Health Informatics Journal

Copyright © 2006 SAGE Publications (London, Thousand Oaks, CA and New Delhi)
Vol 12(2): 107–119 [1460-4582(200606)12:2;107–119; DOI: 10.1177/1460458206063802] www.sagepublications.com

Next-generation applications in healthcare digital libraries using semantic service composition and coordination

Thorsten Möller, Heiko Schuldt, Andreas Gerber and Matthias Klusch

Healthcare digital libraries (DLs) increasingly make use of dedicated services to access functionality and/or data. Semantic (web) services enhance single services and facilitate compound services, thereby supporting advanced applications on top of a DL. The traditional process management approach tends to focus on process definition at build time rather than on actual service events in run time, and to anticipate failures in order to define appropriate strategies. This paper presents a novel approach where service coordination is distributed among a set of agents. A dedicated component plans compound semantic services on demand for a particular application. In failure, the planner is reinvoked to define contingency strategies. Finally, matchmaking is effected at runtime by choosing the appropriate service provider. These combined technologies will provide key support for highly flexible next-generation DL applications. Such technologies are under development within CASCOM.

Keywords

agent systems, compound service execution, service composition planning, service coordination, service matchmaking

Introduction

The paradigm of service-oriented computing encapsulates functionality and makes use of it in a well-defined way by providing standardized interfaces for access and description. Digital libraries (DLs) in general and healthcare digital libraries in particular increasingly exploit services that encapsulate functionality and/or data. 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 (healthcare) DL applications usually require the combination and composition of several (semantic) services into compound services or processes. 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 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) [1] 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 coordinated and pervasively provided to the mobile worker/user by intelligent agents in dynamically changing contexts of open, large-scale, and pervasive environments. Validation and testing of the CASCOM architecture will take place in a real-world healthcare DL setting.

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

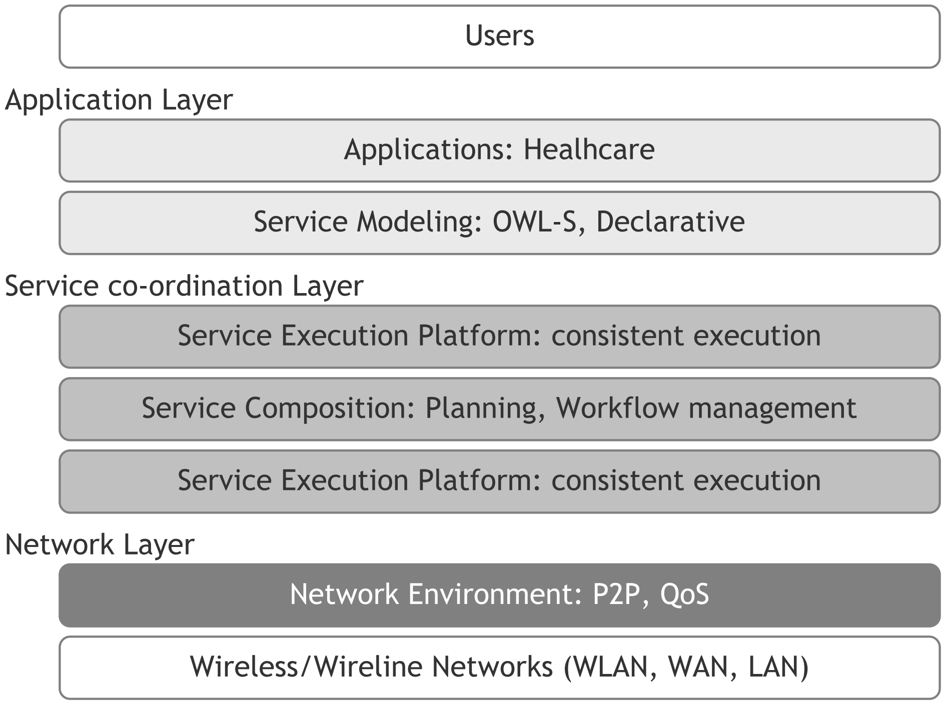
Users
Application Layer
Applications: Healhcare
Service Modeling: OWL-S, Declarative
Service co-ordination Layer
Service Execution Platform: consistent execution
Service Composition: Planning, Workflow management
Service Execution Platform: consistent execution
Network Layer
Network Environment: P2P, QoS
Wireless/Wireline Networks (WLAN, WAN, LAN)

