Context Coupling Techniques for Context-aware Web Service Systems - An Overview[*]

Hong-Linh Truong and Schahram Dustdar

Distributed Systems Group, Vienna University of Technology, Austria
{truong, dustdar}@infosys.tuwien.ac.at

November 28, 2009

Abstract

Distributed services in a context-aware Web service system or in a service composition need to couple context information from different sources in order to be able to interpret context information. Such interpretation is a prerequisite before services can adapt themselves to be context-aware. However, context coupling in context-aware Web service systems is a complex issue. It is related not only to how we represent and transfer context information among Web services, but also to how we support the context storage, and how we ensure security and privacy issues of context information. Context coupling techniques for Web service systems will be different from that of tightly coupled systems because, with Web service systems, context information is typically shared across the boundaries of organizations which host Web services. This chapter aims at presenting an overview of context coupling techniques, their related issues, and their implication for a context-aware Web service system. Our approach is to study existing techniques implemented in current systems, to present open context coupling issues in current and future context-aware Web service systems, and to suggest further research directions.

1 Introduction

At the time of writing, utilizing Web services to develop large-scale systems and distributed applications operating beyond the boundary of a single organization is the norm. Many Internet-based systems include several Web services, which are loosely coupled and owned by different providers. The concept of the Internet of Things and the Internet of Services[45], although still being shaped, puts this loosely coupled applications model further: future Internet-based applications will include various software services, things, and people interacting via standard protocols and models, which are highly dependent on SOA (Service-oriented Architecture) technologies. To date, we have observed the popularity of Web services, and user participation and customization in the Web. Certain types of these Web services based systems and applications require Web services to be aware of context associated with their operations. Such context could be associated with, for example, time, location, profile, and runtime status of services, things, and people. Obviously, such context could be individually handled by a single service and collectively processed by different services in multi-organizational environments.

From our previous study in [42], we have observed great challenging issues for supporting context-aware Web services. In our work, a context-aware Web service is a smart Web service which, defined by Manes, ”can understand situational context and can share that context with other services” [31]. Being smart or context-aware is important for Web services because they could not only effectively match user’s needs to their capabilities, but also be able to adapt themselves with situation changes to improve their availability and reliability. As we discussed in [42], when systems and applications are built from different Web services provided by and hosted in multiple organizations, it is challenging to make the systems and applications context-aware, as this requires services to be aware of each other and aware of the context of customers and applications. This challenge is due to the distributed, large-scale, and diverse nature of Web service-based environments. Unlike past context-aware

[*]This paper is a revised version of the paper entitled *Context Coupling Techniques for Context-aware Web Service Systems - An Overview* which is accepted for the book entitled *Enabling Context-Aware Web Services: Methods, Architectures, and Technologies, Eds: Michael Sheng, Jian Yu, Schahram Dustdar, (c) Chapman and Hall/CRC, 2010.*

systems in which components are tightly coupled and in a closed environment, such as [48, 38], as indicated by [31, 33, 43] and others, solutions for enabling context sharing in Web services-based environments must be open and interoperable to ensure that they can be applicable in Web service-based systems.

To support the concept of context-aware Web services, distributed Web services in a context-aware Web service system or in a service composition need to couple context information from different sources in order to be able to interpret context information. Such interpretation is a prerequisite before services can adapt themselves, a condition to be context-aware. However, context coupling in context-ware Web service systems is a complex issue. It is related not only to how we represent and transfer context information among Web services but also to how we provide context information to different interesting services and how we ensure security and privacy issues of context information. Context coupling techniques for Web service systems will be different from that of tightly coupled systems because, with Web service systems, context information is typically shared across the boundaries of organizations which host Web services.

In order to examine this issue, this chapter aims at presenting an overview of context coupling techniques, their related issues, and their implication to the success of a context-aware Web service system. Our approach is to study existing techniques implemented in current systems, to present open context coupling issues in current and future context-aware Web service systems, and to suggest further research directions. To this end, in Section 2 we will discuss fundamental assumptions, present scenarios, and explain what context coupling means. From well-establish coupling techniques, we concentrate our study on four different models, named structure, data, message and common couplings, which in turn strongly impact the design of existing context-aware Web services. In Section 3, we study how existing context-aware Web service systems support context coupling techniques and analyze strength and weakness of existing techniques. To further analyze how context coupling techniques are implemented in a real-world scenario, we present a case study based on the EU FP6 inContext project. Based on our analysis, we discuss some open issues and suggest few recommendations for future research in Section 5. Related work of this study is also given in Section 6 and we conclude the chapter with an outline of some of our future steps in Section 7.

2 Fundamental Concepts

2.1 Context-aware Web Services

What context information and context-aware systems are, has been defined and discussed in various papers [13, 19, 23, 17] . Context information is dependent on individual systems, as a type of information might be considered as context information in one system but not in another one. Context-aware Web services are a subtype of context-aware systems defined in these papers. As given in [42], we consider context information as any additional information that can be used to improve the behavior of a service in a situation. Without such additional information, the service should be operable as normal but with context information, it is arguable that the service can operate better or more appropriately.

A Web service might be context-aware as it can adapt its operations according to context of its clients and environment. Naturally, context-aware Web services sharing context information will operate in a distributed environment. However, it might never need to share context information, e.g., to other services it depends on. This case is not the focus of the environment we assume in this study. Instead, we consider the case in which different services will share context information. We also consider whether these services belong to the same (virtual) organization or not. By (virtual) organization, we mean that these services will follow certain policies established and enforced by the same (virtual) organization, such as security, privacy and data governance. For example, when services are provided by the same company, they might not worry about how sensitive the context information transferred among the company's services is, thus the services might relax some privacy conditions which must be implemented when the services do not belong to the same organization.

2.2 What is Context Coupling?

Before we discuss context coupling, let us consider the following scenarios. The first scenario is described in Figure 1. Assume that a user would like to use a PDA (Personal Digital Assistant) which is equipped with a GPS (Global Positioning System) to find relevant restaurants with/without open gardens when the user is in

