Automated Context-aware Service Selection for Collaborative Systems

Hong Qing Yu and Stephan Reiff-Marganiec

University of Leicester, Department of Computer Science, Leicester, UK
{hqy1,srm13}@le.ac.uk

Abstract. Service-Oriented Architecture (SOA) can provide a paradigm for constructing context-aware collaboration systems. Particularly, the promise of inexpensive context-aware collaboration devices and context-awareness for supporting the selection of suitable services at run-time have provoked growing adoption of SOA in collaborative systems. In this paper, we introduce an approach for selecting the most suitable service within a SOA based collaboration system, where suitability depends on the user's context. The approach includes context modelling, generation of context-aware selection criteria and a suitable service selection methodology.

Key words: Context-awareness, SOA, Service Selection, Collaborative Systems

1 Introduction

Collaboration systems are becoming more and more important for facilitating team work and e-activities (e.g. e-business, e-government or e-education). In particular, context-aware and dynamically configured collaboration systems are demanded in order to support collaboration activities, which are progressively flexible due to changes in modern work environments. While context-aware features can not be found in some legacy collaboration systems [Dus04], these systems are static in their architecture and hence the functionality that they offer. With widespread deployment of inexpensive context-aware devices and the innovations brought by the SOA paradigm, we get an opportunity to reconstruct collaboration architectures with context-awareness and dynamic configuration at its centre. For example, the inContext project[1] has defined a platform, called PCSA (Pervasive Collaboration Services Architecture) [RMTC+09] to support context-aware collaboration services by working closely with industry users and studying their collaboration needs. The PCSA mainly includes three subsystems:

1. The Access subsystem controls the user and service registration and system access.

[1] www.in-context.eu

2. The service management subsystem, most relevant to this work, is in charge of maintaining the service repository which includes a categorisation of services and details of NFPs (non-functional properties) of the registered services. It also provides functionality to look up services and to obtain service suggestions based on suitability.
3. The Context management subsystem maintains context data of registered users, based on a specified context model.

One of the major challenges of the PCSA is to select collaboration services based on user context information by matching services according to their non-functional properties – clearly the decision is made in the service management subsystem, but is based on the data obtained by the context subsystem.

Before we consider the challenges in more detail and provide an overview of the results of this paper, we consider two motivating examples, which highlight the challenges that modern collaborative systems need to adapt to.

> Organising an emergency meeting is a typical e-business and e-government collaboration activity. Notifying all participants to attend the meeting is an important and difficult task because different participants may be in different context situations including different in locations and time zones, they are available on different devices, have diverse contact preferences/rules. For example, a participant may be on holidays in a foreign country and only has a mobile phone with him, or the participant has switched off his mobile phone to save power but is online using IM (Instant Messenger).

In this scenario, the service selection challenge becomes to select the most suitable one. This decision has to be based on dynamically obtain user' context information, with the user being passive in that they cannot be asked upfront which service is most suitable for them.

In contrast to the first scenario, the second case study is concerned with selecting a medical support service during a park fair:

> It is expected that a large number of people joins the fair and incidents are expected. Two medical tents are prepared in different locations for providing aid. Efficient collaboration between fair assistants and tents is essential. One tent (Service 1) has more staff and is meant to cope with minor injury cases. The other tent (Service 2) has fewer staff, but more advanced equipment to deal with severe incidents as well as minor injuries. One assistant team scours the park to locate incidents and reports to the most suitable medical support service based on injury level, location of the incident, availability of the tents and response times. For example, if the injury has been reported close to Service 2, but it is not severe then which medical support service should receive the report from the assistant?

Successfully supporting such a collaboration system, requires the service selection method to recognize both users' (member of assistant team) current context and the services' current NFPs.

These two examples highlight a number of challenges to be addressed in context-aware collaboration service selection.

1. The users' runtime context information needs to be dynamically gathered and aggregated in a structured form.
2. The context information needs to be an input to the service selection approach, requiring a link between user context and the relevant non-functional properties of services.
3. The service selection method needs to be automatic.

In this paper, we are going to illustrate our novel contributions to address these challenges. Specifically we are presenting the following:

1. An OWL/RDF based user context model, with a link connecting to the service NFPs.
2. A method for dynamically generating context-aware service selection criteria based on the service category.
3. A TLE (Type-based LSP Extension) service selection method using the context-aware criteria.

