In: Proceedings of the 3^{rd} Healthcare Digital Libraries Workshop (HDL 2005), Vienna, Austria, September 2005.

Semantic Service Composition and Co-ordination in *CASCOM* *

Thorsten Möller Heiko Schuldt
UMIT
Information and Software Engineering
Hall in Tyrol, Austria
thorsten.moeller|heiko.schuldt@umit.at

Andreas Gerber Matthias Klusch
German Research Center for
Artificial Intelligence
Saarbrücken, Germany
klusch|gerber@dfki.de

ABSTRACT

Semantic (Web) services allow for a fine-grained description of the functionality of single services and highly facilitate the combination and composition of several services into processes. The traditional workflow and process management approach considers the definition of a process at build-time without taking into account the service instances that are actually available at run-time. Moreover, failures have to be anticipated in order to define appropriate failure handling strategies. In this paper, we present an agent-based approach where process execution is distributed among a set of agents. A dedicated planning component composes semantic services based on the particular goals of an application. In case of failures, the planner is re-invoked in order to define contingency execution strategies. Finally, instance matchmaking is done at run-time by choosing the most appropriate service provider (according to pre-defined quality-of-service constraints). The focus of this paper is on the interaction of planning, matchmaking, and execution of processes (compound services) consisting of invocations of semantic web services. In the EU-funded project CASCOM, these technologies are currently applied to the composition of semantic services from the healthcare domain in order to run individualized applications (processes), thereby providing access to an eHealth digital library of services and data.

Keywords

Agent systems, Service co-ordination, Service matchmaking, Service composition planning, Composite service execution

1. INTRODUCTION

Enriching the conventional syntactic description of a service with information on its semantics allows for a more focused search for appropriate services in a large-scale network. However, complex applications usually require the combination and composition of several (semantic) services into compound services or processes, respectively. The traditional workflow and process management approach considers the definition of a process at build-time but does not take into account the service instances that are actually available at run-time. Failures have to be anticipated and appropriate failure handling also has to be defined already at build-time. In this approach, unforeseen failures cannot be handled. In addition, usually a centralized approach is followed that implies a single point of failure and that does not scale well with the number of processes to be executed and the number of semantic services available.

The goal of the CASCOM project (Context-Aware Business Application Service Co-ordination in Mobile Computing Environments) [5] is to overcome these limitations by implementing, validating, and testing a value-added supportive infrastructure for business application services for mobile workers and users across mobile and fixed networks. The driving vision of CASCOM is that ubiquitous business application services are flexibly co-ordinated and pervasively provided to the mobile worker/user by intelligent agents in dynamically changing contexts of open, large-scale, and pervasive environments.

The CASCOM architecture is divided into several layers (see Figure 1). The planned technological innovations at each layer can be summarized as follows. The main outcome of the network layer is a generic, secure, and open intelligent agent-based peer-to-peer (IP2P) network infrastructure taking into account varying quality-of-service (QoS) of wireless communication paths, limitations of resource-poor mobile devices, and contextual variability of nomadic environments. IP2P environments are extensions to conventional P2P architectures with components for mobile and ad hoc computing, wireless communications, and a broad range of pervasive devices. The main outcomes of the service co-ordination layer are i.) flexible semantic Web service discovery including adaptive service QoS-oriented service matching and usage of distributed semantic Web service directories (DSD), ii.) dynamic context-aware semantic Web service composition including resource-efficient interaction between DSD and service composition planner, fault-tolerant interleaving of planning and service execution, and iii.) secure service execution and monitoring providing service data consistency.

In this paper, we present the agent-based approach to pro-

*This work is supported by the EU in the 6^{th} Framework Programme within the STREP CASCOM (Context-Aware Business Application Service Co-ordination in Mobile Computing Environments), contract no. 511632.

