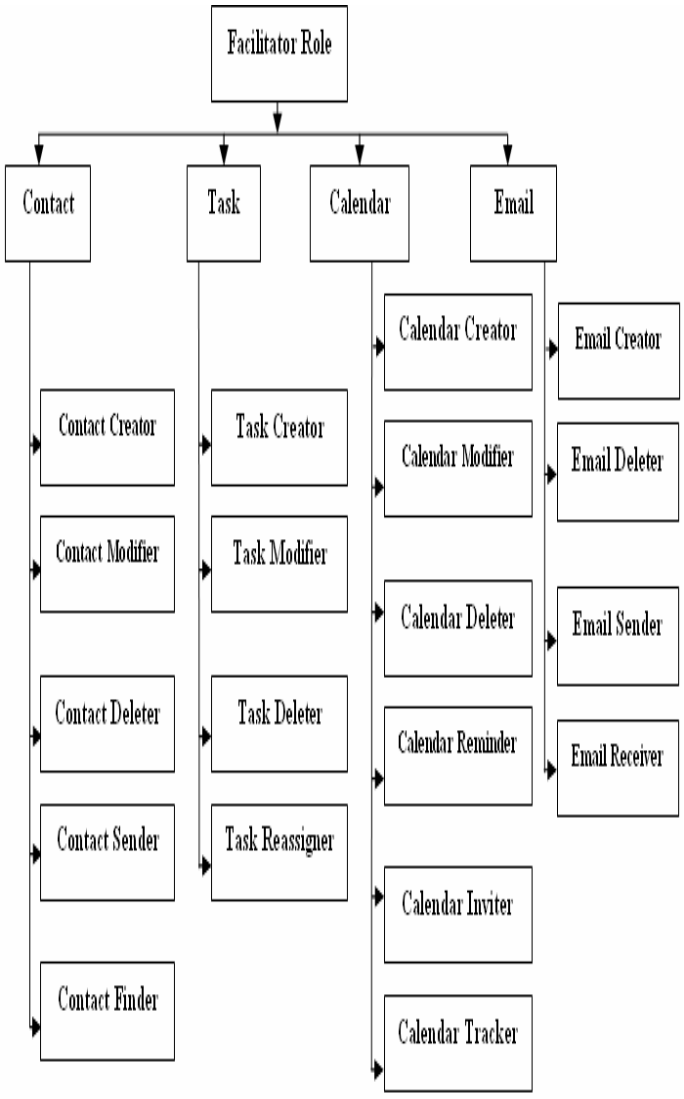

Figure 3: Roles Hierarchy

Role Name: Task
Description: This role manages the communication between the facilitator (level 1) and other task roles in level 3.
Protocols and Activities: AwaitCall, Parse, ReplyToIsItYours
Permissions: This role does not have access to the resources directly.
Responsibilities: Liveness: Task = (AwaitCall, Parse, ReplyToIsItYours) ^{w} Safety: True

Figure 4: Schema for the Task Role

Role Name: TaskCreator
Description: Create a new Task Item for the logged on user.
Protocols and Activities: AwaitCall, CreateTask
Permissions: Reads supplied Logged User //Read all tasks for the user who logged on. TaskItems[LoggedUser] //Create a new TaskItem i for the user who logged on. Generates TaskItems [LoggedUser][i].
Responsibilities: Liveness: TaskCreator = (AwaitCall.CreateTask)^{w} Safety: True

Figure 5: Schema for the Task Creator Role

3.3.3 The Interaction Model

This model captures the dependencies and relationships between the roles of the system. Each interaction between two roles has a protocol definition. A protocol definition consists of these attributes:

- *Protocol Name*: brief description that captures the nature of the interaction.
- *Initiator*: the role(s) responsible for starting the interaction.
- *Partner*: the responder role(s) with which the initiator interacts.
- *Inputs*: information supplied to the protocol responder during interaction.
- *Description*: textual description explaining the purpose of the protocol and the processing activities implied in its execution.

In order to build the interaction model of our PIM, we have defined each interaction (protocol) named in the role model using Gaia's protocol definition template. For example, in the Task role schema defined in Figure 4, we have two protocols 'AwaitCall' and 'ReplyToIsItYours' which are defined in Figures 6 and 7 respectively.

3.4 Design

The objective of the design phase is to transform the abstract models derived during the analysis phase into a sufficiently low level of abstractions in order to implement agents. The design phase includes the generation of the agent model and the acquaintance model. The following sections discuss these models for our PIM.

Role Name: Task		
Protocol Name: AwaitCall		
Initiator:	Partner:	Input:
Facilitator	Task	User's command
Description:		Output:
When the user commands the system to do something, the facilitator will send the command to all 2nd level roles (including task role) asking them if the command is related to them or not.		Command will be parsed

Figure 6: Definition of AwaitCall Protocol

Role Name: Task		
Protocol Name: ReplyToIsItYours		
Initiator:	Partner:	Input:
Task	Facilitator	None
Description:		Output:
This interaction occurs as a reply to the facilitator in his request about if the command is related to this role or not. If it is related then a responsible role from level 3 will be sent as output.		Yes/No, and the responsible agent from level 3.

Figure 7: Definition of ReplyToIsItYours Protocol

3.4.1 The Agent Model

The purpose of the agent model is to document the various agent types that will be used in the system and the agent instances that will realize these agent types at runtime. An agent type is a set of agent roles. However, sometimes there is a one-to-one correspondence between roles and agent types. A designer can choose to package a number of closely related roles in the same agent type for the purpose of convenience or efficiency. The designer may want to optimize the design by aggregating a number of agent roles into a single type that carries out all the functionalities required by all roles. Later, only this agent type needs to be delivered. In our case, there is a one-to-one correspondence between roles and agent types. That means we will deliver as many agent types as roles defined in Figure 3.

3.4.2 The Acquaintance Model

This model defines the communication links that exist between agent types and simply indicates that communication exists. However, it does not define what messages are sent or when they should be sent. The purpose of this model is to identify any potential communication bottlenecks that may cause problems at runtime. The acquaintance model is a directed graph with nodes in the graph corresponding to agent types and arcs corresponding to communication pathways. Figure 8 below shows the acquaintance model of the system.

3.5 Prototype Implementation

We have developed a prototype for the model for partial implementation of Tasks and Contacts features. The prototype was developed using the Microsoft SQL Server 2005 Express Edition as a back-end database and the AgentBuilder as a multi-agent development tool. Using Gaia methodology in the analysis and design phases helps us developing the prototype in minimal effort. The artifacts produced in the analysis and design phases can be directly transformed to code. For example, if we consider the TaskCreator role produced in the analysis phase and defined in Figure 5, we find that the live-ness expression is:

$$\text{TaskCreator} = (\text{AwaitCall} \cdot \text{CreateTask})^{w}$$

where AwaitCall is a protocol involving communication with another agent, and CreateTask is an activity that the agent can do without any communication.

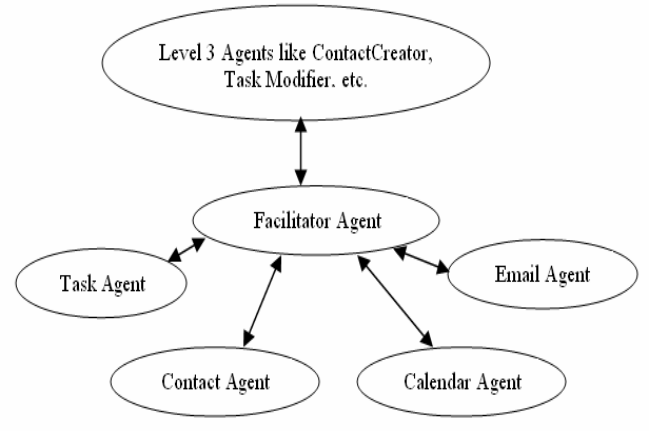

Figure 8: Acquaintance Model

Similarly, in order to find the definition of the AwaitCall protocol, we refer to the interaction model developed also in the analysis phase [Figure 9]. The definition of the protocol specifies that the initiator for the communication is the facilitator agent, while the TaskCreator agent is the partner. It also specifies that the input to that interaction is the Task details and the output expected after the interaction is inserting the task into the database table. This discussion is directly mapped into the agent's rules in the AgentBuilder tool as shown in Figure 10.

Role Name: TaskCreator		
Protocol Name: AwaitCall		
Initiator:	Partner:	Input:
Facilitator	TaskCreator	Task Details
Description: In this interaction, the Facilitator agent sends the task details to the TaskCreator in order to create a new task in the database.		Output: New task created.

Figure 9: Definition of AwaitCall Protocol

The interpretation of the snippet in Figure 10 is as follows. If the KQML message received is from the "Facilitator" (line 1) and asks the receiver (line 2) to execute the command embedded into the message (line 3), then the receiver executes a CreateTask method of object $myTask (line 3 in THEN section). Lines 4 in the THEN section are used to print to the agent's console.

WHEN:
1. (%message.sender EQUALS "Facilitator")
2. (%message.performative EQUALS "ask-one")
3. (%message.replyWith EQUALS "doIt")
THEN:
1. DO SystemOutPrintln ("TaskCreator Agent is going to create the task")
2. SET_TEMPORARY $myTask TO %message.content.myTask
3. DO CreationReturnedValue=$myTaskCreateTask (%message.content)
4. DO SystemOutPrintln ("Task is created successfully")

Figure 10: Behavioral Rules of TaskCreatorAgent

4. MODEL EXTENSIBILITY

One important objective of this model; in addition to benefiting from agents' characteristics; is to build a PIM model that can be extended by researchers in order to test their new practices and algorithms. In this section, we study the ability of the model to extend it by adding two intelligent agents: the 'Speech Recognition Agent' and the 'Calendar Scheduler Agent'.

4.1 Adding a Speech Recognition Agent

The purpose of the speech recognition agent is to transform the speech into a text. The speech recognition agent does not have direct access to the system resources. Instead, it sends the command as a text to the facilitator agent. Figure 11 shows the role's schema while Figure 12 shows the definition of the unique protocol involved: 'SendTextCommand'.

4.2 Adding a Calendar Scheduling Agent

This agent schedules the meetings of the user on his behalf based on his preferences and the usage history. It also negotiates timing with other users until agreement is reached. Users' preferences are stored as profiles in the database. The user needs to answer a number of questions posed by the agent in order to allow supervisory learning. Later, the agent will consult the user's profile to schedule meetings at the user's preferred times. Here are some questions that can be stored in the user's profile:

1- Do you like your meetings to be contiguous or scattered throughout the day? If you like your meetings to be scattered, what is the minimum time needed between meetings?

2- What are your block days of the week (i.e. the days that you least want to have meetings)?

3- What are your block times of the day (i.e. the times that you do not want the agent to schedule meetings)?

4- What is the maximum number of meetings in a day?

5- Who are the people that have high priority? Meetings with high priority people are given higher precedence in scheduling. For example, meetings with the user's manager cannot be negotiated. In addition, in case of conflict, meetings with low priority people will be rescheduled to another time.

6- What is the maximum number of negotiations allowed per meeting without involving the user? This question is asked in order to stop the agent from negotiating meeting times infinitely. When negotiation exceeds a user-defined limit, the user is able to review his schedules manually and negotiate.

Role Name: SpeechRecognition
Description: This role recognizes speech and transfers it to text.
Protocols and Activities: ListenToCommand, ConvertToText, SendTextCommand.
Permissions: *//This agent does not have access to the resources directly*
Responsibilities: Liveness: SpeechRecognition = (ListenToCommand**.** ConvertToText **.** SendTextCommand)^{w} Safety: * True
Figure 11: Schema for the Role SpeechRecognition

Role Name: SpeechRecognition		
Protocol Name: SendTextCommand		
Initiator:	Partner:	Input:
SpeechRecognition	Facilitator	Command as text
Description:		Output:
This interaction occurs to transfer the command to the facilitator after it is converted to text.		Facilitator will continue the execution process of the command.

Figure 12: Definition of SendTextCommand Protocol

In addition to this supervisory learning, the agent learns the user's preferences from the history. Clearly, this agent needs some functionality provided by the model. It needs to access and update the user's calendar and contacts, as well as, it needs the emailing functionality to negotiate meeting times. Hence, the model is going to be extended. Considering the meeting scheduling problem, the user can be either an organizer or an attendee. Being an organizer means he needs to invite other people, whereas an attendee is invited to a meeting. If the user is an organizer, then the scheduling and negotiation process will be as follows:

a. The user asks the agent to schedule a meeting with someone.

b. The agent consults the user's preferences and sends an invitation to other members by using the emailing functionality provided by the model.

c. The agent receives Accept, Reject or Propose New Time replies from the attendees.

d. Steps (b) and (c) are repeated until satisfaction.

On the other hand, if the user is an attendee then the scheduling and negotiation process will be as follows:

a. The user receives a meeting invitation by email.

b. The agent consults the user's preferences and sends Accept, or Reject, or Propose New Time reply to the organizer.

c. Steps (a) and (b) are repeated until satisfaction.

Hence, we have two roles for the CalendarScheduler agent: organizer and attendee. Figure 13 shows the role's schemas for the organizer role while Figure 14 and 15 show the definitions of the protocols involved in its live-ness expression.

Role Name: Organizer
Description: This role organizes a meeting and sends invitations to attendees.
Protocols and Activities: ScheduleIt, InviteAttendees, Receive, Accept, Reject
Permissions: *//This agent does not have access to the resources directly*
Responsibilities: Liveness: Organizer = (ScheduleIt**.**InviteAttendees **.** Receive, Accept, Reject)^{w} Safety: * True

Figure 13: Schema for the Role Organizer

Role Name: Organizer		
Protocol Name: InviteAttendees		
Initiator:	Partner:	Input:
Organizer	EmailSender	Meeting Details
Description: It sends invitations to		Output: Invitation

Figure 14: Definition of InviteAttendees Protocol

Role Name: Organizer		
Protocol Name: ReceiveAcceptReject		
Initiator:	Partner:	Input:
EmailReceiver	Organizer	Invitation response
Description: This interaction occurs to deliver the invitation response to the organizer.		Output:

Figure 15: Definition of ReceiveAcceptReject Protocol

5. EXECUTION SCENARIO

Assume that a user wants to create a new task and inserts it into the database. The following execution steps happen:

1. Once s/he starts the prototype, the facilitator agent does not have any belief about the user's identity. Hence, the facilitator will ask the user to provide the user's name and password. If they are valid, then the facilitator will add this fact to its beliefs and redirect the user to the next step. Otherwise, s/he will be asked to enter the correct user's name and password again.
2. The facilitator agent displays a command frame to allow the user to type his/her command. Assume the user enters the command: 'Can you please create a task "Work on initial conference paper" starting 09/04/2008 and due date is 10/04/2008 with high importance and 30% completed'. Figure 16 shows a snapshot of the command.

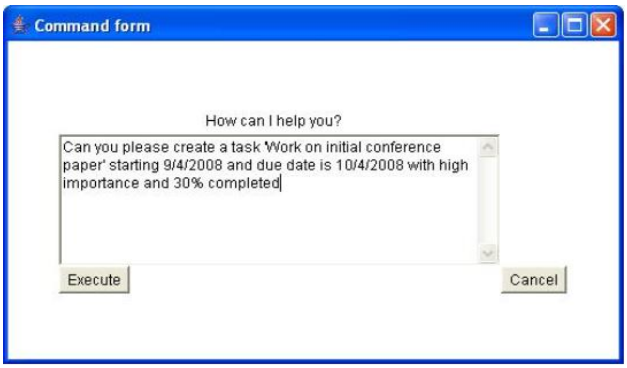

Figure 16: Command Form

3. When the user clicks on the Execute button, the facilitator agent will forward the command as it is to the second-level agents in the hierarchy of the agent model, asking them whether the command is related to any one of them.
4. Each second-level agent will study the command and check if it is related to itself or not. The result of this checking will be sent to the facilitator with the name of the responsible agent in third-level agents. In this scenario, the task agent will reply by saying the command is mine and the specific responsible third-level agent is the "TaskCreator" agent. Other second-level agents will reply by saying that the request is not related to them. In addition, the task agent parses the command and understands it.
5. The facilitator displays a task confirmation screen based on how the task agent has interpreted the command [Figure 17].
6. The user can accept the system's interpretation or change any values. Once the user clicks on the Save button, the facilitator agent will send the command to the 3rd responsible agent to execute it, which in our case is the "TaskCreator".

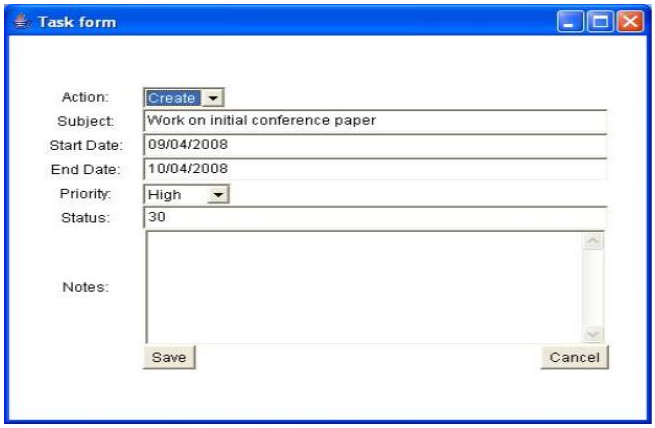

Figure 17: Task Form

6. CONCLUSION

In this paper, we have discussed the importance of following an agent-oriented software engineering methodology to build a high quality PIM model. Using Gaia methodology guided us throughout the different phases of developing an agent-oriented model of PIM and helped us in analyzing and designing the solution. The proposed model represents a foundation for sophisticated PIMs that have high degree of autonomous behavior and ability to learn. In addition, it is shown to be extensible.

7. ACKNOWLEDGEMENT

We would like to thank King Fahd University of Petroleum and Minerals for supporting this research work and providing the computing facilities.

REFERENCES

[1] Boardman, R. Improving Tool Support for Personal Information Management, 2004. Thesis Report for the degree of Doctor of Philosophy, pp. 30-60.

[2] Dumais, S., Cutrell, E., Cadiz, J., Jancke, G., Sarin, R., and Robbins, D. C. 2003. Stuff I've seen: a system for personal information retrieval and re-use. Proceedings of the 26th Annual international ACM SIGIR Conference on Research and Development in information Retrieval (Canada, July 28 - August 01). http://doi.acm.org/10.1145/860435.860451

[3] Mao, X., Wang, J., and Chen, J. 2005. Modeling Organization Structure of Multi-Agent System. In Proceedings of the IEEE/WIC/ACM international Conference on intelligent Agent Technology (Sept. 19-22), pp. 116-119. http://dx.doi.org/10.1109/IAT.2005.102

[4] Nikraz, M., Caire, G., and Bahri, P. A, 2006. A methodology for the analysis and design of multi-agent systems using JADE. Computer Systems Science and Engineering, 21(2).

[5] Luck, M., Ashri, R., and d'Inverno, M. 2004 Agent-Based Software Development. Artech House, Inc.

[6] Pressman, R. S. 2004 Software Engineering: a Practitioner's Approach. McGraw-Hill Science/Engineering/Math.

