The inContext Pervasive Collaboration Services Architecture*

Stephan Reiff-Marganiec[1], Hong-Linh Truong[2], Giovanni Casella[3], Christoph Dorn[2], Schahram Dustdar[2], and Sarit Moretzky[4]

[1] Department of Computer Science, University of Leicester, UK, srm13@le.ac.uk
[2] Distributed Systems Group, Vienna University of Technology, Austria, {truong,dorn,dustdar@infosys.tuwien.ac.at}
[3] Softeco Sismat SpA, Italy, giovanni.casella@softeco.it
[4] Innovation Lab,Comverse, Israel, Sarit.Moretzky@comverse.com

Abstract. Traditional collaborative work environments are often proprietary systems. However, the demands of todays e-worker are such that they use their own tools and services and collaborate across company boundaries making highly integrated solutions less feasible. Service oriented computing provides an obvious solution here, in providing mechanisms to loosely integrate many tools and services. In this paper, we present the inContext PCSA (Pervasive Collaboration Services Architecture), which is a reference architecture for building context aware collaborative systems that are based on service oriented techniques.

1 Introduction

Collaborative systems are tools supporting collaborative work, typical examples are document management systems or customer information systems where different staff of the same organisation can access information and contribute to information in order to jointly bring forward the aim of the organistion. Many of the existing collaborative systems are not integrated with each other, so for example workflow and document management are not connected, or the communications systems are entirely separate from the other two. This means that information either needs to be transferred manually (e.g. logging call activities in a workflow system), or is simply not available where and when it is required. Clearly this calls for an infrastructure that allows for integration of the different activities. The other major disadvantage is that systems are usually used within a single organization, while nowadays collaborative work often spans institutional boundaries calling for platforms that operate across these boundaries. A further disadvantage of existing systems is that they are not context aware, that is the user's context is not automatically available to support the given activities.

Work in the past has addressed some of the above aspects, in particular the issue about context. However, it has not done so by considering a single platform addressing all needs – which partly might have been due to limitations with the available middleware and underlying systems. In the inContext project

* This work is supported by inContext (Interaction and Context Based Technologies for Collaborative Teams) project: IST IST-2006-034718

we have developed a platform, or reference architecture called inContext PCSA (Pervasive Collaboration Services Architecture) which provides a context aware setting in which independent collaborative services, exposed as web services, can be used to build different collaboration tools. The platform provides a novel middleware layer that allows selection and composition of services at runtime to fulfil a users current needs. It is pervasive in both the sense that the user does not need to be aware of how and which services are selected, they simply achieve the desired aim by using systems build on top of the platform and thath services can run on different types of devices or platforms. To achieve this service selection, and in particular to identify the most suitable service for a user in their current context the platform is context aware. Context here covers not only location and devices, but also activities creating the all-essential link for integrating collaborative artifacts and information.

Overview. This paper is structured as follows: the next section provides a motivating example and background information on context aware and collaborative systems. We then turn our attention to the PSCA by providing an overview and discussing the essential subsystems. Before drawing concluding remarks and discussing further work, we present examples of the use of the architecture.

2 Motivation and Background

In terms of motivation, let us consider a scenario that is typical in the area of collaborative work, and which calls for many of the PCSA features presented in this paper. Imagine a user in an internal project A in his company requires a document relevant for his current activity. The PCSA will utilize document search service dsA, through which the documents for the project can be accessed. Later the same user switches to project B – a joint venture between his own company and an external association. Again the user request a document search service, this time the platform will use dsB, a public repository service with access by subscribers. We can see that in this simple example based on the user context (mostly his activity in this case) differing services are selected as most appropriate and are consequently invoked in a transparent manner to achieve the user's aim. Behind the scenes there is much technical activity and usual examples are much larger: the document search might be part of a meeting planning tool or a governmental policy review systems.

