Handbook of Research on Mobile Multimedia

Second Edition

Ismail Khalil Ibrahim
Johannes Kepler University Linz, Austria

Volume II

Information Science REFERENCE

INFORMATION SCIENCE REFERENCE

Hershey · New York

Director of Editorial Content: Kristin Klinger
Director of Production: Jennifer Neidig
Managing Editor: Jamie Snavely
Assistant Managing Editor: Carole Coulson
Typesetters: Jeffrey Ash and Michael Brehm
Cover Design: Lisa Tosheff
Printed at: Yurchak Printing Inc.

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue, Suite 200
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

and in the United Kingdom by
Information Science Reference (an imprint of IGI Global)
3 Henrietta Street
Covent Garden
London WC2E 8LU
Tel: 44 20 7240 0856
Fax: 44 20 7379 0609
Web site: http://www.eurospanbookstore.com

Library of Congress Cataloging-in-Publication Data

Handbook of research on mobile multimedia / Ismail Khalil Ibrahim, editor. -- 2nd ed.
p. cm.
Includes bibliographical references and index.
Summary: "The book is intended to clarify the hype, which surrounds the concept of mobile multimedia through introducing the idea in a clear and understandable way, with a strong focus on mobile solutions and applications"--Provided by publisher.
ISBN 978-1-60566-046-2 (hardcover) -- ISBN 978-1-60566-047-9 (ebook)
1. Mobile communication systems. 2. Wireless communication systems. 3. Multimedia systems. 4. Multimedia communications. 5. Mobile computing. I. Ibrahim, Ismail Khalil.
TK6570.M6H27 2008
384.3'3--dc22
2008013114

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter XLIX
Context Aware Collaborative Working Environments

Stephan Reiff-Marganiec
University of Leicester, UK

Yi Hong
University of Leicester, UK

Hong Qing Yu
University of Leicester, UK

Schahram Dustdar
Vienna University of Technology, Austria

Christoph Dorn
Vienna University of Technology, Austria

Daniel Schall
Vienna University of Technology, Austria

Hong-Linh Truong
Vienna University of Technology, Austria

Sebastien Peray
European Microsoft Innovation Center GmbH, Germany

Marcel Tilly
European Microsoft Innovation Center GmbH, Germany

Giovanni Giuliani
HP European Innovation Center, Italy

Christian Melchiorre
Softeco Sismat SpA, Italy

Simona Stringa
Softeco Sismat SpA, Italy

ABSTRACT

Collaborative Work Environments are software systems that allow teams, which are nowadays often distributed in location and organization to which they belong, to achieve certain projects or activities. In recent years, the available computer tools that can support such activities have grown; however, their integration is not necessarily achieved. Furthermore, users of such systems need to typically provide a large amount of setup information as the systems are not context-aware and hence cannot gather information about user activities in a simple way, and almost certainly will falter when the context of users changes. This chapter describes the inContext approach: a collection of novel techniques and a

reference architecture to support integration of tools and context information to provide collaborative work environments for the mobile worker of today. We will explore in detail how collaborative services are selected and how context is modeled, and consider the details of team forms.

EMERGING COLLABORATIVE SYSTEMS

People for a long time have conducted work in a collaborative manner; and of course, with a growing amount of service work, network connectivity, and software use, computers have started to play a greater role.

More recently, we are encountering an "always-on" ethic of many knowledge workers; that is, people want to be connected all the time, want to be able to check and receive e-mail, work as if they were in the office regardless of where they actually are, want to exchange documents in transparent ways, and so forth. Considering this view, software support for collaborative work needs to address a multitude of requirements; of course, basic collaboration functionality is essential, but this needs to be available in a context-aware manner that on the one hand provides the required transparency for a mobile workforce and on the other also supports the fact that individuals are generally part of many teams working on a multitude of projects simultaneously. These teams might span organizations, and teams might be of different forms as far as their longevity and other aspects are concerned.