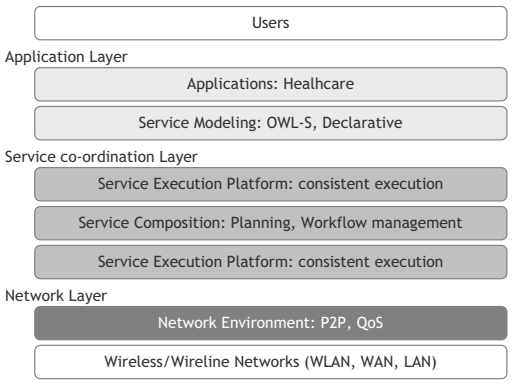

Figure 1: Layered CASCOM architecture

cess generation and execution followed in CASCOM. Process execution is distributed among a set of co-operating service provider agents. Each agent works off its part of a process (i.e., locally invokes the required services) and then forwards control to the next agent that is then in charge of continuing process execution. Processes are not defined statically. Rather, a dedicated planning component composes semantic services based on the particular goals of an application. In case of failures, the planner is re-invoked in order to define contingency execution strategies. Finally, instance matchmaking is done at run-time by choosing the most appropriate agent (according to pre-defined QoS constraints) among a set of agents qualifying for the execution of a particular semantic service.

The focus of this paper is the interaction of planning, matchmaking, and execution of processes consisting of invocations of semantic web services. In particular, we apply these technologies to semantic web service composition in a healthcare scenario (emergency assistance), which supports people traveling in foreign countries with the healthcare services they need when suddenly suffering from illnesses and needing medical treatment and care. This of course requires access to services and data in an eHealth digital library.

The paper is organized as follows: In Section 2, we present an emergency assistance scenario we focus on in CASCOM. The different technologies needed to support this scenario are introduced in Section 3 and Section 4 shows how these can be seamlessly integrated. Section 5 discusses related word and Section 6 concludes.

2. SAMPLE HEALTHCARE APPLICATION

In the following, a sample business application service scenario *Emergency Assistance* is described (see Figure 2). The scenario is based on the fact that people on the move, e.g. travelling in foreign countries for business or holidays, may get into situations where they need medical assistance because of a sudden disease or emergency. Currently, these sorts of episodes are neither tackled nor realised in this form in practice and no software system is presently widespread in use to address them.

Alice and Bob, tourists from Finland, are abroad on a countryside journey in Austria during their summer vacation. They carry a PDA, already equipped with the CASCOM mobile agent suite. Suddenly after some days, Alice is seriously suffering from unknown pain in the upper part of her body. For this reason, she wants to immediately call a hospital or physician. After activation of the PDA, the agent immediately finds out contact information of a local healthcare institution next to them[1]*. Additionally, the agent gives them contact of the Finnish representative of the Emergency Medical Assistance service centre (EMA) that takes care of the remote support of the patient. Alice decides to immediately go to the local hospital. The agent on the PDA also supplies them with information on how to get there. This could be either a map showing their current location and the healthcare centre location, a phone number for a local taxi, or the instructions to get a connection via public transportation. On arrival and check-in at the local hospital, Alice has to manually answer some questions about her personal data because the healthcare institution does not provide the infrastructure and services to plug her PDA into the local information system, i.e., to exchange initial data. During the first examinations by the local emergency physician it bears out that Alice either had a silent heart attack or angina pectoris, but the physician is not sure about the diagnosis and wants to obtain a second opinion. Even Bob and Alice are concerned about the doubtful situation. Bob now uses the PDA to access a second opinion service by forwarding all information available so far. It should be quite obvious that if the hospital had the CASCOM infrastructure installed, the local physician could have used a local PC or PDA to access the second opinion service. However, in this case the agent on the PDA finds out the contact information of a specialized cardiologist and establishes a connection. After assessment of the situation, they both decide that Alice should be transferred soon to a hospital with advanced cardiac life support to undertake thorough examinations. Alice says that she wants to be transferred back to a hospital in her home country Finland. As a result, the EMA service centre will be contacted to organize the transfer by using the PDA – remember that contact information was transferred before. Now, EMA's agent first automatically investigates possible travel arrangements (depending on the medical circumstances and the geographical distance, the agent may eventually come up with a decision on whether using regular flights, a car, or some other form of transportation). Second, the agent informs all people that are involved during the transfer (doctors and escorts). Third, it contacts Alice's insurance company to make sure that her insurance will cover all possible transportation costs. In addition, the agent could possibly contact automatically the Finnish hospital (which participates in the CASCOM network) to make further arrangements. Back in Finland, Alice is treated at a sophisticated cardiac hospital. After two weeks of recovery she finally uses her PDA to send a "Thank you" to all the people involved with her medical case.*