A vast number of tools supporting collaborative work is available today, as indicated in [1–3]. However, many of these tools are used within a single organization's boundaries, while many emerging collaborative work scenarios are spanning these boundaries. When we consider the dynamic and distributed collaboration spanning different organizations and the support of user participation/customization required we find that existing collaborative tools are simply insufficient. There is a number of requirements that emerging collaborative tools need to support. First, there is a need to utilize different collaboration tools which can be provided and hosted by different organizations. Second, there is a need to adapt collaboration tools to the context of the collaboration as we have seen in the motivating example earlier on. Existing collaboration tools provide rich features which work very well when utilised in isolation, for example,

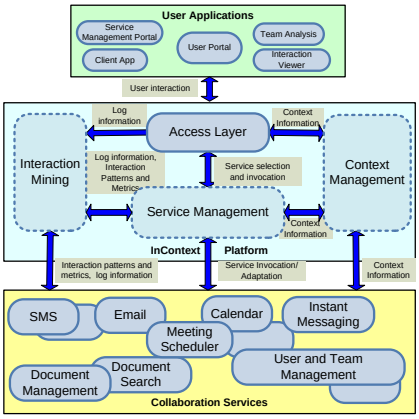

Fig. 1. An overview of the inContext system [4]

for sharing documents or for managing activities. However, those features are not easily integrated into a single collaboration toolset, where various features are required to accomplish the collaboration for various reasons, such as they are propriety collaboration tools, they provide no open interface, or they are tightly-coupled systems.

Dynamic user-defined and user-customized collaboration calls for well-defined, common collaboration services which can be easily composed and adapted for different collaboration contexts. We have found that commodity of CWE services is in increasing use and open standards are widely employed for collaboration tools [3]. However, there are many open questions that need addressing: How can we support diverse collaboration services that can be composed and customized? How can we enable context-awareness and collaboration adaptation across organizational boundaries? How can we support the user to perform collaboration from anywhere with any device? These questions motivated us to investigate the concept of commodity collaboration services which are composable and common. We approach this concept by utilizing Web services technologies and activity-based collaboration techniques to develop a so-called pervasive collaboration services architecture (PCSA).

3 Overview of the PCSA

The inContext Architecture, also referred to as PCSA (Pervasive Collaboration Services Architecture), consists of three main parts: the user applications, the collaboration services and the inContext core platform [4] - we consider these in turn. To provide an overview, the overall architecture is depicted in Figure 3.

User applications have not been at the forefront of the development here, but our case studies have been supported with respective frontends. Applications at this level are meant to be used by the collaborative worker, so some effort has been spent on considering user applications technologies for both mobile and stationary workers.

Underpinning the environemnt are collaborative services: essentially any service (implemented as a web service) or any software whose interfaces have been

exposed as web services that have a meaning for collaborative work. Services at this level are, for example, a Document Service (allowing to retrieve and collate documents that are relevant to the current user activity), an Activity Service (concerned with any information regarding activities), or an User and Team Management Service (providing all sorts of team related information).

The core platform contains four main ingredients: the Access Layer, the Context Management, the Service Management and the Interaction Mining. Each of these is developed as services, providing much flexibility to the platform.

The Access Layer controls access to the platform by checking user credentials – it is the main entry point to using the inContext platform. However, its functionality goes beyond authentication: any transaction across the Access Layer is tracked to gather information on user activities (which in turn enrich the context information) and user decisions (e.g. when a number of services was available it is monitored which was actually selected by the user).

The Context Management sub-system is concerned with gathering, storing, providing and enriching context information, but also provides mechanisms to retrieve context from the store. Furthermore, reasoning techniques allow deriving new, richer context information from the information available in the store. The contex model implementation is based upon RDF triples and the reasoning is based on an enhanced version of the Jena engine. Interaction Mining provides additional information by considering past behavior. Context management and interaction mining are beyond the scope of this paper[1].

The Service Management constitutes the part of the inContext platform most relevant for this paper, as it is here were the collaboration services are drawn together and composed into rich context aware collaboration tools on demand. Service Management is concerned with registration and lookup of services and their execution – this goes beyond standard web service functionality by identifying the best suitable choice for the users current context and activity based on non-functional attributes of services.