[7] Robertson, G., Czerwinski, M., Larson, K., Robbins, D. C., Thiel, D., and van Dantzich, M. 1998. Data Mountain: using spatial memory for document management. In Proceedings of the 11th Annual ACM Symposium on User interface Software and Technology (USA, November 01 - 04). pp. 153-162.http://doi.acm.org/10.1145/288392.288596

[8] Wooldridge, M., Jennings, N. R., and Kinny, D. 2000. The Gaia Methodology for Agent-Oriented Analysis and Design. Autonomous Agents and Multi-Agent Systems (Sep. 2000), pp. 285-312.http://dx.doi.org/10.1023/A:1010071910869

[9] Zambonelli, F., Jennings, N. R., and Wooldridge, M. 2003. Developing multi-agent systems: The Gaia methodology. ACM Trans. Software Engineering. Method. 12, 3 (Jul. 2003), 317-370. http://doi.acm.org/10.1145/958961.958963

[10] Karen Myers, Pauline Berry, Jim Blythe, Ken Conley, Melinda Gervasio, Deborah McGuinness, David Morley, Avi Pfeffer, Martha Pollack , and Milind Tambe, 2007. An Intelligent Personal Assistant for Task and Time Management, AI Magazine Volume 28 Number 2, pp. 47-61.

[11] Tarek Helmy, Makoto Amamiya, 2006. Multi-Agent-Based Adaptive AV Interface. The International Arab Journal of Information Technology, Vol. 3, No. 4, pp. 291-298.

[12] Tarek Helmy, 2006. Towards a User-Centric Web Portals Management, International Journal of Information Technology, Vol. 12 No. 1, pp. 1-15.

Agent-based Context Consistency Management in Smart Space Environment

Wan-rong Jih jih@agents.csie.ntu.edu.tw
Jane Yung-jen Hsu yjhsu@csie.ntu.edu.tw
Han-Wen Chang r96922005@ntu.edu.tw
Umi Laili Yuhana yuhana@its-sby.edu
Department of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan
Department of Informatics
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember (ITS)

ABSTRACT

Context-aware systems in a smart space environment must be aware of the surrounding contexts and adapt to changing contexts in highly dynamic environments. Data managements of contextual information are different from traditional approaches because of the contextual information are dynamic, transient, and fallible. Consequently, capabilities to detect context inconsistency and maintain consistent contextual information are two key issues for managing contexts. We propose an ontology-based model for representing, deducing, and managing consistent contextual information. In addition, we use ontology reasoning to detect and resolve context inconsistency problems, which will be described in a *Smart Alarm Clock* scenario.

1. INTRODUCTION

It's obvious that mobile devices, such as smart phone, personal digital assistants (PDAs), and wireless sensors, are increasingly popular. Moreover, many tiny, battery-powered, and wireless-enabled devices have been deployed in smart spaces for collecting contextual information of residents. Customized information can be delivered across mobile devices, based on specific contexts (location, time, environment, *etc.*) of the user. The Aware Home[1], Place Lab[17], Smart Meeting Room[6],and vehicles[19] provide intelligent and adaptive service environment for assisting users to concentrate on their specific tasks.

Context-awareness is the essential characteristic of a smart space, and using the technologies to achieve context-awareness is a type of intelligent computing. Within a richly equipped and networked environment, users need not carry any devices with them; instead, applications adapt the available resources to their processes for delivering services to vicinity of users, as well as tracking the location of users. Cyberguide[18] uses the user's locations to provide an interactive map service. In the Active Badge[23], every user wears a small infrared device, which generates a unique signal and can be used to identify the user. Xerox PARCTab[24] is a personal digital assistant that uses an infrared cellular network for communication. Bat Teleporting[15] is an ultrasound indoor location system.

In a smart space, augmented appliances, stationary computers, and mobile sensors can be used to capture raw contextual information (*e.g.* temperature, spatial data, network measurement, and environmental factor), and consequently a context-aware system needs to understand the meaning of a context. Therefore, a model to represent contextual information is the first issue of developing context-aware systems. Context-aware services require the high-level description about the user's states and environment situations. However, high-level context cannot be directly acquired from sensors. The capability to entail high-level contexts from the existing knowledge is required in context-aware systems. Consequently, how to derive hight-level contexts is the second issue. As we know that people may move to anywhere at anytime, it is increasingly important that computers develop a sense of location and context in order to appropriately respond to the user's needs. How to deliver right services to right places at the right time will be the third issue. Inconsistent contexts may appear in context-aware systems due to systems should react to the rapid change of contextual information. Any systems with inconsistent knowledge will cause them fail to provide correct services. Therefore, a context-aware system must maintain a consistency knowledge base and react to the dynamic change of contexts, which will be the fourth issue.

In this research, we leverage multi-agent and semantic web technologies that provides the means to express context and uses abstract representations to derive usable context for proactively delivering context-aware service to the user. We propose an ontology-base model for supporting context management, which can provide high-level context reasoning and detect the knowledge inconsistency. In addition, a *Smart Alarm Clock* scenario is help for describing the detailed of our research.

2. BACKGROUND TECHNOLOGIES

An overview of the context models, context reasoning, and ontology are introduced in this section.

2.1 Context Representation

Context is mainly characterized by four dimensions[9]: location, identity, activity and time. Location refers to the

exact position where the user is. If we know a person's identity, we could easily derive related information from several data sources such as birth date, social connectivity, or email addresses. Knowing the location of an entity, we could determine its nearby objects and people.

Many context-aware systems concentrate on location aware services. Ye *et al.*[26] use lattice model to represent spatial structure, which can deal with syntactic and semantic labels. This general spatial model provides both absolute and relative references for geographic positions, both the containment and connection relationships can be determined as well. MINDSWAP Group at University of Maryland Institute for Advanced Computer Studies develops Semantic geoStuff[1] to express basic geographic features such as countries, cities, and relationships between these spatial descriptors.

The RFC 2445[2] defines iCalendar format for calendaring and scheduling applications, which provides users to create personal activities. Google Calendar[3] is a popular web-based calendar supports iCalendar standard and users can share their own personal activities with others. These human activities are related to people, time, and location. Consequently, the contents of persons' schedules can help us to derive their location at a given time.

2.2 Ontology

Strang and Linnhoff-popien[21] concluded that the ontology are the most expressive model. Gruber[13] defines ontology as an "explicit specification of a conceptualization". Ontology is developed to capture the conceptual understanding of the domain in a generic way and provide a semantic basis for grounding the fine-grained knowledge.

COBRA-ONT[5] provides key requirements for modeling context in a smart meeting application. It defines concepts and relations of physical locations, time, people, software agents, mobile devices, and meeting events. SOUPA[7] (Standard Ontology for Ubiquitous and Pervasive Applications) uses some other standard domain ontologies, such as FOAF[4] (Friend of A Friend), OpenGIS, spatial relations in OpenCyc, ISO 8601 date and time formats[5], and DAML time ontology[16]. Clearly, these ontologies provide not only a rich context representation, but also make use of the abilities of reasoning and sharing knowledge.

2.3 Context reasoning

Design and implementation of context reasoning can vary depending on types of contextual information that are involved. Early context-aware systems[8, 25, 3] are tightly coded logics of context reasoning into the behavior of systems. Implementation for understanding the contextual information is bound into the programs. Therefore, developed applications often have rigid implementations and are difficult to maintain.

Rule-based logical inference can help to develop flexible context-aware systems by separating high-level context reasoning from low-level system behaviors. However, context modeling languages are used to represent contextual information and the rule languages are used to enable context reasoning. Accordingly, in most cases, these two types of languages have different syntax and semantic representations; it is a challenge that effectively integrates these distinctive languages to support context-aware systems. A mechanism to convert between contextual modeling and reasoning languages is one of solutions for this challenge. Gandon and Sadeh[11, 12] propose e-Wallet that implements ontologies as context repositories and uses a rule engine Jess[10] to invoke the corresponding access control rules. The e-Wallet using RDF[6] triples to represent contextual information and OWL[7] to define context ontology. Contextual information is loaded into the e-Wallet by using a set of XSLT[8] stylesheets to translate OWL input files into Jess assertions and rules.

Ontology models can represent contextual information and specify concepts, subconcepts, relations, properties, and facts in a smart space. Moreover, ontologies reasoning can use these relations to infer the facts that are not explicitly stated in the knowledge base. Ranganathan*et al.*[20] propose that ontologies can make it easier to develop programs for reasoning about context. Chen[4] proposes that the OWL language can provide a uniformed solution for context representation and reasoning, knowledge sharing, and meta-language definitions. Anagnostopoulos *et al.*[2] adopt the Description Logicas the most useful language for expressing and reasoning contextual knowledge. The OWL DL was designed to support the existing Description Logic business segment and has desirable computational properties for reasoning systems. Typical ontology-based context-aware application is EasyMeeting that uses OWL to define the SOUPA ontology and OWL DL to support context reasoning. Gu *et al.*[14, 22] propose an OWL encoded context ontology CONON in Service Orientated Context Aware Middleware (SOCAM). CONON consists two layers of ontologies, an upper ontology that focuses on capturing general concepts and a domain specific ontology. EasyMeeting and SOCAM are use an OWL DL reasoning engine to check the consistency of contextual information and provide further reasoning over low-level context to derive high-level context.

3. SYSTEM ARCHITECTURE

Figure 1 shows our Context-aware System Architecture, which can continuously proceeds changing contexts and proactively provides services to the user. The top part of Figure 1 depicts a smart space environment, which equipped with devices and applications, such as personal calendar, weather forecasts, location tracking system, contact list, and shopping list, as well as raw sensing data, can provide contextual information and deliver context-aware services. The Context Collection Agents obtain raw sensing data from the context sources and convert the raw context into a semantic representation. Each Context Collection Agent will deliver the sensed contextual information to the *Context Management* after receiving sensing data.

[1] `http://www.mindswap.org/2004/geo/geoStuff.shtml`
[2] `http://tools.ietf.org/html/rfc2445`
[3] `http://calendar.google.com`
[4] `http://xmlns.com/foaf/spec/`
[5] `http://www.w3.org/TR/NOTE-datetime`
[6] `http://www.w3.org/TR/rdf-concepts/`
[7] `http://www.w3.org/TR/owl-features/`
[8] `http://www.w3.org/TR/xslt`

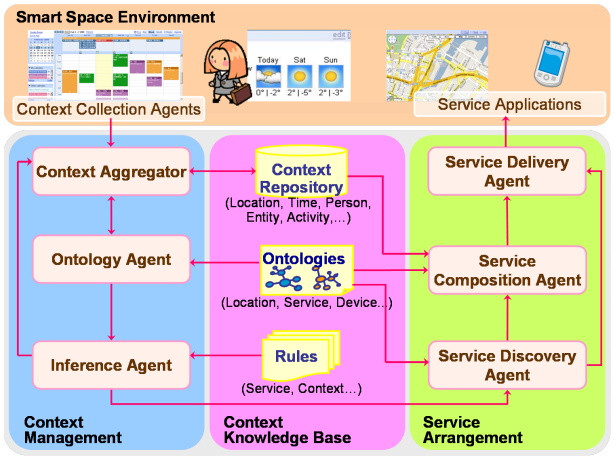

Figure 1: A Multi-agent System Architecture

The lower part of Figure 1 illustrates our context-aware system architecture, which consists of three components: *Context Management*, *Context Knowledge Base*, and *Service Arrangement*. Functions of *Context Management* are monitoring the contextual information and managing the environmental resources. Contextual information, domain knowledge, and service profiles are stored in *Context Knowledge Base*. *Service Arrangement* performs service discovery, composition, and execution. After assign the specified services, the system will invoke Service Applications to provide service in the smart space environment.

4. CONTEXT-AWARE AGENTS

Agents in Figure 1 accomplish the functions of managing contextual information and delivering context-aware services. Functions of *Context Management* include gathers contexts from the surrounding environment, provides methods for querying and storing the contextual information, and provides the context in an ontology-based representation that facilities knowledge representation and inference. *Service Arrangement* checks whether any service operation can match the request under the current situation. If no operation matches the request, it combines operations to match the request.

4.1 Context Aggregator

Context Aggregator collects contextual information from Context Collection Agents and stores the context to the Context Repository for context inference, consistency checking, and knowledge sharing. There are two types of input context, the raw context and the high-level context. Raw context refers to the sensing data which directly obtained from context sources. For example, bed sensors can provide lay-on-bed sensing and weather forecast API can provide forecasting information. Context Aggregator subscribes to the specified Context Collection Agents for retrieving the contextual information, which define in the context ontology. Consequently, Context Collection Agents are the providers of low-level contexts while the high-level contexts are derived from the Ontology Agent and Inference Agent.

4.2 Ontology Agent

Ontology Agent loads and parses an OWL context ontology into RDF triples, which makes other agents able to represent and share context in the system. Context Aggregator sends the current state of contexts to Ontology Agent while the subscribed context changed. The other agents can send their queries to Ontology Agent for retrieving the updated knowledge. According to the structures and relationships between contexts that define in the context ontology, the Ontology Agent performs the subsumption reasoning for deducing new contextual information. For example, it can deduce the superclasses of a specified class and decides whether one class is subsumed by another, *e.g.*, a building may spatially subsume a room.

4.3 Inference Agent

Inference Agent adopts an OWL DL reasoning engine for supporting context reasoning and conflict detection. When Inference Agent receives contextual information from Ontology Agent, the reasoning engine will fire rules and trigger actions that may deduce new high-level contexts and derive service requests. Combining the inferred contexts with the original context ontology, Inference Agent can detect the context inconsistency. Either new high-level contexts or service requests can be derived from Inference Agent and deliver to Context Aggregator or Service Discovery Agent, respectively.

4.4 Service Discovery Agent

Service Discovery Agent maintains the service ontology. An OWL-S[9] file defines the service ontology, which includes three essential types of knowledge about a service: service profile, process model, and service grounding. The OWL-S service profile illustrates the preconditions required by the service and the expected effects that result from the execution of the service. A process model describes how services interact and how the functionalities offer, which can be exploited to solve the goals. The role of service grounding is to provide concrete details of message formats and protocols. According to the description in service ontology, Service Discovery Agent keeps the atomic services information. When the Service Discovery Agent receives a service request, it checks whether any single service satisfies the requirement under the current situation. If an atomic service can accomplish the request, the associated service grounding information will be delivered to the Service Delivery Agent.

4.5 Service Composition Agent

If a service request cannot be achieved by a single service, Service Composition Agent will compose atomic services to fulfill the request. The service profile of a service ontology defines the service goals, preconditions, and effects. According to these semantic annotations, AI planning has been investigated for composing services. The state transition is defined by the operations, which consist of preconditions and effects. Initial states of the AI planner are combined the current contexts and context ontology. The service request is the planning goal. Therefore, giving initial states, goals, and operations, Service Composition Agent will derive a service execution plan, which is a sequence of operations that starts from initial states and accomplishes the given goal.

[9] `http://www.w3.org/Submission/OWL-S/`

4.6 Service Delivery agent

Service ontology defines the information for service grounding, which specifies the details of how an agent can access a service. According to the description of service grounding, Service Delivery Agent invokes the specified Service Application with the required protocol and message contents.

5. CONTEXT ONTOLOGY MODEL

Context-aware applications need a unified context model that is flexible, extendible, and expressive to adapt the variety of context features and dependency relations. The ontology models can fulfill these requirements; therefore, we deploy an ontology context model to represent contextual information in smart space environment. The ontology is inspired by the need to share knowledge about locations, time, and activities so that context-aware applications can infer the environmental contexts and trigger services.

5.1 Context Repository

Context Repository stores a set of consistent context, which including location, person, and activity information. Either raw or high-level context has a unique type identity and value. The associated value is the timestamp represents when the corresponding context is arrived. Context ontology defines the classes of contexts and the relationships between the instances of context objects. A RDF-triple represents a context that contains a subject, a predicate, and an object. Subject is a resource named by a URI with an optional anchor identity. The predicate is a property of the resource, and the object is the value of the property for the resource. For example, the following triple represents "Peter is sleeping".

```
<http://...#Peter>
<http://...#participatesIn>
<http://...#sleeping>
```

Where `Peter` represents subject, `participatesIn` is a predicate, and the activity `sleeping` is an object. We use subject and predicate as the compound key of Context Repository. When a context has been updated, the associated timestamp will be changed accordingly.

5.2 Ontologies

An ontology is a data model that represents a domain and is used to reason about the objects in that domain and their relations. We define a context ontology depicts in Figure 2 as a representation of common concepts about the smart space environment. Context information are collected from real-world classes (Person, Location, Sensor, Time, HomeEntity), and a conceptual class Activity. The class hierarchy represents an `is-a` relation; an arrow points from a subclass to another superclass. A class can have subclasses that represent the concepts more specific than their superclass. For example, we can divide the classes of all locations into indoor and outdoor locations, that is, Indoor Location and Outdoor Location are two disjoint classes and both of them belong to Location class. In addition, the subclass relation is transitive, therefore, the Livingroom is a subclass of Location class because Livingroom is a subclass of Indoor and Indoor is a subclass of Location.

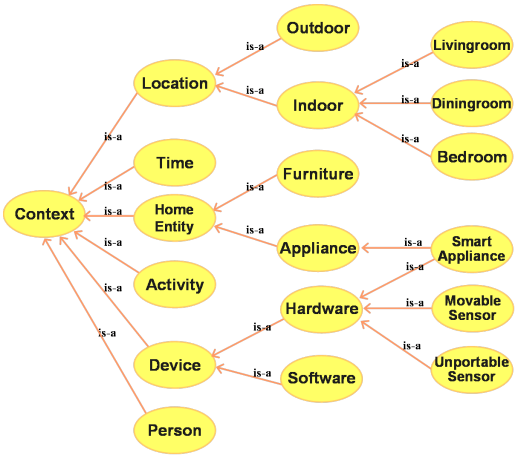

Figure 2: A Context Ontology

The relationship between classes is illustrated in Figure 3. The solid arrows describe relation between subject resources and object resources. For example, isLocatedIn describes the relation between the instances of Person and Location while the instances of Person is the subject resources and instances of Location is the object resources.

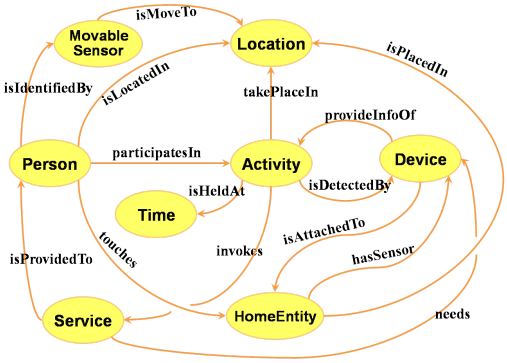

Figure 3: Context Relationship

A service ontology defined by OWL-S is for describing available services that comprises service profile, service model, and service grounding.

5.3 Rules

Rules of a rule-based system serve as IF-THEN statements. Context rules can be triggered to infer high-level context. According to the description of Figure 3, a rule for detecting the location of a user is showed as follows:

```
[Person_Location:
  (?person isIdentifiedBy ?tag)
  (?tag isMoveTo ?room)
->
  (?person isLocatedIn ?room )
]
```

Patterns before -> are the conditions, matched by a specific rule, called left hand side (LHS) of the rule. On the other hand, patterns after the -> are the statements that may be fired, called right hand side (RHS) of the rule. If all the LHS conditions are matched, the actions of RHS will be executed. The RHS statement can be either asserted new high-level contexts or delivered a service request.

Rule `Person_Location` is an example that can deduce high-level context. The `?person` is an instance of class Person, `?tag` is an instance of MovableSensor, and `?room` is an instance of Room, the rule `Person_Location` declares that if any person `?person` is identified by a movable sensor `?tag` and this movable sensor is move to a room `?room`, we can deduce that `?person` is located in `?room`.