Figure 1 Layered CASCOM architecture

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) properties 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. This means that heterogeneous and dynamic systems have to be linked together (e.g. smart phones, PDAs, computers), and that the infrastructure should be robust with no major point of failure while the deployment and maintenance effort should be minimal. Conceptually, the IP2P layer will be built to permit seamless mobility to the user. In contrast to existing wireless technologies, users are then able to roam through different physical network technologies like GSM, UMTS, and WLAN. One essential approach to achieve this is to build a network abstraction (overlay network) on top of the network infrastructure. The main outcomes of the service coordination layer are (1) flexible semantic web service discovery including adaptive service QoS-oriented service matching and usage of distributed semantic web service directories (DSD), (2) 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 (3) secure service execution and monitoring providing service data consistency.

In this paper, we present the novel CASCOM agent-based approach to process generation and execution. Process execution is distributed among a set of cooperating 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, which 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 failure, the planner is reinvoked 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. The combination of these technologies will provide key support for highly flexible next-generation DL applications that make use of semantic service descriptions. In particular, we apply these technologies to semantic web service composition in a healthcare DL application (emergency assistance), which supports people travelling 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 a healthcare DL.

The paper is organized as follows. In the next section we present an emergency assistance scenario which we focus on in CASCOM. Subsequent sections introduce the different technologies needed to support this scenario and show how these can be seamlessly integrated. A discussion of related work and our conclusions complete the paper.

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.

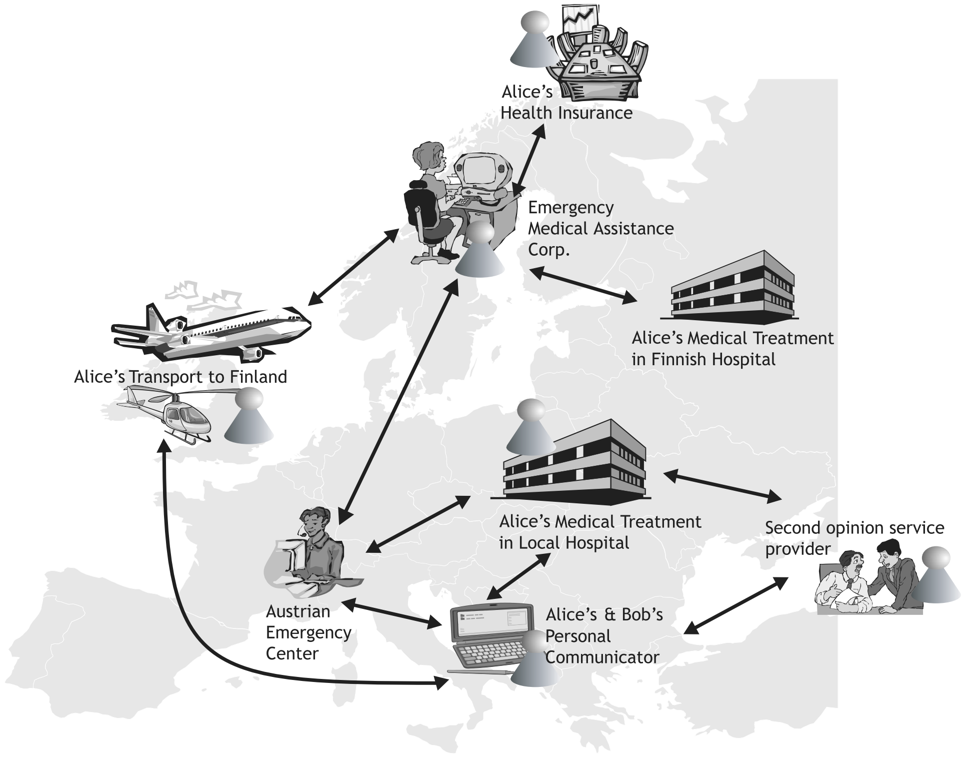
Alice's Health Insurance
Emergency Medical Assistance Corp.
Alice's Medical Treatment in Finnish Hospital
Alice's Transport to Finland
Alice's Medical Treatment in Local Hospital
Second opinion service provider
Austrian Emergency Center
Alice's & Bob's Personal Communicator

