Applying Web Services Technologies to the Management of Context Provisioning

Michael Brenner, Michael Schiffers

Munich Network Management Team, Department of Informatics

Ludwig-Maximilians-University, Munich, Germany

{brenner|schiffers}@informatik.uni-muenchen.de

For a service to be context-aware it must be able to discover and take advantage of contextual information like device capabilities, user surroundings, or the user's current preferences. Ensuring the availability of this information according to predefined quality parameters requires an adequate management of the whole process of sensing, interpreting, iteratively refining and disseminating large amounts of context information in multi-provider environments. These phases are subsumed under the term *context provisioning*. While research in context-awareness so far has been focusing on functionalities, this paper explicitly emphasizes management aspects of context provisioning.

In our projects we are investigating the applicability of Web Services architectures for the provisioning of context information. Advantages are, besides the universal reach, the simple way to describe, encapsulate, advertise and access the services which eventually deliver the respective context information. But using Web Services for context provisioning poses two important questions: Is the Web Services model adequate enough for building a generic context provisioning architecture? And: Is the Web Services model at the same time also adequate enough for the management of context provisioning? This paper addresses both, but emphasizes the management challenges, especially regarding service configuration, inherent to context provisioning and proposes approaches to their solution.

1 Introduction

Context-awareness has recently emerged as a new paradigm for services being reliant on highly dynamic information about aspects of their usage surroundings (e.g. the location, user activity, device capabilities). Various infrastructures and frameworks supporting the construction of such context-aware services (CAS) by assisting in the process of providing the required context information have been proposed [1, 2, 3, 4].

Context Provisioning (CP) is a complex task as it has to cope with challenges like heterogeneity of providers, highly dynamic environments, sensor imperfection, dynamic dependencies between context information, and the quality of context information. A critical issue is the ability to describe how and in which order context information must be collected from widely dispersed resources belonging to different administrative domains. The basic objectives of CP are the dynamic creation and execution of a value chain ensuring the availability of all required context information according to predefined quality parameters [5].

We envision that context information will typically be provided by autonomous organizations (called context providers) which would advertise their services over the Web for seamless integration into CASs by respective service providers. Therefore, we are investigating the applicability of Web Services architectures for the provisioning of context information. But context provisioning also leads to new management challenges [6], due to their real-time and automation requirements, and their high dynamics.

The idea to apply Web Services Technologies to the provisioning and management of context provisioning is, to our knowledge, new. Comparable approaches are hard to find as almost all research projects are concentrating on functionality aspects of device-centric context-aware services and completely neglect the management aspects. [6] is the first paper to address specific management issues of context provisioning. Some investigations have been undertaken, however, in order to understand automatic service composition in general ([7] contains a good overview), the dynamic composition of Web Services in particular ([8, 9]), the semantic creation of workflows [10], and the creation of Web Services workflows ([11]). Although none of these approaches address integrated management aspects, their efforts are valuable when exploring specific management issues like e.g. in configuration management.

Driven by a scenario (section 2) we will discuss in more detail special aspects of context provisioning (section 3). In section 4 we will list some requirements for the management of context provisioning. In section 5 we will explore the suitability of Web Services technologies for the management of context provisioning.

2 Scenario

For illustration purposes, let us consider an Allergy Forecast Service. This (yet fictional) context-aware service delivers a prognosis on how severe the user is expected to be affected by pollen allergy symptoms on a given day, thereby providing her with the opportunity to take precautions, e.g. taking an antihistamine, in due time[1]. Figure 1 depicts the dependency graph of the context information that would contribute to the context-aware service "Allergy Forecast Service". In the context provisioning process all required context information will need to be obtained, be it user-specific data (e.g. the user's profiles or her travel itinerary) or sensed data (e.g. location), or database contents (e.g. bloom periods), or computed information (e.g. expected pollen density).

Managing context provisioning in the Allergy Forecast Service example creates several challenges. Just to mention a few:

- What is the optimal sequence for gathering and combining the required context information? What are good metrics for "optimal"?

[1]Our service would usually be invoked manually by the user herself, e.g. through a web-based interface. It could, however, also be invoked automatically by another service, e.g. a personal medical assistant service. Our service would then not only be a context-aware service, but rather a context provisioning service for the context-aware service "Medical Assistant" (see section 3).