The reminder of this paper is organised as follows. In section 2, we present the context modelling techniques and the details of the model. In section 3, we discuss the connection of the context model to the service's non-functional requirements. In section 4, we explain the TLE method, thus providing a solution to the service ranking issue. Implementation and evaluation results are presented in section 5. We then discuss some current related work and finally draw conclusions.

2 User Context Modelling

To allow for selecting the most suitable service for a user, the user's context needs to be evaluated, which is only realistically feasible if it is well defined and organized. In addition, the context information might be distributed and must be easily retrievable. Based on these requirements, we use OWL [OWL] to model user context information and RDF [Gro04] to store context data. By analysing the motivating scenarios and context information in general, our context model has been divided into 4 packages. A simplified top level OWL context model is shown in Figure 1.

User profile context (Profile) stores a user's personal data. The profile links to other context properties: Language shows which languages this person knows and their proficiency level; ContactInfo connects to possible contact details of a particular person such as postal address, contact number or

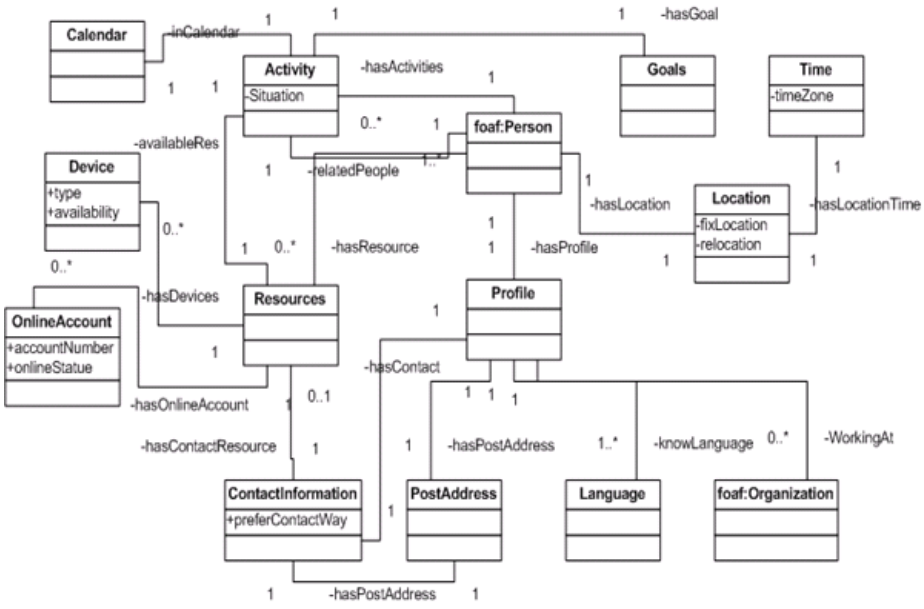

Fig. 1. The integration of 4 packages context model

online contact details. For service selection, Language can be used to filter services that are usable by the user; ContactInfo might indicate which way of contact is preferred by the person. Person details make use of the FOAF ontology [2].

Resource context (Resources) includes both electronic documents and artifacts, as well as physical resource such as devices available. The resource context allows determination of which devices, software and documents are available.

Activity context (Activity) describes everything a person is doing (maybe performed by a service) in order to fulfil a goal. The Goal property uses ontology-based keywords to describe the task and its desired outcomes. It also allows for special variants for e.g. emergency situations. These properties imply the functional requirements of the service. The Situation property influences the importance of different non-functional properties. For example, emergency situation change the weights applied to certain criteria.

The physical location context (Location) is the detailed ontology for Location property. It indicates the location and time related constraints. A fixed location may have different representations such as GPS Coordinate or PostalAddress; [FipD07] provides more detail.

[2] http://xmlns.com/foaf/0.1/

3 Context-aware criteria generation

3.1 Services categories with meta data

Services are traditionally categorised by their functional properties, e.g. in the categorisation system used in UDDI [Org04]. This kind of service categorisation is insufficient for automatic service selection processes because it does not specify NPFs that are essential to differentiate functionally similar services in different situations. We propose to extend the functional properties based categorisation with details of NFPs, following a well-structured data model. We refer to this additional data as service meta data. Different service categories have different sets of relevant meta data. For example, printing services can consider colour options, while communication services might consider the transmission mode (e.g. synchronous).