Figure 2 Emergency assistance application

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 realized in this form in practice and no software system is presently in widespread 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 the contact information of a local healthcare institution near them.[1] Additionally, the agent gives them the contact information of the Finnish representative of the Emergency Medical Assistance (EMA) service centre 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, or a phone number for a local taxi, or instructions for 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

examination by the local emergency physician it turns out that Alice had either 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. If the hospital had 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, the doctors decide that Alice should be transferred soon to a hospital with advanced cardiac life support to undertake thorough examination. 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, the EMA 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 to use 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 automatically contact 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 2 weeks of recovery she finally uses her PDA to send a 'thank you' to all the people involved with her medical case.

As can be seen in this scenario, people (patients) not only need medical treatment, but also need information as well as (sometimes) transportation assistance. Furthermore, assistance in the form of information is also required by the physicians, hospitals, and healthcare professionals involved. One straight implication of these complex requirements is the need for on-demand *initiation*, *composition*, *coordination*, and *supervision* of various activities represented mostly through non-human actors, like agents and services, but also through persons.

Service coordination layer

The process of service coordination is usually considered to encompass all activities that are devoted not only to the description but also to the discovery, composition, and execution of services. In subsequent sections, we introduce the basic technologies needed for the requirements derived from the healthcare application scenario.

One outstanding requirement of the scenario is its demand for coverage of large areas, i.e. it is supposed to spread over many different countries. As a consequence, the number of users (service requesters) and service providers is expected to range from many thousands to millions. Due to this fact, it is impossible to know all services, their functionality, and their distribution beforehand. As an initial step, this requires appropriate methods to discover and register services (either centralized or decentralized). In particular, this demands matchmaking – the selection of appropriate web services using semantic similarity. For instance, in case of emergency, the user needs to find a hospital or emergency centre near him or her. When several distinct but similar service providers exist which are all able to return this information, one has to be chosen. For this, quality-of-service criteria can be exploited (e.g. one service might be able to find emergency centres closer to the current location than another service). As a second step, the scenario requires service composition planning – the composition of several web services into processes. Typical usage cases within the scenario involve interaction with several service providers. For instance, transportation of a patient back to his or her home country may require interaction with different service providers to arrange the most suitable transportation. Finally, the service coordination layer must include execution, i.e. a runtime environment for compound services.

Service matchmaking

The service matchmaking functionality provides the means to compare the semantics specified for services, thus allowing the detection of semantically equivalent/similar services. Several approaches to the sophisticated semantic matchmaking of web services have been proposed that rely on ontology-based languages (e.g. OWL-S [2], WSMO, and Annotated WSDL [3]) and are grounded with formal semantics such as description logics [4]. 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, the OWLS-MX for hybrid matchmaking has been developed at DFKI. 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 is 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 determines pairwise not only the degree of logical match but also the syntactic similarity between the terminological expressions built by unfolding each of the considered query and service input (output) concepts in the local matchmaker ontology. In 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).

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 intend to use OWLS-Xplan [5]. 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 [6]. It combines a 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 than pure hierarchical task-reduction planning (HTN), and 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 of whether the given set of decomposition rules for HTN planning would allow building a plan that contains only atomic actions (services). 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 an HTN planner.

The Xplan system consists of the XML parsing module, a pre-processing module, the planning core, and the replanning module. The latter is used to readjust outdated plans during execution time (see later). 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 efficient lookup table for the actual search.

Service execution

The service execution system (SES) executes compound services 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 [7]. 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 an 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 [8]. 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. the 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 the principles of the OSIRIS (Open Service Infrastructure for Reliable and Integrated process Support) process management system [9]. Within OSIRIS, aspects of agent-oriented systems were introduced to fit into the CASCOM infrastructure. In particular, the execution system consists of one or 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 locally invokes services and which coordinates execution basically by forwarding control and data to the next agent(s). Furthermore, we distinguish between two types of service execution agents (SEAs): service provider agents and standard agents. The difference 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 services 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 direct successors of *s*. In addition, for failure handling purposes, information on the predecessors of *s* is also 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 the 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 few communication efforts between execution agents as possible.