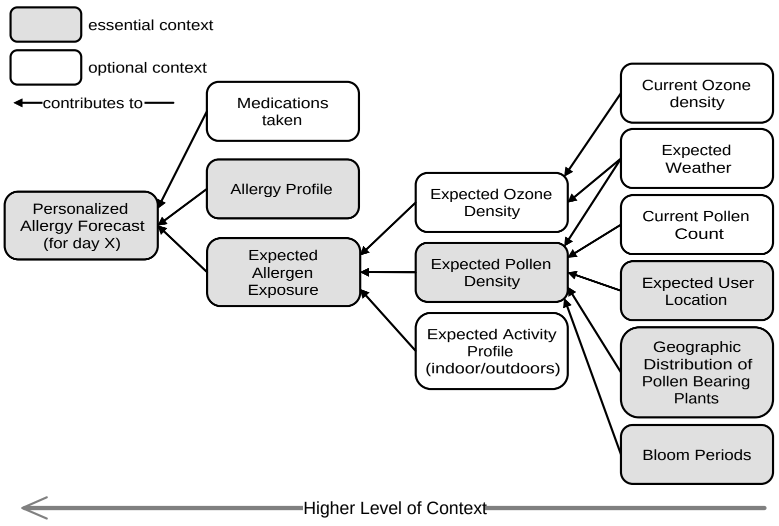


Figure 1: Context information contributing to the Allergy Forecast Service

- How to discover context information if the owner of that information decides to change the provider?
- How to detect a bad quality of a context information? How is this quality being measured? How to recover from bad quality?
- How to secure the whole provisioning process? How to detect intrusion? How to protect the privacy of delicate information?
- How is the cooperation between the providers achieved, maybe even enforced?

Note that for a basic service (the *core service*), a subset of this context information would be sufficient, (the *essential context*). However, the usefulness or quality of the service can be greatly improved by processing *optional context.* For example, the service could include the expected density of ozone, which might make the prediction of pollen allergy symptoms more accurate. Or a forecast for the same day could benefit from current pollen count data. For a forecast three days ahead the current pollen count might be less useful, but considering the weather forecast could make a prognosis about the amount of pollen to be expected in the atmosphere more reliable. Essential context is necessary for the delivery of the core functionality of the context-aware service (CAS). However, optional context may be used to improve service quality, either when e.g. a higher than normal accuracy is requested by the user or in case the essential context is not available with the quality required to achieve the standard CAS quality. Since acquiring optional context might be expensive or would lengthen response times, one of the management tasks would be resolving the trade-off between incorporating it into the context provisioning service or not.

3 Context Provisioning

In the Allergy Forecast Service scenario, context information is needed in order to deliver the service as required. Some of that information is *sensed* directly (e.g. the location using physical sensors, or bloom periods using databases as abstract sensors), while others have to be *refined* from already available information (e.g. the pollen density at a given location for a given time), thereby creating higher level context. Context dissemination will collect (aggregate) context information provided by sub-services in order to enable the further refinement or, in the last step of the context provisioning process, providing the necessary input for adaption of the CAS.

Context provisioning implies a goal-oriented cooperation of several actors in the roles of CAS Users, CAS Providers, Context Providers, and Context Owners. Effectively managing context provisioning aims at providing required context information with a desired level of quality thereby guaranteeing the availability of the CAS according to respective agreement between a CAS Provider and a CAS Customer. Figure 2 depicts this role model, an edge between two roles indicates a contractual or information flow relationship. For a more thorough discussion of this model we refer to [6]. As one actor may play several roles and as one role may be played by several actors, there is an n:m-relationship between actors and roles.

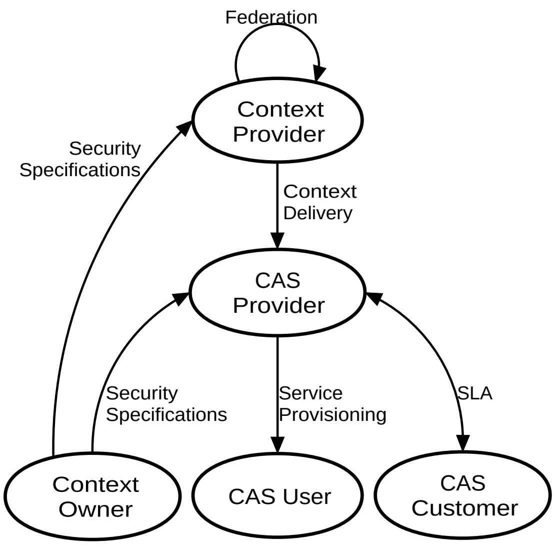


Figure 2: Context provisioning role model [6]

As in the Allergy Forecast Service, the macro phases of context provisioning (i.e. *context sensing*, *context refinement*, *context dissemination*) are arranged along a value chain generating "higher level" context information from "lower level" ones (see figure 3). We consider a value chain as a specific serialization of a workflow associated with some value function. A workflow itself denotes the automation of a process during which documents, information, or tasks are passed from one participant (role) to another for action according to a set of procedural rules [12].