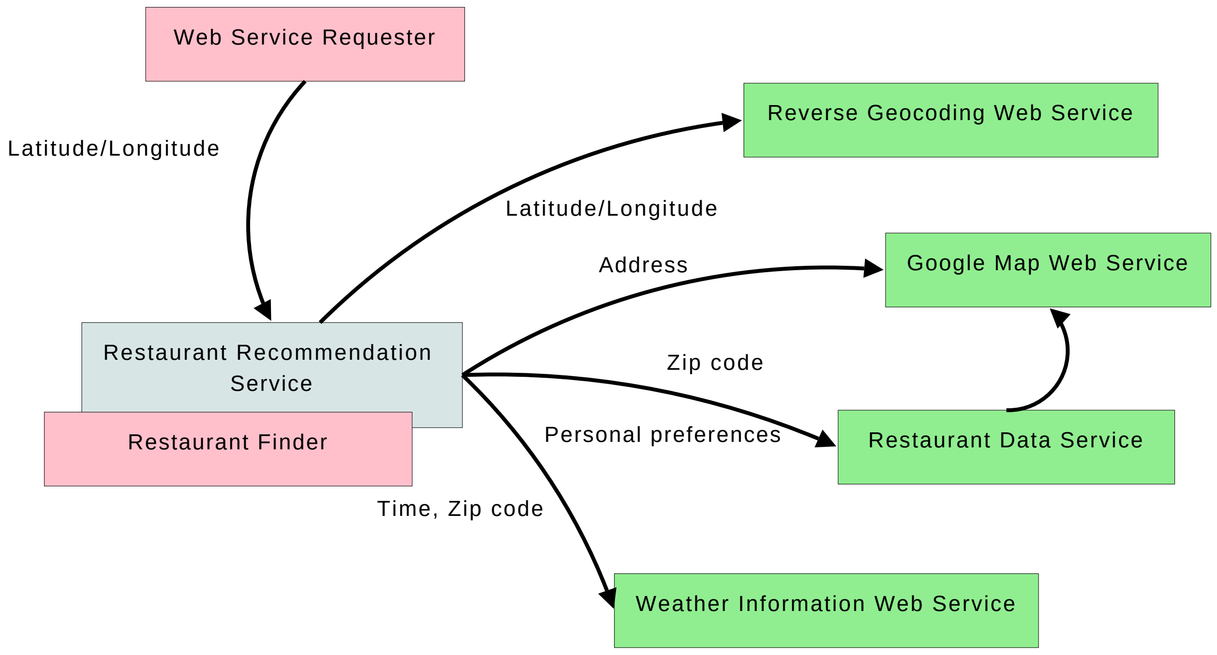

Figure 1: Example of context information sharing

Vienna[1]. To date, all information relevant to this search could be provided by Web services and this user's need could be fulfilled by different means. For example, a `Restaurant Recommendation Service` could offer recommendations to restaurants based on several criteria. The `Restaurant Recommendation Service` can also utilize various other services, such as the `Reverse Geocoding` for mapping GPS information to addresses, the `Google Map` for finding businesses close to an address, the `Restaurant Data Service` for searching restaurants based on user preferences, and the `Weather Information Service` for obtaining weather information. Many types of context information could be exchanged in this scenario. First, the user's PDA has GPS, thus user's location (latitude/longitude) information can be captured and utilized. In a simple form, in order to find a restaurant, a `Restaurant Finder` in the PDA would, for example, be implemented as a Web service requester which directly invokes other services, e.g., by utilizing mashup techniques to aggregating content from different services. The `Restaurant Finder` can automatically obtain GPS information and send this information to the `Reverse Geocoding Web service` to get the address associated with the user's location. Then it uses the address to call the `Google Map Web service` to find restaurants close to the address. Furthermore, it uses the zip information to get the weather information and then recommends restaurants with gardens if the weather is nice. In this case the `Restaurant Finder` has to model and couple all context information used for finding restaurants. In another possibility, a more complex form, the user just uses the `Restaurant Recommendation Service` which is provided by a service provider because the provider can get more benefits (e.g., acquiring more user information or providing the service as an value-added offer). This service requires the user to provide only one parameter which is restaurant search command, but it manages several user-related context information. When the user uses this service, the user's `Web service requester` may automatically pass the user's GPS information to the service via SOAP headers. This service then utilizes the user's location to determine the corresponding address, thus being able to locate relevant restaurants based on the address. Furthermore, it can pass user preferences/behaviors, which it has in its database, to the `Restaurant Data Service` in order to obtain a better match of restaurants to user's profile.

In this scenario, possible types of context information are, for example, location (latitude/longitude), time (when the request is issued), weather status, and personal preferences. Depending on capabilities of services, specific configurations, and software development processes, not all of these information might be used. For example, when the weather status is unknown, it is probably hard to recommend an outdoor restaurant. These types of context information are shared between different clients and services spanning different organizations. Sharing methods are different, depending on available services and compositions. Therefore, some fundamental questions

[1]Although this scenario is imaginary, restaurant recommendation is a popular scenario which can be found in many documents.

about sharing mechanisms or how context is coupled, arise. For example, how latitude/longitude information is transferred to the `Restaurant Recommendation Service` (using parameter invocation or embedded information in a SOAP header)? Does `Restaurant Recommendation Service` manage an ontology and instance information of context information about location, time, weather, etc.? Will the `Restaurant Recommendation Service` pass a structure of personal preferences to the `Restaurant Data Service` or it just gives a link to a person's preferences? If a link is passed, how can we sure that sensitive information, such as user identity, is not accessed by the `Restaurant Data Service`? Generally speaking, as a Web service may manage, process and transfer different types of context information, it has to couple different types of context. How can it deal with this? Which techniques should be used? There are many questions and this chapter aims at answer some of them by studying current state of the art systems. Besides many possible context information might be shared, this sharing is also conducted across different services which are not designed to work together on purposes and may be deployed in different geographic locations. This scenario reflects the case of Web services belonging to different organizations which share context information. Context coupling techniques have to work with the assumption that the policies are governed by different laws, countries, and distributed environments.

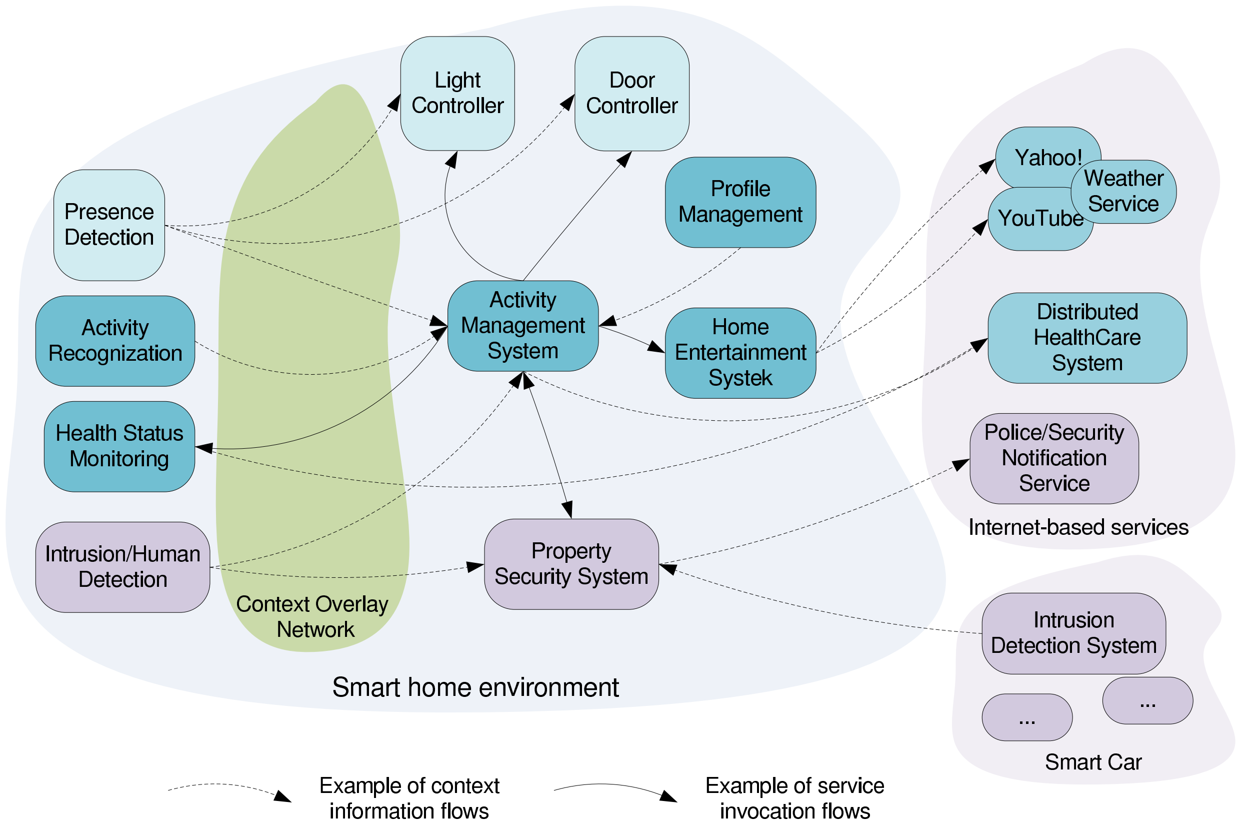

Figure 2: Context providers and consumers in Clara's home and other spaces

The second scenario is about a smart home, shown in Figure 2, and is based on the EU FP7 SM4All project [10]. In Clara's home, there is a system that can operate lights and doors automatically based on the simple presence information so she has bought this system and installed it in her home. Unfortunately, she got an accident and she is now on a wheelchair but she is doing some rehabilitation every day. Thus, she needs more help in order to deal with other difficulties during her rehabilitation. First, she is equipped with a health monitoring system which will connect to a distributed health-care system managed by her caretakers. Second, she installs a wearable activity recognition system which can detect her gesture. This activity recognition system can interact not only with her light and door controllers but also with the distributed health-care system. Furthermore, to support her rehabilitation, a home multimedia system is installed. This system can be controlled by her activity recognition system and interact with external multimedia services provided by, for example, Yahoo, Youtube, and a weather service. Since intruder breaks are increasing, she also install sensors to detect intruders and to control the alarm process, including calling the police, turning on/off lights, and opening/closing doors. This human recognition system also allows her to control the access to her house (automatic block or access). Furthermore, her other properties, such as her car, are also monitored and monitoring information will influence her activities.