Integration of service matchmaking, composition planning, and execution

As noted earlier, the SCPA acts as the client for SES. Consequently, our combined interaction and execution model consists of the following steps (see Figure 3 for the model and step numbers). 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 the 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 [10]. 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 a 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 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

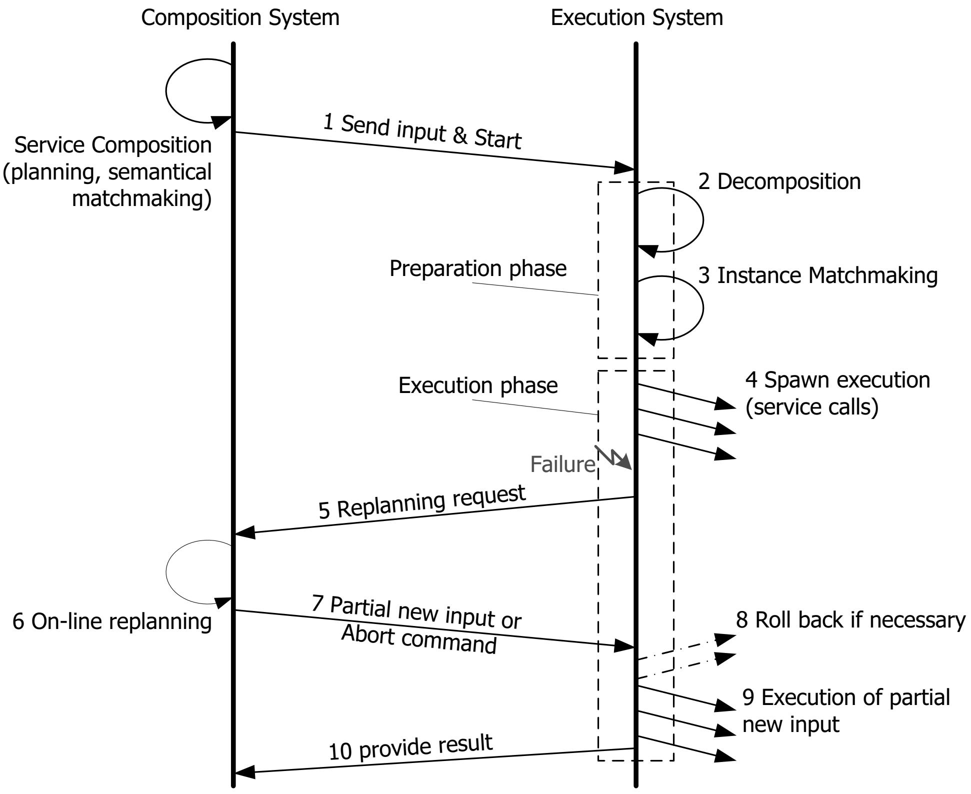

Figure 3 Interaction and execution model

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 would be undone or compensated) and some external logic would have to decide what to do next. In our approach, a crash failure situation does not necessarily end 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* online contingency *replanning* (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 alternative path (6) – usually by employing semantically similar services. If SCPA succeeds 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, SCPA sends an *abort* command to SES. Consequently, SES is then obligated to *roll back* 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 take place (i.e. update of the execution plan). Afterwards, *execution* continues (9). Replanning is also required if the original goal for which the process has been generated is altered. If the new process fragment requires the partial undoing of side effects because of its changes (i.e. utilization of other services), this will be done immediately before continuation. Finally, when execution has finished, the *result* will be sent back to SCPA (10). In order not to block resources endlessly during online replanning, we use a timeout-based approach: when there is no reply from SCPA to SES until the timeout (because SCPA has 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 replanning phases. By allowing indefinite replanning phases it is evident that execution theoretically might never terminate. On the other hand, and with the presented emergency assistance scenario in mind, the probability of high numbers of replanning 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 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 replanning 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 replanning phases should not be made without taking the target application into account.