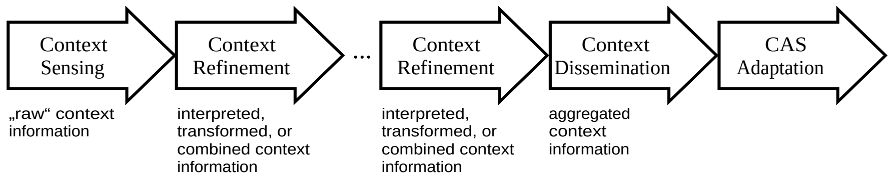

Context Sensing
Context Refinement
...
Context Refinement
Context Dissemination
CAS Adaptation
„raw" context information
interpreted, transformed, or combined context information
interpreted, transformed, or combined context information
aggregated context information

Figure 3: The context provisioning value chain

4 Requirements for the management of context provisioning

This section introduces some challenges for the management of context provisioning, that were analyzed and discussed in [6].

Context provisioning needs to cope with flexibility in at least three key aspects leading to what is called an aggregation problem in [13]:

- The potential use of context information cannot always be anticipated and a CAS cannot always know in advance from where to gather the context information necessary for its adaptation.
- The composition of context information must be flexible enough to accommodate changing characteristics of context information sources, the mobility of sensors and actors, and the dynamic nature of required network resources.
- Context provisioning (i.e. the provider) must be able to flexibly choose context information sources and methods of composition in order to satisfy scalability requirements and at the same time achieve Quality-of-Service (QoS) and Quality-of-Context (QoC, see [5]) objectives.

In order to cope with these requirements, context provisioning management needs to be able to automatically generate a complete context provisioning workflow from a machine readable description and execute it by installing an appropriate value chain.

Context provisioning management must cope with the imperfection of context information due to sensor defects or incomplete information due to sudden sensor losses or network faults. A requirement for an effective fault management in context provisioning is thus the provisioning of adequate recovery mechanisms which may result in scheduling alternative value chains through configuration management with an equivalent or similar QoC in order to guarantee the survivability of at least the core service. The main task of this management function is to establish the most appropriate value chain by taking into account quality parameters and resource availabilities. In other words, value chain creation is itself a context-aware service spanning heterogeneous systems and various context provider domains. For accounting purposes all data related to the usage of a context-aware service, especially to the usage of context information, need to be gathered. The usage must be assignable to the roles to be accounted. Accounting becomes an even more critical issue when considering roaming: in foreign domains the subscribed context provisioning services may not be available at all, or only in parts, or with a bad quality of context information.

From a user's perspective, a CAS "performs well" if she will always receive the anticipated deliverables with a perceived quality. From a management perspective, a CAS provider must therefore be able to tag the value chain (and update the tags) with appropriate QoC information, see [5] for a discussion of QoC related issues.

5 Applying Web Services Technologies and Concepts to the Management of Context Provisioning

5.1 Shared characteristics of context-aware Services and Web Services

At the same time that mobile phone network operators have been thinking about business models and architectures for context-aware services, the concept of XML-based *Web Services* has emerged.

XML-based Web Services make use of the all but ubiquitous infrastructure for HTTP communication, by using HTTP for transport of XML-encoded data other then HTML pages. Web Services interact by sending SOAP messages via HTTP[2]. SOAP uses a basic XML message format that includes optional header elements and a body element within its envelope. A SOAP body can contain any type of XML document. The standard itself does not put any restrictions on message flow, therefore many different interaction patterns between Web Services, e.g. one-way (1-to-1 or 1-to-many), request/response or solicit response (request without input data) can be realized.

The typical scenarios for Web Services and context-aware services share some common aspects. In most scenarios trying to illustrate the benefit of Web Services for end-users, the desired result could also be achieved if the user accessed a number of web-sites herself. But doing so would require finding suitable web-sites, browsing them and interacting with them, manually taking information from one web-site as input for another service offered through a WWW-interface and so on. Typical examples are the scenarios where users want to convert results from a stock ticker into another currency and calculate the value of their holdings, or the travel booking scenario, where a user has to access several airline web-sites to find the cheapest flight, then book a rental car and a hotel room and so on.

So what Web Services do (for the end-user), is mostly the automation of a more or less complex information retrieval workflow that involves invoking numerous services, some of which might again be dependent on others and so on. This is very much like the process of context provisioning that we discussed earlier in section 3.