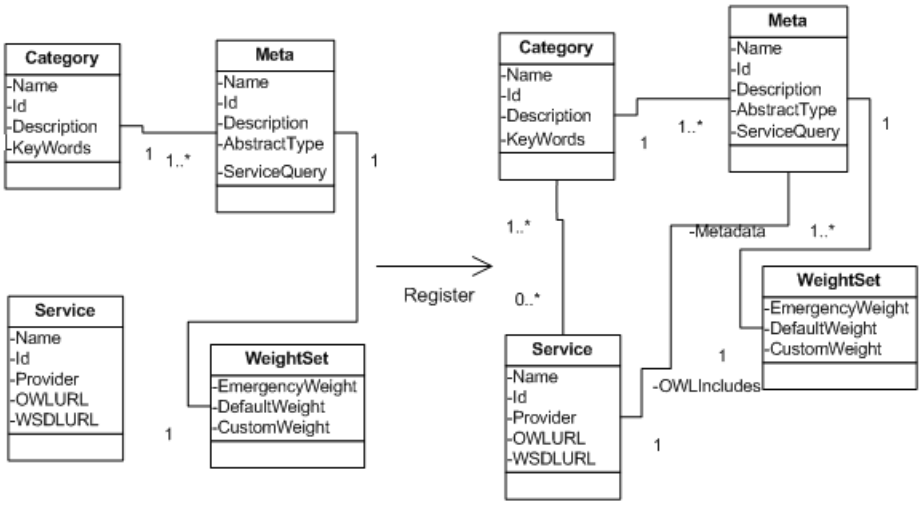

Fig. 2. The conceptual model of category, service and service registration

The service registration process (see Figure 2) builds a link between the service and the category. Meanwhile, the OWL-S description of the service should specify the meta information data which is defined by the category. In the following we provide more details about the category, meta data (Meta) and service.

Each category has a name, which identifies the category (there is also an identifier for computer rather than human use). This is useful for service developers who wish to register a new service, however for searches and finer grained

understanding of what the category represents a number of keywords describing the functional properties of services (or better operations) in the category are provided. A detailed description adds further detail for human use. Each category has a set of meta data associated to it, which captures the non-functional properties.

Each meta data element has an AbstractType, which is used to identify the correct evaluation function for this type of data. The associated WeightSet reflects the importance of this particular non-functional property in the category, from a service provider point of view. However, different situations require a shift in importance, as do individual users. So weights are more flexible in that they provide the default weight as specified by the provider, they allow for an emergency weight (usually defined for the application domain) and custom weights (usually defined by the end user).

In order to indicate that a small value is desirable, weights take on negative values in the range of]0,-1], if a larger value is desirable values come from the]0, 1] interval. In addition, an absolute value of 1, means that the criteria is a hard constraint (that is it must be satisfied, or we are not interested in the service). Examples for these cases are cost (the smaller the better), speed (the faster the better) and availability in a certain country (e.g. a retail service not shipping to the UK would be of no interest to a UK customer).

While some meta data can be found in the service profile (for example the speed of a printer, or whether it prints in colour), there are some criteria that depend on the service context and need to be more up to date (e.g. the length of a print queue). To obtain such data the ServiceQuery specifies a SPARQL [Gro08] query statement which can be used to locate this kind of information from the service context.

A service is described by typical elements, such as information about its provider. the service. OWLURL is a link to the location of the OWL-S description file of the service, which should contain the required data for the non-functional attributes. WSDLURL provides a link to the services WSDL file, as is required for using the service in current web service technologies.

Registering a service involves linking this to the service category model, that is assigning a category for each service (or operation). This has the side effect of linking the service to typical non-functional criteria for which users might require values and this data is populated from the services OWL-S file.

3.2 Automated criteria generation

Based on the user context model and the service category model, we define context-aware criteria that link the two sides of user context and service non-functional properties and are generated automatically. Context-aware criteria consists of a number of criteria that are initialised from the meta data of the correct service category.

The idea is that this brings together the data required to evaluated the service. For example when considering a transport service from Leicester, it is clear that the user would specify some values for locations, services would provide

to be queried on those and more over the two values need to be available for evaluation. The same has to be done for every other criteria of relevance.