Based on various criteria such as team goal, coupling, time span, and so forth, we classify emerging team forms into Nimble, Virtual, and Mobile teams (N/V/M teams). A nimble team quickly gathers to work on problems that may emerge unexpectedly. Team members can be distributed or collocated in terms of physical space. Team leadership is established in an ad hoc fashion, while peers may take up multiple roles simultaneously. Examples for nimble teams are task forces of specialists for crisis mitigation in health care (e.g., SARS) or scientists organizing a conference at a new location.

Virtual team members collaborate across geographical distance and organizational boundaries and have a somewhat stable team configuration with roles and responsibilities assigned to the team members. Exemplary virtual teams are technical consultants for a mechanical engineering project or a production team for a movie.

Members of nomadic teams are typically involved in several projects at the same time in a loosely coupled fashion.

As the name suggests, the concept and model of mobile teams aims to characterize and support team members that are highly mobile and frequently change their locations and move to different places where they may meet other collaborators. Collocation of peers, without being explicitly planned or scheduled, yields the need to opportunistically collaborate by exchanging data and artifacts in an ad hoc fashion. Experts in a political conflict resolution, musicians providing a composition of soundtracks, or actors providing stunt or dubbing services are some real-world examples.

Hence, modern collaborative working environments need to provide solutions for these issues. They should also be delivering increases in productivity; that is, they must support people in what they do and not introduce an extra burden. To that extent, they must integrate the existing tools of relevance that team members are using (be they public services or proprietary ones) and support complex forms of interactions occurring in the various team forms.

The research focus of the inContext project centers on how to exploit and combine novel techniques in the fields of context modeling and

reasoning, service management, interaction mining, and service-oriented architecture technologies to develop a novel pervasive collaborative working environment for emerging team forms. Those research fields are already well established, but their applications in CWEs are not well understood. Basic collaborative tools, such as document sharing, co-office, calendars, instant messaging, and so forth, provide the basic elements that are wrapped and integrated into the inContext environment. The inContext environment provides an architecture that follows service-oriented computing ideas, especially Web service technology, to provide loose and flexible coupling with on-demand binding of the relevant collaboration services. The inContext architecture allows easy composition and automatic selection of services to support demands of various teams.

The architecture makes use of a range of techniques to automatically select the most appropriate services based on the user's context. For this to work, enhancements have been made to the service management structures of SoA to rank services by suitability with the use context information. Also, development on the modeling and gathering of and reasoning about context information has addressed the specific requirements of collaborative teams.

This chapter presents details of the inContext architecture and then discusses in detail the three main components: context, collaboration, and service management. We round the chapter off with a small case study before concluding and making suggestions for future directions.

STATE OF THE ART AND RELATED WORK

Many groupware systems such as the file-oriented BSCW (Bentley, Horstmann, Sikkel & Trevor, 1995) and the virtual officelike Groove (www.groove.net) have emerged. These are usually rigid, tightly integrated systems, and while they address the collaboration functionalities required, they rarely consider context information about the user, activities, or tasks. On the other hand, systems such as the process-aware ad hoc collaboration system Caramba (Dustdar, 2004) allow activity-centric collaboration but lack the notion of service-oriented computing and a dedicated context management.

On the other hand, much work has been conducted in the area of context frameworks that are targeted at specific groups such as mobile users (Bardram & Hansen, 2004; Tang, Yankelovich, Begole, Van Kleek, Li & Bhalodia, 2001) or small mobile groups (Pokraev et al., 2005) acting independently of others. More generic context frameworks try to cover a wider area but lack explicit support for group interaction (e.g., CASS) (Fahy & Clarke, 2004), CoBra (Chen, Finin & Joshi, 2003), CORTEX (Biegel & Cahill, 2004), Gaia (Roman, Hess, Cerqueira, Ranganathan, Campbell & Nahrstedt, 2002), Hydrogen (Hofer, Schwinger, Pichler, Leonhartsberger & Altmann, 2002), and SOCAM (Gu, Pung & Zhang, 2004). A more complete overview of such frameworks can be found in Baldauf, Dustdar, and Rosenberg (2006). These context-aware middleware systems and applications provide and exploit various types of contextual information about location, time, user activities, user preferences, profiles of users, devices, networks, and so forth (Abowd, Dey, Brown, Davies, Smith & Steggles, 1999; Raento, Oulasvirta, Petit & Toivonen, 2005; Solarski, Strick, Motonaga, Noda & Kellerer, 2004). However, those models do not address the rich set of context information associated with collaborations. They focus mainly on user-related context and device capabilities.