5.2 Web Services as Middleware for Context Provisioning

Since SOAP can carry any XML data, using it for the transport of context information (or requests for them) or any other data exchange between actors in a context provisioning process, requires only that the appropriate data is encoded in XML and that a common understanding of the message format and interaction exits between all actors. This can achieved by defining the access to the services offered in a WSDL [14] document. The obvious choice of an interaction pattern between context providers and context users would be RPC-oriented[3], realized by exchange of messages conforming to the SOAP RPC Representation [15]. The acquirement of context by a *context consumer* (a user context information, e.g. a CAS provider or a context provider needing input for refining context) can then be realized by accessing a Web Service published by the context provider. If context providers then also use SOAP for internal communication, by enabling access to their sensors through SOAP via integration servers and publishing their refinement functions as SOAP RPCs (see figure 4 on the following page), then the whole provisioning value chain will be realized by the chaining of Web Services.

Each context provisioning service published by a context provider will then supply one type of context information, which is either sensed or the refinement of one or several other context information items (sensed or acquired or from other services). How this context information is exactly procured by the context provider, should be transparent for the context user, who will only need to define the QoC and QoS levels needed for its

[2]Bindings of SOAP to other protocols are possible and definition proposals have been made for a number of them, but to this day only the binding to HTTP is part of the SOAP standard.

[3]with the context user as the invoker of the service

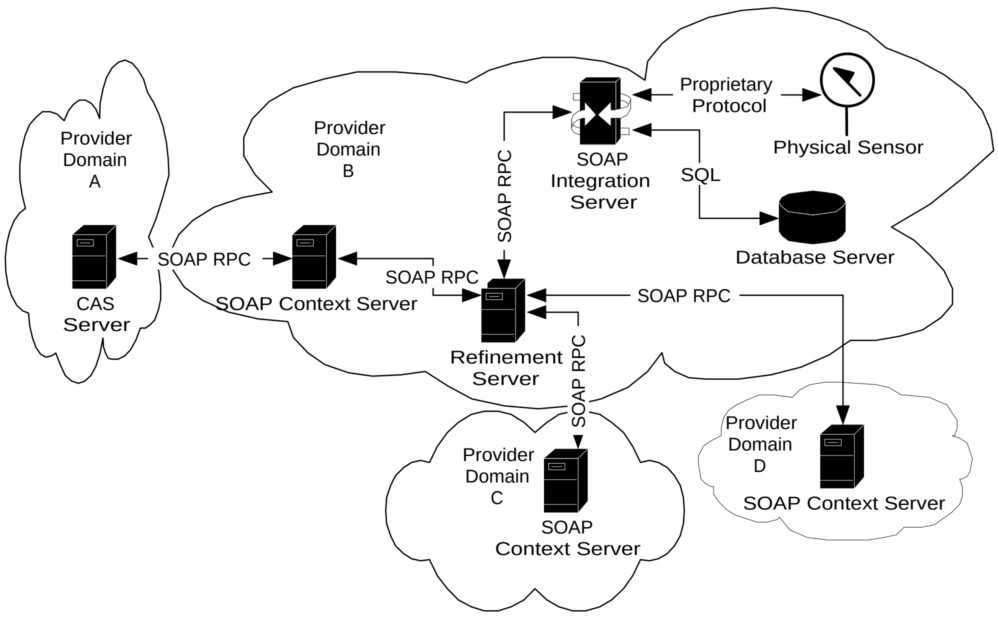


Figure 4: Example of a context provisioning infrastructure

own purposes. QoC parameters can include, among others, *up-to-dateness* (how old can the context information be), *trustworthiness* and *precision* (accuracy) [5]. The ability of a context consumer to specify levels for these quality parameters when requesting a certain type of context information, liberate him from the complexity of having to detail the workflow of how a context item is to be procured - much like the specification of QoS in a SLA liberates the context consumer of "classic" IT services from having to deal with the complexities of its technical implementation. This is therefore a prerequisite if modularity is to be achieved in context-aware systems.

The only drawback of using SOAP for communication along the context provisioning value chain could be its "verbosity", that might make it seem unsuitable for transport across networks like GSM, where bandwidth is sparse and expensive. On the other hand it is simple, can rely on widely used HTTP, and might offer easier interoperability with Web Services accessible through the Internet, which might become numerous and convenient sources for some types of context information in the future.