This illustrative scenario presents many possibilities in exchanging and consuming context information. For example, to ensure that she selects suitable multimedia contents according to her health status, health status information can be used to filter the content. To ensure that external multimedia services offer best selections for a patient, her profile and status can be used to optimize the content delivery. Furthermore, when she has a severe health condition or based on her gesture, health caretakers will be informed and doors are automatically opened when they enter her house based on pre-defined profiles. This scenario shows that different types of context information are coupled over the time, For example, at the beginning, in Clara's home, we may have only a presence sensor, a service to turn lights (built based on a light controller) and a service to open/close doors. Then, as an activity recognition system is installed, we have wearable sensors, an activity management system, and a user profile management system. This activity management system will have to interact with light/door controllers and planned multimedia entertainment system. With this scenario, we see that (1) a context model covering many context sources cannot be pre-defined due to the diversity and opportunistic usage of context sources, (2) context middleware cannot assume all context sources use the same protocol (e.g., push or pull) to provide context information. Thus, they raise the questions of how context information could be encapsulated and propagated over such a network of services.

In this paper, context coupling refers the degree how a Web service/client relies on another Web service/client with respect to context information. In our view, context coupling is a special topic of *coupling* [2, 1]. Understanding context coupling techniques is not only useful for selecting suitable techniques to transfer context but also for ensuring concerns associated with context information, such as privacy, to be guaranteed.

2.3 Basic Models of Context Coupling in Context-aware Web Services

While there are many forms of coupling, in this study of context-aware Web services, we divide context coupling models into the following types:

- *Context data coupling*: it is a type of data coupling in which context information is passed, as simple parameters, through Web service invocations.
- *Context structure coupling*: it is a type of stamp coupling, also called type use coupling. In this type of coupling, context data is described by a data structure which is passed among services through Web service invocations.
- *Context common coupling*: it is a type of common coupling in which context data is stored in a common space (e.g., storage or service) and Web services access the context information through the common space. The common space could be also extended to a distributed space (e.g., a distributed system of repositories).
- *Context message coupling*: it is a type of message coupling in which context-aware Web services use messages to transfer context information. It can be built based on publish-subscribe and P2P (Peer-to-Peer) systems.

The main reason for considering only four types of context coupling is because we observed those as the main types used existing systems. Other coupling models are either not used in or not applicable for context-aware Web service systems. Note that these types of coupling are not orthogonal, because, as we discuss them in Section 3, one type of coupling might use another one.

Context coupling techniques are related to several other techniques in a context-aware system, such as context representation, context dissemination, and context storage. For example, context representation describes how context data is structured. Thus, it has a strong impact on context structure coupling. Context dissemination protocols describe which protocols are used to conduct context message coupling. Context storage can be part of coupling techniques, for example, context message coupling can be performed via centralized context storage. A Web service entity stores context in the storage which is then accessed by another. In addition, context coupling techniques for context-aware Web services deal with the transfer of context information of which many types are sensitive, such as user information and location. Therefore, security and privacy issues in context sharing are of paramount importance [14, 34]. Thus, when studying context coupling techniques, we will also discuss possible privacy and security issues. However, we will not discuss how Web services will utilize context information to become "smart" (refer to [42] for context adaptation techniques, for example).

3 Context Coupling Techniques in Current Context-aware Web Service Systems

3.1 Structure and Data Coupling

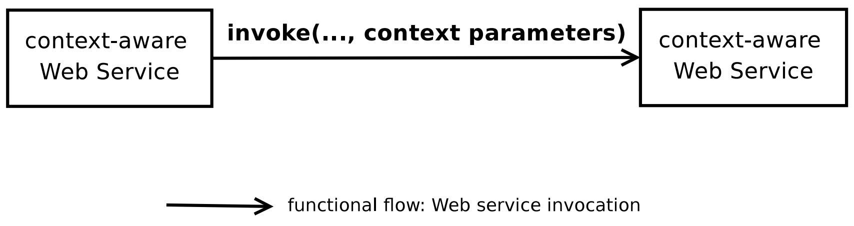
context-aware Web Service
invoke(..., context parameters)
context-aware Web Service
functional flow: Web service invocation

Figure 3: Data coupling model

In structure and data coupling techniques, context information is described in structured and/or simple parameters. A context-aware Web service will explicitly pass context information to another context-aware Web services by invoking appropriate service invocations. Figure 3 describes the interaction between two context-aware Web services supporting structure and data coupling techniques in which `context parameters` can be simple or structured input variables. Since context information is shared through service invocation, a context-aware Web service has to model and express its input context parameters explicitly in its service operations, although it is possible that values of these parameters might never be set in service invocations.

One technique to support such couplings is to develop context-aware Web services based on the Model-driven Engineering (MDE) approach. The MDE approach is increasingly used for structure and data coupling of context information in various context-aware Web services systems. The basic idea is to model the context parameters and their structures explicitly and to associate them with Web services descriptions. Examples of systems utilizing this way are CoWSAMI [16], ContextUML[39], CSOA [46], and their applications and works built atop them. Another technique of structure and data coupling is to develop/utilize agreed context models to pass context information. In this case, mostly a context model is developed based on ontologies [22, 43] or XML schemas [44], and context-aware Web services just provide service operations that accept context information following the model. Typically, this case of structure and data coupling is utilized together with common coupling techniques (see Section 3.3).

Using structure and data coupling techniques, a developer of a callee context-aware Web service will focus on describing what his/her context-aware Web services need and let his/her caller deal with the integration issue. Thus, it is more flexible with respect to the development process, in particular, when context-aware Web service systems are built from existing ones. For example, it is relatively straightforward to compose several context-aware Web services because their operations are known. However, when context parameters are changed, the composition has to be reworked. When a context-aware Web service needs to be improved, e.g., able to handle a new type of context information, its developer freely performs his/her work, but the developer of the composition has to rebuild the composition. Thus, it is also not suitable for large-scale context-aware systems, e.g., envisaged in the Internet of Services. Because in such a Internet, diverse types of context sources exist and modeling all of them is very difficult.

With respect to privacy and security issues, these types of couplings offers many possibilities to strongly enforce privacy and security controls. The caller has a total control over what kind of context information it could share. Security enforcement is the same as what is for other functional invocations of the service, for example WS-Security standards [11] can be used for SOAP-based context-aware Web services while HTTP Authentication [4] and HTTPS [5] can be used for REST-based context-aware Web services. Thus, security enforcement does not impose a large cost for security implementation for context information. Although, many systems studied do not deal with privacy issues or do not describe how they address privacy issues, data and structure coupling techniques could utilize MDE techniques and pre-/post-conditions assertion techniques to enhance the assurance of privacy issues (e.g., eliminating user identity from context parameters). Some basic work [28, 47] have been done in this direction. In the privacy-aware context profile [28], context is associated with owner and authorization for accessing context information can be associated with group or user. Thus, this model can be used to describe some privacy-related information. While privacy-related information can be described, technically, privacy compliance can only be performed at the caller side because there is no way to ensure that the callee will follow the same policy. In fact, the systems studied in this section do not describe how they ensure privacy compliance. Some work, such as [30], have presented the use of information about privacy concerns and policies to enforce the privacy compliance within Web services. However, the connection between context modeling (at design time) and privacy compliance (at runtime) in an end-to-end software engineering process is not well established in existing work.

3.2 Message Coupling

In context message coupling techniques, context information is exchanged via messages. A context-aware Web service passes context information to another Web service by sending a message consisting of context information or links to the information. The message can be relayed through a message middleware.