In detail the context-aware criteria consists of data from the service profile as well as data from the user context. It presents itself with an AbstractType which is used to identify the related evaluation function. In terms of user data it contains a value and a weight set, which are both derived automatically from the user's context using the context query. The context query is a SPARQL query extracting context information from the user context repository (the repository is structured according to the context model presented earlier). The AbstractType and name of criteria as well as the service query are extracted from the service profile. This process of extracting and merging data is completely automatic.

4 The TLE Service Selection Method

In sections 2 and 3 we have discussed the context model and how it is linked to the services' meta data. We will now focus on the service selection process. There are two major steps in the service selection process: First we need to evaluate each criterion of each service. Then we need to aggregate all criteria evaluation results to get an overall score for each service in order to select the most suitable service for the user. To complete these two steps, we use a Type-based LSP (Logic Scoring Preference [Duj96]) Extension (TLE) method which has been introduced in our previous work [YRM08]. The TLE method includes a type-based single criterion evaluation process and an extended LSP aggregation function.

4.1 Type-based evaluation process

Most current criteria evaluation functions strongly rely on human input; usually this means that evaluation functions are designed and assigned to each criterion by hand, providing excellent results by allowing fine tuning of measurements. However, in the dynamic context, the evaluation function often requires to be adapted at runtime as the relevant criteria change. Therefore, human interaction is not acceptable and the method needs to be automized.

The type-based evaluation process is designed to automatically match evaluation functions to the criteria at runtime based on each criteria's abstract type. Various types can be defined for different evaluation contexts and environments to extend this type-based evaluation process. Currently, there are four abstract types which proved sufficient for our work.

The *Numerical type* is used for criteria which take numerical input to the evaluation method such as cost, time and other quantitative measurement values. The evaluation function is given by Formula 1:

$$\varepsilon = \begin{cases} \frac{1-(v_{max}-v)}{v_{max}-v_{min}} & \text{iff } W \geq 0, \\ \frac{v_{max}-v}{v_{min}-v_{min}} & \text{otherwise} \end{cases} \tag{1}$$

where w is the weight of the criterion. v_{max} is the maximum value of all competing services, v is the value for the service under evaluation. v_{min} is the minimum value of all competitive services (if user context does not indicate a minimum value constraint, in which case that value is used). For example, the price criterion or service response time.

The *Boolean type* is used for criteria which are evaluated to 1 or 0. The function is:

$$\varepsilon = \begin{cases} 1 & \text{if criterion is met,} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The *Set overlap type* is used to define criteria which are measured by matching on instances on am enumerated set:

$$\varepsilon = \frac{\varepsilon_1 + \varepsilon_2 + ... + \varepsilon_i}{n} \tag{3}$$

with ε_i being a score for each element of the set. For example, the constraint value of the available devices criterion is $C_{ad} = mobile, PAD, laptop, IM$ in our first notification service selection scenario.

The Distance type is used to evaluate the criteria which are measured by distance between two locations expressed by latitude and longitude.

$$\varepsilon = \begin{cases} R \times c & \text{iff } c \geq 1 \\ R \times 2 \times \arcsin(1) & \text{otherwise} \end{cases} \tag{4}$$

with $c = 2 \times \arcsin \sqrt{\sin 2(\frac{|L2-L1|}{2}) + \cos(L1) \times \cos(L2) \times sin2(\frac{|G2-G1|}{2})}$, $L1 =$ latitude of the first point, $G1 =$ longitude of the first point, $L2 =$ latitude of the second point, $G2 =$ longitude of the second point and $R =$ the Earth's mean radius of 6371 km. For example, the distance between injured person and the two medical support services is the crucial selection criterion in the medical support service scenario.

We found these types sufficient for our case studies, but we do not claim them to be complete. However, more types can be simply added by specifying an evaluation function and type name; the type name is then used in the service meta data definition.

This method provides a link between the evaluation function and the specific criteria under investigation, and the appropriate function can be chosen at runtime. The table 1 shows the criteria examples of select a notification service.