5.3 The Challenge of Providing QoC and QoS Guaranteed Context Services

Using Web Services for context provisioning does not, by itself, provide answers for any of the questions regarding its management. For Web Services, too, no standardized techniques for managing QoS, and naturally not for managing QoC, have been established. But the relevant standards allow extensions, and some work has already discussed possibilities for using Web Services to support management functionalities (e.g. in [16]).

For meeting the required QoC and QoS levels in the delivery of context item, a context provider needs to build the correct value chain. This might be anything but a trivial task, even if we assume, that a context provider knows about the QoS and QoC achievable by its own sensors. Consider again the example described in section 5.2 and depicted in figure 4. What if the services invoked at providers C and D were also available from a large number of other providers, each offering a service with a particular combination of QoS, QoC and cost?

If some sub-service fails, can the invoking service still be provided in the necessary quality? Is an alternative refinement function achievable if the QoC delieverd by the core service does not match the demanded level?

As argued below, for answering these questions, information about the qualities of the third party services will be needed.

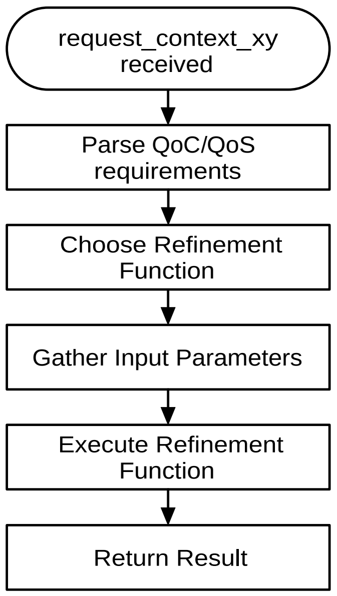
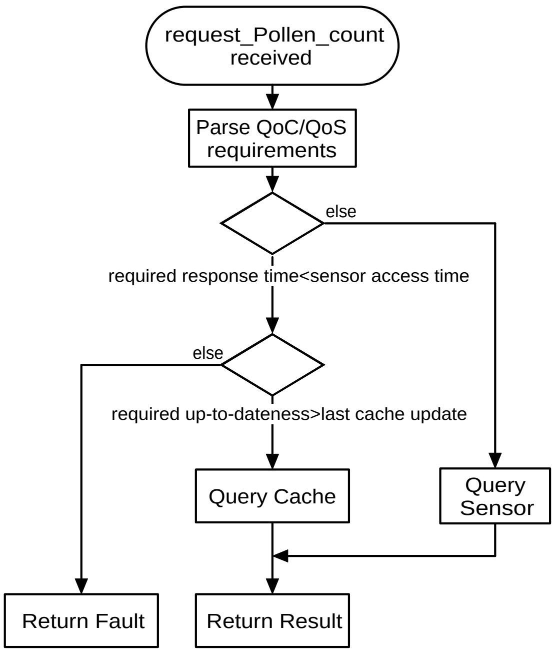

request_context_xy received
Parse QoC/QoS requirements
Choose Refinement Function
Gather Input Parameters
Execute Refinement Function
Return Result
Figure 5: First Level Workflow in CP

Figure 5 shows the first level activities that are executed by a context provider. Two of these activities, choosing the refinement function and gathering the necessary input parameters, can involve complex workflow patterns. Consider the example of the expected pollen density service in the Allergy Forecast Service value chain (figure 1 on page 3), where the basic service would produce its result by checking which plants bloom at the given time, and then map the expected user location the geographic distribution of these plants.

Higher QoC might be achieved in two ways: Either by choosing an alternative refinement function or gathering inputs with better QoC for the original refinement function. But achieving high QoC might incur trade-offs in achieving low response time and vice versa. Consider for example the context information *current pollen count* from our scenario. A context provider offering this information (which, we assume, needs no refinement) for a given area might obtain this context by accessing a context cache, assuring a short response time – but thereby not being able to guarantee the best possible *up-to-dateness*, for which directly querying the sensors would be necessary (see figure 6).


request_Pollen_count received
Parse QoC/QoS requirements
else
required response time<sensor access time
else
required up-to-dateness>last cache update
Query Cache
Query Sensor
Return Fault
Return Result
Figure 6: Simplified Workflow Example

When a refinement of context is executed, deciding on the optimal value chain becomes more difficult and requires predicting QoC of the output of a refinement function. This can be mapped to the QoC of input through a mapping peculiar to each function. In such a case, information about the input QoC levels, i.e. that of the context source (sensor or another provider) con-

```
<env:Envelope
 xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
    <env:Header>
         <qoc:qoc-pollen
          xmlns:qoc="http://contextsomething.org/pollen-count/qoc"
          env:mustUnderstand="1">
              <qoc:up-to-dateness value="2003-08-08-15:00" />
              <qoc:presicion value="98" />
              ...
         </qoc:qoc-pollen>
    </env:Header>
      ...
</env:Envelope>
```

Figure 7: Including QoC requirements into a invocation message for the “Pollen Count” service

sumed by the refinement function, is needed. Obviously, QoS parameters like response times of sensors or other context sources will need to be known, too.