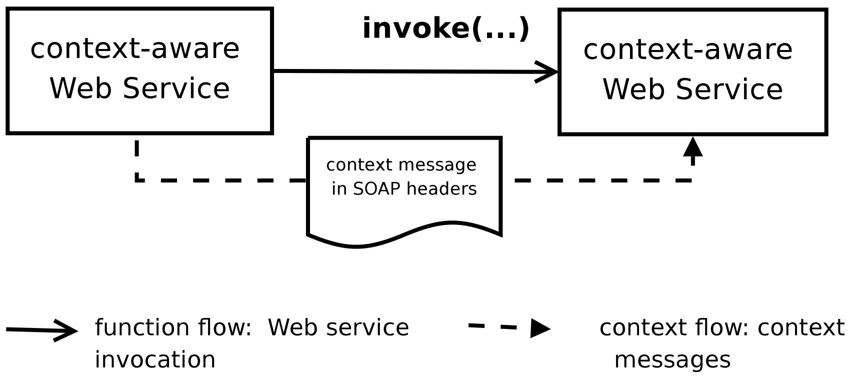
context-aware Web Service
invoke(...)
context-aware Web Service
context message in SOAP headers
function flow: Web service invocation
context flow: context messages

Figure 4: Model of context message coupling using SOAP headers

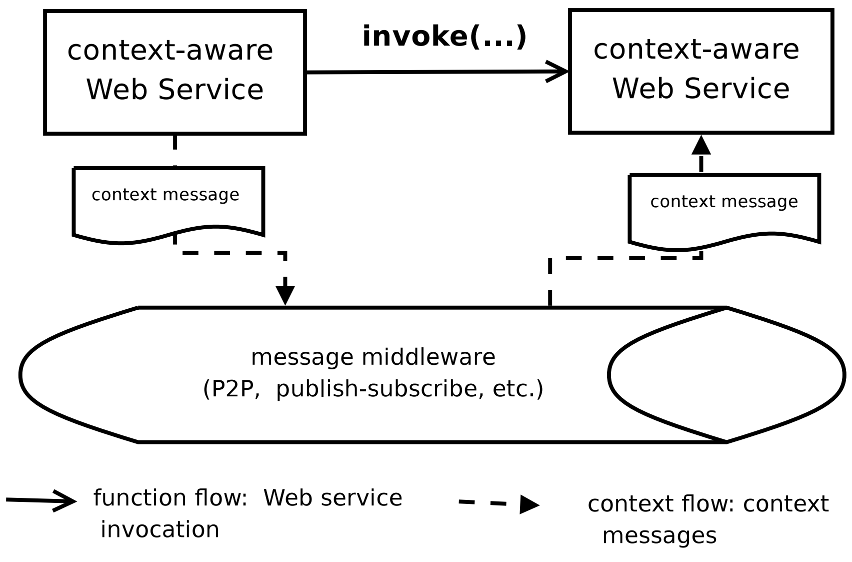
context-aware Web Service
invoke(...)
context-aware Web Service
context message
context message
message middleware (P2P, publish-subscribe, etc.)
function flow: Web service invocation
context flow: context messages

Figure 5: Model of context message coupling through message middleware

Using SOAP headers to transfer context information is one technique in this type. In this technique, context information or links to the information are encapsulated in SOAP header messages which will be transferred from a caller Web service to its callee Web service. As shown in Figure 4, context message is separated from messages of functional calls. Context messages can be directly sent from one service to another one (see Figure 4) or indirectly relayed through message middleware (see Figure 5). This technique has been proposed and implemented in several systems. Keild et al. present a generic framework to support the development of context-aware adaptable Web services [29]. This framework separates clients/Web services from the context framework which supports clients and services. The transfer of context information is performed through SOAP message header. Context information can be explicitly and directly processed by clients or Web services or be automatically handled by the context framework. Similarly, in the CASD (Context-aware Service Directory) system [21] SOAP headers are used to transfer context information which influences the operation of CASDs. In the inContext system [41], URIs (Uniform Resource Identifiers) specifying the location of context information are transferred and based on that context information can be obtained. The above-mentioned systems can actually be considered as particular cases that WS-Context [12] aims to support. WS-Context specifies mechanisms to transfer context messages among Web services, supporting transferring context messages as well as references to context messages. In the first case, context information is wrapped into a message and transferred among Web services using SOAP headers. In the second case, only the references are transferred. The real context information will be obtained from an external service (context manager). WS-Context, therefore, supports different message coupling techniques. Similar to the above-mentioned tools, when transferring messages, WS-context implementations will face similar issues. When transferring references, corresponding context information could be retrieved from dedicated repositories or services. There are some implementations utilizing WS-Context concepts, such as [35].

While, this technique does not require change of service interfaces, thus it will not require changes of service composition, it requires a mutual understanding between the caller and the callee. Context-aware Web services sharing similar context information have to know schema of the message in advance. Thus, it is suitable for service providers who have a strong agreement (e.g., are in the same organization). One advantage of this technique is that it supports very loosely coupling by means of messages. The use of references instead of message content allows a context-aware Web services to access context information from many context providers. However, this technique works only with SOAP-based Web services.

Another message coupling technique is to support the exchange context information through overlay networks (shown in Figure 5). There are many possibilities in this model. First, context-aware services may know the structure of the context information and just send different query or subscription requests over a network of services in order to obtain the context information. Second, context information may be published through an overlay network, possibly, of context management services. For example, the ERMHAN system [36] distributed context management services relay context information to a central context management service which then propagates selected context information met specific conditions to appropriate applications. Using messages to couple context information through overlay networks could also be useful for context-aware Web services which have a low temporal coupling degree, such as mobile Web services, because it does not require both services have to be present at the same time.

With respect to privacy and security issues, context providers have a flexibility to enforce which information to be shared and protected. Even though encryption can be applied to SOAP headers, most existing systems do not apply security methods to context information in SOAP headers. In message middleware for context coupling, security can be established based on WS-Security, HTTP Authentication and HTTPS. However, security enforcement could be complicated in case of transferring references because the context provider might not be the same as the caller (thus this requires some forms of security configuration between the caller, the provider and the callee). The systems in our study do not show how they solve this problem. In this respect, some protocols could be utilized for context-aware Web services. For example, the OAuth protocol [6] can be used for authorizing resource access in Web services. With the OAuth protocol, a user can grant service consumers to access user's private resources hosted in another service. Therefore, it can be used to grant access to privacy context information among context-aware Web services. The OpenID protocol [7] is another one that can be used for managing identities associated with the caller, the provider, and the callee when transferring context references is employed.

3.3 Context Common Coupling

The common coupling model for context information is widely used in practice, especially when Web service providers agree on context information to be shared, or when services are managed by a single organization. This type of coupling is typically achieved by using a common repository to store, manage, and provide context information. The repository can be implemented as a Web service, such as in [26, 15, 35, 27, 32, 8, 37, 3], a tuple space, such as in The Ubiquitous Semantic Space (USS) [40], and agents, such as in [18].

This model could serve as a fast solution for context sharing in Web services. The common repository can provide well-defined interfaces for storing, querying, and manipulating context information. In many cases, context common coupling techniques are implemented based on a centralized model. However, this type of coupling is not limited to a centralized model. For example, in the ESCAPE framework [44], context information is hosted

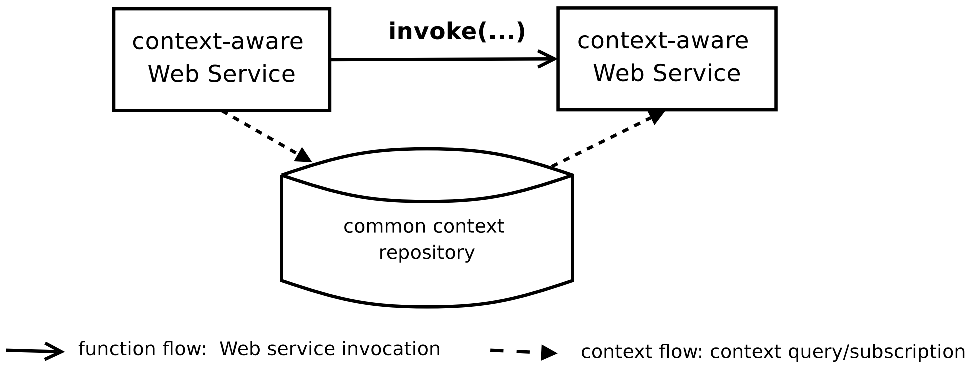
context-aware Web Service
invoke(...)
context-aware Web Service
common context repository
function flow: Web service invocation
context flow: context query/subscription

Figure 6: Common coupling model

by different context management services. In order to retrieve context information, various different languages and service operations can be utilized, depending on specific implementations and context data representations.