6. CONTEXT MANAGEMENT AND REASONING MECHANISM

In order to make our research easier to understand, we use a simple example to describe the detail mechanism of context management and reasoning.

> *In a smart space, a Smart Alarm Clock can check Peter's schedule and automatically set the wake-up alarm for helping him not miss his daily first task. If Peter does not wake up within 5-minute period after the alarm is sent, send another sound of alarm and increases its volume. If Peter wake up earlier then the alarm time, the alarm will be disabled.*

6.1 Context Reasoning

In order to archive *Smart Alarm Clock*, we have to collect Peter's schedule to decide the alarm time and should reasoning whether Peter is awake or not. Google Calendar Data API[10] can support the information of Peter's calendar events. The position-aware sensors, bed pressure sensors, *etc.* can help to detect whether user on the bed or not. For example, RFID technologiescan be used to recognize and identify the activities of Peter. Using a wireless-based indoor location tracking system can determine Peter's location with room-level precision.

Figure 4 shows the instance relationships for detecting whether Peter is currently sleeping or not. The word within an oval represents a class and the box represents an instance of the corresponding class. For example, bed is an instance of Furniture class. Dashed line indicates the connection of a class and its instance. Each solid arrow reflects the direction of object property relationship that directs from domain to range. In addition, an inverse property can be declared while reverse the direction of a line. For example, the inverse object property of `isAttachedTo` is `hasSensor`.

A boolean data type property isOn is associated with Sensor class for detecting whether the value of instances is on or off. If someone is on the bed, value of the sensor bed_sensor will be on, that is, the value of isOn is `true`. Otherwise, when nobody touches the bed, the value of isOn has to be `false`. When an event of wake-up call has been triggered, a rule

[10] `http://code.google.com/apis/calendar/`

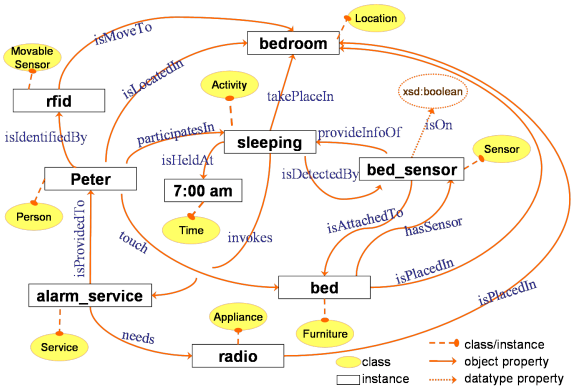

Figure 4: A Context Snapshot

for detecting the value of bed_sensor can be used to decide whether it is necessary to deliver the alarm_service or not.

For reasoning high-level contexts, we apply rule-based reasoning with horn clauses into the ontology model. The rule `Person_activity` can deduce what's the activity the user currently involved. For example, in Figure 4, when the time is up, given the location of Peter and the status of bed sensor, the rule `Person_activity` will be triggered and can deduce whether Peter is sleeping or not. The rule `Invoke_service` can deduce what's the service for delivering to the user. Given the instances of Figure 4, the rule `Invoke_service` reflects "if Peter is sleeping, deliver smart alarm service".

```
[Person_activity:
  (?person touch ?entity)
  (?entity hasSensor ?sensor)
  (?sensor providesInfoOf ?activity)
->
  (?person participatesIn ?activity) ]


[Invoke_service:
  (?person participatesIn ?activity)
  (?activity invokes ?service)
->
  (?service isProvidedTo ?person)]
```

6.2 Context Management

Changes of environmental contexts are transient in the sense of that any context may appear and vanish at anytime. Algorithm 1 shows how the Context Aggregator manages the contextual information.

We use RDF-triple to represent a context while an associated compound key comprises the subject and object. When a new context is arrived, Context Aggregator uses the key of new context to query Context Repository. If a context exists in Context Repository and the associated predicate represents one-to-one relationship, the new context will replace the old one. Otherwise, the new context will be inserted into Context Repository. Functions $update(key_c, c)$ and $insert(key_c, c)$ perform the context replacement and insertion, respectively. When a context is vanished, it should

Algorithm 1 Maintaining Context Repository

Input: c is the new context
C: Context Repository
rdf_i: RDF-triple (s_i, p_i, o_i) of a context i
key_i: key of context i in Context Repository
for all $i \in C$ **do**
  if $isOutdated(i)$ **then**
    $delete(i)$
  end if
end for
if $\exists i \in C$ s.t. $key_i = key_c$ and $isOne2One(p_c)$ **then**
  $update(key_c, c)$
else
  $insert(key_c, c)$
end if

be removed. function $delete(i)$ can remove the specified context from Context Repository. We use a decay function to determine the existence of a context. Different context is associated with a different decay function. This function can either be an objective function for predicating a specified activity or simply be a constant function. The function $isOutdated(i)$ apply the context decay function to decide whether the context is existed or not.

6.3 Inconsistency Resolution

The Context Repository is dynamically updated for reflecting the change of context. Therefore, we must ensure incorrect or outdated contexts are not existed in Context Repository. If a raw context is changed, some of the inferred high-level contexts may be changed. For example, if Peter walks from living room to bedroom, the corresponding RDF-triple will be changed from `<Peter isLocatedIn living_room>` to `<Peter isLocatedIn bedroom>`. Context Aggregator will update the location context of Peter because the property isLocatedIn is one-to-one relationship. If a predicate allow multiple relationships, the original context will be reserved.

It is a challenge that when a raw context is changed, we need to updated the associated high-level contexts. However, it is hard to find the corresponding high-level contexts by using the context dependency of inference rules. Updating a context may easily trigger the infinite context dependency checking and can lead to unpredictable situations. We categorize the data in Context Repository to three types: core knowledge, raw-level context, and high-level context. The OWL ontologies define the contents of core knowledge that are static and persistent. The raw-level context is the raw sensing data that delivers from the Context Collection Agents in Figure 1. Using the core knowledge and raw-level context, a rule-based reasoning can deduce high-level contexts. When a raw context has been removed, we discard the original set of high-level context and perform context reasoning. Clearly, the deduced high-level contexts are consistent with the current raw contexts and the Context Repository can maintain the context consistency. This approach is simple, but can efficiently resolve the context inconsistency without recursively check the context dependency.

7. IMPLEMENTATION

Our agent is deployed on JADE[11] (Java Agent DEvelopment Framework), which is a FIPA-compliant software framework for multi-agent systems, implemented in Java and comprised several agents. Jena[12] is a Java framework for building Semantic Web applications, is used for providing a programmatic environment for RDF, RDFS, and OWL. Moreover, we use an open-source OWL Description Logics (OWL DL) reasoner Pellet[13] that developed by Mindswap Lab at University of Maryland, to infer high-level contexts and detect context conflicts.

8. CONCLUSION AND FUTURE WORK

This research presents a context management mechanism in a smart space. We integrate context-aware technologies, semantic web, and logical reasoning for providing context-aware services. An ontology-based model supports reasoning mechanism, which can deduce high-level contexts and detect context consistency.

We use a simple scenario to demonstrate the mechanism of context management. However, this simple case does not show the power of context reasoning. Therefore, design other scenarios that can explain and evaluate our approach is one of our future direction.

9. REFERENCES

[1] G. D. Abowd, C. G. Atkeson, A. F. Bobick, I. A. Essa, B. MacIntyre, E. D. Mynatt, and T. E. Starner. Living laboratories: the future computing environments group at the georgia institute of technology. In *Proceedings of Conference on Human Factors in Computing Systems (CHI '00): extended abstracts on Human factors in computing systems*, pages 215–216, New York, NY, USA, 2000. ACM Press.

[2] C. B. Anagnostopoulos, A. Tsounis, and S. Hadjiefthymiades. Context awareness in mobile computing environments. *Wireless Personal Communications: An International Journal*, 42(3):445–464, 2007.

[3] L. Capra, W. Emmerich, and C. Mascolo. CARISMA: Context-aware reflective mIddleware system for mobile applications. *IEEE Transactions on Software Engineering*, 29(10):929 – 945, 2003.

[4] H. Chen. *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*. PhD thesis, University of Maryland, Baltimore County, 2004.

[5] H. Chen, T. Finin, and A. Joshi. An ontology for context-aware pervasive computing environments. *The Knowledge Engineering Review*, 18(3):197–207, September 2003.

[6] H. Chen, T. Finin, A. Joshi, L. Kagal, F. Perich, and D. Chakraborty. Intelligent agents meet the semantic web in smart spaces. *IEEE Internet Computing*, 8(6):69–79, November – December 2004.

[7] H. Chen, F. Perich, T. Finin, and A. Joshi. SOUPA: Standard ontology for ubiquitous and pervasive applications. In *The First Annual International Conference on Mobile and Ubiquitous Systems:*

[11] `http://jade.tilab.com/`
[12] `http://jena.sourceforge.net/`
[13] `http://pellet.owldl.com/`

Networking and Services (MobiQuitous'04), pages 258–267, August 2004.

[8] M. H. Coen. Building brains for rooms: designing distributed software agents. In *Proceedings of the Conference on Innovative Applications of Artificial Intelligence (IAAI'97)*, pages 971–977. AAAI Press, 1997.

[9] A. K. Dey. *Providing architectural support for building context-aware applications*. PhD thesis, Georgia Institute of Technology, 2000. Director-Gregory D. Abowd.

[10] E. Friedman-Hill. *Jess in Action: Java Rule-Based Systems*. Manning Publications, Greenwich, CT, USA, 2003.

[11] F. L. Gandon and N. M. Sadeh. A semantic e-wallet to reconcile privacy and context awareness. *Lecture Notes in Computer Science: The SemanticWeb (ISWC 2003)*, 2870:385–401, October 2003.

[12] F. L. Gandon and N. M. Sadeh. Semantic web technologies to reconcile privacy and context awareness. *Journal of Web Semantics*, 1(3):241–260, 2004.

[13] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, June 1993. Special issue: Current issues in knowledge modeling.

[14] T. Gu, X. H. Wang, H. K. Pung, and D. Q. Zhang. An ontology-based context model in intelligent environments. In *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*, pages 270–275, 2004.

[15] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster. The anatomy of a context-aware application. *Wireless Networks*, 8(2 – 3):187 – 197, March – May 2002.

[16] J. R. Hobbs and F. Pan. An ontology of time for the semantic web. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(1):66–85, 2004. Special Issue on Temporal Information Processing.

[17] S. S. Intille. Designing a home of the future. *IEEE Pervasive Computing*, 1(2):76–82, April 2002.

[18] S. Long, D. Aust, G. Abowd, and C. Atkeson. Cyberguide: prototyping context-aware mobile applications. In *Conference companion on Human factors in computing systems (CHI '96)*, pages 293 – 294, Vancouver, British Columbia, Canada, April 13 – 18 1996. ACM Press.

[19] G. Look and H. Shrobe. A plan-based mission control center for autonomous vehicles. In *IUI '04: Proceedings of the 9th international conference on Intelligent user interfaces*, pages 277–279, New York, NY, USA, 2004. ACM Press.

[20] A. Ranganathan, J. Al-Muhtadi, and R. H. Campbell. Reasoning about uncertain contexts in pervasive computing environments. *IEEE Pervasive Computing*, 3(2):62–70, 2004.

[21] T. Strang and C. Linnhoff-popien. A context modeling survey. In *Workshop on Advanced Context Modelling, Reasoning and Management at The Sixth International Conference on Ubiquitous Computing (UbiComp 2004)*, Nottingham, England, 2004.

[22] X. H. Wang, D. Q. Zhang, T. Gu, and H. K. Pung. Ontology based context modeling and reasoning using OWL. In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW '04)*, page 18, Washington, DC, USA, 2004. IEEE Computer Society.

[23] R. Want, A. Hopper, V. Falcão, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems (TOIS)*, 10(1):91–102, January 1992.

[24] R. Want, B. N. Schilit, N. I. Adams, R. Gold, K. Petersen, D. Goldberg, J. R. Ellis, and M. Weiser. An overview of the PARCTAB ubiquitous computing experiment. *Personal Communications*, 2(6):28 – 43, December 1995.

[25] H. Wu, M. Siegel, and S. Ablay. Sensor fusion for context understanding. In *Proceedings of IEEE Instrumentation and Measurement Technology Conference*, Anchorage, AK, USA, May 21 – 23 2002.

[26] J. Ye, L. Coyle, S. Dobson, and P. Nixon. A unified semantics space model. In J. Hightower, B. Schiele, and T. Strang, editors, *Proceedings of the 3rd International Symposium on location- and Context-Awareness (LoCA 2007)*, volume 4718 of *Lecture Notes in Computer Science*, pages 103–120, September 2007.

Agent-Based Support for Context-Aware Provisioning of IMS-Enabled Ubiquitous Services

Ana Petric
University of Zagreb
Faculty of Electrical
Engineering and Computing
Unska 3, Zagreb, Croatia
ana.petric@fer.hr

Krunoslav Trzec
Ericsson Nikola Tesla
Krapinska 45, Zagreb, Croatia
krunoslav.trzec@
ericsson.com

Kresimir Jurasovic
University of Zagreb
Faculty of Electrical
Engineering and Computing
Unska 3, Zagreb, Croatia
kresimir.jurasovic@fer.hr

Vedran Podobnik
University of Zagreb
Faculty of Electrical
Engineering and Computing
Unska 3, Zagreb, Croatia
vedran.podobnik@fer.hr

Gordan Jezic
University of Zagreb
Faculty of Electrical
Engineering and Computing
Unska 3, Zagreb, Croatia
gordan.jezic@fer.hr

Mario Kusek
University of Zagreb
Faculty of Electrical
Engineering and Computing
Unska 3, Zagreb, Croatia
mario.kusek@fer.hr

ABSTRACT

Multimedia applications executed on mobile devices allow users to be present and communicate with other users, anywhere and anytime, through wide area cellular networks, wireless local area networks (WLAN), or fixed networks. In order to enable ubiquitous personalized services, communication systems should allow users to specify their context, or their mobile devices and network infrastructures should automatically sense their context and offer enhanced service provisioning solutions. In this paper, we propose an agent-based solution that supports context-aware provisioning of IP multimedia subsystem (IMS)-enabled ubiquitous services. Using agent technology, multimedia communication, controlled by SIP and enriched with context-related events delivered by SIP's event mechanism, can be seamlessly provisioned, taking into account not only mobility issues and context-awareness, but also the semantics of exchanged events. This paper proposes a multi-agent system that will enable users to consume context-aware ubiquitous services in a seamless (i.e., automated) way, providing optimal provisioning of SIP-based multimedia services according to user preferences.

Categories and Subject Descriptors

D.2.11 [**Software Architectures**]: Domain-specific Architectures; H.3.4 [**Systems and Software**]: Distributed systems; I.2.11 [**Distributed Artificial Intelligence**]: Intelligent Agents, Multiagent Systems

General Terms

Management, Performance, Design

Cite as: Agent-Based Support for Context-Aware Provisioning of IMS-Enabled Ubiquitous Services, Ana Petric, Krunoslav Trzec, Kresimir Jurasovic, Vedran Podobnik, Gordan Jezic, Mario Kusek and Igor Ljubi, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

Keywords

context-aware service provisioning, IMS-enabled services, session mobility, multi-agent system

1. INTRODUCTION

The advent of the Internet and the development of the Next Generation Network (NGN) is enabling a lifestyle aspiring to digital humanism where people's daily activities are becoming more digitalized, convenient and intelligent [20]. Actors on telecom markets are pursuing innovations and launching new value-added services (VAS) [4] in order to increase revenue. This new market demand and technological development has led to the convergence of different domains (i.e., telecommunications, information technology (IT), the Internet, broadcasting and media), all involved in the telecom service provisioning process. The ability to transfer information embodied in different media into digital form to be deployed across multiple technologies is considered to be the most fundamental enabler of convergence [8].

The evolved network should aim at taking changing customer demands into account and creating spontaneous, adaptive services that can be delivered anytime, anywhere, to any device the user prefers. Therefore, the realization of the full potential of convergence will make it necessary for operators to deploy a dynamic, cooperative and business-aware consistent knowledge layer in the network architecture in order to enable ubiquitous personalized services. Providing such context-aware services transparently to the user is not only challenging from a network point of view, but also places severe requirements on service provisioning. This is particularly true when the service is to be accessed across several administrative domains, i.e., in a multi-provider environment.

The number of telecom value-added service consumers is rising continuously. Moreover, the competition among stakeholders in the telecom market, as well as the NGN concept which introduces a whole new spectrum of services, enables consumers to be very picky. Consequently, realization of the full potential of the NGN will make it necessary for service providers to offer dynamic, ubiquitous and context-

aware personalized services. Moreover, a large part of NGN services will provide multimedia sessions which will be composed of different audio and/or video communications with a certain quality of service (QoS) [6]. Providing such services to consumers transparently is challenging from the technical, business and social points of view.

The rest of the paper is structured as follows. Section 2 addresses the position of our multi-agent system in the telecommunications environment it is placed in. Furthermore, it gives an overview of the main technological foundations used in our proof-of-concept prototype which enables advanced service provisioning options in the NGN. Section 3 describes the architecture and the main features of our multi-agent system, while Section 4 presents the agent-based personalized session mobility service. Section 5 concludes the paper and gives an outline for future work.

2. TECHNOLOGICAL FOUNDATIONS

Our proof-of-concept prototype is placed in a telecommunications environment and it is based on the following technological foundations presented in this section: the IP Multimedia System (IMS), the Session Initiation Protocol (SIP), the Java Agent DEvelopment (JADE) framework, the Java Expert System Shell (JESS) and Asterisk. In the designed multi-agent system, intelligent software agents use different technologies (e.g., IMS, JADE, JESS, Asterisk) to enable personalized service provisioning, automated coordination of network operator's operations and automated interaction between all entities in an NGN environment.

2.1 Telecommunications Environment

Today we are witnessing the fusion of the Internet and mobile networks into a single, but extremely prominent and globally ubiquitous, technology: the Network [15]. The Network will enable the transformation of physical spaces into computationally active and intelligent environments [19], characterized by ambient intelligence where devices embedded in the environment provide seamless connectivity and services at all times. The vision of the Network is becoming a reality with the new generation of communication systems: the NGN [12].

One of the fundamental principals in the NGN is the separation of services from transport [1]. This separation represents a horizontal relationship in the NGN where the transport stratum and the service stratum can be distinguished. The transport stratum encompasses the technical processes that enable three types of connectivity: user-to-user, user-to-service platform and service platform-to-service platform connectivity. On the other hand, the service stratum is comprised of business processes that enable (advanced) telecom service provisioning assuming that the earlier stated types of connectivity already exist. Each stratum can have multiple layers, representing vertical relationships, where each layer can be distinguished into a data (or user) plane, a control plane and a management plane. In our model, we introduce intelligent agents into the control plane and the management plane of the service stratum. These agents are in charge of gathering context information that is required for service personalization (i.e., service management) and facilitating personalized application-level mobility (i.e., service control).

2.2 IP Multimedia System

The first step in seamless provisioning of personalized ubiquities services is enabled by deployment of the IP Multimedia System (IMS) with the aim of offering IP-based real-time multimedia services. IMS enables the convergence of mobile and wireline networks into a single unified infrastructure in all its forms by supporting services independent of access. It has a layered structure comprised of 1) a service layer, 2) a control layer, and 3) a connectivity layer. The service layer consists of application and content servers which execute value-added services for a user. The control layer is composed of network control servers for managing call or session set-up, modification and release. The most important is the Call Session Control Function (CSCF). This layer also contains a full suite of support functions, such as provisioning, charging and operations and maintenance (O&M). Interworking with other operators' networks and/or other types of networks is handled by border gateways. The connectivity layer is comprised of routers and switches, both for the backbone and for the access network. IMS takes the concept of layered architecture one step further by defining a horizontal architecture where service enablers and common functions can be reused for multiple applications.