The required QoS and QoC should therefore be included in a service invocation and achieved QoS and QoC should be included in service responses. The drawback to this is, that a service will have to be invoked, and therefore possibly paid for, before its quality can be evaluated. Another approach could be the publishing of QoC/QoS-information services by the providers that would return the currently guaranteed quality levels for each service without the context information itself. Even when such a QoC/QoS-information service is offered, the response to a context information request should include the realized QoC, since it could be better than what was guaranteed.

For transport of this information over SOAP, “inline” with the message flow of a context request or explicitly when invoking a QoS/QoC-information service, an appropriate XML encoding (and namespace) for required QoS and QoC parameters for every type of service need to be defined (see figure 7). In both cases this information can be included in the SOAP header. A response message to a QoS/QoC-information request would therefore have the same header[4] as the response to a context provisioning request but carry an empty body element. Also some convention about the semantics of these parameters (“how and on what scale is the precision of a pollen count service measured?”) would need to be agreed upon. Context services not offering support for QoC or QoS could still be invoked by setting the “must understand” attribute in the corresponding header elements to “0”. But if required elements are marked with “mustUnderstand=1”, a SOAP server that cannot parse any of them will have to return a SOAP fault.

QoC and QoS levels of a the external context sources can primarily be supplied only by the context providers hosting those sources. Clearly, the accuracy of this information is essential to the consumers of the context service in question, while the provider might

[4]Of course, despite identical formats, the semantics of the information would be slightly different. A response to a QoS/QoC information request describes minimum achievable quality, while the QoS/QoC information in the header of a context service response describes the actually achieved quality.

be tempted to manipulate this data in order to attract more business. This takes us to the QoC-parameter of "trustworthiness", a quality that is plainly not suited to be inquired from the service[5] itself. Therefore a mechanism will be needed that encourages context providers to publish management information to their customers in an accurate and faithful way.

5.4 Gaining Information about Trustworthiness

A possible solution could be the addition of a *rating provider* [17]. After a context consumer in search of a context information has acquired access data for a number of suitable services from a service repository run by a *context broker*, she could query a rating service which of the services in question is recommendable in terms of their trustworthiness. The trustworthiness levels required for these recommendations could be compiled from ratings supplied by the customers of these providers themselves, thus creating a system that, despite its well-known shortcomings, has been successfully used by eBay for many years now. Of course this could still be augmented by active performance monitoring by the rating provider, e.g. by launching test requests.

In a Web Services based federation of context providers, such a rating service could be also be accessed through SOAP Server, serving as a frontend to the ratings database, while a *context broker* would run a UDDI (Universal Description, Discovery and Integration, see) registry (also accessible through SOAP).

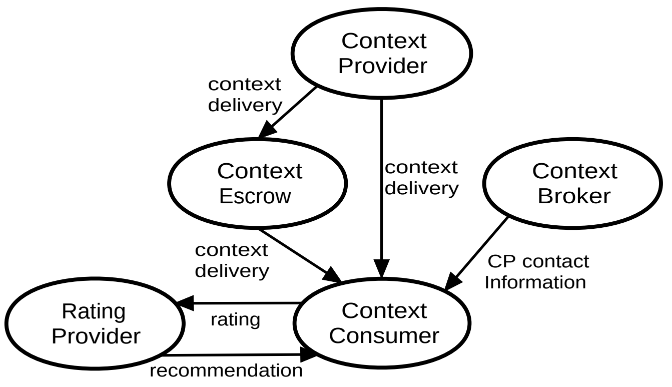


Figure 8: Roles in Context Provisioning Federation

In case a rating repository would not be sufficient or not suitable (e.g. when a context provider is new and therefore unrated), another actor, trusted by both parties (the context provider and consumer) could act in the role of a context escrow, realizable by a SOAP intermediary acting as a proxy, through which context service invocations and responses would be routed (see figure 8). Both parties would direct their SOAP messages to the escrow service, including a header containing, besides the address of the business partner to which the message should be forwarded, processing instructions, regarding what kind of escrow service is requested (e.g. just counting exchanged messages or also caching them for audit purposes). This way, service response times could also be approximately measured. Overseeing a large number of transactions, a context escrow would then be an excellent source for ratings, allowing new context providers to build a reputation quickly.