This model is also usually used together with other models in two main ways. In the first case, it can be used with message coupling techniques. Context-aware services exchange only URIs indicating the context information and the information is retrieved from the common repository. One example of this case is the inContext system [41]. In the second case, it can be used with context structure/data coupling. For example, in the CATIS system [37], the user's location is stored into a context manager. An application server serving user's request will retrieve the context information from the manager and passes the context information to relevant services.

With this type of couplings, in principle, Web services can exchange context information regardless of how the repository is implemented. However, when the repository is not based on Web services, such as in [18], from the integration point of view, there is a problem to support context-aware Web service systems comprising different services from different providers. One of main issues for this type of coupling is the performance and scalability of the common repository, in particular, when ontology-based repository is used. To overcome this issue, distributed repositories are proposed. Another issue is that all services have to agree on the context structure. Furthermore, querying context information will bring overheads and more interactions are needed in order to retrieve context information. On the other hand, with this model, context information is easily coupled and strong reasoning capabilities can be built. Moreover, historical context information might be stored for other purposes.

With respect to privacy and security issues, this type of coupling allows a strong control of accessing and sharing context data at a central point. In addition to transport security enforcement which can be based on common techniques, as discussed in other types of couplings, such as WS-Security, HTTP Authentication and HTTPS, context access control can be based on rules. Such rules can also be strongly related to privacy preferences. For example, the Google Latitude [3] supports different privacy policies for a particular instance or type of context information to a particular people/application. In [47], privacy preferences, privacy rules and context ownership information are used in the context manager to ensure the privacy-aware context sharing. In [26], access policies can be established based on user's time, location and event participation. Overall, this common repository model allows utilizing rich semantic representations for context information and well-integration of access control rules.

3.4 Summary

Table 1 presents the summary of how systems studied utilize context coupling techniques. Overall, structure, data and common coupling techniques are widely employed in current context-aware Web services. It is understandable because it is fast and easy to perform the integration with these techniques. Furthermore, they allow the developer to be flexible in developing their services without worrying much on common agreement in sharing context information. Until now only WS-Context is proposed as a standard for exchanging context in Web services, but it is not widely adopted yet.

In general, security and privacy issues are not well addressed in studied context-aware systems. Some of systems studied provide authentication and authorization access controls, but most of them neglect privacy issues. This is probably due to the fact that many types of context information in these systems are (i) not classified as private and sensitive information, and (ii) actually used only within a single organization. However, these issues have to be addressed when context coupling is performed via open, Internet-wide context-aware services.

Systems	Types of Coupling				Privacy	Multi-organization
	structure	data	common	message		
MDE approach	+	+				
inContext [41]	+			+		+
WS-Context [12]				+		+
USS [40]		+				+
CATIS [37]		+	+			+
CASD [21]				+		
ESCAPE [44]	+		+			
CA-SOA [20]	+		+		+	
Akogrimo [8]			+			+
ConServ [26]			+		+	+
ERMHAN [36]				+		
Keild et al. [29]				+		

Table 1: Summary of context coupling techniques supported in existing context-aware Web services. We use **+** for a feature when a system explicitly supports the feature. When it is unknown or unclear (some systems mention a feature but do not show how they implement the feature), the feature was left blank.

Category	Types of Coupling			
	Structure	Data	Common	Message
Development flexibility	high	high	medium	low
Integration complexity	low	low	medium	high
Security enforcement effort	low	low	medium	high
Privacy enforcement effort	low	low	medium	high
Interoperability degree	low	low	high	high

Table 2: Perspectives on the utilization of context coupling techniques

Table 2 summarizes our perspectives on the utilization of context coupling techniques. Overall, structure/data and common couplings offer less complexity and effort for the implementation but the interoperability degree is low, while message coupling is complex but offers better interoperability degree.

4 A Case Study: Context Coupling in the inContext Project

The EU FP6 inContext project [9] aims at supporting highly dynamic forms of human collaboration such as Nimble (short-lived collaboration to solve emerging problems), Virtual (spanning different geographical place and having diverse professionals) and Mobile (collaboration with mobility capabilities) teams. Therefore, it deals with many types of context information. Furthermore, the inContext project proposes its solutions based on the SOA model: context-aware services enabling team interactions are built as Web services [43]. Thus, in the inContext project, various types of context coupling techniques are employed, namely context structure, context data, message, and common coupling techniques, and context-aware Web services are strongly supported.

As partially described in [43, 41], the inContext project proposes four core capabilities dealing with context coupling. Figure 7 describes these capabilities:

- Manage Structure: this capability is actually the context coupling based on the context structure technique. Within this capability, the inContext project describes and associates different types of context, such as team activities, user profile, resource information, etc.
- Create Correlation and Extract Correlation: these two capabilities are dealing with context message techniques. The inContext project uses SOAP headers to transfer URIs indicating context information among Web services. These capabilities provide libraries for handling context URIs extraction, propagation and correlation.
- Manage Context: this capability supports context common coupling. The inContext project provides tools for storing, managing and querying context information.

The first capability is realized at design-time context coupling whereas the last three capabilities are achieved through runtime context coupling.

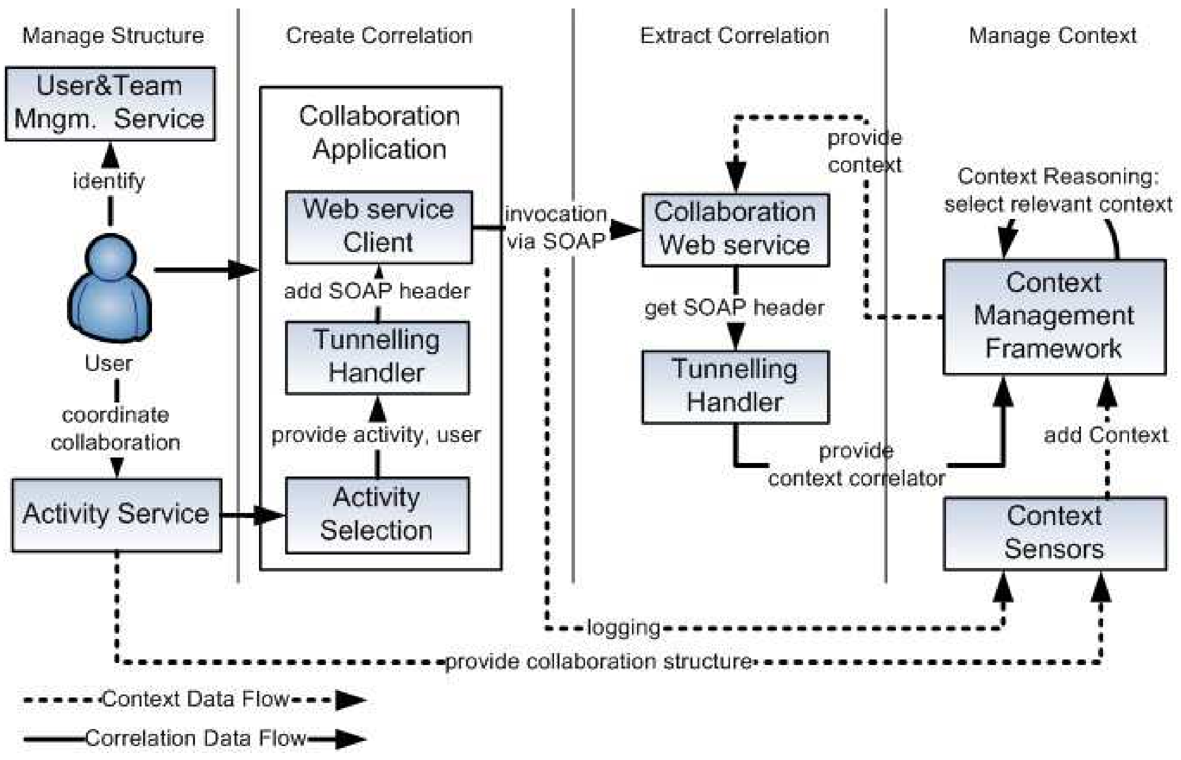

Figure 7: Required core capabilities for supporting context coupling in the inContext project [41]