As results of the scenario described above, persons (patients) not only need medical treatment, they also need informational as well as (sometimes) transportation assistance. Furthermore, assistance in the form of information is also required by the physicians, hospitals, or healthcare professionals involved. One straight implication of these complex requirements is the need for on-demand **initiation**, **composition**, **coordination**, and **supervision** of various ac-

[1]Their location is found out either by using the GSM network cell identifier, or by GPS.

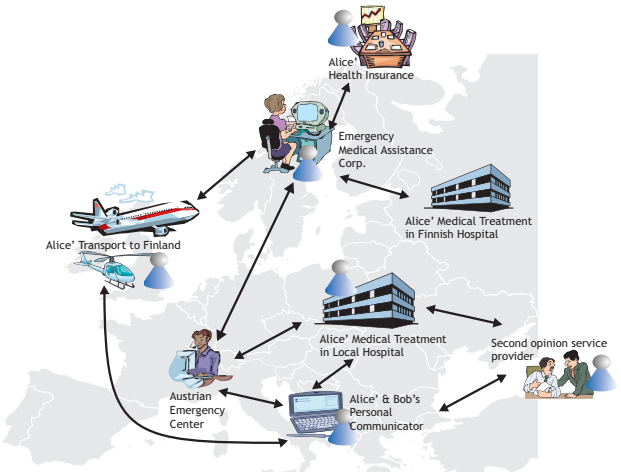

Figure 2: Emergency Assistance application

tivities represented mostly through non-human actors, like agents and services, but also though persons.

3. SERVICE COORDINATION LAYER

In this section, we introduce the basic technologies needed for the requirements derived from the application scenario. These requirements include i.) matchmaking – the selection of appropriate semantic Web Services (using semantic equivalence/similarity), ii.) composition planning – the composition of several semantic Web services into processes, and iii.) execution – a runtime environment for compound services.

3.1 Service Matchmaking

The Service Matchmaking functionality provides the means to compare the semantics specified for services, thus allowing to detect semantically equivalent/similar services. Several approaches for sophisticated semantic matchmaking of Web services have been proposed that rely on ontology-based languages (e.g., OWL-S [6], WSMO, and Annotated WSDL [19]) and are grounded with formal semantics such as Description Logics [2]. The most important principle of semantic matchmaking is that semantics of words used in the description of Web services are formally defined in ontologies. Those ontologies can be exploited by matchmaker agents to determine the degree of semantic matching of advertised services with a given service request.

For semantic matching of services specified in OWL-S, we have developed the OWLS-MX for hybrid matchmaking. It takes any OWL-S service description as a query, and returns an ordered set of relevant services that match the query, each annotated with its individual degree of matching (DOM), and syntactic similarity value. The user may extend the query by specifying the desired DOM, and a syntactic similarity threshold. OWLS-MX first classifies the service query I/O concepts into its local service I/O concept ontology. As usual, we assume that the type of computed terminological subsumption relation determines the degree of semantic relation between pairs of input and concepts. Attached to each concept in the concept hierarchy are auxiliary information on whether it is used as an input or output concept by any service that has been registered at the matchmaker. The corresponding I/O lists of unique service identifiers for input and output concepts are then used by the matchmaker to compute the set of relevant services that match the given query according to its five matching filters. In particular, OWLS-MX does not only pair-wisely determine the degree of logical match but syntactic similarity between the terminological expressions built by unfolding each of the considered query and service input (output) concept in the local matchmaker ontology. This way, logical subsumption failures produced by the integrated description logic reasoner of OWLS-MX are tolerated, if the syntactic similarity value computed by means of a specific information retrieval similarity metric is sufficient (i.e., exceeds the given threshold).