Of course a context escrow, context broker and rating repository service could all be run in cooperation or even as services of a single context broker, allowing for some synergies. E.g. services from providers with low trustworthiness could be removed

[5]Trustworthiness is technically attributed to a service. In reality, however, it is a quality of a context provider. It can therefore make sense to, instead of querying the level of trustworthiness of a single service, to take the average level of all services offered by that provider.

from the UDDI context registry, while the ratings database could be updated with data gathered by the escrow service.

5.5 Controlling the Workflow

We consider effective (re-)configuration management, i.e. the ability to create, monitor, and (re-)configure optimal value chains, the first major challenge that needs to be overcome in order to realize federated context provisioning. It also is an important prerequisite for successfully managing fault handling and performance tuning.

Configuring a context service is about initializing its value chain. If all possible elements of a value chain are, as proposed, accessible through SOAP RPC services, then Web Services orchestration languages, e.g. BPEL4WS (Business Process Execution Language for Web Services, [18]) offer an elegant approach for controlling the workflow. BPEL4WS (like other Web Services orchestration languages) describes Web Services workflows in XML allowing for the automation of setting up "best" value chains. This can be achieved by documenting the context provisioning workflow including all necessary conditional branches for achieving the best value chain, in a BPEL4WS document. Such a workflow is then executable on an orchestration server. BPEL4WS is pure XML, therefore a workflow could easily be transferred to other serves, as the realization of a context provisioning process would be independent of the technologies (e.g. J2EE or .NET) that the orchestration servers are based upon.

A suitable design of the workflow can support performance management through achieving a high QoS-level. Obviously, the workflow design can also ba a way to yield better QoC or a way to improve stability through fault handling, even though some compromise will be needed when reconciling these aims.

BPEL4WS not only allows sequential, but also parallel invocations of other Web Services and asynchronous message flows [19], allowing, for example, when QoC is valued higher then response time, to simultaneously invoke two (or even more) value chains yielding different QoCs. If execution of all value chains were successful within the given time constraints the result with the highest QoC could be used, otherwise a timeout would trigger returning the best result achieved thus far. When on the other hand response time is most important, the first response, regardless of its QoC could be immediately used.

Timeouts are also useful for detecting faults in proper time, before QoS guarantees are violated, and reconfigure the value chain. Parallel execution, too, could obviously improve fault tolerance. Caches like proposed in [6] are a possible way to enhance overall efficiency of a context-aware system. These could again be realized by using orchestration servers acting as service proxies based on the a SOAP intermediary. An appropriate workflow for controlling cache access dependent on QoC has already been discussed in section 5. In the same manner SOAP intermediaries can also be used to route SOAP messages for load balancing purposes. Current load could either be determined directly through including current load information in the header of service responses (in the same manner as including QoC/QoS information) or indirectly by measuring response times.

6 Concluding Remarks and Outlook

In this paper we have discussed some issues in managing context provisioning in multi-provider environments by applying Web Services technologies. We have presented some requirements for the management and we have argued that using Web Services technologies might be a promising approach for the management as well as the provisioning of context aware services. This is not least because of some similarities between context provisioning and Web Services. Both share many concepts but they also share many challenges. Both offer visions of emerging services but business models are yet to evolve. Differences may be found mostly in the type and the dynamics of information processed, but not so much in the underlying principles. As of today we would venture to predict that if the vision of provider-independent and interoperable context-awareness is to be achieved, this will most likely incorporate the use of Web Services. If, on the other hand, providers will offer their own proprietary context-aware services and regard ease of interoperability as of minor concern, then proprietary "binary" protocols, due to their bandwidth efficiency, will be given preference over SOAP.

This work is intended as a starting point for future work. In a next step we will study in more detail the suitability of BPEL4WS as well as competing Web Services orchestration languages like Business Process Modelling Language (BPML) [20] for managing the workflow of context provisioning. We are working on a prototypical realization of the Allergy Forecast Service discussed in section 2. We will also continue our investigations on how QoC guarantees can be achieved and what impact this will have on the applicability of Web Services technologies. An interesting question in this "context" is: What is the impact of using Web Services technologies on the "last- and first-mile", i.e. when communicating with rather simple sensors?. SOAP, for example, might require too much bandwidth and appropriate gateways might be needed. We will also investigate aspects of both accounting and security management, as discussed in [6], in more detail. We are especially interested in understanding the impact of context information, which has been consciously manipulated by the context owner, on the management process.