4.1 Context Structure and Data Coupling

The approach to support context structure and data coupling techniques in the inContext project is to design an ontology describing all possible types of context information used in teamwork. To this end, the inContext project has reused many existing ontologies and developed new ones and linked them together in the so-called inContext context model. Figure 8 shows the inContext ontology that is able to seamlessly unify individual, team and activity context. Individual context includes most of the traditional context types such as location, available devices, communication online status, but also more collaborative work related information such as team membership, activities (within the different teams), available resources, skills, or team members (from different teams). Team context includes information about interactions, projects, organizations, and locations that are associated with members of a team. Activity context describes tasks and their associated information in different levels of detail, for example, work breakdown structures of a project or user current activities.

The inContext context model is described and implemented using the RDF (Resource Description Framework) and OWL (Ontology Web Language). The main advantage of the inContext approach is that it allows for flexibility and extensibility of the context model, for instance, by inclusion of domain-specific data or reuse of Web data already available in common RDF formats. Furthermore, the ontology-based model provides common coupling with reasoning capabilities that will be discussed in Section 4.3.

4.2 Message Context Coupling

Message context coupling techniques in the inContext project are developed based on the propagation of URIs indicating context information via SOAP headers. Unlike [29], only URIs specifying the location of context information are transferred. This assumes that by using the URIs, a context-aware Web service can retrieve context information from appropriate context providers. Figure 9 depicts the message context coupling implemented in the inContext project. URIs specifying context information, such as `ActivityURI` - for activity information, and

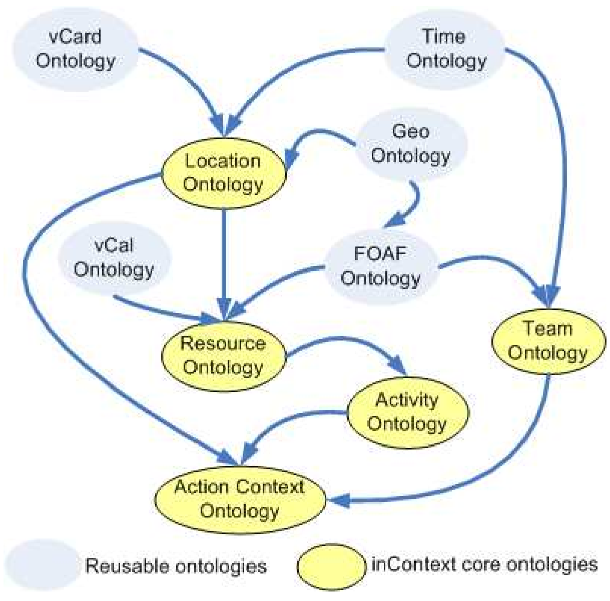

Figure 8: The inContext context model used to describe individual, team and activity context [43]