The closest match is probably the Kimura system (Voida, Mynatt, MacIntyre & Corso, 2002), which monitors user's interaction during the collaboration by integrating and providing various types of context information. However, Kimura is targeted to an office environment and does not address issues posed by emerging team forms.

An SOA-oriented approach to context awareness (Gu et al., 2004) and collaboration (Jørstad, Dustdar & van Do, 2005) is very promising, but the notion of context as proposed by Dey and Abowd (1999) needs to be extended beyond involved services (Dorn & Dustdar, 2006) to explicitly include teams as a first order entity. The exhaustive review of current literature by Powell, Picolli, and Ives (2004) reveals that research efforts have focused merely on distributed teams as a whole without analyzing the internal interaction.

OVERVIEW OF THE INCONTEXT PERVASIVE AND COLLABORATIVE WORKING ENVIRONMENT

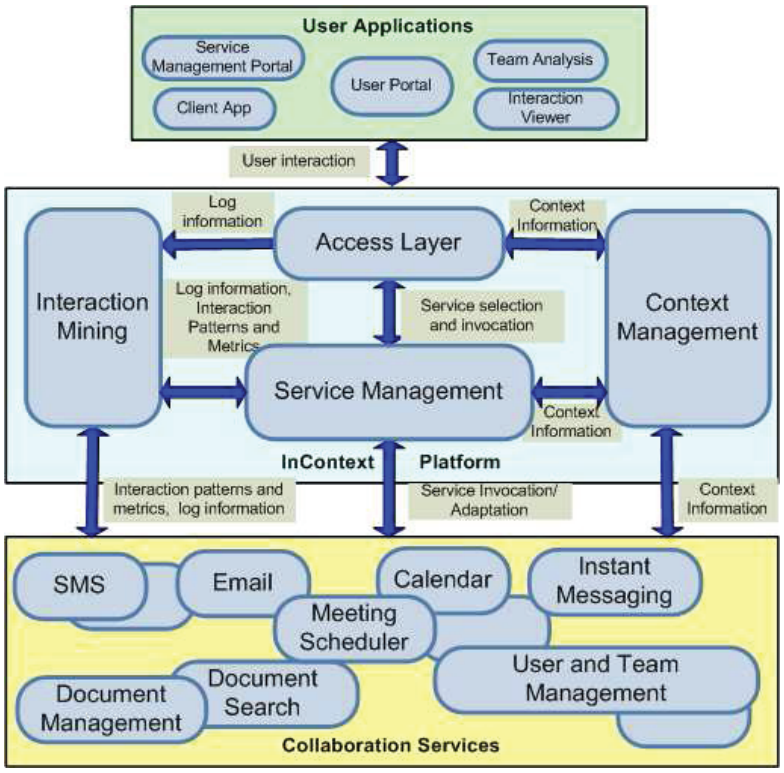
The inContext environment comprises three main parts: *Collaboration Services*, the core *inContext Platform*, and *User Applications*. An overview of the architecture is depicted in Figure 1. The User Applications essentially provide an interface to the users of the system (they tend to be either Web-based, based on specific device capabilities of small devices, or based on a specific computer application/plugin) and shall not be discussed in more detail here.

Collaboration Services are services that are normally required in team collaboration. These are either special services specifically built for use in the collaborative architectures or general services (e.g., calendars that team members use anyway). Furthermore, they can be proprietary, company intern with specific interfaces, or open source and public. We make the assumption that any service useful for team collaboration is networked and has a well-defined interface, either as a Web service or such that it can be wrapped into a Web service interface. More examples of collaboration services are document sharing (e.g., Document Management or Document Search), communication (e.g., SMS [Short Message Service], Instant Messaging [IM], E-mail, etc.), and team and project management (e.g., User and Team Management and Activity Management). Those services could be specific to particular projects, but many are generic services that can be reconfigured to fit into particular purposes.