4 Common SOA-based Collaboration Services

The inContext PCSA is based on SOA to allow for collaboration services to be dynamically located and invoked. The platform supports flexible and dynamic collaborative working environments able to aggregate heterogeneous services while at the same time considering and exploiting the user's context.

Common collaboration services can be developed and provided by different organizations, following well-defined interfaces to support fundamental tasks typically required by collaboration tools. Based on the inContext's case studies, we have analyzed requirements for collaboration platforms and identified various common collaboration services and have as part of the platform provided a set of such services. Although, these services were demanded by the inContext case studies, they are not specific to these. The services are more general and can be used in a wide variety of emerging collaborative work scenarios as well

[1] inContext D2.2 and D2.3 discuss these in detail; both are available at www.incontext.eu

Collaboration Services	Description
`User and Team Management Service`	provides a list of projects related to a user with detailed information on project members, timeline and team structure.
`Member Search Service`	searches for relevant people based on specific role or expertise.
`Personal Document Search Service`	allows searching for text patterns inside a set of documents stored on specific hosts which were declared as shared for the inContext platform.
`Document Service`	manages virtual 'shared areas' for documents.
`Short Message Service`	enables to send SMS to mobile users.
`Meeting Scheduling Service`	is a composed service allowing setting up a meeting.
`Activity Service`	manages the activities associated to a project, enabling the creation and the organization of such activities in an activity tree and assigning them to users, documents, locations, etc.

Table 1. Examples of common collaboration services

as be served as basic services for building different collaboration tools. Table 1 presents a selection to show the flavour of typical collaboration services.
Common collaboration services can be atomic or composite – as is typical for web services. By utilizing well-developed publish-registry techniques in Web services, common collaboration service will be registered in a repository named Service Registry. However, when registering a service some extra information is required: All services (actually each operation) are organized in categories, within the Service Registry. Each registered service belongs to at least one category. The category is important, as it is this which is used for lookup; each category also has associated non-functional attributes and a well defined generic service interface. Then, common collaboration services can be used to build collaboration tools which could discover and execute the services based on collaboration context.

5 Context-aware Service Management

The context-aware service management identifies the most appropriate services based on the user's context and current needs and composes these into a collaborative tool-set. There are three aspects here that go beyond what standard service oriented techniques offer: (1) we have addressed the need to select the most appropriate service automatically, (2) we have addressed the requirements for selecting services as part of a worklfow and (3) we have provided techniques to invoke services through a standard interface by automatically mapping specific service interfaces to more generic interfaces that are exposed to the user application. We will consider these 3 aspects in the next few sections.

Selecting Services. One of the challenging aspects in service oriented computing is selecting the best service if a number of services are on offer. We devised a ranking mechanism called the RelevanceEngine which ties in with service lookup in the service management subsystem. The RelevanceEngine has one job: to consider a list of suitable services that all seem to functionally address the user's requirements and rank these so that the user can see which service is most appropriate for supporting their activity in their current situation.

The ranking mechanism makes use of a number of inputs: it queries the context management system to obtain information about the user's context; it queries the data mining component to gain an insight into historical handling of a similar situation and it of course explores the services profile – the extra meta data in the registry that is associated with the service category. For example, if the user is looking for a printing service, the meta data will tell us static information such as whether the service is colour or not and how fast the printer is; it will also provide a query URL to find out about the current service use (e.g. queue length). The user's context will amongst others provide insight into whether he requires the printout very quickly and whether the document is very long. All these facts are combined and a rank value is calculated for each service by using an automated, type based version of LSP (Logic Scoring for Preferences). This paper provides a wider overview of the architecture and a very detailed discussion of the ranking mechanism is beyond the scope, however this has extensively been described in [5]. Briefly, the LSP mechanism can handle large numbers of criteria while maintaining an assurance that even factors with a small weight but which a user cares strongly about are given appropriate weighting in the resulting score. It can also act as both "ranker" and "filter" – that is we consider hard and soft criteria using the same mechanism and ensure that services which do not meet hard criteria are shown as inappropriate in the ranking (essentially a score of 0): for example a hard criteria might be "the service must be free", while the related soft criteria would be "the service should cost as little as possible".