4.2 Extended LSP aggregation function

Having defined how individual criteria cam be evaluated, we turn our attention to the criteria aggregation function: vital for computing overall scores for a service. The LSP aggregation function [**?**] modifies the traditional weighted sum aggregating function, to capture concepts such as replaceability (the fact that one criteria might be replaced, that is ignored, if another criteria is extremely well

Table 1. Selection Criteria and Service NFPs Meta

Criteria ID	C1	C2	C3	C4	C5	C6
Criteria Name	Location	Devices	Price	Time	Privacy	PCW
Criteria Type	Boolean	Set overlap	Numerical	Numerical	Numerical	Set overlap
Weight	0.9	0.9	0.5	0.3	0.7	0.8

PCW = Prefer Contacting Way.

suited) or an mandatory-ness (the fact that a criteria can under no circumstances be ignored no matter how low the score is compared to other criteria). These concepts are captured in the power (r) that is applied to each factor (see 5).

$$E = (\sum_{i=1}^{n} W_i E_i^r)^{\frac{1}{r}} \tag{5}$$

LSP was developed for manual evaluation, we extended the LSP function with an automatic process to determine the correct value of r based on the weight values of different criteria. The extended function is shown in formula 6, where W_i can be less than 0 to express that a smaller value is desirable for numerical type criteria (e.g. think about minimizing cost).

$$E = (\sum_{i=1}^{n} \mid W_i \mid E_i^r)^{\frac{1}{r}} \tag{6}$$

If we refer back to the table 1, then the evaluation function should like equation 7

$$E = \mid W_{C1} \mid E_{C1}^r + \mid W_{C2} \mid E_{C2}^r + ... + \mid W_{C6} \mid E_{C6}^r \tag{7}$$

Details of the evaluation function have been introduced in [YRM08]. In this paper we added the link to context information, so that the service selection decision can be made automatically.

5 Evaluation

While the ranking approach is implemented in terms of the relevance engine in the inContext platform, the engine itself is developed as a Web service so allows for easy embedding in different environments. For more detailed analysis we have developed a testbed that also considers the generation of the criteria as described here. The testbed includes an OWL/RDF context store, a repository that is organised by service category, but is enhanced with the meta data model and information and the relevance engine performing the TLE selection process. The evaluation reported here considers scalability and was conducted through 3 evaluation cases for notification service selection scenarios. Within inContext the ranking method has been evaluated further on real case studies, focusing on functional correctness rather than scalability.

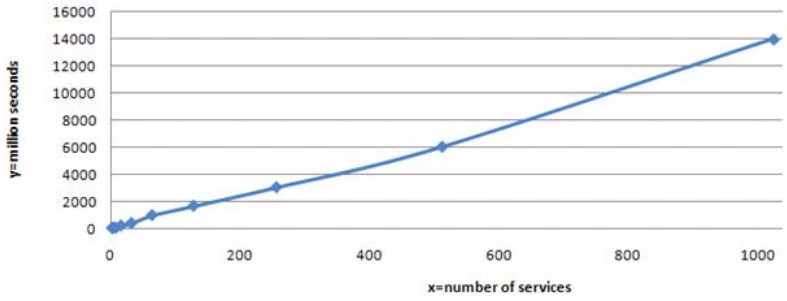

Fig. 3. Evaluation results for increasing numbers of services with a fixed number of criteria.

The first series of tests focuses on measuring the selection time when the number of services increases. There was a fixed number of criteria that was used to evaluate the services here (there were 6 criteria). We considered up to 1000 services.

Figure 3 shows that the approach is essential linear with respect to the number of services.

The second evaluation case was to evaluate the selection time in the light of increasing the number of criteria. We fixed the number of services to 4, but tested up to 192 criteria. The test results are shown in Fig. 4. We can again see that the approach is linear with respect to the number of criteria.

In the last test we evaluated the selection time against both an increasing numbers of criteria and services. We defined a number of test groups with different service numbers and evaluated these against an increase in criteria. The evaluation results are shown in Fig. 5.

From the results, we can see that the scalability will be not dramatically decreased with an increasing number of criteria if the number of services is not too large (e.g. less than 16 services) or if we consider large number of services but smaller numbers of criteria (less than 96 criteria).