The inContext platform is the central and most interesting part of the inContext environment. It integrates the algorithms and methods that provide the context awareness and collaborative functionality bridging and binding the services. It includes novel services that support advanced, dynamic collaboration of emerging teams based on a context and interaction model. The *Access Layer* acts as an intermediary receiving requests from the client side and invoking services; all interactions with the core system are passed through the access layer to allow for logging and subsequent mining of interactions. The Interaction Mining is used to extract and analyze interactions inherent within collaborations of teams, but also the historic decisions on which services were most appropriate, given a certain context. The Context Management manages context associated with humans, services, teams, and activities conducted. It supports reasoning mechanisms to infer new context information and can enrich existing context information. The Service Management is responsible for selecting the right services, ranking the services and invoking the services according to requests from Access Layer. All the aforementioned components can be deployed in and operate in a distributed manner.

The architecture shown in Figure 1 is a reference implementation of the so-called Pervasive Collaboration Service Architecture (PCSA) that we have developed in the inContext project. By introducing new core services that support context- and interaction-based collaboration, the inContext platform is able to integrate various existing collaboration services to establish a network of PCSAs deployed in multiple organizations.

Figure 1. The InContext environment



CONTEXT MODELING, GATHERING, AND REASONING

Context-aware systems have been discussed for some time as a way to enhance computer systems; this happens mostly at two levels: either at the system management level or to enhance the user experience. In both cases, relevant context information is being gathered, and the system adapts to the respective context. One example of context use to enhance the user's experience stems from the telecommunications domain. The aim was to develop a framework to allow end users to express how they want their communications to be conducted, and it was identified that this depends very much on the context of the users (Reiff-Marganiec & Turner, 2002).

Until now, context models discussed in the literature have either targeted concrete situations or have been very abstract. However, we believe that a context model should provide the scope for encapsulating all kinds of context information such as individual settings and team environments, short-lived coordination activities, and long-running complex projects. At the same time, the model needs to provide ways to manage context; the model should have notions of how contexts can change and which changes influence other areas of context. For example, if users change their location by going home from work, they might also change their roles, for instance, from a team leader to a wife and mother.

Hence, our suggested context model consists of several layers, ranging from concrete instantiations (the system level; M0) via a domain layer (M1) to a meta model layer (M2). An overview is shown in Table 1. The meta model layer represents concepts that are relevant for all collaborative

Table 1. Layered context model

M3	**M2** Model of Meta Classes	**M1** Model of System Level Objects	**M0** System Level
	CWE	Domain Specific	Concrete
	Role	**Managers, Technical Staff**	**Manager: Jim**
Language for system models	All possible system model	Concrete system model	Concrete System

working environments, while the domain layer and instantiation layer contain concepts that are required for Collaborative Working Environments (CWE) in a specific domain. Finally, the instance layer provides concrete instances for a particular situation. We also have a layer (M3) that provides notations for expressing the concepts at level M2; however, this is very generic, and UML or OWL, for example, provide the relevant mechanisms.

To identify the relevant concepts at the domain model layer M1, we identified five major team views. These views combined make up the Team Characteristics Model.

- **Spatial view.** Physical space can be characterized by geometric quantities such as volume, area, point, and extension of space. We look at measures that characterize entities such as users within space and their relation to one another. A position in space is expressed by location information that can also provide a semantic description or representation of space. The spatial distribution of team members is the geographical distance of individuals as well as distance between subteams. From a team's point of view, a proximity measure may be used to indicate distance or distribution. Temporal aspects such as transition of entities to a new location or frequency of location changes are expressed by location dynamics. Dynamics applied to teams include information such as the likelihood of intersecting trajectories or variations due to entities traveling at different speed. Awareness of location information and dynamics is a prime concern of the different team forms. Being aware of location is not only a question of observing the location context of other entities, but also the ability to recognize places or discover situated entities such as infrastructure elements and computing devices.
- **Organizational view.** The organizational level defines the structure of an organization. Specifically, it defines a topology, which denotes links and relationships between workers, employees, and departments. Staff members take up different (organizational) roles within organizations/departments such as supervisor, manager, and CEO. Relationships and roles are essential for trust building among individuals as well as between subdivisions. An organization may define rules and guidelines at different levels in the topology that have impact, for instance, on how people communicate and execute projects in different team forms. These rules can be manifested in the form of policies and are referred to as enterprise culture.
- **Project view.** The project view aims at organizing and managing resources. It includes the definition of scope of required work (project goals) and planning and monitoring of constraints such as time, cost, and risk. Projects are usually instantiated