It is worthwhile pointing out that we are using a pragmatic extension to service models in a standard UDDI repository: we have developed a model for capturing key non-functional data about services in this way. This mechanism is lightweight and requires little technologythat goes beyond standard web services; in particular it only requires for a service developer to register a few extra values in the repository. Of course one could consider semantic web technologies here, which provide mechanisms to express properties but these are a more fundamental shiftin the technology used and hence we decided against them. Independent of this, the data about services is an input to the ranking mechanisms and the same would be still appropriate in the context of semantic web services with richer service descriptions.

The Composition Context. The mechanism just described was initially developed to find the most appropriate service for a given activity, considering the user's context but not necessarily the context of execution of the service. However, usually services are not required in isolation but are often invoked as part of a workflow. The ranking mechanism has been extended by what is called composition context, essentially information gathered about the last stages in the workflow that we have executed: did services from a certain provider fail? Are there policies that disallow us to select (or would mean preferential treatment if we chose) a future service from a specific provider? The concept and related ranking mechanism have been described in [6], but the idea is probably best explained with a small example: consider ordering a book. You have a choice of two providers, provider A charges €10 for the book, provider B €13. If we select the most optimal single service, we would select provider A. But in our

workflow context, we know that the book also has to be shipped to us and find that provider A charges €5 for shipping, while provider B offers it for free, thus overall provider B is the better choice. This example is simple but it is only meant to show that local optima differ from global ones; further one could argue that some websites currently already provide such functionality: they usually do so by considering services offered by the same provider. The composition context explored here is not bound to just relating information by the same provider, but can be applied to services from different providers.

Mapping Interfaces. Considering that we retrieve services from all sorts of providers at runtime, we must ensure that they can be invoked by the platform in a coherrent way. In order to achieve this, we have assigned to each category a common interface and each service in that category has to provide mapping information on how to map the service interface to the common interface. This way we enable transparent execution of various services with different interfaces for a specific category.

To support this mechanism of service interface mapping, we implemented a service called Interface Mediator. This service forwards web service calls after applying a transformation in order to match the destination web service interface. As already mentioned, if a service is registered under a certain category the consumer should use the common interface to use service of this category. During runtime the interface mediator relies on XSLT style sheets to map the common interface to a service interface. This allows the service providers to perform some complex data manipulation to match their requirements. In order to expose the PCSA capabilities, we created some XSLT templates that can be directly used to (1) query and use context data, (2) gather user preferences / information or (3) Query any service metadata.
While offering high flexibility and enabling scenarios like translating content based on user's language, sending an SMS to the relevant phone number, or converting data to the right format (e.g. currency units, or temperature scales), this transformation implies of course an overhead compared to a direct call. Its impact on the performance is very limited for several reasons.

- The style sheet is compiled to machine code allowing for very fast execution.
- The data is manipulated at the XML level without being marshaled to any programming model which removes expensive conversions.
- The external service calls go through the Access Layer anyway in order to be logged and to provide feedback to the system. Integrating the interface mediator there implies no network overhead for the transformation itself.
- XSLT is standard technology with very fast engines emerging.

The overheads are outweighed by the benefits of being able to dynamically select a service based on the current situation. This mechanism makes the platform more robust (being able to use another service if one fails), but also increases its adaptability by offering to add/ remove services to existing categories.

6 Context-aware Collaboration Services

In typical service composition one considers functional and QoS parameters. Collaborative work scenarios require in addition for context to be considered as

a main criteria. In this section we address two issues: context-aware composition and adaptation support for collaboration services.