`UserURI` - for user identifiers, are embedded into SOAP headers in SOAP messages exchanged between context-aware Web services. A service will use the URI to access the context information which is stored in a separate service (e.g., a `Context Store` - see Section 4.3) in the inContext's `Context Management Framework`. Listing 1 presents a simplified example in which context information related to activity `act1` and user `Rossi` is transferred. The `ActivityURI` and `UserURI` are `http://www.in-context.eu/pcsa#act1` and `http://www.in-context.eu/pcsa#Rossi.E54`, respectively. The context information itself is stored in the `Context Store` of the `Context Management Framework`.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 <soapenv:Header>
   <ns1:ctxtunnelling soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"
       soapenv:mustUnderstand="0" xmlns:ns1="www.in-context.eu">
   <ns1:Activity>
      http://www.in-context.eu/pcsa#act1
   </ns1:Activity>
   <ns1:User>
      http://www.in-context.eu/pcsa#Rossi.E54
   </ns1:User>
  </ns1:ctxtunnelling>
 </soapenv:Header>
 <soapenv:Body>
 </soapenv:Body>
</soapenv:Envelope>
```

Listing 1: Simplified example of SOAP header message including context coupling information in the inContext project [41]

The approach of sending URI only facilitates the access to context information from any services. Thus it is possible to support runtime binding: each time a context can be accessed from a different service. Currently, the inContext project has not supported any privacy protection for this sharing.

4.3 Context Common Coupling

Context common coupling is achieved in the inContext project by means of a centralized `Context Store` which is implemented atop an RDF store with added OWL and RDFS inference capabilities. With a strong reasoning capability, the inContext approach offers better retrieval mechanisms for context-aware Web services.

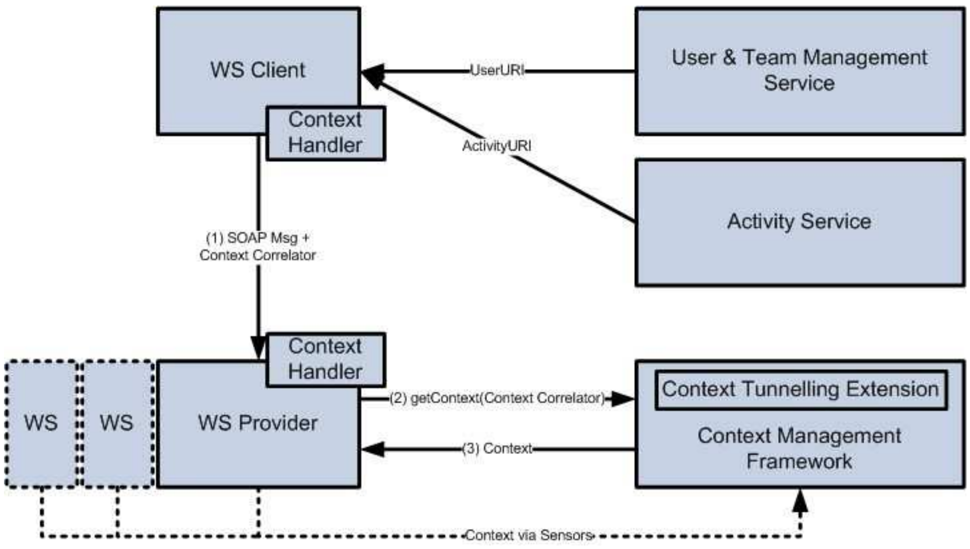
WS Client
Context Handler
UserURI
User & Team Management Service
ActivityURI
Activity Service
(1) SOAP Msg + Context Correlator
Context Handler
WS
WS
WS Provider
(2) getContext(Context Correlator)
(3) Context
Context Tunnelling Extension
Context Management Framework
Context via Sensors

Figure 9: Components and steps in the inContext's context message coupling [41]

However, as noted in [41], it requires Web services being able to process RDF content, which is not common in Web services yet, and is hard to achieve in mobile applications. To master this problem, the inContext project also provides transformation solution that returns the results in XML format. While offering a strong reasoning capability, the inContext's `Context Store` does not provide any privacy policies at the time of wring.

4.4 Summary

The inContext case study shows that for large scale and complex systems dealing with various types of context information, a single technique might not be enough. On the one hand, the inContext project employs structure and common coupling techniques together to support a context storage which can be used by any context-aware Web services. On the other hand, it utilizes message coupling to exchange context information. Using multiple types of couplings together could help to deal with the diversity of context-aware Web services in different situations and configurations. For example, the common context storage would be utilized by context-aware Web services which are deployed in strong platforms and require complex types of context information, possibly, inferred from other types of information. It is because the context storage has strong capabilities to support advanced reasoning. On the other hand, message coupling could be used to share context information to mobile Web services which are deployed in mobile devices or which require simple context information and minimum numbers of interactions. For example, a notification service just needs to be aware of if the user is available in an instant messaging service or not in order to use instant messages to notify the user with some information. This does not need to acquire complex context information using reasoning techniques. Combining structure, common and message couplings could be, for example, suitable for the scenario of smart homes in which there are multitude of varying services.

5 Open Issues and Recommendations

From our study, we draw some conclusions. First, software engineering techniques for building context-aware Web services should take into account privacy and security issues in the whole development and deployment of these services. On the one hand, privacy and security features should be provided to the user when context information being shared is sensitive to the user. This is particularly important for common coupling techniques. On the other hand, in case of using structure and data coupling techniques, constraints could be added into the design of context-aware Web services, e.g., based on the MDE approach, to automatically check and ensure privacy issues when context information is passed through service invocation.

Second, it is hard to achieve a common context model but the use of rich semantic representations, agreed concepts, and extensible message specification to describe context information would facilitate the context coupling in the Internet of Services. It could enhance the runtime context structure coupling by means of reasoning and enrichment. Using reasoning techniques one could correlate many different types of context information from a common repository and couple these types into a message specification.

Third, currently there is a lack of techniques to couple existing context information described in different specifications and provided by different sources during runtime. For example, in structure and common coupling, context information is mostly assumed to follow the same specification, forcing, for example, a new context-aware Web service to use the same specification for its sharing context information, even if its context information is described in a different form. Consider the scenario of smart homes in which each service (e.g., light control, entertainment, and activity recognition systems) has its own context model. Then, a newly-built context-aware Web service retrieves context information from existing services would need to solve many interoperability issues among context specifications.

Fourth, related to the third issue is the need to have open Internet-scale context exchange protocols and models. The WS-Context could be a good starting point but its assumption of transferring context information using SOAP would be a limitation as certain context-aware Web services will not be based on SOAP. We suggest to agree, like WS-Context proposed, that operations of obtaining context are well-defined. However, context messages could be transferred by different means based on SOAP and REST (Representational State Transfer). Furthermore, these protocols and models should also address the interoperability of different context-aware Web service systems. A similar approach to a recent work on bridging context management systems [24] could be investigated for context-aware Web services.

Fifth, existing protocols for authorizing resources access in Web services and for supporting identity management and single sign-on, such as the OAuth and OpenID protocols, could be useful for context coupling in a large-scale network of context-aware Web services. We suggest to integrate and extend these protocols for context coupling among context-aware Web services, especially those provided by different vendors.

6 Related Work and Further Reading

In our previous work [42], we have analyzed different techniques employed in context-aware Web services. This paper is a further step to detail how context information could be coupled. However as [42] gives a broad view of existing techniques, it also covers some systems studied in this chapter. Furthermore, in [42], we mainly study existing techniques based on context supporting components, such as context presentation, context sensing, context storage, context distribution and adaptation. In this chapter, we examine only context coupling techniques which are of course related to different context supporting components. Thus, some results found in [42] are also given in this chapter.

Coupling techniques are well-known [2, 1], thus they have presented in several papers. When study coupling techniques in Web services, in general, and in context-aware Web service systems, in particular, we are able to find only four common types of coupling techniques namely structure, data, message and common couplings. Context coupling techniques are subset of these techniques, thus they have some common properties. However, to our best knowledge, there exists no study of context coupling techniques in context-aware Web services.

The interaction models between context-aware Web services in context coupling techniques are also related to SOA and enterprise integration patterns [25]. Thus generic concerns associated with these models with respect to integration issues could also be learned from these patterns. However, these patterns are generic, while using them for coupling context information should be driven by the properties of and compliance rules applied to context information.

7 Conclusion

Enabling context-aware Web services, or “smart Web services”, is important as this will substantially improve how services could adapt to complex, situational behaviors of humans, things and services on the Internet. Coupling context information across multiple Web services is non-trivial problem because it requires well-agreed context models and protocols. In this chapter, we have studied a particular topic - context coupling - that is a must for enabling context-aware Web services. We have analyzed existing systems and presented a case study. Overall, many open issues have not been addressed yet in current literature. This is understandable because context-aware Web services research is a relatively new. We need to invest more efforts on establishing open protocols for exchanging context information in a large-scale system. Furthermore, privacy and security issues for context coupling should be treated as first entities.