With IMS, user personal services are achieved via a dynamically associated, user-centric, service independent and standardized access point, the CSCF. The service architecture is user-centric and is highly scalable. However, IMS does not have a common framework for context-awareness across all functions in the control layer in order to automatically adapt service availability and delivery to heterogeneous networks and dynamically changing environments. In order to enable seamless provisioning of personalized ubiquities services in a converged telecom network, we propose an agent-based common framework for context- awareness.

2.3 Session Initiation Protocol

By applying SIP for call/session control in the mobile Internet, personalized ubiquitous services can incorporate all types of media, from continuous media to application sharing. Besides support for multimedia information, SIP enables seamless integration of user devices such as 3G mobile phones and laptops. Moreover, SIP enables integration of devices with resources embedded in the users environment, such as video projectors, video cameras, and loudspeakers. Active multimedia sessions can be moved from one device to another or can be split across devices [16, 2].

In addition to mobility support, SIP enables deployment of context-aware services by utilizing its event mechanism extension which scales to a large number of users spread across different administrative domains. SIP event mechanism can be used for exchange of context-related information such as a user's current location, device capabilities, network characteristics, a user's interests, presence, time of day, etc. It is important that the exchange of context information is as privacy-conscious as possible (i.e., users should control the which part of their context information is revealed to others). Furthermore, wherever possible, context-aware services should take advantage of user defined policies, rather than requiring direct user interaction. For example, policies can be triggered dynamically by presence and location information. By using agent technology and SIP, multimedia communication can be seamlessly provisioned, taking into account not only mobility issues and context-awareness, but

also the semantics of exchanged events. This can be accomplished through the introduction of formally (i.e., ontology-based) described metadata representing SIP events which enables intelligent agents to understand context information and dynamically trigger provisioning operations. Therefore, we propose building intelligent agents (Personal Assistants) that will enable users to consume context-aware ubiquitous services in a seamless way, providing enhanced provisioning of SIP-based multimedia services according to user preferences.

2.4 Java Agent DEvelopment Framework

The proposed multi-agent system was implemented using the Java Agent DEvelopment (JADE[1]) framework. JADE is a software framework used for developing agent-based applications in compliance with the Foundation for Intelligent Physical Agents (FIPA[2]) specifications. These specifications define standard agent interaction protocols and other key aspects of a multi-agent system allowing interaction between different agent platforms and software components.

JADE agents are identified by a unique name (Agent IDentifier - AID). Assuming that the agents know each others names, they can communicate transparently regardless of their actual location. Communication is performed using Agent Communication Language (ACL) messages which agents exchange between themselves. The architecture is based on two agents that are initiated each time the platform is started - the Agent Management Service (AMS) agent and the Directory Facilitator (DF) agent. The AMS agent exerts supervisory control over access and use of the platform. The DF agent provides a Yellow Pages service which enables agents to find other agents providing the services they requires in order to achieve their goals. The third component of the platform is the Agent Communication Channel (ACC). The ACC is a Message Transport System that controls the exchange of all messages within the platform, including messages to/from remote platforms.

2.5 Java Expert System Shell

Rule-based programming is appropriate for problems that are difficult to solve using traditional algorithmic methods [9]. Consequently, it has to be executed by a of run-time system that understands how to control its flow and how to use declarative information to solve problems. A rule-based program does not consist of one long sequence of instructions; instead, it is made up of discrete rules, each of which applies to a subset of the problem. A rule engine determines which rules apply at any given time and executes them accordingly.

Our service provisioning solution has adopted the Java Expert System Shell (JESS)[3] as both a rule engine and a scripting language for the specification of rules. JESS was developed at Sandia National Laboratories in Livermore, California in late 1990s [9]. Using Jess, Java software can be built with the capacity to "reason" using knowledge that the programmer supplies in the form of declarative rules. JESS rules are very similar to 'if-then' statements in traditional programming languages. The rule-based system uses rules to reach conclusions from a set of premises. A JESS rule-based system consists of a working memory, an inference engine and a rule base. The rule base contains all the rules that the system knows, while the working memory contains all the facts that the system works with. The inference engine has three components: a pattern matcher, an agenda and an execution engine. The pattern matcher decides which rules should be activated during the current cycle by comparing all of them with the facts currently present in the working memory. This list of rules is then stored in the agenda. The execution engine executes the first rule from the list, along with adding removing or modifying existing facts in the working memory if necessary. The entire process is then repeated.

[1]http://jade.tilab.com
[2]http://www.fipa.org
[3]http://herzberg.ca.sandia.gov/jess/

2.6 Asterisk

Asterisk[4] is an open source telephony engine and toolkit. It was originally created in 1999 and it represents an implementation of a telephone Private Branch eXchange (PBX[5]). Telephones connected to Asterisk can access the Public Switched Telephone Network (PSTN) and Voice over Internet Protocol (VoIP) services. It provides SIP and PSTN-based phones with call and session functionalities. Every SIP phone can register through it and can use it to call other SIP and PSTN phones. Asterisk also provides handlers for defining presence and session information since it can send event messages to components in the system when such events occur. In this paper, we consider the following two compliant Asterisk interfaces which we use in our prototype: The Asterisk Management Interface (AMI) which allows our multi-agent system to control and monitor the Asterisk system; and the Asterisk Gateway Interface (AGI) which allows our multi-agent system to control the Asterisk dial plan.

3. A-STORM MULTI-AGENT SYSTEM

The Agent-based Service and Telecom Operations Management (A-STORM) multi-agent system is part of the prototype that deals with agent-based service provisioning. The prototype has been developed in order to explore the possibilities of implementing ontology-based user profiling/clustering, context-aware service personalization and rule-based software deployment in the 3G mobile network. Figure 1 shows the implemented multi-agent system in the proof-of-concept prototype.

The Business Manager Agent and the Provisioning Manager Agent belong to the group of business-driven provisioning agents whose task is to perform provisioning operations according to business strategies defined in a rule-based system. These strategies take into account business related information (such as user categories, service tariffs, season period, location of service execution, type of service content, business-to-business (B2B) relationships, etc.).

The Deployment Coordinator Agent and Remote Maintenance Shell agents can be categorized as service deployment agents that provide end-to-end solutions for efficient service deployment by enabling software deployment and maintenance at remote systems. Moreover, they provide so-called software component mobility (i.e., software components can seamlessly be relocated from one network node to another).

The Charging Manager Agent, Group Manager Agent, Session Manager Agent and Preference Manager Agent form

[4]http://www.asterisk.org
[5]http://en.wikipedia.org/wiki/Asterisk_(PBX)

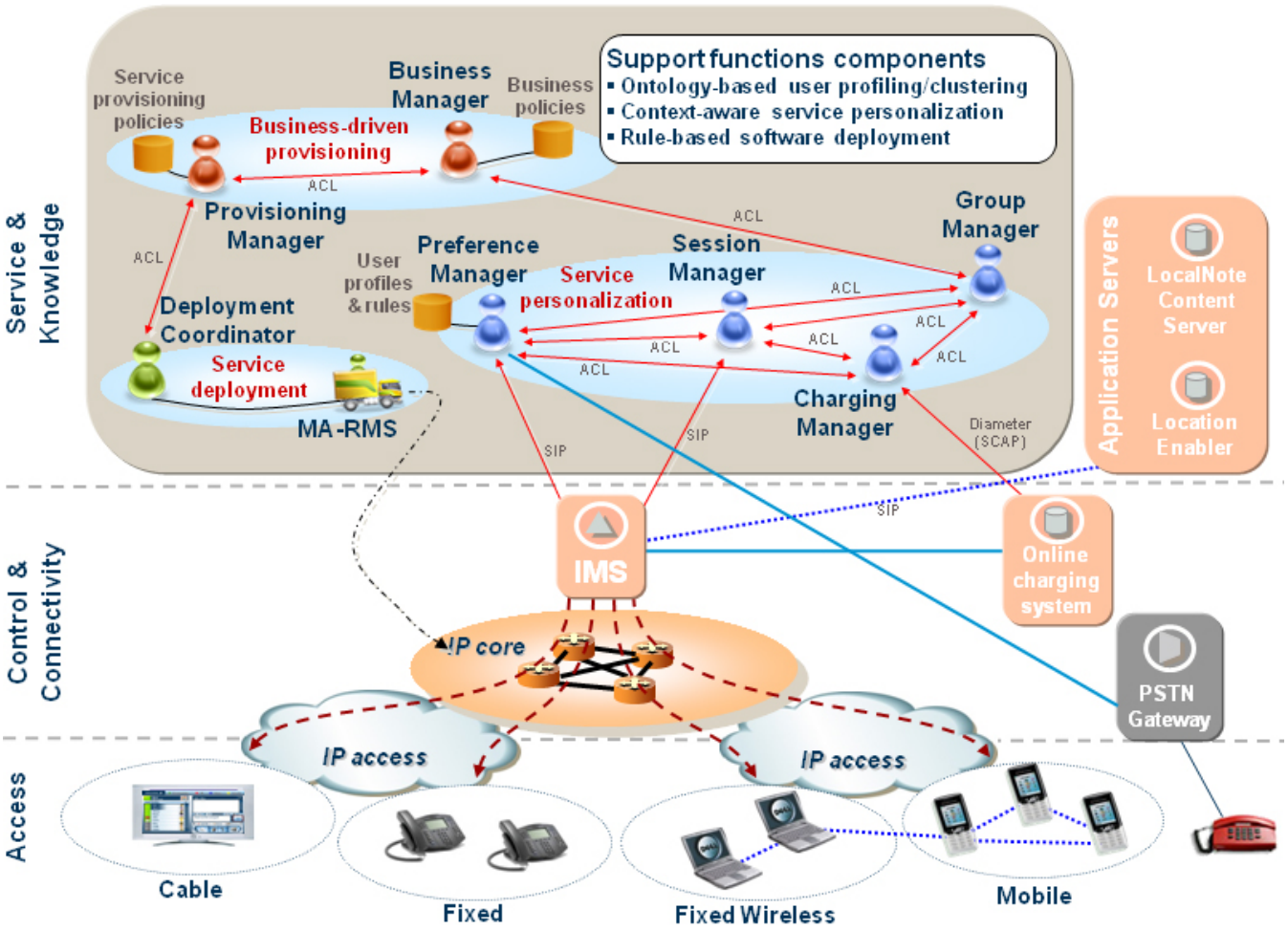

Figure 1: A-STORM proof-of-concept prototype architecture

a group of context management agents. They gather context information from network nodes and terminals (e.g., trigger events in SIP/PSTN call model, balance status, terminal location) and enable user personalization through the execution of context-dependent personal rules.

The Business Manager Agent, Charging Manager Agent, Provisioning Manager Agent, Deployment Coordinator Agent and Remote Management Shell Agents' tasks are beyond the scope of this paper. For a more detailed description of these agents and their functionalities can be found in [14, 18, 5, 7, 10, 13]. Their features are prerequisites for context-aware provisioning of IMS-enabled ubiquitous services. Two agents (the Preference Manager Agent (PMA) and the Session Manager Agent (SMA)) discussed later in this paper are incorporated in the prototype in order to facilitate personalized application-level mobility features supported by SIP. Furthermore, they provide semantic interoperability of context information delivered by the SIP event mechanism.

The context information that is required for service personalization is managed by the PMA which follows personal rules specified by the user it represents. The personal rules depend on context information (e.g., user location, session/balance status, presence) gathered from network nodes that are, for example, part of an Online Charging System[6] or an IMS, enhanced with a location enabler and a media gateway towards traditional PSTN terminals. Moreover, the PMA handles the knowledge base which contains profiles of user preferences and terminal capabilities, enabling service personalization in different types of user devices. Each user in the NGN should have its own PMA. The SMA is created at the beginning of the session and is in charge of monitoring the session in progress.

3.1 Agent-Based Support for Session Mobility

Device mobility in an all-IP mobile network is elegantly enabled by the mobile IP protocol. However, in order to exploit the full power of mobility in the Mobile Internet, personal and session mobility also have to be addressed. Such mobility can be enabled in the application level with SIP support. In particular, personal (pre-call) mobility occurs when a mobile user moves to another network prior to receiving or making a call. After the mobile user has obtained a new IP address, it registers with a SIP server allowing incoming invitations to be re-directed to the mobile user's current location. Session (mid-call) mobility, on the other hand,

[6]http://www.ericsson.com/mobilityworld/sub/open/technologies/charging_solutions/tools/diameter_charging_sdk

occurs when the user changes a terminal, moves to another network, or switches to another network interface during an ongoing session. After the mobile user has obtained a new IP address it re-invites the correspondent host in order to re-establish communication. However, addressing personal and session mobility in a dynamic heterogeneous environment, in which both resources and services vary in terms of availability and configuration, is a real challenge.

3.2 SIP-Based Knowledge Exchange of Service Context

Context-awareness refers to the ability to detect and incorporate information regarding the user, network conditions, terminal capabilities, etc. The SIP event mechanism represents an elegant and scalable solution for the delivery of all types of context information. This information can be spread over different devices, can involve several pieces of service logic, and may reside in several administrative domains. However, the SIP event mechanism does not provide any application-level semantic description of the delivered events between distributed parties. We propose a solution for semantic interoperability by building an agent-based knowledge exchange system that will enable the sharing and unambiguous understanding of context-related information through the use of ontologies.

The intelligent agents used in our prototype are equipped with reasoning capabilities enabling them to understand information delivered to them by the SIP event mechanism. Namely, they are able to understand ontology-based context information by utilizing a description logic (DL)-based reasoning engine. Consequently, there is a high degree of automation in knowledge exchange between software agents in the prototype. The agents (e.g., the GMA and the PMA) use OWLS-MX [11], a hybrid semantic matching tool which combines logic-based reasoning with approximate matching, based on syntactic IR similarity computations. The GMA uses this for semantic clustering of users, while the PMA uses it for the discovery of eligible services according to user preferences.

4. CASE STUDY

Figure 2 shows entities included in the case study described in this section. All entities communicate with each other by exchanging ACL or SIP messages.

The LocalNote service [3] can be described as a location-triggered instant messaging service. It provides a mechanism for sending short text messages whereby the sender can specify the area in which the recipient must reside in order to receive the message. The LocalNote Content Server (LCS) is the core server of the LocalNote service and is used to trigger redirection of the established SIP session. It contacts the IMS enabler, known as the Location Enabler (LE), which is used to obtain information regarding recipients' positions. The PMA subscribes to a user's location, while the LE notifies the PMA when the user enters the area specified in his subscription.

4.1 SIP Entities

In this subsection, we describe the SIP elements that are used in our proof-of-concept prototype. Basic SIP elements present in a typical SIP network are: User Agents, SIP Proxy Servers, SIP Registrars and Redirect Servers. These elements are often only logically separate entities that are physically co-located.

User Agents are Internet end points that use SIP to find each other and negotiate session characteristics. They usually reside on users' computers in the form of applications, but can also reside in mobile phones, PSTN gateways, PDAs and so on. In our prototype, we consider two users: Alice and Bob. Bob's User Agent is on his office desktop PC, while Alice has two User Agents: one on her mobile phone and the other on her office PSTN phone.

SIP allows creation of an infrastructure of network hosts called SIP Proxy Servers. They are important entities in the SIP infrastructure because they perform routing of session invitations according to invitee's current location, authentication, accounting and many other important functions. The most important task of a Proxy Server is to route session invitations "closer" to the callee.

Redirect Servers respond to a SIP request with an address where the SIP message should be redirected. It maps a destination address (in the SIP message) to one or more addresses and returns the new address list to the originator of the SIP request. The location of the intended recipient is retrieved from the location database maintained by the SIP Registrar.

The Registrar is a special SIP entity that receives registrations from users, extracts information regarding their current location and stores this information in a location database. The location database is then used by the Proxy Servers. Due to their tight coupling with Proxy and Redirect Servers, Registrars are usually co-located with them.

Proxy Servers are referred to as CSCFs [17]. We distinguish between Proxy CSCFs (P-CSCF), Serving CSCFs (S-CSCF), and Interrogating CSCFs (I-CSCF). A P-CSCF represents the point of contact between the user terminal and the rest of the network. A S-CSCF provides services to the user while an I-CSCF's role is to find the proper S-CSCF for a particular user.

In our proof-of-concept prototype we used the Ericsson Service Development Studio[7] to simulate an IMS environment and run Application Servers (e.g., PCS, LE).

4.2 Session Mobility

In order to make or receive a SIP call, the user must register, after which Asterisk informs the user's PMA about the registration. The PMA updates the user's rule engine with information that the user is available.

User Bob initiates a SIP session with user Alice. Asterisk establishes the session and creates the SMA in charge of the created session. Bob is connected with his desktop PC, while Alice is in town talking to Bob from her mobile phone. The SMA informs Alice's and Bob's PMAs, which must update their databases. When Alice's PMA updates her database with the information that she is in an ongoing session on her mobile phone, a rule is triggered. This rule states that when Alice enters her office talking on her mobile phone, the session should be redirected to her office PSTN phone. Consequently, the PMA sends a message to the LE, requesting notification when Alice enters her office. The LCS subscribes to information regarding Alice's location with the LE.

During an ongoing session, Alice arrives to her office. The

[7]http://www.ericsson.com/mobilityworld/sub/open/technologies/ims_poc/tools/sds_40

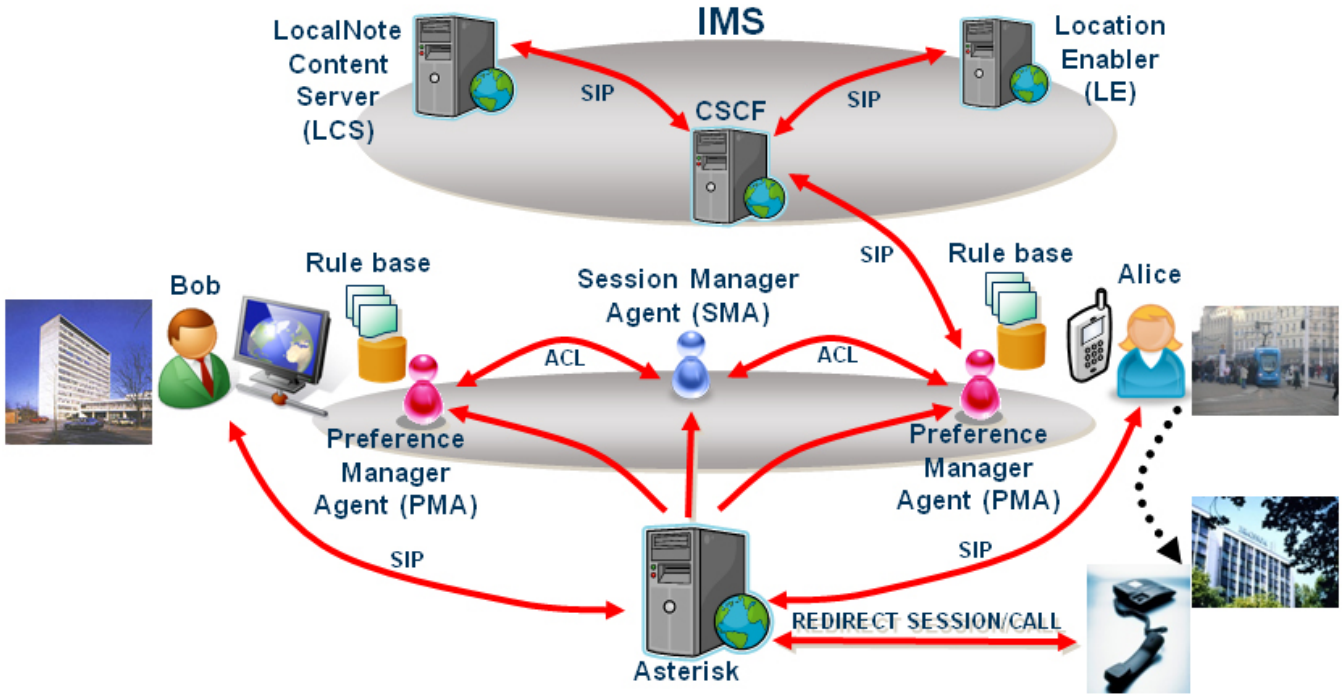

Figure 2: Session mobility

LE sends a notification message to the LCS, which in turn forwards this message to Alice's PMA. Her new location is written in her database, triggering another rule which ensures that the PMA informs the SMA that the session should be redirected. The SMA then sends a redirect request to Asterisk which initiates redirection of the ongoing session from Alice's mobile phone to her office PSTN phone. After the session has been redirected to the PSTN phone, the mobile phone is excluded from the rest of the session.