under a certain premise; for instance, a company's mission or the mission and goals of nimble teams. The mission is shaped by the management unit and team leaders. The responsibility of a leader is to ensure that the actual outcome is created such that defined constraints and desired quality are satisfied (i.e., by monitoring and coordinating project-related activities). People exercise various roles in a project. We note that people increasingly work on more than one project or set of activities simultaneously. The role might depend on a particular task in a certain project (peer membership and cohesion of teams).

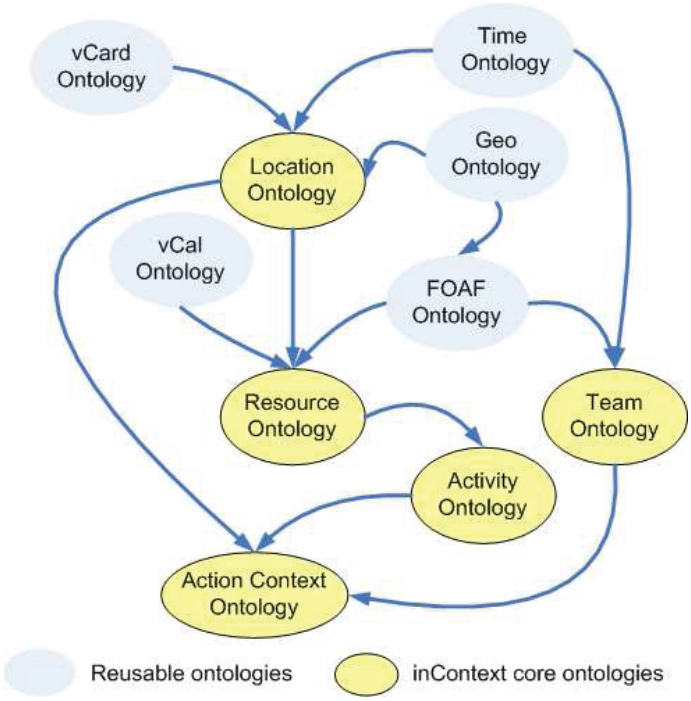
- **Interaction view.** Interaction patterns are activities or activity steps that are frequently repeated and can be observed by means of pattern detection between human actors. Such patterns may impact a team in terms of being aware of dynamics, status updates, and so forth, as well as the measures that can be taken to support collaboration. A communication pattern in human collaboration shows how distributed teams exchange information by means of synchronous or asynchronous communication channels. Whether or not individual interactions have great or only limited impact on a team is limited by the scope of an interaction.
- **Service view.** Services are means for supporting the user/teams in project-dependent activities and tasks. Service interactions are related to situations that arise when services engage in concurrent and interrelated interactions. Services may be consumed in combination by sequential, parallel, or conditional execution of tasks. Patterns provide the foundation for aggregating a number of services that are used in combination. Providing a service from a pool of available services, considering the consumer's context, is defined as relevance-based service provisioning. A service is provided upon request (reactive) or provided based on context (proactive), thereby considering collaborative activities or a task at hand. By understanding service interactions, a set of aggregated services could be provided so that human collaborators are able to complete an activity or progress toward an objective.

For the instantiation level M0, let us consider two examples: roles and locations. At the meta model layer, we are aware that users play certain roles; however, roles differ very much from domain to domain. In a medical context, we might have doctors and nurses; in academia, we find teachers and students. For locations we might have grid coordinates when considering outdoor locations, or we might have room numbers if we are inside a building. At the instance level, we find that Jim might be a manager, and that he is currently in a meeting in a specific building or room.