Context-aware Composition The composition of collaboration services is based on collaboration context. By utilizing collaboration services, collaboration tools can be built. In our PCSA, a collaboration is described by collaboration activities which are performed by a set of users. Consequently, a collaboration context will be determined when the activities are specified and the context will be updated by the user or by the services which monitor actions within activities.

A tool supporting the end user to collaborate can utilize collaboration services, thus it has to manage compositions for collaborations. During the collaboration, the user will specify activities which include information about who is involved in them, which artifacts are needed, the type of collaboration services used, etc. Specified activities are managed by the Activity Service. All context information related to activities are managed, for example, the location of involved people and the status of services being used for the activities. When collaboration services are required, the most appropriate services will be determined and composed based on the current collaboration context; we discussed the technical mechanism for service ranking in the previous section.

Supporting Adaptation based on Collaboration Context Collaboration services are deployed as web services. These services must be aware of changes in the collaboration context, and therefore they need access to the current collaboration context. The PCSA supports two types of context-based adaptation: (1) service-level adaptation focuses on improving the behavior of the invoked service depending on provided context information and (2) composition-level adaptation aims at selecting and combining the most suitable set of services to fulfil the user's requirements in the given situation.

To this end, we provide a generic solution for distributed collaboration context sharing. The sharing mechanism remains context model agnostic. While the actual context information is managed by the context management framework, the context sharing mechanism is responsible for providing correlation information. This correlation information acts as the initial context entry point, thereby allowing a service to retrieve the relevant information from the context store. To remain independent of service interface and Web service stack, we insert the correlation information in the header part of a SOAP message. Whenever a service operation is invoked, the message header includes the URI of the invoking user and the user's current activity. This provides sufficient correlation for a service to obtain the context information and adapt its operation, if needed.

One of the main advantages of using Web services technologies for common collaboration services is the ability to loosely couple context information: (1) services do not need to explicitly pass along large sets of context information of which each individual service requires potentially only a small subset; (2) services need to understand only that part of the overall context model that they require; and (3) extensions for domain specific collaboration services (e.g., health care) can place additional correlation information in the SOAP header without having to update existing services.

Utilizing the SOAP header extension for context correlation yields another benefit to simplify cross-organizational collaboration. SOAP intermediaries such as the Access Layer but also message routers, security checkpoints, and governance mechanism for SLAs in general can access the header information and subsequently base their decision on the available context instead of inflexible policies and rules. Thus, adaptation at the service composition level becomes increasingly feasible and manageable. For example, consider the following adaptation supported by the PCSA: (a) The Access Layer forwards a notification request to the most suitable communication service based on user preferences, costs, and available devices. (b) The service interface mediator is able to extract missing data from the context to invoke a service demanding for additional input parameters not specified by the generic service interface. (c) Once a shared document space is selected for an activity, all subsequent service calls are forwarded to the same service endpoint.

7 Experimental Evaluation

In this section, we discuss how we utilize PCSA to build different collaboration tools. We present our experiences based on the development of two real case studies proposed by our end-user partners:

Scheduling a meeting: The Electrolux Group is one of the world's largest manufacturers of white goods. Within the company secretaries must often organize meetings which is a difficult task. In fact, managers are often unavailable to participate to meetings due to the multiple activities in which they are involved, in which case the secretary can select his/her project proxy which requires an understanding of the project team structure. Moreover it is often difficult to communicate with them (e.g. to establish a meeting date) due to their travels (they are frequently in different time zones, are not always able to access the web or to answer phone calls).

Wolverhampton Fair: Every year the West Midlands Local Government Association organizes and manages the Wolverhampton fair. In particular a manager must build the staff to manage the fair (the workers belong to different departments and have different skills and experiences), must assign the activities related to the fair and must check that during the fair coordination and communicating between all staff is as expected, which requires tools to exchange documents and to communicate quickly.