5. CONCLUSION AND FUTURE WORK

Agent technology presents a promising solution for service provisioning problems associated with 3G mobile networks. The A-STORM multi-agent system, which consists of intelligent and mobile agents, provides possibilities for implementing proactive optimized service deployment, charging-aware service provisioning, and support for multi-provider environments in the 3G mobile network. We propose a multi-agent architecture which, in conjunction with SIP, can realize the mobility of SIP-based ubiquitous services. The functionalities of two agents (the Preference Manager Agent (PMA) and the Session Manager Agent (SMA)) are presented. The PMA follows personal rules, specified by its user, and manages the context information that is required for service personalization. After a session between two users has been established, the SMA is created and is in charge of monitoring and managing the ongoing session. In cooperation with SIP entities, these two types of agents enable personalized session mobility.

Further development of our multi-agent system is aimed at enabling context- aware provisioning of group-oriented services that will use context information, not only in IMS-based networks, but also in emerging sensor networks.

6. ACKNOWLEDGMENTS

The work presented in this paper was carried out within research projects 036-0362027-1639 "Content Delivery and Mobility of Users and Services in New Generation Networks", supported by the Ministry of Science, Education and Sports of the Republic of Croatia, and "Agent-based Service & Telecom Operations Management", supported by Ericsson Nikola Tesla, Croatia.

7. ADDITIONAL AUTHORS

Igor Ljubi, University of Zagreb, Faculty of Electrical Engineering and Computing, Unska 3, Zagreb, Croatia, email: `igor.ljubi@fer.hr`

8. REFERENCES

[1] General principles and general reference model for next generation networks. Technical Report ITU-T Recommendation Y.2011, Telecommunication Standardization Sector, 2004.

[2] S. Berger, H. Schulzrinne, S. Sidiroglou, and X. Wu. Ubiquitous computing using SIP. In C. Papadopoulos and K. C. Almeroth, editors, *NOSSDAV*, pages 82–89. ACM, 2003.

[3] A. Brajdic, O. Lapcevic, M. Matijasevic, and M. Mosmondor. Service composition in IMS: A location based service example. In *3rd Int. Symposium on Wireless Pervasive Computing, 2008. ISWPC 2008*, pages 208–212, Santorini, Greese, 2008.

[4] J. Damsgaard and L. Marchegiani. Like Rome, a mobile operator's empire wasn't built in a day!: a journey through the rise and fall of mobile network operators. In M. Janssen, H. G. Sol, and R. W. Wagenaar, editors, *ICEC*, volume 60 of *ACM*

International Conference Proceeding Series, pages 639–648. ACM, 2004.

[5] G. Dumic, V. Podobnik, G. Jezic, K. Trzec, and A. Petric. An agent-based optimization of service fulfillment in next-generation telecommunication systems. In Z. Car and M. Kusek, editors, *Proceedings of the 9th International Conference on Telecommunications ConTEL 2007*, pages 57–63, Zagreb, Croatia, 2007.

[6] F. Ghys and A. Vaaraniemi. Component-based charging in a next-generation multimedia network. *IEEE Communications Magazine*, 41(1):99–102, January 2003.

[7] D. Grubisic, K. Kljaic, I. Ljubi, A. Petric, and G. Jezic. An agent-based user selection method for business-aware service provisioning. In *Proceedings of the 15th International Conference on Software, Telecommunications and Computer Networks*, pages 1–5, Split, Croatia, 2007.

[8] H. Hanrahan. *Network Convergence: Services, Applications, Transport, and Operations Support*. John Wiley & Sons, Inc., New York, NY, USA, 2007.

[9] E. F. Hill. *Jess in Action: Java Rule-Based Systems*. Manning Publications Co., Greenwich, CT, USA, 2003.

[10] G. Jezic, M. Kusek, S. Desic, O. Labor, A. Caric, and D. Huljenic. Multi-agent system for remote software operations. In V. Palade, R. J. Howlett, and L. C. Jain, editors, *KES*, volume 2774 of *Lecture Notes in Computer Science*, pages 675–682. Springer, 2003.

[11] M. Klusch, B. Fries, and K. P. Sycara. Automated semantic web service discovery with OWLS-MX. In H. Nakashima, M. P. Wellman, G. Weiss, and P. Stone, editors, *AAMAS*, pages 915–922. ACM, 2006.

[12] I. Ljubi, V. Podobnik, and G. Jezic. Cooperative mobile agents for automation of service provisioning: A telecom innovation. In *ICDIM*, pages 817–822. IEEE, 2007.

[13] I. Lovrek, G. Jezic, M. Kusek, I. Ljubi, A. Caric, D. Huljenic, S. Desic, and O. Labor. Improving software maintenance by using agent-based remote maintenance shell. In *ICSM*, pages 440–449. IEEE Computer Society, 2003.

[14] A. Petric, I. Ljubi, K. Trzec, G. Jezic, M. Kusek, V. Podobnik, and K. Jurasovic. An agent based system for business-driven service provisioning. In B. O'Sullivan and K. Orsvarn, editors, *Proceedings of the AAAI'07 Workshop on Configuration*, pages 25–30. AAAI Press, 2007.

[15] V. Podobnik, A. Petric, K. Trzec, and G. Jezic. Software agents in new generation networks: Towards the automation of telecom processes. In L. C. Jain and N. T. Nguyen, editors, *Knowledge Processing and Decision Making in Agent-Based Systems*, chapter 4, pages 71–99. Springer-Verlag, Berlin Heidelberg, 2009.

[16] H. Schulzrinne, X. Wu, S. Sidiroglou, and S. Berger. Ubiquitous computing in home networks. *IEEE Communications Magazine*, 41(11):128–135, 2003.

[17] H. Sinnreich and A. B. Johnston. *Internet communications using SIP: delivering VoIP and multimedia services with Session Initiation Protocol*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[18] A. Vrancic, K. Jurasovic, M. Kusek, G. Jezic, and K. Trzec. Service provisioning in telecommunication networks using software agents and rule-based approach. In *Proceedings of the 30th International Convention MIPRO 2007*, pages 159–164, 2007.

[19] J. S. Weiser. The coming age of calm technology. In R. Denning, editor, *Beyond Calculation: The Next Fifty Years of Computing*. Copernicus, 1998.

[20] J. L. Yoon. Telco 2.0: a new role and business model. *IEEE Communications Magazine*, 45(1):10–12, 2007.

Agent-based Framework for Personalized Service Provisioning in Converged IP Networks

Vedran Podobnik
University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3
HR-10000 Zagreb, Croatia
vedran.podobnik@fer.hr

Maja Matijasevic
University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3
HR-10000 Zagreb, Croatia
maja.matijasevic@fer.hr

Ignac Lovrek
University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3
HR-10000 Zagreb, Croatia
ignac.lovrek@fer.hr

Lea Skorin-Kapov
Ericsson Nikola Tesla, R&D Center
Krapinska 45
HR-10000 Zagreb, Croatia
lea.skorin-kapov@ericsson.com

Sasa Desic
Ericsson Nikola Tesla, R&D Center
Krapinska 45
HR-10000 Zagreb, Croatia
sasa.desic@ericsson.com

ABSTRACT
In a global multi-service and multi-provider market, the Internet Service Providers will increasingly need to differentiate in the service quality they offer and base their operation on new, consumer-centric business models. In this paper, we propose an agent-based framework for the Business-to-Consumer (B2C) electronic market, comprising the Consumer Agents, Broker Agents and Content Agents, which enable Internet consumers to select a content provider in an automated manner. We also discuss how to dynamically allocate network resources to provide end-to-end Quality of Service (QoS) for a given consumer and content provider.

Categories and Subject Descriptors
D.2.11 [**Software Architectures**]: Domain-specific Architectures, H.3.3 [**Information Search and Retrieval**]: Information Filtering, Selection Process, I.2.11 [**Distributed Artificial Intelligence**]: Intelligent Agents, Multiagent Systems, K.4.4 [**Electronic Commerce**], K.6.4 [**System Management**]: Quality Assurance.

General Terms
Management, Performance, Economics.

Keywords
Agent-based B2C e-market, Quality of Service, Internet business environment, Provider selection.

1. INTRODUCTION
The Internet service providers (ISPs) and IP-based telecom network operators are turning towards new business opportunities in a global multi-service and multi-provider market. With consumers typically having several multi-purpose end-user devices, the number and variety of personal, work, and home related services offered will also grow. As "plain broadband" wired/wireless Internet access is likely to become a commodity in the next 10 years or so [1], the ISPs will have to differentiate in the service quality they offer, and base their operation on new, consumer-centric business models. Such models involve a number of actors involved in service delivery, from the user (consumer of the service) to the end service provider, where the selection of the service provider is a non-trivial issue, considering an electronic market (e-market) with a number of service providers offering the same or similar service. The challenge is twofold: first, how to select "the best" service provider, given the user preferences and semantic service descriptions; and second, once the selection is made, how to dynamically allocate network resources on an end-to-end basis. The main contribution of this paper focuses on the first issue by proposing a novel agent-based framework for service provider selection. This process is based on service discovery which considers not only the semantic matching, but also the price and reputation of the service provider, in which it differs from other approaches found in literature. Section 2 gives the problem formulation, while Section 3 presents the model. We discuss the second issue, end-to-end Quality of Service (QoS) in Section 4. Section 5 concludes the paper.

Cite as: Agent-based Framework for Personalized Service Provisioning in Converged IP Networks, Vedran Podobnik, Maja Matijasevic, Ignac Lovrek, Lea Skorin-Kapov, and Sasa Desic, *Proc. of International Workshop on Service-Oriented Computing: Agents, Semantics, and Engineering @ AAMAS 2009*, May, 11, 2009, Budapest, Hungary.

2. ROLES AND RELATIONSHIPS IN THE ELECTRONIC MARKET
There are a number of actors present in the Internet business environment who need to establish relationships in order to provide consumers with converged services. An actor may take on a number of roles in a particular scenario, and furthermore a number of actors can play the same role. The key roles and relationships, shown in Figure 1, include *Consumer*, *Access Line Provider*, *Primary Service Provider* (PSP), *Internet Service Provider* (ISP), *Content Provider* (CP), and *Transport Provider* (TP) [14].

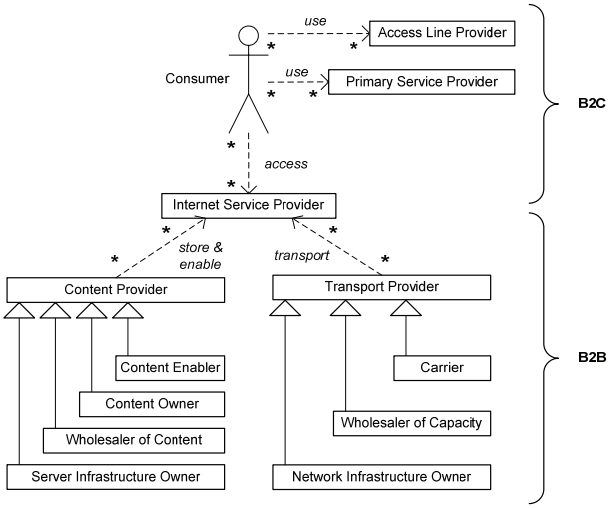

Figure 1. Roles and relationships of actors in the Internet business environment.

The *Consumer* is a role which typically represents the human user. The *Access Line Provider* is a role representing the owner of the access line. The ISP, in the most general sense, is a business entity providing a user with service(s). To differentiate between the responsibilities of an ISP which involve dealing with (e.g., multimedia) content, and those related to managing the transport of the content over the network infrastructure, we introduce the roles of CP and TP, respectively. The role of ISP as CP will be relevant for the first problem addressed in this paper – the selection of the service provider, while the role of TP will be relevant for the second issue – ensuring the end-to-end QoS. From now, we will refer to CP and TP, instead of just "ISP", to disambiguate roles. The PSP is an ISP which provides to a consumer the service of Internet access and consequently has a business relationship with that consumer. It may be noted that a particular consumer can have multiple PSPs, but only one PSP can be active at any one time. By adopting the "one-stop responsibility" concept [5], the PSP is also perceived as being responsible for coordinating the QoS negotiation and adaptation process, while further relying on the services of sub-providers in order to secure an end-to-end service and quality level to the consumer. Figure 1 also shows the relationships between roles as Business-to-Customer (B2C) and Business-to-Business (B2B).

Our proposed agent based framework addresses the many-to-many relationship between Consumer and ISP in the B2C electronic market, as shown in Figure 2. The roles are modeled by using an agent paradigm [15], as follows: 1) the *consumer representation* is represented by a Consumer Agent, 2) the *content provisioning* is represented by a Content Agent, and 3) the *brokering* between Consumer Agents and Content Agents is represented by a Broker Agent. The Broker Agent belonging to a certain ISP offers not only its own content (acting as a CP), but also the content offered (advertised) by other ISPs or CPs to which this ISP has established business relationships, as shown in Figure 2.

The issue of selecting a content provider has significant implications for all aspects of service provisioning. In the multi-provider network, a consumer will typically have a choice of a (possibly large) number of ISPs/CPs for a given content, as shown in Figure 2. He or she may also have personal preferences regarding service options, device, and/or a particular wireline/wireless access network. We assume that the Consumer Agent containing these preferences is formed at the time of signing a contract with a PSP (e.g., fixed access via xDSL at home, and mobile access via HSPA in a 3G mobile network), and it resides within the PSP. Given all that, it is the task of the Broker Agent to discover the content and select the best match for the particular content request that corresponds to the given consumer preferences. It is also assumed that the scale of the problem is such that it cannot be solved by exhaustively querying all possible CPs.

As a running example we use the following problem, illustrated in Figure 3: the user (Consumer) wants to view video-clips with Bayern goals from the latest Bundesliga round on her dual 3G/WLAN mobile phone. Her current (active) PSP is ISP_1. There are several CPs offering and advertising their service of providing *video clips of European football matches* (CP_i, ISP_j, ISP_k) to ISP_1. The ISP_1 has a business and technical relationship (SLA [5]) with ISP_j, and ISP_j has one with ISP_k. After receiving the Consumer

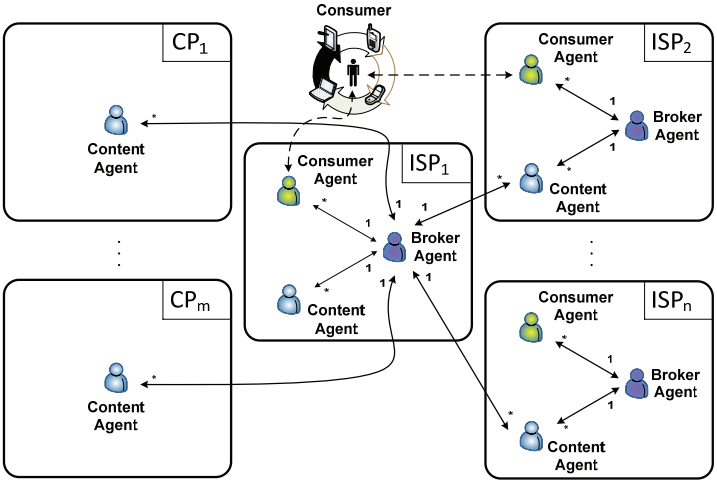

Figure 2. The proposed agent framework for B2C electronic market.

request for content, the content from ISP_k is selected as the best match (the most eligible content). Then the QoS negotiation and adaptation takes place on end-to-end basis for a given consumer, service, and ISPs involved in service delivery, and having completed that, the service provisioning starts.

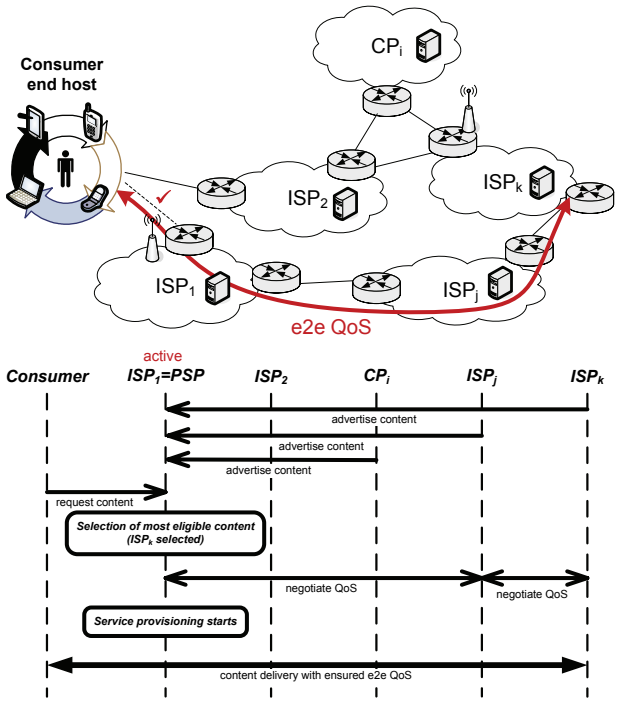

Figure 3. Problem illustration by example

3. SELECTION OF THE CONTENT PROVIDER

The first step in the process of selecting the content provider is discovery. The state-of-the art discovery mechanisms are based on matching the semantics of resource (e.g., content) descriptions (i.e., semantic matchmaking) [7][10][18], rather than keyword matching [6]. The semantic dimension of resources such as multimedia content has been exploited in order to evaluate "interesting" inexact matches [3]. Most approaches suggested for semantic discovery use standard DL (*Description Logic*) reasoning to automatically determine whether one resource description matches the other. Our discovery mediator in the B2C e-market differs from previous approaches in that it considers the actual performance of businesses which act as ISPs/CPs (with respect to both price and reputation) in addition to semantic matchmaking. The mechanism on which the mediator is based is the Semantic Pay-Per-Click Agent (SPPCA) auction, a novel auction mechanism based on Pay-Per-Click (PPC) advertising auctions [6], but adapted for agent environment and enhanced with a semantic dimension [13].