3.2 Service Composition Planning

The Service Composition functionality supports the context dependent composition of compound, value-added services whenever no appropriate single service can be found during matchmaking. In CASCOM we have different approaches of semantic web service composition, and OWLS-Xplan [16] is one opportunity we intend to use. OWLS-Xplan takes a set of available OWL-S services, related OWL ontologies, and a query as input, and returns a plan sequence of composed services that satisfies the query goal. For this purpose, it first converts the domain ontology and service descriptions in OWL and OWL-S, respectively, to equivalent problem and domain descriptions. The problem description contains the definition of all types, predicates and actions, whereas the domain description includes all objects, the initial state, and the goal state. Both descriptions are then used by the AI planner Xplan to create a composition plan that solves the given problem in the actual domain.

Xplan is a heuristic hybrid search planner based on the FF-planner [13]. It combines guided local search with graph planning, and a simple form of hierarchical task networks to produce a plan sequence of actions that solves a given problem. This yields a higher degree of flexibility opposed to pure Hierarchical Task-reduction Planning (HTN) whereas the use of predefined workflows or methods improves the efficiency of the FF-planner. In contrast to the general HTN planning approach, a graph-plan based planner is guaranteed to always find a solution independent from whether the given set of decomposition rules for HTN planning would allow to build a plan that contains only atomic actions. In fact, any graph-plan based planner would test every combination of actions in the search space to satisfy the goal which, of course, can quickly become prohibitively expensive. Xplan combines the strengths of both approaches. It is a graph-plan based planner with additional functionality to perform decomposition like a HTN planner.

The Xplan system consists of the XML parsing module, a pre-processing module, the planning core, and the re-planning module. After the domain and problem definitions have been parsed, Xplan compiles the information into memory efficient data structures. A connectivity graph is then generated, which contains information about connections between facts and instantiated operators, as well as information about numerical expressions which can be connected to facts. This connectivity graph is maintained during the whole planning process and serves as a kind of effi-

cient lookup table for the actual search.

Xplan uses an enforced hill-climbing search method to prune the search space during planning, and a modified version of relaxed graph-planning that allows to use (decomposition) information from hierarchical task networks during the efficient creation of the relaxed planning graph, if required, such as in partially hierarchical domains. Information on the quality of an action (service) are utilized by the local search to decide upon two or more steps that are equally weighted by the used heuristic. In addition, Xplan includes a re-planning component which is able to re-adjust outdated plans during execution time (see Section 4).

3.3 Service Execution

The service execution system (SES) executes compound services (in what follows, we will use the term process synonymously) as they are generated by the service composition planner agent (SCPA). For process execution, we first assume that a compound service contains an arbitrary number of service invocations whereby the composition structure is equal to an acyclic ordered graph, i.e., combined sequential and parallel flows together forming processes as denoted in [22]. Second, as a basis for correct process execution, each service invocation is assumed to be atomic and compensatable. This means that the effects of a service can be undone later Otherwise, unwanted side effects of aborted or compensated executions may remain and at-most-once execution semantic could not be guaranteed. For services which do not comply with the atomicity requirement, we assume that a wrapper can be built which adds this functionality. Third, we assume that services are stateless, i.e., that they never have to remember anything beyond interaction. In our approach, process state (i.e., intermediate results) is solely stored by the execution system. Finally, our approach considers the crash failure model, which means that components such as services and machines may fail by prematurely halting their execution.