Context information plays an important role in adapting services suitable for emerging team forms. Unlike existing context-awareness systems in HCI or location-based services that utilize limited context information related to devices, user preferences, user presence, and location, the context associated with human collaboration is much more complex. Context of emerging teams will be associated with human (e.g., *person, organization, skill,* etc.), services (e.g., SMS and Document Management), location (e.g., site and address), teams (e.g., membership role and department), activities (e.g., project and communication), and interactions among human and services. Therefore, to describe the context model for inContext, we have to not only utilize many existing concepts and but also develop new ones suitable for emerging team forms.

Our approach in inContext is that we rely on ontology, named Web Ontology Language (OWL) (http://www.w3.org/TR/owl-ref/) to model context information. To this end, we incorporate existing ontologies with newly developed ones. Figure 2 depicts the hierarchy of existing and inCon-

Figure 2. Structure of the inContext context model



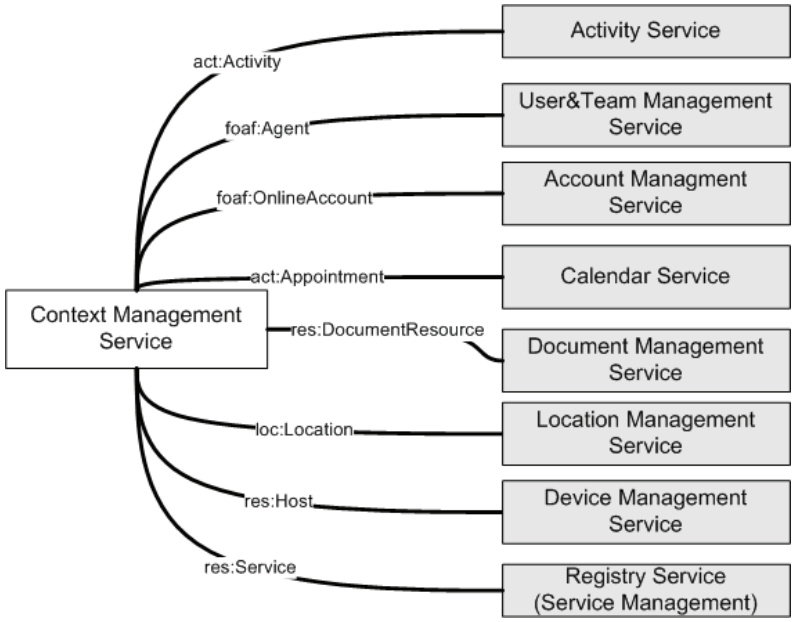
text ontologies. We partially reuse concepts in standard ontologies for describing user profiles, location information, time information, and so forth. Examples of such ontologies are the FOAF vocabulary (http://xmlns.com/foaf/spec/), which provides concepts such as Person, Organization, Group, Document, Project, or the Basic Geo Data (http://www.w3.org/2003/01/geo/) for concepts related to locations such as latitude and longitude. In addition to those reusable ontologies, we develop five new core ontologies for collaborative work-specific issues:

- **Location.** Describes various types of location information, including mobility one, because Basic Geo and vCard ontologies are not enough to express relocation.
- **Activity.** Describes the basic nature of activities and how they relate to users, resources, artefacts, and other activities.
- **Team.** Extends FOAF concepts to describe teams in more detail.
- **Resource.** Describes usual input for an activity such as documents, services, and devices.
- **Action.** Models the highly dynamic context that is subject to permanent changes.

Based on the context model, we have developed a set of software sensors that capture relevant context information. The context information is captured and stored whenever context is changed. Context information is collected from various sources and is not stored at any centralized place. As shown in Figure 3, the *Context Management* subsystem does not store context information into a central repository. Instead, context information is stored into and retrieved from distributed services. A core model is stored in a dedicated store within the *Context Management*, and from that model, different types of context information are linked by using RDF.

By using ontologies, context information can be inferred based on rules in order to provide

Figure 3. Sources of context information



value-added information about the context associated with people, teams, services, and activities. Our context reasoning techniques are built on the SPARQL++ engine named dlvhex, which processes ontological context data collected in the *Context Management*.