3.1 The Architecture of Electronic Market for Content Trading

A description of the proposed agent-mediated B2C e-market architecture (Figure 2) follows along with a demonstration of how it operates.

3.1.1 The Content Agent

In the proposed B2C e-market agents trade with various types of content (formally defined as a set $\mathcal{IC}$):

$$\mathcal{IC} = \{ic_1, ic_2, \ldots, ic_{|\mathcal{IC}|}\},$$

which is provided by different Content Providers (formally defined as a set $\mathcal{CP}$):

$$\mathcal{CP} = \{cp_1, cp_2, \ldots, cp_{|\mathcal{CP}|}\}.$$

Content Providers are represented in the e-market by Content Agents (formally defined as a set $\mathcal{A}_{\mathcal{CP}}$):

$$\mathcal{A}_{\mathcal{CP}} = \{a_{cp_1}, a_{cp_2}, \ldots, a_{cp_{|\mathcal{CP}|}}\}.$$

An a_{cp_i} represents a cp_i which offers a certain content ic_i that is described by content ontology, whose fragment (describing *video clips of European football matches*) is presented later in this work. Initially, a_{cp_i} wishes to advertise its content (advertised ic_i is denoted as ic_{adv}) at discovery mediator (i.e., the Broker Agent). An a_{cp_i} accomplishes that by participation in the SPPCA.

3.1.2 The Consumer Agent

Consumers of $\mathcal{IC}$ (formally defined as a set $\mathcal{C}$):

$$\mathcal{C} = \{c_1, c_2, \ldots, c_{|\mathcal{C}|}\},$$

are represented on the e-market by Consumer Agents (formally defined as a set $\mathcal{A}_{\mathcal{C}}$):

$$\mathcal{A}_{\mathcal{C}} = \{a_{c_1}, a_{c_2}, \ldots, a_{c_{|\mathcal{C}|}}\}.$$

An a_{c_i} acts on behalf of its human owner (i.e., consumer) in the discovery process of suitable ic_{adv} and subsequently negotiates the utilization of that content. An a_{c_i} wishes to get a best-ranked advertised content which is appropriate with respect to its needs (requested ic_i is denoted as ic_{req}).

3.1.3 The Broker Agent

Mediation between content requesters and content providers is performed by Broker Agents (formally defined as a set $\mathcal{A}_{\mathcal{B}}$):

$$\mathcal{A}_{\mathcal{B}} = \{a_{b_1}, a_{b_2}, \ldots, a_{b_{|\mathcal{B}|}}\}.$$

There is one a_{b_i} located at every ISP and it mediates between c (i.e., $a_{\mathcal{C}}$) to whom this ISP is PSP and all cp (i.e., $a_{\mathcal{CP}}$) which advertised its content at this a_{b_i}. An a_{b_i} enables $\mathcal{A}_{\mathcal{CP}}$ to advertise their content descriptions and recommends the most eligible content to $a_{\mathcal{C}}$ in response to their requests. It is assumed that a_{b_i} is a trusted party which fairly mediates between content requesters and content providers.

3.2 The Content Discovery Model

Figure 4 presents interactions between a_{c_i} and a_{b_i} which enable content discovery in the proposed B2C e-market. The a_{c_i}, by sending CFP (Call for Proposal) to a_{b_i}, requests two-level filtering of advertised content descriptions to discover which is the most adequate for its needs. Along with the description of requested content ic_{req}, the CFP includes the set of matching parameters (to be explained later) that personalize the discovery process according to the consumer preferences. First-level filtering ($f_1 : \mathcal{IC} \rightarrow \mathcal{IC}$) is based on semantic matchmaking between descriptions of content requested by c_i (i.e., a_{c_i}) and

those advertised by cp (i.e., $a_{\mathcal{CP}}$). Content which passes the first level of filtering ($ic_{ff_1} \subset \mathcal{IC}$) is then considered in the second filtering step. Second-level filtering ($ff_2 : \mathcal{IC} \rightarrow \mathcal{IC}$) combines information regarding the actual performance of cp_{ff_1} (cp which offer ic_{ff_1}) and prices bid in SPPCA by corresponding $a_{\mathcal{CP}_{ff_1}}$ ($a_{\mathcal{CP}}$ that represent cp which offer ic_{ff_1}). The performance of cp_{ff_1} (with respect to both price and reputation) is calculated from the previous $\mathcal{A}_{\mathcal{C}}$ feedback ratings. Following filtering, the most eligible content ($ic_{ff_2} \subset ic_{ff_1} : |ic_{ff_2}|=1$) is chosen and recommended to the a_{c_i} in response to its request.

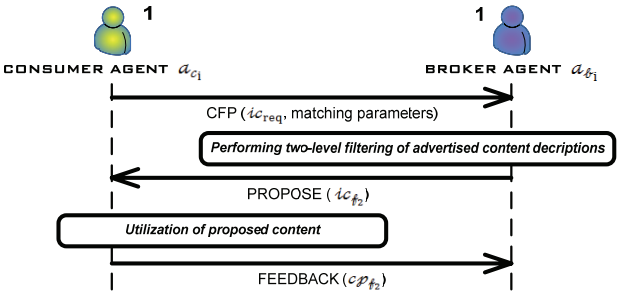

Figure 4. The a_{c_i} discovers the most eligible content advertised at a_{ff_i}.

Figure 5 explains how the SPPCA auction, which is part of the discovery process, operates. The SPPCA auction is divided into rounds of fixed time duration. To announce the beginning of a new auction round, the a_{ff_i} broadcasts a CFB (Call for Bid) message to all the $a_{\mathcal{CP}}$ which have registered their ic_{adv} for participation in the SPPCA auction. Every CFB message contains a status report. In such a report, the a_{ff_i} sends to the a_{cp_i} information regarding events related to its advertisement which occurred during the previous auction round. The most important information is that regarding how much of the a_{cp_i} budget was spent (i.e., the advertisement bid price $\text{bid}_{ic_{adv}}$ multiplied by the number of recommendations of its ic_{adv} to various $a_{\mathcal{C}}$). In response to a CFB message, an a_{cp_i} sends a BID message.

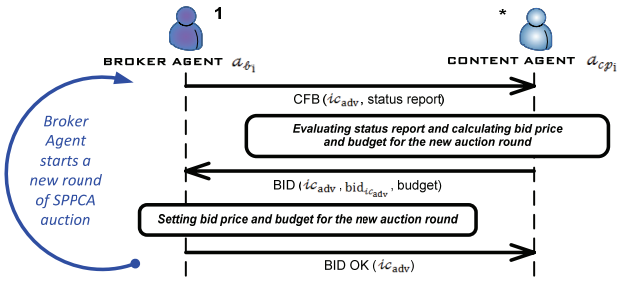

Figure 5. The SPPCA auction.

3.2.1 Semantic Matchmaking of Content Descriptions

In the multi-agent system implementing the proposed B2C e-market model, the Semantic Web technology [9] is used to describe content. Namely, for describing content we use W3C's OWL-S (*Web Ontology Language for Services*), which is an OWL-based (*Web Ontology Language*) (Figure 6) technology for describing the properties and capabilities of Web Services in an unambiguous, computer interpretable mark-up language.

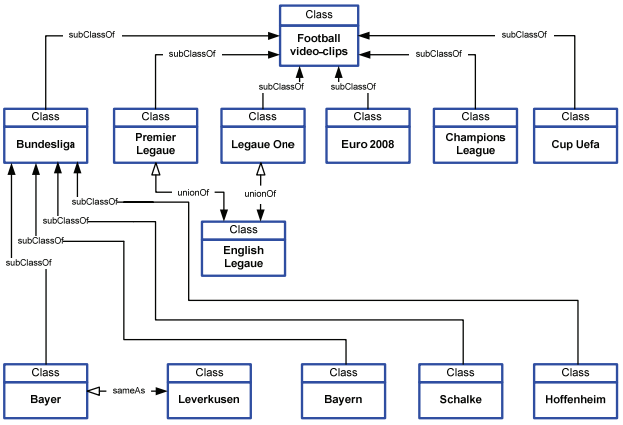

Figure 6. The OWL ontology fragment describing *video clips of European football matches*

Our a_{ff_i} uses OWLS-MX [8], a hybrid semantic matching tool which combines logic-based reasoning with approximate matching based on syntactic IR similarity computations. As the notion of match rankings is very important, OWLS-MX enables computation of the degree of similarity between compared service descriptions, i.e., the comparison is assigned a *content correspondence factor* (M). Namely, the OWLS-MX matchmaker takes as input the OWL-S description of a_{c_i} desired content ic_{req}, and returns a set of relevant content which match the query ic_{ff_1}. Relevant content is annotated with its individual degree of matching similarity value (i.e., $M_{ic_{req},ic_{adv}}$). There are six possible levels of matching [8]. The first level is a perfect match (also called an EXACT match) which is assigned a factor $M = 5$. Furthermore, we have four possible inexact match levels which are as follows: a PLUG-IN match ($M = 4$), a SUBSUMES match ($M = 3$), a SUBSUMES-BY match ($M = 2$) and a NEAREST-NEIGHBOUR match ($M = 1$). If two content descriptions do not match according to any of the above mentioned criteria, they are assigned a matching level of FAIL ($M = 0$). An a_{c_i} specifies its desired matching degree threshold (i.e., the M_{min}), defining how relaxed the semantic matching is.

3.2.2 The Performance Model of Content Providers

A performance model tracks the past performance of $\mathcal{CP}$ in the B2C e-market. Our model monitors two aspects of a cp_i performance – the reputation of the cp_i and the cost of utilizing the ic that cp_i is offering.

After utilizing the recommended content, an a_{c_i} gives an a_{ff_i} feedback regarding cp_{ff_2}, both from the reputation viewpoint (called the *reputation rating* ($Q \in [0.0, 1.0]$)) and the cost viewpoint (called the *price rating* ($P \in [0.0, 1.0]$)). A rating of 0.0 is the worst (i.e., the cp_{ff_2} could not provide the content at all and/or utilizing the content is very expensive) while a rating of 1.0 is the best (i.e., the cp_{ff_2} provides a content that perfectly corresponds to the c_i needs and/or utilizing the content is cost-efficient). The overall ratings of cp_i can be calculated in a

number of ways. In our approach, we use the EWMA-based (*Exponentially Weighted Moving Average*) learning [11].

3.2.3 Calculating a Recommended Ranked Set of Eligible Services

After an $a_{\mathcal{f}_i}$ receives a CFP message from an a_{c_i} (Figure 4), the discovery mediator finds the best-suitable content $ic_{\mathcal{f}_2}$ and recommends it to the a_{c_i} in response to its request. The final rating $R_{ic_{adv}}$ of a specific ic_{adv} at the end of discovery process is given by:

$$R_{ic_{adv}} = \frac{\alpha \times \frac{M_{ic_{req},ic_{adv}}}{5} + \beta \times Q_{cp_{adv}} + \gamma \times P_{cp_{adv}}}{\alpha + \beta + \gamma} \times bid_{ic_{adv}} \quad (1)$$

A higher rating means that this particular ic_{adv} is more eligible for the consumer's needs (i.e., ic_{req}); α, β and γ are weight factors (i.e., matching parameters from CFP message in Figure 4) which enable the a_{c_i} to personalize its request according to its owner's (i.e., c_i's) needs regarding the semantic similarity, reputation and price of a ic_{adv}, respectively; $M_{ic_{req},ic_{adv}}$ represents the *content correspondence factor* (M), but only ic_{adv} with M higher than threshold M_{min} are considered; $Q_{cp_{adv}}$ and $P_{cp_{adv}}$ represent the quality and price ratings of a particular cp_{adv}, respectively; $bid_{ic_{adv}}$ is the bid value for advertising an ic_{adv} in the SPPCA auction.

Since our performance model monitors two aspects of the cp_{adv} performance (i.e., its reputation and price), the $a_{c_{req}}$ defines two

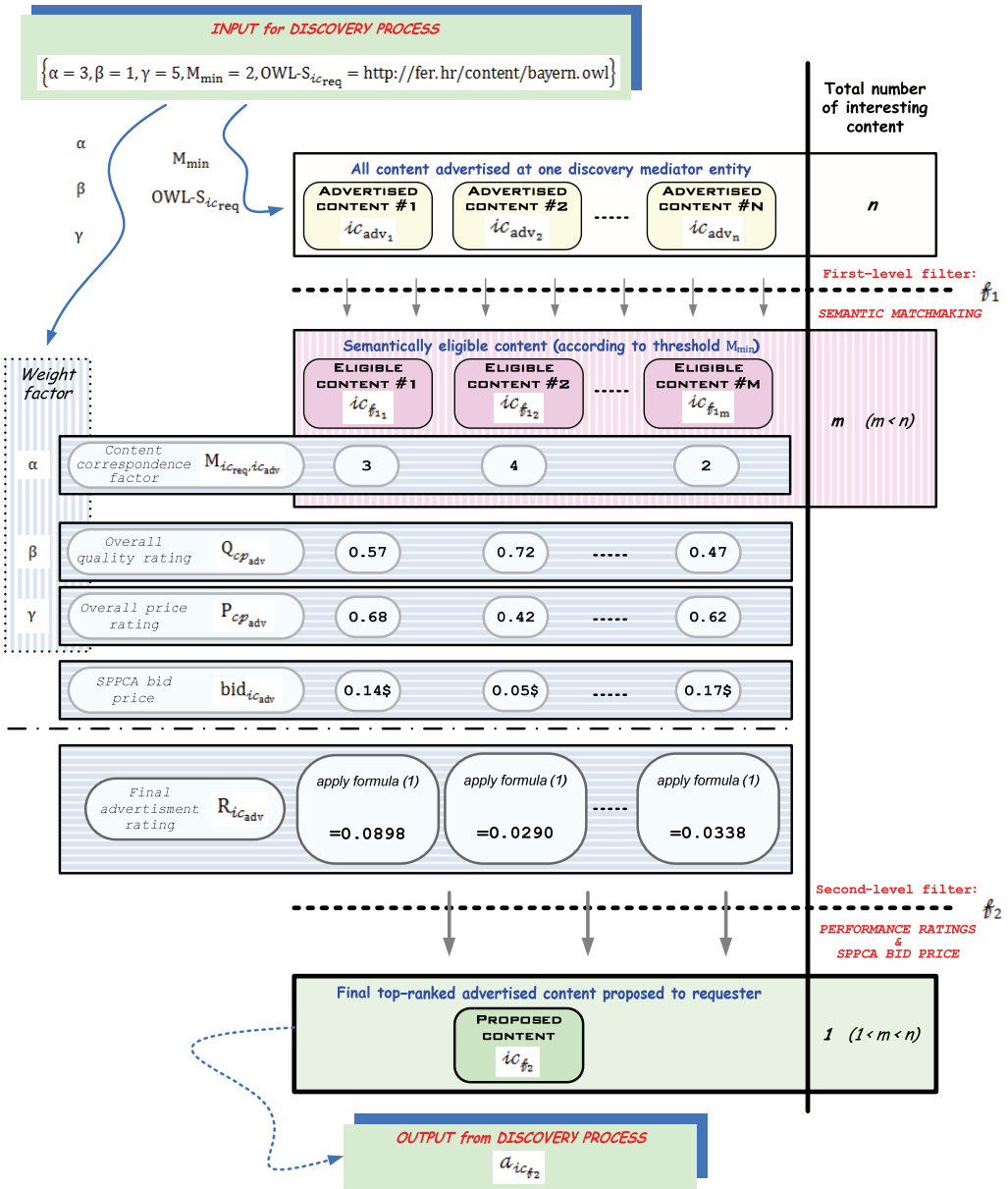

Figure 7. An example of the discovery process.

weight factors which determine the significance of each of the two aspects in the process of calculating the final proposal (β represents a weight factor describing the importance of cp_{adv} reputation while γ represents a weight factor describing the importance of content prices at cp_{adv}). Furthermore, an $a_{c_{\text{req}}}$ can specify whether information regarding the semantic similarity of ic_{req} and ic_{adv} is more important to it or information regarding an cp_{adv} performance. Thus, the $a_{c_{\text{req}}}$ also defines parameter α which is a weight factor representing the importance of the semantic similarity between ic_{ic} and ic_{adv}. In our example (Figure 7) where requested content are video-clips with Bayern goals from the latest Bundesliga round (OWL-$S_{ic_{\text{req}}}$ = http://fer.hr/content/bayern.owl), $\alpha = 3$, $\beta = 1$ and $\gamma = 5$. This means that the $a_{c_{\text{req}}}$ is looking for an inexpensive ic_{adv} and it is not very concerned with the cp_{adv} reputation.

4. ENSURING END-TO-END QOS

Once the consumer, by using the mechanism described in the previous section, selected the CP, the end-to-end QoS needs to be negotiated with her PSP. For ensuring end-to-end QoS, support in the network is needed to negotiate and adapt QoS to match the consumer preferences, service profile, and network capabilities; and thus create a basis for service and price differentiation [2]. A general QoS negotiation scenario involves four steps: 1) a host initiating a service on another host; 2) the addressed host providing a service offer/answer; 3) the initiating host responding to the offer/answer; and 4) service delivery. We assume that all ISPs are QoS-aware, i.e., that they control and administer the necessary infrastructure for providing QoS-based services, regardless of which lower layers mechanisms are used. The selection of QoS provisioning mechanisms in the access and core network is performed in the TPs. Depending on the type of the network, this may involve various service control entities that handle QoS signaling, QoS policy control, and interaction with underlying network QoS mechanisms, as well as typical "support functions" (if and when needed), such as consumer authorization, authentication, accounting, auditing, and charging. During the process of QoS (re)negotiation, signaling flows typically traverse a number of functional network entities along the end-to-end path between communication endpoints, as shown in Figure 8 [16]. It should be noted that the signaling (control) and data flows are separated, and that the resource managers in the data plane are "vertically" controlled by session control functions, which interface with the end-points and the internal databases related to consumers and services.