This merits some more general discussion: in the real world service selection scenarios, we do not expect there to be vast numbers of criteria, so around 100 seems a good pragmatic upper bound. Also, in terms of competing services, while we expect these to increase with more services becoming available, it seems safe

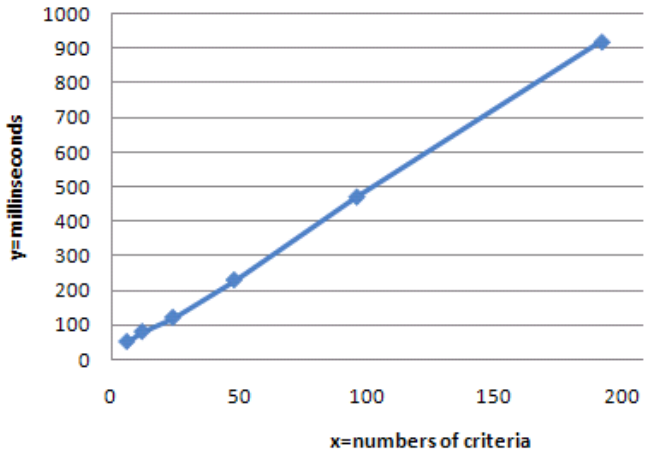

Fig. 4. Evaluation results for increasing numbers of criteria with a fixed number of services.

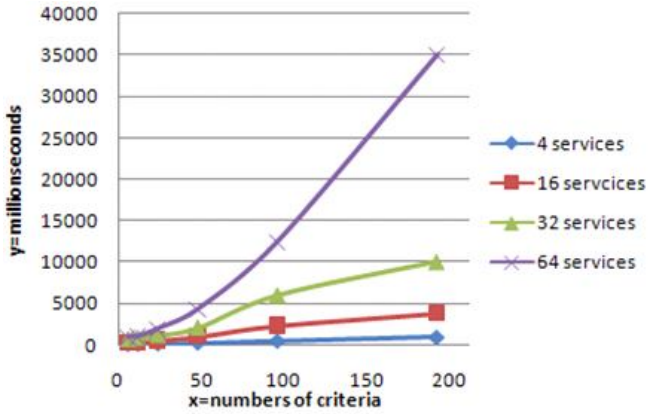

Fig. 5. Results for increasing number of services and criteria.

to claim that a choice of 20 services fulfilling our functional requirements and hence been drawn into the comparison should be already a significant number.

Additionally, we should see that the services are chosen in a matter of seconds based on the prototype implementation (which has not been designed with performance in mind, but rather with functional correctness). When considering real service selection scenarios, the runtime of services usually exceed this time by far, and being presented with the best possible service will be 'worth the wait', even more so if we consider this selection to form part of a longer running business process possibly containing human tasks.

It may be more helpful to evaluate the selection correctness from user point of views. However, correctness is difficult to be defined in general because it depends on different views and concerns. It is more like a multiple criteria decision problem, you can tell which decision is wrong, but it is very hard to say which one is much better than others from human and that is a key reason we need assistant from machine to get a tough decision. Therefore, we will not make a evaluation on the correctness from user side.

6 Related Work

Several research approaches have considered using context information for selecting suitable services for the end-user. Location, which is introduced in Cooltown project [Pac04], [RT06] and the Jini [KEKW04] service discovery protocol, is the earliest form of context information used for service selection. These approaches can discovery and select the service nearest to the user. Nerveless, the context information is limited to location context only.

[CKL05] and [LH03] extended the context information by adding so called dynamic and static service attributes. The dynamic service attributes are those characteristics of a service whose values change over time. Other attribute are said to be static. Since there is more than a single context constraint, these works make use of weighted vector based aggregation functions for ranking the services and returning the top matches to the user. However, there are two main drawbacks:

1. the work is reliant on a syntactic representation of contextual information of services. Consequently, it is very difficult to apply more advanced semantic level searching, matching and reasoning techniques.
2. they only focus on modelling services' attributes/context information without specifying the user's context information. Thus, context-awareness means service non-functional properties awareness.

Work in [ESB06] addressed the first drawback by utilising concepts from the Semantic Web. However, it does not address the second problem of considering and modelling user context information. In contrast, [SVC+03] makes a lot of efforts on defining user's context information in details and identifies nine categories expressing user context: User information, Personal Information, Activity Information, Social Information, User Defined Rules, Environment Information,

Application Information, Terminal Information and Network information. However, this work then expects service developers to build suitable new services in order to satisfy these context constraints.

In summary, current context-aware service selection methodologies do not bridge the gap between user's context and service's context. Few approaches provide a clear picture of using the user's context information for generating the service selection criteria/constraints. Furthermore, the ranking methods are far more simplistic than what is really required to cope with the context mapping between user and services. The presented work bridges between service and user context and provides a powerful, yet scalable ranking approach.