Our future work is to focus on message specifications that can be used to transfer different types of context information and open protocols for context coupling over overlay networks. In particular, we will target our work in multiple spaces connecting smart homes to distributed healthcare systems and Internet-based Web services.

Acknowledgments

This work has been partly funded by the European Commission through the EU FP6 inContext (Interaction and Context-based Technologies for Collaborative Teams) [9] and the EU FP7 SM4All (Smart Homes for All) [10] projects. We thank all inContext members for their contribution on the discussion and the implementation of the context coupling in inContext which serves as the case study in this chapter. We thank Fei Li for fruitful discussion on open issues and the scenario of smart home which is built based on the SM4All research. inContext's coupling techniques are partially published in [41]. We thank Christoph Dorn, Giovanni Casella, Axel Polleres, and Stephan Reiff-Marganiec for their contribution in [41] which is extracted for the case study in this chapter.

References

[1] Coupling. `http://www.site.uottawa.ca:4321/oose/coupling.html`. Last access: 27 May 2009.

[2] Coupling (computer science). `http://en.wikipedia.org/wiki/Coupling_(computer_science)`. Last access: 27 May 2009.

[3] Google latitude. `http://www.google.com/latitude/intro.html`.

[4] HTTP Authentication: Basic and Digest Access Authentication. `http://tools.ietf.org/html/rfc2617`. Last access: 24 July 2009.

[5] HTTP Over TLS. `http://tools.ietf.org/html/rfc2818`. Last access: 24 July 2009.

[6] OAuth. `http://oauth.net`. Last access: 24 July 2009.

[7] OpenID. `http://openid.net`. Last access: 24 July 2009.

[8] The Akigrimo project. `http://www.mobilegrids.org/`.

[9] The inContext project,`http://www.in-context.eu`.

[10] The SM4All project: Smart Homes for All, `http://www.sm4all-project.eu`.

[11] WS-Security. `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss`. Last access: 24 July 2009.

[12] Web services context specification (ws-context) version 1.0. `http://docs.oasis-open.org/ws-caf/ws-context/v1.0/wsctx.html`, April 2007. Last access: 27 May 2009.

[13] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In Hans-Werner Gellersen, editor, *HUC*, volume 1707 of *Lecture Notes in Computer Science*, pages 304–307. Springer, 1999.

[14] Mark Ackerman, Trevor Darrell, and Daniel Weitzner. Privacy in context. *Hum.-Comput. Interact.*, 16(2):167–176, 2001.

[15] Chimay J. Anumba and Zeeshan Aziz. Case studies of intelligent context-aware services delivery in aec/fm. In Ian F. C. Smith, editor, *EG-ICE*, volume 4200 of *Lecture Notes in Computer Science*, pages 23–31. Springer, 2006.

[16] Dionysis Athanasopoulos, Apostolos V. Zarras, Valerie Issarny, Evaggelia Pitoura, and Panos Vassiliadis. Cowsami: Interface-aware context gathering in ambient intelligence environments. *Pervasive Mob. Comput.*, 4(3):360–389, 2008.

[17] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *International Journal of Ad-Hoc and Ubiquitous Computing*, Jan 2006.

[18] Harry Chen Baltimore and Harry Chen. Semantic web in a pervasive context-aware architecture. In *In Artificial Intelligence in Mobile System 2003 (AIMS 2003), in conjuction with Ubicomp*, pages 33–40, 2003.

[19] Guanling Chen and David Kotz. A survey of context-aware mobile computing research. Technical report, Hanover, NH, USA, 2000.

[20] Irene Y. L. Chen, Stephen J. H. Yang, and Jia Zhang. Ubiquitous provision of context aware web services. In *SCC '06: Proceedings of the IEEE International Conference on Services Computing*, pages 60–68, Washington, DC, USA, 2006. IEEE Computer Society.

[21] Christos Doulkeridis, Vassilis Zafeiris, Kjetil Norvag, Michalis Vazirgiannis, and Emmanouel A. Giakoumakis. Context-based caching and routing for p2p web service discovery. *Distrib. Parallel Databases*, 21(1):59–84, 2007.

[22] Tao Gu, Hung Keng Pung, and Da Qing Zhang. A service-oriented middleware for building context-aware services. *J. Netw. Comput. Appl.*, 28(1):1–18, 2005.

[23] Karen Henricksen, Jadwiga Indulska, Ted McFadden, and Sasitharan Balasubramaniam. Middleware for distributed context-aware systems. In Robert Meersman, Zahir Tari, Mohand-Said Hacid, John Mylopoulos, Barbara Pernici, Özalp Babaoglu, Hans-Arno Jacobsen, Joseph P. Loyall, Michael Kifer, and Stefano Spaccapietra, editors, *OTM Conferences (1)*, volume 3760 of *Lecture Notes in Computer Science*, pages 846–863. Springer, 2005.

[24] Cristian Hesselman, Hartmut Benz, Pravin Pawar, Fei Liu, Maarten Wegdam, Martin Wibbels, Tom Broens, and Jacco Brok. Bridging context management systems for different types of pervasive computing environments. In *MOBILWARE '08: Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications*, pages 1–8, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[25] Gregor Hohpe and Bobby Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[26] Gearoid Hynes, Vinny Reynolds, and Manfred Hauswirth. Enabling mobility between context-aware smart spaces. In *The 5th International Symposium on Web and Mobile Information Services, IEEE Workshop at IEEE international conference on advanced information networking and applications*, Bradford, England, 2009.

[27] Carlos R. E. Arruda Jr., Renato Bulcão Neto, and Maria da Graça Campos Pimentel. Open context-aware storage as a web service. In *Middleware Workshops*, pages 81–87. PUC-Rio, 2003.

[28] Georgia M. Kapitsaki and Iakovos S. Venieris. PCP: privacy-aware context profile towards context-aware application development. In *iiWAS '08: Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*, pages 104–110, New York, NY, USA, 2008. ACM.

[29] Markus Keidl and Alfons Kemper. A framework for context-aware adaptable web services. In Elisa Bertino, Stavros Christodoulakis, Dimitris Plexousakis, Vassilis Christophides, Manolis Koubarakis, Klemens Böhm, and Elena Ferrari, editors, *EDBT*, volume 2992 of *Lecture Notes in Computer Science*, pages 826–829. Springer, 2004.

[30] Zakaria Maamar, Qusay H. Mahmoud, Nabil Sahli, and Khouloud Boukadi. Privacy-aware web services in smart homes. In Mounir Mokhtari, Ismail Khalil, Jérémy Bauchet, Daqing Zhang, and Chris D. Nugent, editors, *ICOST*, volume 5597 of *Lecture Notes in Computer Science*, pages 174–181. Springer, 2009.

[31] Anne Thomas Manes. Enabling open, interoperable, and smart web services - the need for shared context, April 2001. `http://www.w3.org/2001/03/WSWS-popa/paper29`.

[32] Renato Bulco Neto, Carlos Jardim, Jos Camacho-Guerrero, and Maria da Graa Pimentel. A web service approach for providing context information to cscw applications. *Web Congress, Joint Conference Brazilian Symposium on Multimedia and the Web & Latin America*, 0:46–53, 2004.

[33] Katsumi Nihei. Context sharing platform. *NEC Journal of Advanced Technology*, pages 200–204, 2004.

[34] C. O'Driscoll. Privacy in context: Privacy issues in ubiquitous computing applications. pages 827–837, Nov. 2008.

[35] Sangyoon Oh and Geoffrey Fox. Optimizing web service messaging performance in mobile computing. *Future Generation Comp. Syst.*, 23(4):623–632, 2007.

[36] Federica Paganelli, Emilio Spinicci, and Dino Giuli. Ermhan: a context-aware service platform to support continuous care networks for home-based assistance. *Int. J. Telemedicine Appl.*, 2008(5):1–13, 2008.

[37] Ariel Pashtan, Remy Blattler, Andi, Andi Heusser, and Peter Scheuermann. Catis: A context-aware tourist information system, 2003. `http://www.ece.northwestern.edu/~peters/IMC.CATIS.pdf`.

[38] Manuel Romn, Christopher Hess, Renato Cerqueira, Anand Ranganathan, Roy H. Campbell, and Klara Nahrstedt. A middleware infrastructure for active spaces. *IEEE Pervasive Computing*, 1(4):74–83, 2002.

[39] Quan Z. Sheng and Boualem Benatallah. Contextuml: A uml-based modeling language for model-driven development of context-aware web services development. In *ICMB '05: Proceedings of the International Conference on Mobile Business*, pages 206–212, Washington, DC, USA, 2005. IEEE Computer Society.

[40] R. Sudha, M. R. Rajagopalan, M. Selvanayaki, and S. Thamarai Selvi. Ubiquitous semantic space: A context-aware and coordination middleware for ubiquitous computing. In *COMSWARE*. IEEE, 2007.

[41] Hong-Linh Truong, Christoph Dorn, Giovanni Casella, Axel Polleres, Stephan Reiff-Marganiec, and Schahram Dustdar. incontext: On coupling and sharing context for collaborative teams. In *Proceedings of the 14th International Conference of Concurrent Enterprising (ICE 2008)*, pages 225–232, 2008.

[42] Hong-Linh Truong and Schahram Dustdar. A survey on context-aware web service systems. *International Journal of Web Information Systems*, 5(1):5–31, 2009.

[43] Hong-Linh Truong, Schahram Dustdar, Dino Baggio, Stephane Corlosquet, Christoph Dorn, Giovanni Giuliani, Robert Gombotz, Yi Hong, Pete Kendal, Christian Melchiorre, Sarit Moretzky, Sebastien Peray, Axel Polleres, Stephan Reiff-Marganiec, Daniel Schall, Simona Stringa, Marcel Tilly, and HongQing Yu. incontext: A pervasive and collaborative working environment for emerging team forms. In *SAINT*, pages 118–125. IEEE Computer Society, 2008.

[44] Hong Linh Truong, Lukasz Juszczyk, Atif Manzoor, and Schahram Dustdar. Escape - an adaptive framework for managing and providing context information in emergency situations. In Gerd Kortuem, Joe Finney, Rodger Lea, and Vasughi Sundramoorthy, editors, *EuroSSC*, volume 4793 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 2007.

[45] Georgios Tselentis, John Domingue, Alex Galis, Anastasius Gavras, David Hausheer, Srdjan Krco, Volkmar Lotz, and Theodore Zahariadis, editors. *Towards the Future Internet - A European Research Perspective*. IOS Press, May 2009. ISBN 978-1-60750-007-0.

[46] Samyr Vale and Slimane Hammoudi. Towards context independence in distributed context-aware applications by the model driven approach. In *SIPE '08: Proceedings of the 3rd international workshop on Services integration in pervasive environments*, pages 31–36, New York, NY, USA, 2008. ACM.

[47] Ryan Wishart, Karen Henricksen, and Jadwiga Indulska. Context privacy and obfuscation supported by dynamic context source discovery and processing in a context management system. In Jadwiga Indulska, Jianhua Ma, Laurence Tianruo Yang, Theo Ungerer, and Jiannong Cao, editors, *UIC*, volume 4611 of *Lecture Notes in Computer Science*, pages 929–940. Springer, 2007.

[48] Hao Yan and Ted Selker. Context-aware office assistant. In *IUI '00: Proceedings of the 5th international conference on Intelligent user interfaces*, pages 276–279, New York, NY, USA, 2000. ACM.