The two usage scenarios are addressing very specialised needs which are very different in their requirements. We were able to exploit the PCSA successfully to build two user applications to handle these scenarios. The two user tools exploit a subset of common collaborative services, which are composed and adapted in different ways to satisfy the user needs. Figure 2 shows the GUI of the Event Management Tool as an example. The Event Management Tool offers a set of tools to organize events but is also applicable to manage general projects. The Meeting Scheduling Tool addresses the needs of the Electrolux case study.
The following shows some collaboration services used for each of the two tools, where SM and EM are used for the *Schedule a Meeting Tool* and the *Event Management Tool* respectively; the last three servcies are composite.

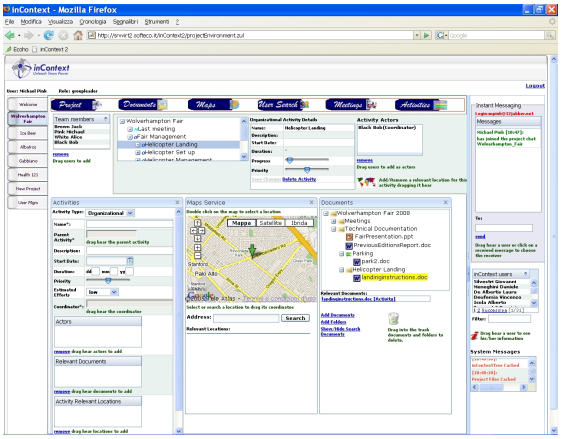

Fig. 2. The Event Management Tool

User & Team Management Service is used to retrieve participants details and project team structures (SM) and to maintain the event stuff structure and to retrieve user details, skills, experiences, etc. (EM)

Activity Service is used to explore the project activity tree in which the meeting is created (SM) and to create the activities that must be handled in the event and to assign such activities to the staff members (EM).

Document Service is used to retrieve information about relevant documents for a meeting and to store the meeting agenda (SM) and to share and organize documents (EM).

The Relevant Documents Finder is a composition of the activity service and the document service. Based on relations between the current user activity (e.g. the meeting that is scheduled and a general activity) and others project activities it retrieves the documents associated to these activities. Some reasoning rules are applied to identify the relevant documents.

The Relevant User Finder is a composition of of the activity service and the User & Team management service. Based on the relations between the current user activity and others project activities it provides information about users involved in these activities. Again, some reasoning rules help to ensure that all appropriate users are identified.

The Notification Service is a composition of the Instant Messaging Service, the SMS and the Mail Service. By exploiting the user context this service decides which is the best way to notify a user about something (e.g. an important message informs that he must attend a meeting) and sends the message using the best service selected.

Collabration services become sometimes unavailable (this is unavoidable in a distributed, dezentralised environment). In these cases the PCSA lookup mechanism enables selection of an alternative service in a manner transparent to the user by using context and the interface mediator.

Our experimental evaluation shows that the PCSA enables the exploitation of a set of collaboration services to build heterogeneous collaboration tools addressing different requirements and functionalities. Moreover, by using the ser-

vice common interfaces composition of simple services to offer new complex services which are able to satisfy the user needs is possible. Finally the dynamic management of services enables integration of different services with the same functionalities and to exploit them according to their availability.

While we considered two specific cases here – both taken from the domain of collaborative work – these are only meant to illustrate the ideas. The platform developed addresses the need of collaborative systems which we have analysed and briefly introduced at the start of the paper and hence allows for the dynamic building of tools for collaborative work environments which tend to require flexible system structures where the system takes much of the burden of providing the right service at the right time based on the users activity. Many of the developed mechanisms can be applied outside collaborative systems, for example the approach for service selection has not been developed with only collaboration services in mind, but rather with a wider view of service slection.

8 Related Work

Many basic collaboration tools and services, such as document sharing, calendars, and instant messaging have been developed. However, currently it is not easy to compose these services and make them interoperable for Web-based, user-customized collaborations because most of them lack well-defined Web services interfaces. Web services support have been incorporated into few collaboration services such as BCWS, Google Doc, and Microsoft Sharepoint for document sharing. In our work, we not only propose solutions for the interoperability of collaboration services but also present how common collaboration services can be composed suitable for different collaborations based on context.