For example, let's assume we want to set up a team of civil engineers on demand for work at a particular site. To find suitable engineers, the SPARQL query shown in Listing 1 can be used.

```
PREFIX team:<http://www.in-context.
eu/team.owl#>
SELECT ?engineer
WHERE{
 ?engineer :hasProfile ?profile.
 ?profile :hasSkill ?skill.
 ?skill :name ?sname.
 ?engineer :locatedAt :''Genoa sea
port''
  FILTER  regex(?sname,"civil
engineer","i")
}
```

Listing 1. Finding suitable engineers using SPARQL.

Any services and clients can invoke the *Context Management* to query context information. Furthermore, context reasoning techniques can be used to aggregate context information from external sources, and evaluate and query rules defined over context information.

COLLABORATIVE TEAMS: TEAM FORMS AND INTERACTIONS

The term *interaction pattern* refers to a common, reoccurring interaction scenario between actors. The term *relation* refers to a tie or link between two actors within a pattern. We take three initial interaction patterns that are well known in the domain of Software Engineering (SE) and apply them to the domain of human collaboration.

Proxy Pattern

Originally, the proxy pattern was introduced by Gamma, Helm, Johnson, and Vlissides (1994) as a structural pattern in software design. The intention for using a proxy is "to provide a surrogate or placeholder for another object to control access to it." Besides forwarding the clients' requests and sending back the responses, a proxy can do pre- or postprocessing, depending on its type. A real-life example of a proxy in human collaboration is a secretary. A secretary receives e-mails, phone calls, messages, and so forth, which are actually intended for a different entity (i.e., the boss). The secretary preprocesses these client requests, for example, by filtering out unwanted requests (protection proxy) or even answering simple requests without having to involve the boss (cache proxy) (Dustdar & Hoffmann, 2006).

A proxy pattern usually describes a 1:1 relationship between proxy and original. However, there are two exceptions, remote proxies and firewall proxies, where a proxy is responsible for multiple originals.

Broker Pattern

The broker architectural pattern can be used to structure distributed software systems with decoupled components that interact by remote invocations. "A broker component is responsible for coordinating communication, such as forwarding requests, as well as for transmitting results and exceptions" (Buschmann, Meunier, Rohnert, Sommerlad & Stal, 1996). According to Dustdar and Hoffmann (2006), "a broker's foremost goal is to achieve location transparency of servers/services. … The broker is responsible to locate a server/service that can handle a given request. Then the broker forwards the request to the appropriate component, receives its response and delivers the response to the client." In contrast to a proxy, a broker does not perform any pre- or postprocessing.

Master/Slave Pattern

The SE domain defines a Master/Slave (M/S) pattern as follows: "The Master-Slave design pattern supports fault tolerance, parallel computation and computational accuracy. A master component distributes work to identical slave components and computes a final result from the results these slaves return" (Buschmann et al., 1996).

Understanding interaction among team members and services sheds light on characteristics of team members; for example, the role of a team member, which type of communications a team member prefers, and the performance of a service. Quantitative information associated with interactions can then be used to enrich context information and utilized as inputs directly by the service selection and ranking.

Because in emerging team collaboration many activities are defined on demand without any predefined processes, interactions are detected from log information based on correlation techniques. Various types of interactions associated with humans and services are inherent within collaborative environments. We categorize three kinds of interactions:

- **Service-to-service interaction.** The interaction between two services (e.g., a service might call another service)
- **Human-to-service interaction.** The interaction between a human and a service (e.g., how which services are used by a team).
- **Human-to-human interaction.** The interaction between human and human (e.g., how a team member interacts with another one in order to perform activities).

For each type of interaction, interaction mining is applied at multiple levels such as individual (human or service), group (a team or a set of services), and the whole system (all services and/or teams). In order to provide metrics associated with interactions, we have collected log information of

collaboration services and performed the mining. Using aggregation techniques, higher-level metrics can be determined from lower-level ones.

The amount of information provided by the Interaction Mining is vast, and the information ranges from low-level such as historical metrics associated with a service, to high-level, such as detected patterns associated with a team. To provide such information to *Context Management* and *Service Management* as well as to other clients, the *Interaction Mining* provides APIs and languages for accessing the information through Web services. We are currently working on a query language that allows the client to specify *concepts* in inContext ontologies and *duration* for which the *Interaction Mining* should provide mining information associated with the concepts.