The execution system is based on principles of the OSIRIS (Open Service Infrastructure for Reliable and Integrated process Support) process management system [24]. Upon that, aspects of agent-oriented systems were introduced to fit into the CASCOM infrastructure. In particular, the execution system consists of one ore more federated execution agents organized in a peer-to-peer manner, meaning that no central execution coordinator is required. To accomplish this, every agent implements a process manager which coordinates execution basically by forwarding control and data to the next agent(s). Furthermore, we distinguish between two types of execution agents (SEA): service provider agents and standard agents. The difference between both is that the former is locally attached to one or more service instance(s) on the same machine (i.e., agent and service(s) run on the same device) whereas the latter may run on any computing device – especially mobile devices – and calls service(s) remotely. Nevertheless, both implement execution functionality completely according to our execution requirements.

The execution first involves decomposing the process model into its atomic execution units. An execution unit contains a service invocation s and links to all the services that are the

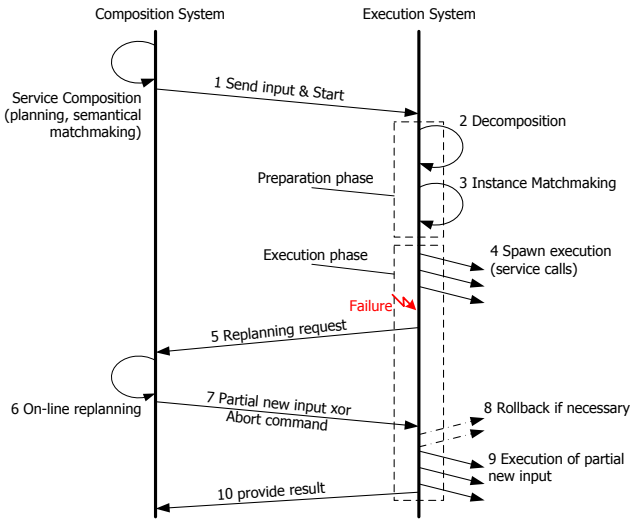

Figure 3: Interaction and execution model

direct successors of s. In addition, for failure handling purposes, also information on the predecessors of s is needed. This is important in order to determine which services need to be compensated (i.e., which effects need to be undone) when a failure during process execution occurs. This means that for every service only *links* to adjacent services are of interest. All in all, the units provide execution agents with all information they require to execute a service and to do forward navigation afterwards. The explicit distinction between control and data flow enables optimal interaction paths with as less communication efforts between execution agents as possible.

4. INTEGRATION OF SERVICE MATCHMAKING, COMPOSITION PLANNING, AND EXECUTION

As noted in Section 3.3, the SCPA acts as the client for SES. Consequently, our combined interaction and execution model consists of the following steps (see Figure 3). Before actual execution starts, the SCPA creates a new process using a planning algorithm and semantic matchmaking to employ some of the services in the domain according to their service descriptions. It then sends input (the newly generated process type) to SES and orders execution start (1) – note that the process type contains all necessary information for instantiation; just the individual service types still need to be bound to instances. Now, the execution preparation phase starts. Since a process instance is not suitable for execution on the physical layer, a detailed execution plan has to be created [23]. The most important part of this plan is the decomposition of the process into its execution units (2). The following step is called *instance matchmaking at runtime* (3), where a concrete service provider instance of given type will be selected based on most current QoS criteria like average load or execution costs[2].