In the scenario presented earlier, Alice and Bob are first required to state the goal of the process they wish 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 the case of failures or changes in the environment, planning is reinvoked and the process is changed (and executed) accordingly.

Related work

Similar to CASCOM, the ARTEMIS project (Semantic Web Service-Based P2P Infrastructure for the Interoperability of Medical Information) [11] aims at supporting healthcare applications by means of dedicated semantic web services. However, ARTEMIS focuses on 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 coordination 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 [12], OWLS/UDDI [13], MAtchMAker-Service [14], RACER [15], and HotBlu [16]. Other approaches are either process based (e.g.

High-Precision Service Retrieval Service [17]), peer based (e.g. Semantic Web Services P2P Discovery Service [18]) or hybrid (e.g. the Recursive Tree Matchmaker [19]). Alternative approaches include graph-based matching methods such as those presented in [20] and [16]. Furthermore, there are currently very few approaches and software tools available for OWL-S-based service composition planning, such as OWL-S Composition Planner using SHOP2 [21], logic-based DAML-S composition planning [22], the DAML-S workflow composer [23], and a Petri-net approach in which an OWL-S service description is automatically translated into Petri nets [24].

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 [25], ideas from process management, peer-to-peer networks, database technology, and Grid [26] 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 [27] are service providers acting as peers. Finally, Pleisch and Schiper [28] provide a general overview on fault-tolerant agent-based (process) execution.

Conclusions

In this paper, we have presented the CASCOM approach to providing access to functionality and data, encapsulated by semantic services, in a healthcare digital library. *Ad hoc* process-based applications 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 cooperation with TILAK, the umbrella organization of the state hospitals of the Austrian state Tyrol. In future work, we aim – among other things – to address more sophisticated transaction and failure models, especially by considering malicious failures (i.e. Byzantine failures as they might appear in untrusted environments).

Acknowledgements

This work is supported by the EU in the 6th Framework Programme within the STREP CASCOM (Context-Aware Business Application Service Coordination in Mobile Computing Environments), contract no. 511632. The work presented in this article has been done while Thorsten Möller and Heiko Schuldt have been affiliated with the Information & Software Engineering Group at the University for Health Sciences, Medical Information and Technology in Tyrol, Austria.

Notes

1 Their location is found either by using the GSM network cell identifier or by GPS.
2 This is important when there is more than one service instance available with equal signatures.