In an actual network architecture, functional entities may be mapped to one or more network nodes. The end points are shown as hosts (Host A, Host B) or application servers (Content Server, 3rd party Content/Application Servers). The additional functionality which must be implemented in the host may include, for example: a GUI for consumer preferences management, capability to negotiate session QoS (e.g., SIP (*Session Initiation Protocol*) interface), resource management capability (e.g., DiffServ), and mobility (e.g., Mobile IP's Mobile Host entity). Having in mind the CP selection procedure described in the previous section, we assume that a Consumer A, attached to the NGN by using Host A, has selected a PSP here shown as PSP

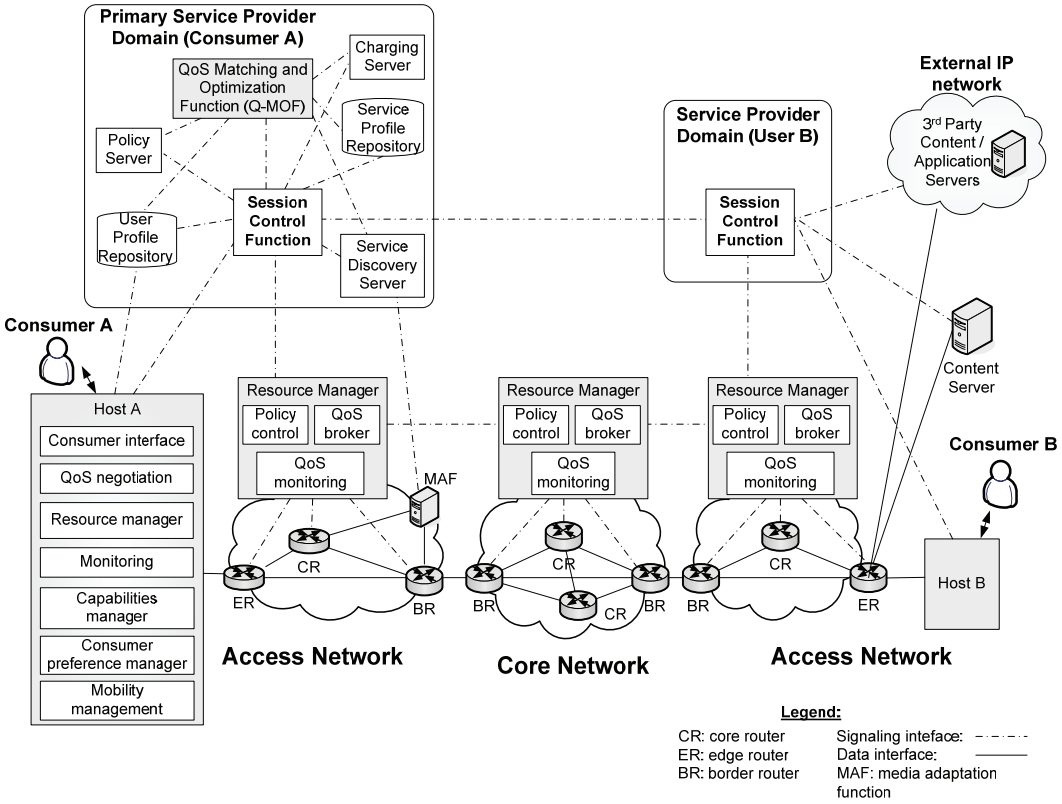

Figure 8. End-to-End QoS provisioning

Domain (Consumer A), to perform service control functions and offer access to 3rd party applications and services.

This service provider is then responsible for AAA functions (consumer authentication, authorization, and accounting), service provisioning, and maintaining a database for storing consumer-related data. It further interacts with an underlying network provider, for example, a 3G mobile network provider, which provides media connectivity functions. In a real life scenario, a single operator may take on multiple roles, including that of both a service and a network provider.

While the initial service matching and resource mapping may be based on QoS classes and SLA, more advanced mechanisms are needed to take into account dynamic changes in service profile (e.g., consumer willingness to pay for the service), network capabilities (e.g., due to handover), and service parameters (e.g., types of media streams comprising the service). Although the architecture proposed here is generic, in our previous work we considered a converged IP-based network based on 3GPP IP Multimedia Subsystem (IMS) [17]. In a pure-IP approach, SIP can be applied for session signaling, Diameter for policy control and AAA, and any QoS enabling mechanism may be applied at the network layer, including those for IP QoS interconnection [2][4][12].

5. CONCLUSION

In this paper, we proposed an agent-based framework for the B2C e-market where interactions between Consumer Agents, Broker Agents and Content Agents enable Internet consumers to select the most eligible content provider in an automated manner. The main benefit of the proposed approach is that in a situation with many ISPs/CPs offering the same or similar content, the user could not search for the content manually, nor exhaustively, nor could the best match be found based solely on semantic descriptions. Finally, we have discussed how end-to-end QoS could be negotiated once the content provider is selected.

6. ACKNOWLEDGMENTS

The authors acknowledge the support of research project “Content Delivery and Mobility of Users and Services in New Generation Networks” (036-0362027-1639), funded by the Ministry of Science, Education and Sports of the Republic of Croatia, and projects "Agent-based Service & Telecom Operations Management" and "Future Advanced Multimedia Service Enablers" of Ericsson Nikola Tesla, Croatia.

7. REFERENCES

[1] Anderson, J.Q., and Rainie, L. 2006. The future of the Internet II. Technical report, Pew Internet and American Life Project, http://www.pewinternet.org .

[2] Briscoe, B., and Rudkin, S. 2005. Commercial models for IP quality of service interconnect. BT Technology Journal 23, 2, 171–195.

[3] Di Noia, T., Di Sciascio, E., Donini, F.M., and Mong, M. 2004. A System for Principled Matchmaking in an Electronic Marketplace. International Journal of Electronic Commerce 8, 4, 9–37.

[4] Howarth, M.P., et al. 2005. Provisioning for interdomain quality of service: the MESCAL approach. IEEE Communications Magazine 43, 6, 129–137.

[5] ITU-T Recommendation E.860. 2002. Framework of a service level agreement.

[6] Jansen, B.J. 2006. Paid Search. IEEE Computer 39, 7, 88–90.

[7] Keller, U., Lara, R., Lausen, H., Polleres, A., and Fensel, D. 2005. Automatic Location of Services. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. Lecture Notes in Computer Science 3532, pp. 1–16. Springer, Heidelberg.

[8] Klusch, M., Fries, B., and Sycara, K. 2006. Automated Semantic Web Service Discovery with OWLS-MX. In: 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp. 915–922. ACM, Hakodate.

[9] Leuf, B.: The Semantic Web. 2006. Crafting Infrastructure for Agency. John Wiley & Sons, New York.

[10] Li, L., and Horrock, I. 2003. A Software Framework for Matchmaking Based on Semantic Web Technology. In: 12th International World Wide Web Conference (WWW), pp. 331–339. ACM, Budapest.

[11] Luan, X. 2004. Adaptive Middle Agent for Service Matching in the Semantic Web: A Quantitive Approach. PhD Thesis, University of Maryland.

[12] Masip-Bruin, X. et al. 2007. The EuQoS System: A Solution for QoS Routing in Heterogeneous Networks, IEEE Communications Magazine 45, 2, 9–103.

[13] Podobnik, V., Trzec, K., and Jezic, G. 2006. Auction-Based Semantic Service Discovery Model for E-Commerce Applications. In: Meersman, R., Tari, Z., Herrero P., et al. (eds.) OTM 2006. Lecture Notes in Computer Science 4277, pp. 97–106. Springer, Heidelberg.

[14] Podobnik, V., and Lovrek, I. 2008. Multi-Agent System for Automation of B2C Processes in the Future Internet. In: 27th IEEE Conference on Computer Communications (INFOCOM) Workshops, pp. 1–4. IEEE Press, Phoenix.

[15] Podobnik, V., Jezic, G., and Trzec, K. 2008. Towards New Generation of Mobile Communications: Discovery of Ubiquitous Resources. Electrotechnical Review 75, 1-2, 31–36.

[16] Skorin-Kapov, L. 2007. A framework for service-level end-to-end quality of service negotiation and adaptation. PhD Thesis, University of Zagreb.

[17] Skorin-Kapov, L., Mosmondor, M., Dobrijevic, O., and Matijasevic, M. 2007. Application-level QoS negotiation and signaling for advanced multimedia services in the IMS. IEEE Communications Magazine 45, 7, 108–116.

[18] Sycara, K., Paolucci, M., Anolekar, A., and Srinivasan, N. 2004. Automated Discovery, Interaction and Composition of Semantic Web Services. Journal of Web Semantics 1, 1.

Business Modeling via Commitments

Pankaj R. Telang and Munindar P. Singh

Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA

prtelang@ncsu.edu, singh@ncsu.edu

ABSTRACT

Existing computer science approaches to business modeling offer low-level abstractions such as data and control flows, which fail to capture the business intent underlying the interactions that are central to real-life business models. In contrast, existing management science approaches are high-level but not only are these semiformal, they are also focused exclusively on managerial concerns such as valuations and profitability.

This paper proposes a novel business metamodel based on commitments that considers additional agent-oriented concepts, specifically, goals and tasks. It proposes a set of business patterns and algorithms for checking model completeness and verification of agent interactions. Unlike traditional models, our approach marries rigor and flexibility, providing a crisp notion of correctness and compliance independent of specific executions.

1. INTRODUCTION

Real-life service engagements generally involve long-lived, complex interactions among two or more autonomous business partners. We define a *business model* as a specification of a way in which a service engagement is carried out. We address the problem of creating, enacting, and verifying business models from a high-level, yet rigorous standpoint.

Service organizations form complex business relationships with other organizations to exchange value. Competition continually forces organizations to improve their operations. Such improvements include out-sourcing or in-sourcing business tasks based on appropriate strategic considerations. Mergers, acquisitions, and alliances change the partners of a value network. The business processes needed to support such dynamic interactions tend to be complex.

Existing techniques for modeling, operationalizing, and evolving such processes are inadequate, because they are based on low-level abstractions at the level of data and control flows, expressed in orchestrations or choreographies. These specifications do not capture the business intent of the interactions. They tend to over-constrain business behavior by mandating the exchange of a predetermined set of messages usually in an unnecessarily restrictive temporal order.

This paper proposes a commitment-based business metamodel, which captures value exchanges among business partners in terms of their commitments. This paper defines patterns based on the above metamodel as well as algorithms to verify the correctness of service engagements with respect to their designs.

Contributions. The main contributions of this paper are (1) a commitment-based metamodel that describes value exchanges among business partners, (2) a set of business modeling patterns, and (3) algorithms for verifying (a) implemented agent interactions with respect to a business model and (b) the completeness of a business model.

Organization. Section 2 presents the business metamodel and a set of business patterns. Section 3 applies the patterns to create a model for an insurance claim processing scenario. Sections 4 and 5 introduce notions of compliance and completeness respectively, and provide algorithms for checking them. Section 6 compares our approach with related work.

2. METAMODEL AND PATTERNS

A business model seeks to capture value exchanges and the evolution of commitments among business partners. We characterize a business model via a set of *business relationships*, the participants of which we term *(business) partners*. The partners execute tasks for each other that enable achieving their respective *goals*. Importantly, our approach defines relationships in terms of the creation and manipulation of *commitments* among the partners. To enter into a business relationship, each partner takes on the commitments that the relationship specifies. The partner presumably possesses the *capabilities* that the relationship requires—these are presumably required to perform the *tasks* that would discharge the specified commitments.

We associate interaction protocols with business relationships in two main ways. Interaction protocols are crucial both to (1) creating or modifying a business relationship, such as via negotiation and (2) to enacting a business relationship. Figure 1 illustrates our metamodel. The following paragraphs describe the key concepts.

Agent: a computational representation of a business partner. An agent captures the autonomy and heterogeneity of a real-world business. An agent has goals and possesses a set of capabilities that enable it to execute business tasks. For each business relationship in which an agent participates, it enacts one or more roles in that relationship.

Role: an abstraction over agents that helps specify a business relationship. Each role specifies the commitments expected of the agents who play that role along with the capabilities they must possess to function in that role.

Goal: a state of the world that an agent desires to be brought about [3]. In simple terms, an agent's goals are its ends. An agent achieves a goal by executing appropriate tasks.

Task: a business activity viewed from the perspective of an agent.

Cite as: Business Modeling via Commitments, Pankaj R. Telang, Munindar P. Singh, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

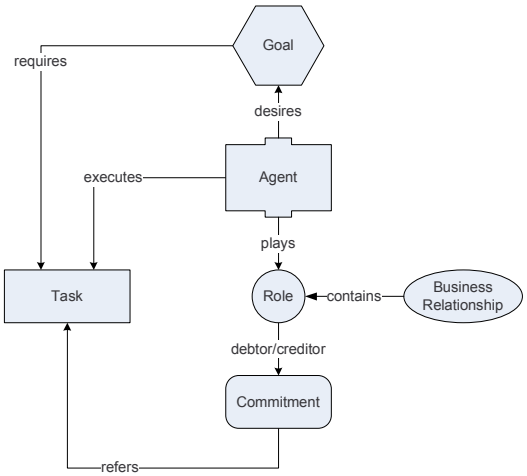

Figure 1: Business model

Value transfers between the agents when they execute tasks for one another.

Capability: an abstraction of the tasks that an agent can perform.

Commitment: a directed obligation from a debtor to a creditor [8]. A commitment C(DEBTOR, CREDITOR, antecedent, consequent) denotes that the DEBTOR is obliged to the CREDITOR for bringing about consequent if antecedent holds. A commitment C(BUYER, SELLER, goods, pay) means that buyer commits to paying the seller if goods are delivered. When the seller delivers goods, the buyer becomes unconditionally committed to paying. In the event that the buyer makes a payment, this commitment is discharged.

At run-time, commitments arise between agents, but at design-time we specify them between roles. Being able to manipulate commitments yields the flexibility needed in open interactions. A commitment may be *created*. When its consequent is brought about, regardless of whether antecedent holds or not, it is *discharged*, i.e., satisfied. If its antecedent is brought about then it is *detached*. The creditor may *assign* a commitment to another agent. Conversely, a debtor may *delegate* a commitment to another agent. A debtor may also *cancel* a commitment and a creditor may *release* the debtor from the commitment. Further, a commitment moves among four main states: *active* (when it is created and (presumably) being worked upon), *pending* (when it has been delegated and is not being worked upon), *satisfied*, and *violated*.

Business relationship: a set of interrelated commitments among two or more roles that describe the value to be exchanged among the roles.

In simple terms, each agent's main motivation behind forming a business relationship is to access the capabilities of others.

2.1 Running Example

We evaluate the proposed metamodel and patterns via a real-world insurance claim processing use case involving AGFIL, an insurance company in Ireland [4]. AGFIL underwrites automobile insurance policies. Fig. 2 shows the parties and processes involved in the business service of (emergency) claim processing that AGFIL provides.

To provide this service, AGFIL must provide claim reception and vehicle repair to the policy holders. Additionally, it needs to assess claims to protect against fraud. AGFIL depends on its partners,

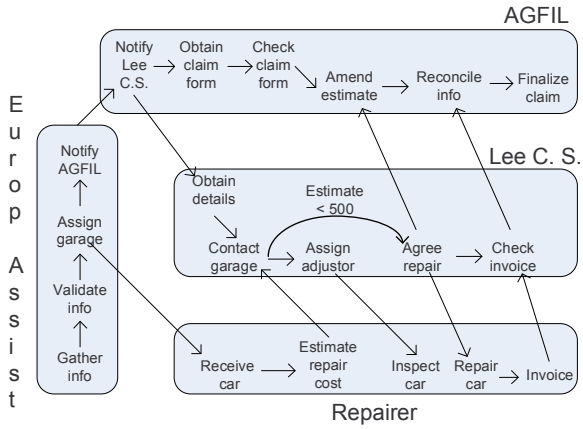

Figure 2: Insurance claim processing [4]

Europ Assist (EA), Lee Consulting Services (CS), and repairers, for executing these tasks. EA provides a 24-hour helpline for customers to report a claim and provides an approved repairer garage. CS assesses and presents invoices to AGFIL on behalf of the repairers. A network of approved repairers provide repair services. AGFIL retains the authority for final claim approvals.

2.2 Patterns

A pattern, in the context of this paper, is a recipe for modeling recurring business scenarios. This section describes a key set of such patterns, which could seed a potential business model pattern library. Section 3 demonstrates the effectiveness of this simple set of patterns on an existing use-case based on a real-life scenario.

Of the 13 attributes in the classical template for object oriented design patterns [6], we use *name*, *intent*, *motivation*, *implementation*, and *consequences* to describe our patterns. Here the *consequences* of a pattern allude to the practical consequences of applying the pattern, i.e., the assumptions underlying the model. The pattern figures use the notation of Fig. 1, and additionally show two directed edges for each commitment: from the debtor to the commitment and from the commitment to the creditor. The subscript on a commitment indicates its state: A for active, D for detached, S for satisfied, and P for pending. The patterns are expressed in terms of roles and would be instantiated by the agents who adopt the specified roles. Each role of a pattern must be adopted by some agent in order for the resulting business relationship to be executed.

2.3 Unilateral Commitment

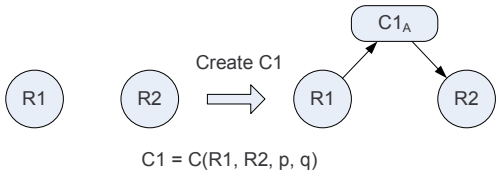

Figure 3: Unilateral commitment

Intent: A performer commits to a beneficiary for value transfer. There is no commitment from the beneficiary.

Motivation: For example, a conference committee member commits to a program chair to review a paper that the program

asks the member to review it. The chair makes no converse commitment.

Implementation: A commitment is created from the performer (R_1) to the beneficiary (R_2) for a value transfer. Figure 3 shows this pattern.

Consequences: This presumes a side benefit to the performer.

2.4 Commercial Transaction

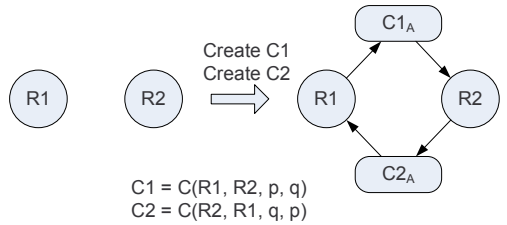

Figure 4: Commercial transaction

Intent: This pattern expresses a value exchange between two trading partners. The trading partners negotiate and, upon agreement, commit to each other for the specified value transfers.

Motivation: A typical barter motivates this pattern. For example, a seller and a buyer agree to exchange goods for payment. A more conventional barter would be when the parties exchange goods and services rather than money for goods or services.

Implementation: A pair of reciprocal commitments between the trading partners (R_1 and R_2, treated symmetrically) specify the pattern. Figure 4 shows this pattern.

Consequences: In general, the antecedents and consequents of the commitments are both composite expressions. Importantly, we need a mechanism to ensure progress by in essence breaking the symmetry, e.g., via a form of concession [10].

2.5 Outsourcing

Intent: An outsourcer delegates a task to a subcontractor, typically because the outsourcer lacks the necessary capabilities or expects some other benefit such as a more efficient solution or a lower risk of failure.

Motivation: Many business organizations outsource noncore activities. As an example, consider a customer who signs up for cable television service. The cable operator commits to the customer for installation. Instead of staffing its entire service area, the cable operator outsources the installation task in several regions to its local partners in those regions.

Implementation: The outsourcer is the current debtor (R_1). The current debtor and the new debtor (R_2) create a relationship, following which the current debtor delegates the commitment to the new debtor. The existing commitment becomes pending; the new commitment becomes active. The creditor is unchanged. Figure 5 shows this pattern.

Consequences: The business relationship between the new and old debtors would be a standing arrangement, which must have a scope and lifetime no smaller than that of the delegated commitment. The commitment from the old debtor is pending and must either be considered discharged or reactivated depending on how the new debtor performs.