The preparation phase is finished by distributing required information to the execution agents, such that they can

[2]This is important in case there is more than one service instance available with equal signature among them.

forward control on their own during execution. Then SES spawns service execution on behalf of the input (4), i.e. the execution phase starts. During execution, failures might happen, for example service instances or other infrastructure components might crash. In such a situation, execution cannot terminate or at least cannot continue without some recovery mechanism. In classical transactional systems, this would lead to an abort of the global transaction (i.e., all side effects created so far will be undone or compensated) and some external logic has to decide what to do next. In our approach, a crash failure situation does not necessarily end up in the abortion of process execution. After a failure, SES temporarily freezes process execution. In particular, if parallel execution paths exist, all of them will be frozen. Furthermore, SES knows about the current process state and the side-effects created. Then, SES transmits this information to the SCPA and requests on-line contingency re-planning (5) – remember that the original process execution goal still holds. Starting from the stop point the planner now tries to fix the problem by searching for an alternate path (6) – in all likelihood by employing semantically similar services. If SCPA succeeded in composing a partial new process of remaining activities, this new process fragment will be sent to SES (7). Otherwise, if it was not possible to find an alternative path, SCS sends an abort command to SES. Consequently, SES is then obligated to rollback the process side effects completely (8) – which is possible according to our assumption of atomic, compensatable services. In the former case, SES can continue execution with the new process fragment. In order to be able to do this, it first has to replace the old remaining process fragment (which is now obsolete) with the new one. This is accomplished by starting a new sub-preparation phase, whereby decomposition, instance matchmaking, and distribution to the service provider agents again takes place (i.e., update of the execution plan). Afterwards, execution continues (9). Replanning is also required in case the original goal for which the process has been generated is altered. If the new process fragment requires to partially undo side effects because of its changes (i.e., utilisation of other services), this will be done right before continuation. Finally, when execution has finished, the result will be sent back to SCPA (10). In order not to block resources endlessly during on-line re-planning, we use a timeout based approach: In case of no reply from SCPA to SES until the timeout (because SCPA crashed or connection has been lost), SES aborts the current process execution and tries to notify SCPA about that.

One aspect of our interaction model that is still open for discussion is whether we allow for indefinite re-planning phases. By allowing indefinite re-planning phases it is evident that execution theoretically might never terminate. On the other hand, and with the presented emergency assistance scenario in mind, probability for high numbers of re-planning cycles falls with the number of cycles: For emergency service providers it is crucial that their services are constantly available. If not, nobody would develop trust in such applications. However, because of different service providers, similar services (alternatives) are expected to exist. Thus, it is empirically evident that either an alternative is available early, which eventually leads to success or no alternative exists and execution stops entirely. A simple approach to address this issue is to fix a maximum re-planning cycle count for the implementation. A more sophisticated approach would be the definition of *execution progress*. If there is no significant progress towards the execution goal even though both SCPA and SEA are not inactive (i.e., execution stagnates) its value converges to zero. Thus, it is possible to detect stagnated executions and abort them eventually. All in all, the decision about which policy should be used for re-planing phases should not be made without taking the target application into account.

In the scenario presented in Section 2, Alice and Bob are first required to state the goal of the process they aim to be executed (e.g., transfer to a hospital, receive treatment from there while giving the local physician access to Alice's health record). Then, by combining matchmaking and planning, a process tailored to Alice's needs is generated and executed. In case of failures or of revised goals (e.g., second opinion is needed or later on transportation), planning is re-invoked and the process is changed (and executed) accordingly.

5. RELATED WORK

Similar to CASCOM, the ARTEMIS Project (A Semantic Web Service-based P2P Infrastructure for the Interoperability of Medical Information) [1] also aims at supporting healthcare applications by means of dedicated semantic Web services. However, ARTEMIS focuses at providing single semantic Web services and addresses standards and interoperability issues of these services, while the goal of CASCOM is to provide value-added, composite services in order to support sophisticated ad hoc process-based healthcare applications in IP2P environments.