References

1 The CASCOM project. http://www.ist-cascom.org.
2 T.O.-S. Coalition. OWL-S 1.0 (Beta) Draft Release. Autonomous Agents and Multi-Agent Systems, 2003.
3 Patil A, Oundhakar S, Sheth A, Verma K. Meteor-s web service annotation framework. In *Proceedings of the World Wide Web Conference, 2004*.
4 Baader F, Nutt W. Basic description logics. Chapter in *The Description Logic Handbook: Theory, Implementation and Applications* 47–100. Cambridge: Cambridge University Press, 2003.
5 Klusch M, Gerber A, Schmidt M. Semantic web service composition planning with OWLS-Xplan. To appear in the First International Symposium on Agents and the Semantic Web, 2005.
6 Hoffmann J, Nebel B. The FF planning system: fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 2001; **14**; 253–302.
7 Schuldt H, Alonso G, Beeri C, Schek H-J. Atomicity and isolation for transactional processes. *ACM Transactions on Database Systems (TODS)* 2002; **27** (1); 63–116.
8 Schuldt H, Schek H-J, Alonso G. Transactional coordination agents for composite systems. In *Proceedings of the 3rd International Database Engineering and Applications Symposium (IDEAS 1999), August, Montréal* 321–31. IEEE Computer Society.
9 Schuler C, Weber R, Schuldt H, Schek H-J. Scalable peer-to-peer process management: the OSIRIS approach. In *Proceedings of the 2nd ICWS 2004, July, San Diego* 26–34. IEEE Computer Society.
10 Schuler C, Schuldt H, Türker C, Weber R, Schek H-J. Peer-to-peer execution of (transactional) processes. To appear in *International Journal of Cooperative Information Systems (IJCIS)* 2005.
11 The ARTEMIS project. http://www.srdc.metu.edu.tr/webpage/projects/artemis.
12 Sycara K, Widoff S, Klusch M, Lu J. *LARKS: Dynamic Matchmaking among Heterogeneous Software Agents in Cyberspace*. Boston: Kluwer, 2002.
13 Paolucci M, Kawamura T, Payne T R, Sycara K P. Semantic matching of web services capabilities. In *Proceedings of the First International Semantic Web Conference on the Semantic Web* 333–47. Springer, 2002.
14 Colucci S, Noia T D, Sciascio E D, Donini F, Mongiello M, Piscitelli G, Rossi G. An agency for semantic-based automatic discovery of web-services. In *Artificial Intelligence Applications and Innovations: Proceedings of IFIPWCC-04* 315–28. Boston: Kluwer, 2004.
15 Li L, Horrocks I. A software framework for matchmaking based on semantic web technology. In *Proceedings of the 12th International Conference on the World Wide Web* 331–9. ACM Press, 2003.
16 Constantinescu I, Faltings B. Efficient matchmaking and directory services. In IEEE/WIC International Conference on Web Intelligence, 2003.
17 Klein M, Bernstein A. Towards high-precision service retrieval. *IEEE Internet Computing* 2004; **8** (1); 30–6.
18 Banaei-Kashani F, Chen C, Shahabi C. WSPDS: Web Services Peer-to-Peer Discovery Service. In ISWS 2004.
19 Bansal S, Vidal J. Matchmaking of web services based on the DAML-S service model. In AAMAS2003, Melbourne.
20 Trastour D, Bartolini C, Gonzalez-Castillo J. A semantic web approach to service description for matchmaking of services. In *Proceedings of SWWS 2001*.
21 Wu D, Parsia B, Sirin J H E, Nau D. Automating DAML-S web services composition using SHOP2. In *Proceedings of 2nd ISWC2003, Sanibel Island, FL* 20–3.
22 Sheshagiri M, desJardins M, Finin T. A planner for composing services described in DAML-S. In *Proceedings of AAMAS 2003 Workshop on Web Services and Agent-Based Engineering*.
23 Tarkoma S, Laukkanen M. Adaptive agent-based service composition for wireless terminals. In Klusch M et al. eds *Proceedings of CIA VII, August 2003, Helsinki* 16–29. Berlin: Springer, 2003.
24 Hamadi R, Benatallah B. A Petri-net-based model for web service composition. In *Proceedings of the 14th Australasian Database Conference: Database Technologies* 191–200. ACM Press, 2003.
25 Schek H-J, Schuldt H, Schuler C, Weber R. Infrastructure for information spaces. In *Advances in Databases and Information Systems: Proceedings of the 6th East European Symposium, ADBIS 2002, September, Bratislava* 23–36.
26 Foster I, Kesselman C eds. *The Grid: Blueprint for a New Computing Infrastructure* 2nd edn. San Francisco: Morgan Kaufmann, 2004.

27 Dogac A, Tambag Y, Tumer A, Ezbiderli M, Tatbul N, Hamali N, Icdem C, Beeri C. A workflow system through cooperating agents for control and document flow over the internet. In *Proceedings of the 7th International Conference on Cooperative Information Systems (CoopIS 2000), September 2000, Eilat* 138–43.
28 Pleisch S, Schiper A. Approaches to fault-tolerant and transactional mobile agent execution: an algorithmic view. *ACM Computing Surveys* 2004; **36** (3); 219–62.

Correspondence to: Thorsten Möller

Thorsten Möller
University of Basel
Department of Computer Science
Database and Information Systems Group
Bernoullistraße 16, 4056 Basel,
Switzerland
E-mail: thorsten.moeller@unibas.ch

Heiko Schuldt
University of Basel
Switzerland
E-mail: heiko.schuldt@unibas.ch

Andreas Gerber
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
E-mail: gerber@dfki.de

Matthias Klusch
DFKI, Germany
E-mail: klusch@dfki.de