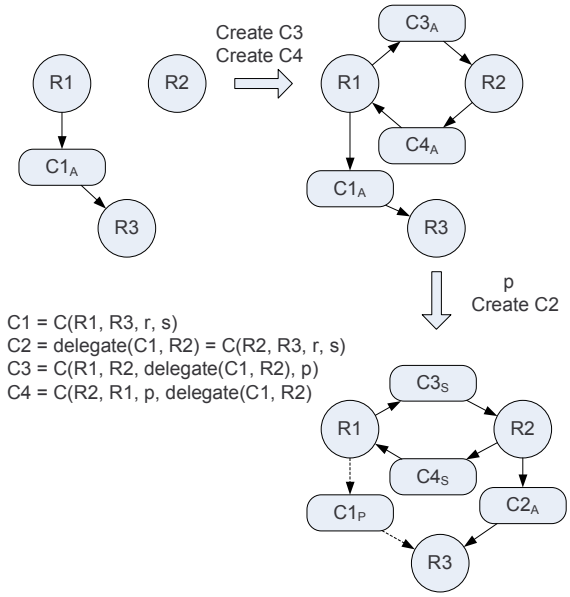

Figure 5: Outsourcing

2.6 Standing Service Contract

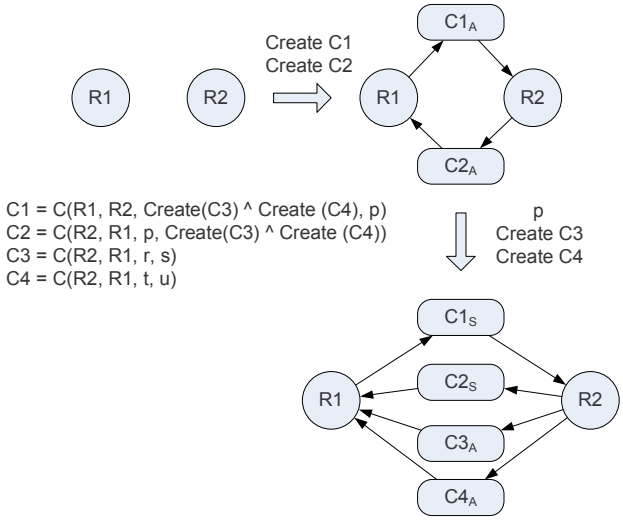

Figure 6: Standing service contract

Intent: A service provider negotiates with a consumer for providing service over a specified duration, and creates a pair of commitments. The consumer's request for a service instance detaches the standing commitment. The provider then creates one or more commitments for providing the service instance.

Motivation: A business service such as plumbing maintenance or a line of credit from a bank refers to (potentially) numerous service instances. Whenever the faucet leaks (within specified limitations), the plumber will fix it. Whenever the customer submits a check for an amount up to the specified credit limit, the bank will disburse funds.

Implementation: The service provider (R_1) and consumer (R_2) enter into the following commitments. Here, C_1 and C_2 are reciprocal commitments (as in the commercial transactions pattern) that describe the standing service contract. C_3 and C_4 arise from the consumer exercising the service contract. Figure 6 shows this pattern.

Consequences: The standing contract must be of sufficiently large scope to cover the cases of interest but should generally be bounded in the effort it requires. This pattern can be applied multiple times as when a consumer pays a subscription every month to obtain a continuing plumbing warranty.

3. AGFIL BUSINESS MODEL

This section applies the patterns to the AGFIL scenario and describes the resulting business model. AGFIL, an insurer (I), has a goal to provide emergency service, which requires the capabilities for claim reception, claim assessment, claim finalization, and vehicle repair. Except claim finalization, which it possesses locally, AGFIL acquires the remaining capabilities from its partners.

The insurer delegates to the call center its claim reception commitment to the policy holder. Although the commitment from the insurer to the policy holder for claim reception is not created yet, the insurer chooses to set up the delegation earlier. The *outsourcing pattern* models this scenario. The insurer selects EA as a call center provider (C). The selection process is out of our present scope. Figure 7 shows how the outsourcing pattern applies.

C1. C(C, I, payCallcenter, create(C3))

C2. C(I, C, create(C3), payCallcenter)

C3. C(C, P, reportAccident, receiveClaim)

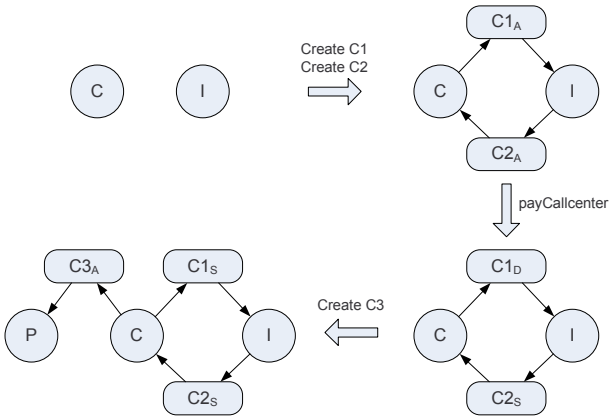

Figure 7: Claim reception: Outsourcing

The insurer and the call center agree upon the payment that the insurer makes to the call center, for providing claim reception to the policy holder, and create commitments C1 and C2. The commitment C1 means the call center commits to the insurer for creating commitment C3, which is to receive claims from the policy holder, provided the insurer pays the call center. The commitment C2 means the insurer commits to the call center for payment if the call center creates C3. The insurer pays the call center, and therefore discharges C2 and detaches C1. Later, the call center creates C3 and discharges C1.

The insurer outsources the claim assessment capability to Lee CS, an assessor. In this case, the *outsourcing pattern* does not apply since the insurer is not delegating a commitment. That is, the insurer requires claim assessment for itself. Instead, the *commercial transaction pattern* models this scenario.

C4. C(A, I, payAssessor ∧ reqAssessment, agreeToRepair)

C5. C(I, A, agreeToRepair, payAssessor)

The commitment C4 means the assessor commits to the insurer, for negotiating repair cost and to bring about the agreement to repair with the repairer, provided the insurer pays the assessor and makes a request for assessment. The commitment C5 means the insurer commits to the assessor for the payment provided the assessor brings about agreement to repair.

The assessor outsources the vehicle inspection to an adjuster (D). The *commercial transaction pattern* models this scenario. Since this scenario is similar to the claim assessment scenario, to save space, we do not describe it in detail.

A policy holder (P) desires to get insurance. Through a directory service, the policy holder locates AGFIL, the insurer. The policy holder and the insurer interact to setup the insurance service contract. The *service contract pattern* models this scenario. Figure 8 shows how the service contract pattern applies.

C8. C(P, I, insurance, payInsurer)

C9. C(I, P, payInsurer, insurance)

C10. C(I, P, reportAccident, receiveClaim)

C11. C(I, P, requestService, repairVehicle)

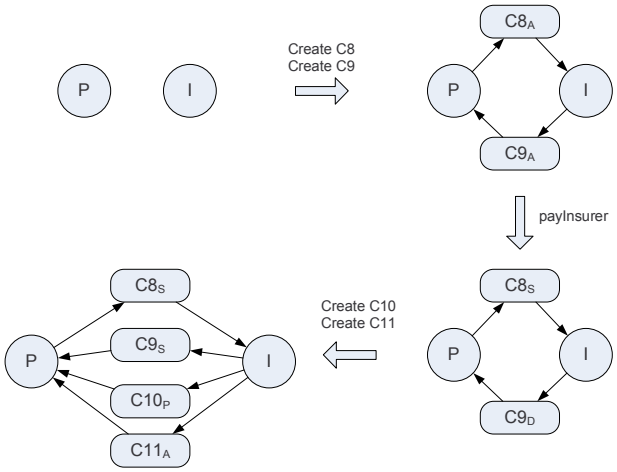

Figure 8: Insurance purchase: Service contract

Commitment C8 means the policy holder commits to the insurer for payment if insurance is provided, and commitment C9 means the insurer commits to the policy holder for insurance if the policy holder pays the insurer. To provide insurance, the insurer creates the commitments C10 and C11, that is, insurance = create(C10) ∧ create(C11). Commitment C10 means the insurer commits to receiving claim if the policy holder reports an accident, and in commitment C11, the insurer commits to repairing the (insured) vehicle if the policy holder requests repair service for it. The insurer changes the status of commitment C10 to pending, since it has delegated that commitment to the call center. Recall that C3 results from the delegation of C10. That is, C3 = delegate(C10, C).

To assess a claim, the assessor has the adjuster inspect the vehicle. The assessor negotiates with the repairer. By bringing about an agreement to repair, the assessor satisfies its commitment to the insurer C4. Figure 9 shows how the *outsourcing pattern* now applies between the insurer, the repairer, and the policy holder.

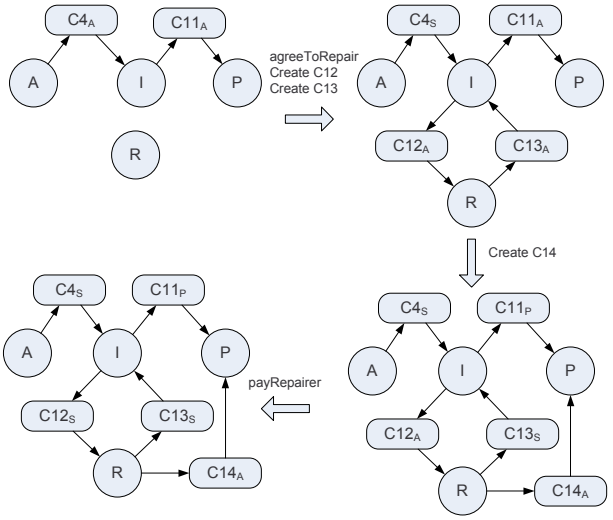

Figure 9: Vehicle repair: Outsourcing

C12. C(I, R, delegate(C11, R) ∧ agreeToRepair, payRepairer)

C13. C(R, I, payRepairer, delegate(C11, R))

C14. delegate(C11, R) = C(R, P, requestService, repairVehicle)

Commitment C12 means the insurer commits to the repairer for paying the repair charges, if the repairer accepts the delegation of C11 and creates C14. Commitment C13 means the repairer commits to accepting the delegation of commitment C11 if the insurer pays. In the delegated commitment C14, the repairer commits to the policy holder for vehicle repair when the policy holder requests for repair. The repairer satisfies the commitment C13 by creating C14, and detaches C12. Later the insurer discharges C12 by paying the repairer. Note that it is not necessary for the insurer to pay the repairer at this time, and other evolutions are possible. For example, the repairer may repair the vehicle, that is, satisfy the commitment C14, before the insurer pays. We describe one possible model evolution above.

4. VERIFYING AGENT INTERACTIONS

This section presents an algorithm for verifying if each partner complies with a business model. An agent complies with a business model if it discharged each detached commitment of which it is the debtor. We consider a UML sequence diagram as a low-level model for agent interactions. The agents may exchange multiple messages for executing one task. For example, the policy holder may report an accident by sending a message to the insurer; the insurer may request additional information, leading to further messages. In the interaction model (based on a sequence diagram), we assume that upon completing a task, the executor of the task sends a message asserting its completion.

Given a business model and an interaction model, Algorithm 1 returns a set of violated commitments. We assume that the interaction model captures all agent interactions. The algorithm iterates over the commitments from the business model and evaluates the antecedent and consequent of each using the tasks asserted in the interaction model. The antecedent and consequent of a commitment are formulae each containing a disjunction of tasks. The $eval$ procedure evaluates these based on the tasks asserted in the interaction model. The commitments whose consequent evaluates to true are satisfied, whereas the commitments whose antecedent evaluates to true, but whose consequent evaluates to false, are detached

Algorithm 1: verifyInteractions(m, i): Verify agent interaction model i with respect to business model m

```
1  C = m.C; // Model Commitments
2  CS = (); // Satisfied commitments
3  CV = (); // Violated commitments
4  T = i.T; // Tasks completed in the interaction model
5  foreach c ∈ C do
6      if (eval(c.consequent, T) = true) then
7          CS.add(c);
8  foreach ((c ∈ C) ∧ (c ∉ CS)) do
9      if (eval(c.antecedent, T) = true) then
10         CV.add(c)
11 return CV;
```

commitments that are violated. The debtors of the violated commitments are the agents that do not comply with the given business model (within the scope of the given interaction model).

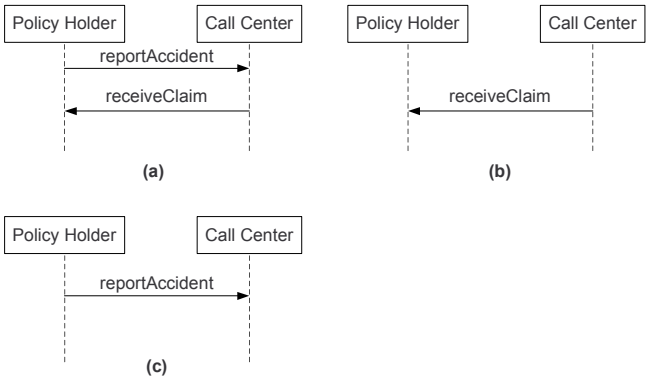

Figure 10: Verifying agent interactions

For example, in the AGFIL business model, consider the commitment C10 = C(C, P, reportAccident, receiveClaim). An interaction model in which neither of the tasks, reportAccident and receiveClaim, are asserted, is a trivial case where both agents, the policy holder (P) and the call center (C), comply with the business model. In Fig. 10(a), the policy holder reports an accident, and detaches the commitment C10. The call center receives the claim, and therefore, satisfies the detached commitment C10. In this case, both the agents comply with the business model. In Fig. 10(b), the call center receives the claim and satisfies the commitment C10. This is another case where both the agents comply with the business model. In Fig. 10(c), the policy holder reports an accident, but the call center does not receive the claim. The call center violates the detached commitment C10, and lacks compliance with the business model.

5. COMPLETENESS

Agents enter in a business relationship for achieving their respective goals. A business model in which all agents achieve their goals is complete. It is important to check for model completeness, since in its absence, some agents will not achieve goals and desire to leave the relationship. That is, the business model will not be stable.

The Algorithm 2 checks a model for completeness. For each agent, the algorithm checks if the agent can achieve all of its goals.

Algorithm 2: verifyCompleteness(m): Verify completeness of business model m

```
1  C = m.C; // Model Commitments
2  A = m.A; // Agents
3  foreach (a ∈ A) do
4      G = a.G; // Agent goals
5      foreach (g ∈ G) do
6          GT = g.T; // Tasks for goal
7          AT = a.T; // Agent tasks
8          task: foreach ((t ∈ GT) ∧ (t ∉ AT)) do
9              foreach (c ∈ C) do
10                 if ((c.creditor = a)∧
                   (t ∈ tasks(c.consequent)) ∧
                   (tasks(c.antecedent) ⊂ AT)) then
11                     next task;
12             return false;
13 return true;
```

An agent a can achieve a goal g, if it can execute all the tasks required for that goal. In case where the agent cannot execute all the tasks required for its goal, the model must contain commitments from other agents to execute the remaining tasks. Additionally, the agent a should be able to execute the tasks specified in the antecedents of those commitments. In the model, if there is an agent who cannot achieve a goal, then the algorithm returns false indicating that the model lacks completeness. Otherwise, the algorithm returns true.

For example, consider the AGFIL business model. The assessor has the goal of claim assessment. To assess a claim, the assessor needs to inspectVehicle and agreeToRepair. The assessor has the capability of bringing about agreeToRepair, but it lacks the capability to inspectVehicle. In this case, for completeness, the model must contain commitment from some other agent to inspectVehicle. Additionally, the assessor should be able to bring about the antecedent of that commitment. For example, C(D, A, payAdjuster, inspectVehicle) is a commitment required for model completeness, assuming the assessor can payAdjuster.

6. DISCUSSION

This section compares our approach with some existing approaches. Existing high-level approaches capture business organizations and value exchanges among them [1]. Many of these approaches are semiformal and are developed for valuation and profitability analysis. They lack a rigorous treatment of business relationships (as via commitments) and lack a corresponding business-level notion of compliance.

Gordijn and Wieringa [7] propose the e^3-value approach, which captures a business organization as an actor. This is similar to the notion of an agent from our model. Actors execute value activities similar to the tasks in our model. In e^3, a value interface aggregates related in and out value ports of an actor to represent economic reciprocity. This concept is close to our concept of commitment, but it lacks formal semantics and doesn't yield equivalent flexibility. For example, commitments can be delegated unlike value interfaces. Due to this, an e^3 model may capture value exchange among two actors, but during execution, the exchange and interaction may take place between two different actors.

Tropos [2] is an agent-oriented software methodology based on concepts of actor, goal, plan, and actor dependencies. The concepts of role, goal, and task from our model are similar to the Tropos concepts of actor, goal, and plan, respectively. A key difference between our model and Tropos is the concept of commitment. In Tropos, a dependency means that a depender actor depends on a dependee actor, for executing a plan or achieving a goal. This concept of dependency does not model what is required of the depender, and the dependee unconditionally adopts the dependency. Our debtor, creditor, and consequent are similar to the Tropos dependee, depender, and dependum, respectively. Unlike a dependency, a commitment includes an antecedent that brings it into full force. This allows modeling of reciprocal relationships between economic entities, which is lacking in the concept of dependency.

Opera is a framework for modeling multiagent societies [9], though from the perspective of a single designer or economic entity. In contrast, we model interactions among multiple entities. Opera's concepts of landmark, scene, and contract are close to our concepts of task, protocol, and commitment, respectively. However, Opera uses traditional obligations, which lack the flexibility of commitments.

Amoeba [5] is a process modeling methodology based on commitment protocols. This methodology creates model in terms of fine-grained messages and commitments. In contrast, our model is at a higher level of abstraction containing business goals, tasks, and commitments.

Conclusion: The main contributions of this paper are a business metamodel, a set of modeling patterns, and algorithms for verifying compliance and completeness of service engagements to business models. Our set of business model patterns is clearly not exhaustive; nor do we expect any set of patterns to be exhaustive—hundreds of patterns exist for programming and for software architecture, and the domain of business models is at least as complex as those. However, our core set of patterns shows how we may construct additional patterns. Future work includes development of a methodology for business modeling, model formalization and complexity analysis, and graphical tools for creating business models.

7. REFERENCES

[1] Birger Andersson, Maria Bergholtz, Ananda Edirisuriya, Tharaka Ilayperuma, Paul Johannesson, Jaap Gordijn, Bertrand Grégoire, Michael Schmitt, Eric Dubois, Sven Abels, Axel Hahn, Benkt Wangler, and Hans Weigand. Towards a reference ontology for business models. In David W. Embley, Antoni Olivé, and Sudha Ram, editors, *Conceptual Modeling - ER 2006, 25th International Conference on Conceptual Modeling, Tucson, AZ, USA, November 6-9, 2006, Proceedings*, volume 4215 of *Lecture Notes in Computer Science*, pages 482–496. Springer, 2006.

[2] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.

[3] BRG. The business motivation model, 2007.

[4] Sinead Browne and Michael Kellett. Insurance (motor damage claims) scenario. Document Identifier D1.a, CrossFlow Consortium, 1999.

[5] Nirmit Desai, Amit K. Chopra, and Munindar P. Singh. Amoeba: A methodology for modeling and evolution of cross-organizational business processes. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 2009. To appear.

[6] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable*

Object-Oriented Software. Professional Computing Series. Addison-Wesley, Reading, MA, 1995.

[7] Jaap Gordijn and Roel Wieringa. A value-oriented approach to E-business process design. In Johann Eder and Michele Missikoff, editors, *Advanced Information Systems Engineering, 15th International Conference, CAiSE 2003, Klagenfurt, Austria, June 16-18, 2003, Proceedings*, volume 2681 of *Lecture Notes in Computer Science*, pages 390–403. Springer, 2003.

[8] Munindar P. Singh. An ontology for commitments in multiagent systems: Toward a unification of normative concepts. *Artificial Intelligence and Law*, 7:97–113, 1999.

[9] Hans Weigand, Virginia Dignum, John-Jules Ch. Meyer, and Frank Dignum. Specification by refinement and agreement: Designing agent interaction using landmarks and contracts. In Paolo Petta, Robert Tolksdorf, and Franco Zambonelli, editors, *ESAW*, volume 2577 of *Lecture Notes in Computer Science*, pages 257–269. Springer, 2002.

[10] Pınar Yolum and Munindar P. Singh. Enacting protocols by commitment concession. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 116–123, May 2007.