7 Acknowledgement

The authors wish to thank the members of the Munich Network Management (MNM) Team for helpful discussions and valuable comments on previous versions of the paper. The MNM Team directed by Prof. Dr. Heinz-Gerd Hegering and Prof. Dr. Claudia Linnhoff-Popien is a group of researchers at the University of Munich, the Munich University of Technology, and the Leibniz Supercomputing Center of the Bavarian Academy of Sciences. Its web-server is located at http://wwwmnmteam.informatik.uni-muenchen.de.

References

[1] A. Schmidt, K.A. Aidoo, A. Takaluoma, U. Tuomela, K. Van Laerhoven, W. Van de Velde. Advanced Interaction in Context. In H. Gellersen, editor, *Proceedings of the International Workshop on Handheld and Ubiquitous Computing (HUC99)*, volume 1707 of *LNCS*, Heidelberg, Germany, 1999. Springer.

[2] A.K. Dey. *Architectural Support for Building Context-Aware Applications.* PhD thesis, College of Computing, Georgia Institute of Technology, December 2000.

[3] Jason Hong. Context Fabric: Infrastructure Support for Context-Aware Systems - Qualifying Exam Proposal. http://www.cs.berkeley.edu/ jasonh/research/index.html, March 2001.

[4] Terry Winograd. Architectures for Context. *Human-Computer-Interaction*, 16(2), December 2001.

[5] T. Buchholz, A. Küpper, and M. Schiffers. Quality of Context: What It Is And Why We Need It. In *Proceedings of the 10th Workshop of the OpenView University Association: OVUA'03*, Geneva, Switzerland, July 2003.

[6] H.G. Hegering, A. Küpper, C. Linnhoff-Popien, and H. Reiser. Management Challenges of Context-Aware Services in Ubiquitous Environments. submitted to 14th IFIP/IEEE Workshop on Distributed Systems: Operations and Management (DSCOM 2003), Heidelberg, Germany 2003.

[7] Dipanjan Chakraborty and Anupam Joshi. Dynamic Service Composition: State-of-the-Art and Research Directions. Technical Report TR-CS-01-19, Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, Baltimore, MD, 2001.

[8] V. Tosic, B. Pagurek, B. Esfandiari, and K. Patel. On the Management of Compositions of Web Services. In *Proceedings of the OOWS01 (Object-Oriented Web Services 2001) Workshop at OOPSLA 2001*, Oct 2001.

[9] B. Benatallah, M. Dumas, Q. Sheng, and A. Ngu. Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services. In *Proceedings of the 18th International Conference on Data Engineering (ICDE 2002)*, San Jose, CA., Feb 2002. IEEE Computer Society.

[10] A. J. S. Cardoso. *Quality of Service and Semantic Composition of Workflows.* PhD thesis, Graduate Faculty of The University of Georgia, Athens, Georgia, 2002.

[11] W.M.P. van der Aalst. Don´ t go with the flow: Web services composition standards exposed. *IEEE Intelligent Systems*, 18(1), Jan/Feb 2003.

[12] H.-G. Hegering, S. Abeck, and B. Neumair. *Integrated Management of Networked Systems.* Morgan Kaufman Publishers, 1999.

[13] Norman H. Cohen, Apratim Purakayastha, John Turek, Luke Wong, and Danny Yeh. Challenges in Flexible Aggregation of Pervasive Data. IBM Research Report RC 21942 (98646), IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY., Jan 2001.

[14] W3C. Web Services Description Language (WSDL) 1.1, MAR 2001. W3C Note, `http://www.w3.org/TR/2001/NOTE-wsdl-20010315`.

[15] W3C. SOAP Version 1.2 Part 2: Adjuncts, MAY 2003. W3C Proposed Recommendation, `http://www.w3.org/TR/2003/PR-soap12-part2-20030507/`.

[16] W3C. XML Protocol Usage Scenarios, Jun 2002. working draft, http://www.w3.org/TR/2002/WD-xmlp-scenarios-20020626/.

[17] Scott Weller. Web Services Qualification. Technical report, IBM Developer Works, Apr 2002. ftp://www6.software.ibm.com/software/developer/library/ws-qual.pdf.

[18] Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic, and Sanjiva Weerawarana. Business Process Execution Language for Web Services - Version 1.1, May 2003. ftp://www6.software.ibm.com/software/developer/library/ws-bpel11.pdf.

[19] Collaxa, Inc. *Programming BPEL*, 2003. version 4.27.3.

[20] Intalio Assaf Arkin. Business Process Modeling Language, 2002. Version 1.0 - Last Call Working Draft, http://www.bpmi.org/bpml.esp.