Recently, various researchers have advocated the standardization of (basic) collaboration services. A CoCoS Working Draft[2] proposes common collaboration services in terms of message representation, service operations and service behaviors. CoCoS addresses a subset of common collaboration services proposed in our PCSA but does not discuss the composition and execution aspects of collaboration services. The ECOSPACE project[3] also investigates various common collaboration services. However, it focuses on document sharing services. The OCA-WG (Open Collaborative Architecture Working Group[4]) aims at defining a reference architecture for collaboration services.

In the area of service selection [7] propose the addition of a broker component to the service selection architecture which essentially sits between the UDDI repository and the invoker and monitors service invocations and the resulting quality. However they, like most other approaches (e.g. [8]), only consider service quality as criteria for service ranking. We have signifcantly added to this with the more complex context model that is used in our work. Also, we have extended the Data model in the repository itself to get richer service information.

[2] http://www.ubicollab.net/images/stories/UbiCollab/Standards/ CoreCollaborationServices_v0.1.pdf

[3] http://www.ip-ecospace. org

[4] http://www.oca-wg.org

9 Conclusion and Further Work

The lack of common collaboration services and an open architecture for collaborative working environments hinders the integration and reusability of diverse collaboration tools. We have presented the inContext PCSA – a reference architecture for context aware collaborative systems that defines and provides an open platform with various common collaboration services. The PCSA allows to combine collaboration services into larger platforms that fulfil the needs of an organisation or user. What should be noted is that the combination is loose, in the sense that the actual service for a specific task is only selected at runtime based on the users activites and current needs.

What constitutes a collaboration service is open: in principle any service could be used as part of a collaboration and can be easily introduced to the platform by the creator of the service registering the same and providing some meta data. The PCSA itself is also built from services and hence can be used in its entirety, or selected components can be used within other contexts.

The resulting technical platform has been used to implement two quite diverse toolsets for two real case studies and initial test results have been discussed. Currently we are running extensive end-user tests by exposing the toolsets to larger audiences. Another line of future work is improvements at the level of individual system components, such as further refinement of the ranking mechanisms or enhancement of service profiles.

References

1. O'Leary, D.E.: Wikis: From each according to his knowledge. Computer **41**(2) (2008) 34–41
2. Optaros, Inc.: Unleashing the power of open source in document management. Optaros Whitepaper (2006) http://www.optaros.com/system/files/optaros_wp_os_crm_20060316%282%29.pdf.
3. Skopik, F., Truong, H.L., Dustdar, S.: Current and Future Technologies for Collaborative Working Environments (2008) ESA Report, https://www.vitalab.tuwien.ac.at/autocompwiki/index.php/Main_Page.
4. Truong, H.L., Dustdar, S., Baggio, D., Corlosquet, S., Dorn, C., Giuliani, G., Gombotz, R., Hong, Y., Kendal, P., Melchiorre, C., Moretzky, S., Peray, S., Polleres, A., Reiff-Marganiec, S., Schall, D., Stringa, S., Tilly, M., Yu, H.: incontext: A pervasive and collaborative working environment for emerging team forms. In: SAINT, IEEE Computer Society (2008) 118–125
5. Reiff-Marganiec, S., Yu, H.Q., Tilly, M.: Service selection based on non-functional properties. In: NFPSLASOC 2007. LNCS (2007) (in press)
6. Yu, H., Reiff-Marganiec, S., Tilly, M.: Composition context for web services selection. In: ICWS 2008. (2008) (in press)
7. Al-Masri, E., Mahmoud, Q.: Discovering the best web service. In: Proceedings of the 16th international conference on World Wide Web, ACM (2007) 1257 – 1258
8. Seo, Y., Jeong, H., Song, Y.: A study on web services selection method based on the negotiation through quality broker: A maut-based approach. In: Proceedings of International Conference on Embedded Software and Systems. (2004) 65 – 73