7 Conclusion and Future Work

In this paper, we developed a context model including four aspects typical for collaborative systems: user profile, resources, activities and physical environment. In order to link the user context information to the service selection, we defined a meta data based category system and context-aware criteria which can be automatically generated by querying both user context and service non-functional meta data. Context-aware service selection also requires a suitable selection method to use the criteria. Therefore, we introduced the TLE selection method, which is based on type-based evaluation functions and an extended LSP aggregation method. Our evaluation results show capability in both increasing numbers of services as well as increasing numbers of criteria. We illustrated the need for this work through two typical scenarios.

There are some issues worth future exploration. On one hand, it would be worthwhile to explore how other services selected as part of a workflow provide additional context and requirements. In fact, we have recently started to look at this issue and initial result has been presented in [YRMT08]. On the other hand, it is worthwhile to enable reasoning on context information to determine criteria weights automatically rather than statically providing weights for each user or service domain.

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

References

[CKL05] S. Cuddy, M. Katchabaw, and H. Lutfiyya. Context-aware service selection based on dynamic and static service attributes. Proceedings of IEEE International Conference on Wireless And Mobile Computing, Networking And Communications,Vol. 4, 2005.

[Duj96] J.J. Dujmovic. A method for evaluation and selection of complex hardware and software systems. Proceedings of 22nd International Conference for the Resource Management and Perfromance Evaluation of Enterprise Computer Systems, Turnersville, New jersey, 1996.

[Dus04] Schahram Dustdar. Caramba—a process-aware collaboration system supporting ad hoc and collaborative processes in virtual teams. *Distrib. Parallel Databases*, 15(1):45–66, 2004.

[ESB06] A-R. El-Sayed and J.P. Black. Semantic-based context-aware service discovery in pervasive-computing environments. Proceedings of IEEE Workshop on Service Integration in Pervasive Environments (SIPE), 2006.

[FipD07] European Union FP6 Framework and inContext project Deliveries. Design and proof-of-concept implementation of the incontext context model version 1 wp2.2, 2007.

[Gro04] RDF W3C Working Group. Rdf/xml syntax specification, 2004. http://www.w3.org/TR/rdf-syntax-grammar.

[Gro08] W3C SPARQL Standard Group. Sparql query language for rdf, 2008. http://www.w3.org/TR/rdf-sparql-query.

[KEKW04] N. Klimin, W. Enkelmann, H. Karl, and A. Wolisz. A hybrid approach for location-based service discovery. Proceedings of International Conference on Vehicular Ad Hoc Networks, 2004.

[LH03] C. Lee and S. Helal. Context attributes: An approach to enable context-awareness for service discovery. Proceedings of SAINT'03, 2003.

[Org04] OASIS Organisation. Uddi version 3 specification, oasis standard, 2004.

[OWL] Owl web ontology language. http://www.w3.org/TR/owl-ref/.

[Pac04] Hewlett Packard. Cooltown project, 2004. http://www.cooltown.com/cooltown/.

[RMTC+09] Stephan Reiff-Marganiec, Hong-Linh Truong, Giovanni Casella, Christoph Dorn, Schahram Dustdar, and Sarit Moretzki. The in-context pervasive collaboration services architecture. Proceedings of Service Wave 2009; to appear, 2009.

[RT06] O. Riva and S. Toivonen. A hybrid model of context-aware service provisioning implemented on smart phones. Proceedings of ACS/IEEE International Conference on Pervasive Services, 2006.

[SVC+03] I. Sygkouna, S. Vrontis, M. Chantzara, M. Anagnostou, and E. Sykas. Context-aware services provisioning on top of active technologies. Book Series Lecture Notes in Computer Science: Mobile Agents for Telecommunication Applications, Category Service Management - Service Provisioning, Subject Collection Computer Science, 2003.

[YRM08] H.Q. Yu and S. Reiff-Marganiec. A method for automated web service selection. Proceedings of 2nd IEEE International Workshop on Web Service Composition and Adaptation (WSCA-2008) Special Theme: Dynamic Services Composition and User Steering held in conjunction with 6th IEEE International Conference on Services Computing (SCC-2008), Honolulu, USA., 2008.

[YRMT08] H.Q. Yu, S. Reiff-Marganiec, and Marcel Tilly. Composition context for service composition. Proceedings of IEEE International Conference on Web Service, WIP Track, 2008.