Issues of service composition (planning) and co-ordination are currently widely addressed in research, especially if extension to semantic description of Web services comes into play. Some ontology-based approaches to semantic service matchmaking that have been proposed in the literature are LARKS [26], OWLS/UDDI [18], MAtchMAker-Service [7], RACER [17], Parameterized Semantic [10], HotBlu [8], and Signature-Specification [15]. Other approaches are either process-based (e.g., High-precision Service Retrieval Service [14]), peer-based (e.g., Semantic Web Services P2P Discovery Service [3]) or are hybrid approaches (e.g., the Recursive Tree Matchmaker [4]). Alternate approaches include graph based matching methods such as those presented in [28], [15], and [8]. Furthermore, there are currently only very few approaches and software tools available for OWL-S based service composition planning such as, for instance, OWL-S Composition Planner using SHOP2 [29], Logic-based DAML-S composition planning [25], the DAML-S workflow composer [27], a Petri-net approach in which an OWL-S service description is automatically translated into Petri-nets [12].

Finally, the issue of service execution is widely addressed in classical research domains like transactional information systems and process management. The OSIRIS infrastructure on which the SES is built provides a scalable distributed process navigation platform. To achieve this, it combines a rich set of aspects. Based on the hyperdatabase vision [21], ideas from process management, peer-to-peer networks, database technology, and Grid [11] infrastructures have been combined. Similar to OSIRIS where processes are running within

a peer-to-peer community that is established by the individual service providers, the MARCAs presented in [9] are service providers acting as peers. [20] provides a general overview on fault-tolerant agent based (process) execution.

6. CONCLUSIONS

In this paper, we have presented the CASCOM approach to providing access to services and data in an eHealth digital library. Ad hoc applications by means of processes are supported by seamlessly combining sophisticated service composition planning, service matchmaking, and agent-based distributed service execution. The binding of service types to service instances during runtime integrates well with the dynamic nature of the healthcare application domain, i.e., provides a high degree of flexibility.

The CASCOM infrastructure that is currently being built will be evaluated in detail in a real-world setting in a cooperation with TILAK, the umbrella organization of the state hospitals of the Austrian state Tyrol. In future work, we aim –among others– at addressing more sophisticated transaction and failure models, especially by considering malicious failures (i.e., Byzantine failures as they might appear in untrusted environments).

7. REFERENCES

[1] The ARTEMIS project. `http://www.srdc.metu.edu.tr/webpage/projects/artemis`.

[2] F. Baader and W. Nutt. The Description Logic Handbook-Theory. *Implementation and Applications, chapter Basic Description Logics, Cambridge University Press*, pages 47–100, 2003.

[3] F. Banaei-Kashani, C. Chen, and C. Shahabi. Wspds: Web services peer-to-peer discovery service. *ISWS 2004*, 2004.

[4] S. Bansal and J. Vidal. Matchmaking of web services based on the DAML-S service model. *AAMAS2003, Melbourne, Australia*, 2003.

[5] The CASCOM project. `http://www.ist-cascom.org`.

[6] T. O.-S. Coalition. OWL-S 1.0 (Beta) Draft Release. *Autonomous Agents and Multi-Agent Systems*, 2003.

[7] S. Colucci, T. D. Noia, E. D. Sciascio, F. Donini, M. Mongiello, G. Piscitelli, and G. Rossi. An agency for semantic-based automatic discovery of web-services. *In Artificial Intelligence Applications and Innovations. Proc. of IFIPWCC-04, Kluwer Academic Publishers*, pages 315–328, 2004.

[8] I. Constantinescu and B. Faltings. Efficient matchmaking and directory services. *The 2003 IEEE/WIC International Conference on Web Intelligence*, 2003.

[9] A. Dogac, Y. Tambag, A. Tumer, M. Ezbiderli, N. Tatbul, N. Hamali, C. Icdem, and C. Beeri. A Workflow System through Cooperating Agents for Control and Document Flow over the Internet. In *7th International Conference on Cooperative Information Systems (CoopIS 2000)*, pages 138–143, Eilat, Israel, Sept. 2000.

[10] P. Doshi, R. Goodwin, R. Akkiraju, and S. Roeder. Parameterized semantic matchmaking for workflow composition. *Technical Report RC23133, IBM Research, T.J. Watson Research Center, NY*, 2002.

[11] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, San Francisco, 2nd edition, 2004.

[12] R. Hamadi and B. Benatallah. A Petri-Net-Based Model for Web Service Composition. *Proc. 14th Australasian Database Conf. Database Technologies, ACM Press 2003*, pages 191–200, 2003.

[13] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, (14):253–302, 2001.

[14] M. Klein and A. Bernstein. Towards high-precision service retrieval. *IEEE Internet Computing, 8(1)*, pages 30–36, January 2004.

[15] M. Klein and B. Koenig-Ries. Coupled signature and specification matching for automatic service binding. *ECOWS*, pages 183–197, 2004.

[16] M. Klusch, A. Gerber, and M. Schmidt. Semantic Web Service Composition Planning with OWLS-Xplan. *First International Symposium on Agents and the Semantic Web*, 2005. To appear.

[17] L. Li and I.Horrocks. A software framework for matchmaking based on semantic web technology. *Proceedings of the twelfth international conference on World Wide Web, ACM Press*, pages 331–339, 2003.

[18] M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara. Semantic matching of web services capabilities. *Proceedings of the First International Semantic Web Conference on The Semantic Web, Springer-Verlag*, pages 333–347, 2002.

[19] A. Patil, S. Oundhakar, A. Sheth, and K. Verma. Meteor-s web service annotation framework. *Proceeding of the World Wide Web Conference*, 2004.

[20] S. Pleisch and A. Schiper. Approaches to Fault-tolerant and Transactional Mobile Agent Execution—an Algorithmic View. *ACM Comput. Surv.*, 36(3):219–262, 2004.

[21] H.-J. Schek, H. Schuldt, C. Schuler, and R. Weber. Infrastructure for Information Spaces. In *Advances in Databases and Information Systems, Proc. of the 6th East-European Symposium, ADBIS'2002*, pages 23–36, Bratislava, Slovakia, Sept. 2002.

[22] H. Schuldt, G. Alonso, C. Beeri, and H.-J. Schek. Atomicity and Isolation for Transactional Processes. *ACM Transactions on Database Systems (TODS)*, 27(1):63–116, Mar. 2002.

[23] C. Schuler, H. Schuldt, C. Türker, R. Weber, and H.-J. Schek. Peer-to-Peer Execution of (Transactional) Processes. *International Journal of Cooperative Information Systems (IJCIS)*, 2005. To appear.

[24] C. Schuler, R. Weber, H. Schuldt, and H.-J. Schek. Scalable Peer-to-Peer Process Management - The OSIRIS Approach. In *Proceedings of 2nd ICWS'2004*, pages 26–34, San Diego, CA, USA, July 2004. IEEE Computer Society.

[25] M. Sheshagiri, M. desJardins, and T. Finin. A planner for composing services described in DAML-S. *Proceedings of AAMAS 2003 Workshop on Web Services and Agent-Based Engineering*, 2003.

[26] K. Sycara, S. Widoff, M. Klusch, and J. Lu. LARKS: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace. *Kluwer Academic Press*, 2002.

[27] S. Tarkoma and M. Laukkanen. Adaptive agent-based service composition for wireless terminals. *In M. Klusch et al., editor, Proceedings of CIA VII, Helsinki, Finland, August 2003. Springer Verlag, LNAI 2782*, pages 16–29, 2003.

[28] D. Trastour, C. Bartolini, and J. Gonzalez-Castillo. A semantic web approach to service description for matchmaking of services. *Proceedings of SWWS*, 2001.

[29] D. Wu, B. Parsia, J. H. E. Sirin, and D. Nau. Automating DAML-S web services composition using SHOP2. *Proceedings of 2nd ISWC2003, Sanibel Island, Florida, USA*, pages 20–23, 2003.