COLLABORATION SERVICES: DESCRIPTION, LOOKUP AND SELECTION

Service-Oriented Architecture (SOA) is becoming a more and more established paradigm. The fundamental concept of SOA is the notion of a *service* that is defined by the following basic properties:

- The service provides well-defined functionality
- The interface contract for the service is platform-independent
- The service can be dynamically discovered and invoked
- The service is self-contained and loosely coupled

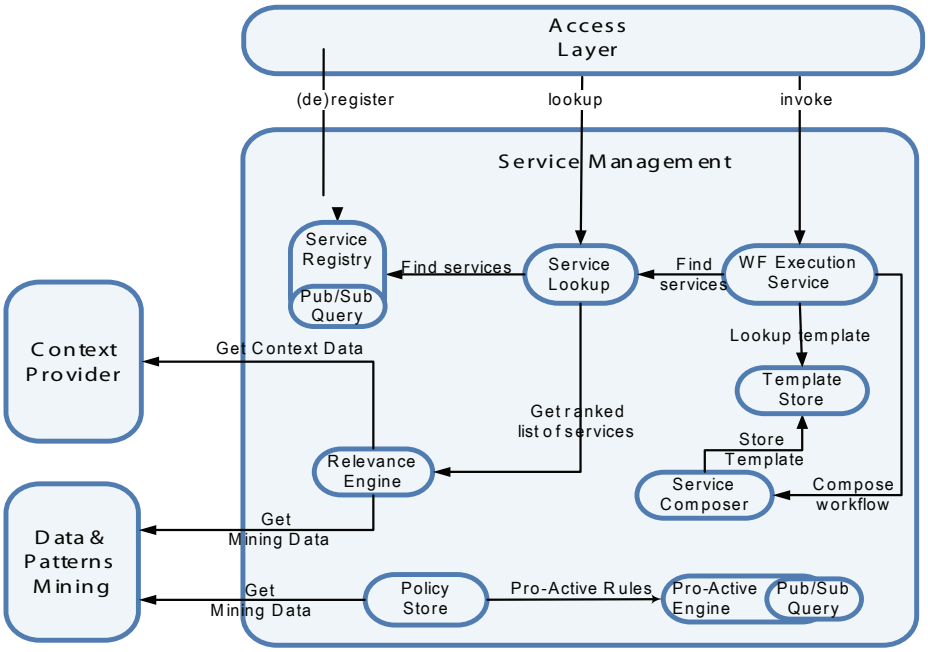
SOA and Web services-based solutions have been used in many domains and have proved the advantages promised in the definitions already mentioned (flexibility, interoperability, loose coupling) and have become quite mature technologies. Still, due to the dynamic nature of teams and the large variety of group types, additional research efforts need to be put in the following areas:

- Enabling flexibility in terms of dynamic discovery of available services, which can frequently change based on user and team settings
- Extended description of services (syntax not sufficient) for context-based autonomous selection of alternative services
- Support mechanisms to facilitate composition and orchestration of higher-level services to support the typical operations of collaboration
- Dynamic adaptation of service compositions based on the context of users and teams

Before we can approach dynamic, context-aware service composition, we need to enable context-aware service selection. Existing approaches do not take context into account at all, or they focus only on individuals (Cuddy, Katchabaw & Lutfiyya, 2005) rather than teams.

Through the dynamic selection of services, it is possible to cater to a different requirement of collaboration, depending on the team structure, interaction pattern, or context. This approach allows composition at different levels, such as concrete and abstract workflows and noncoupled tasks that have to be combined during runtime. Thus, an architecture supporting collaborative workspaces must be able to react to changing context settings (e.g., to contact a user depends on the user's location, current online status, and the device currently in use). In a situation where the user is sitting in front of a PC, it might be possible to contact the user directly via e-mail, but if the user is currently on the move (the context would say that the user is off-line), a possible notification for contacting would be to send an SMS to the user's mobile phone. Hence, as SOA helps to abstract from the underlying technology and infrastructure, a platform will offer different contact services, such as e-mail, SMS, or instant

Figure 4. Service lookup and ranking



messaging depending on context information.

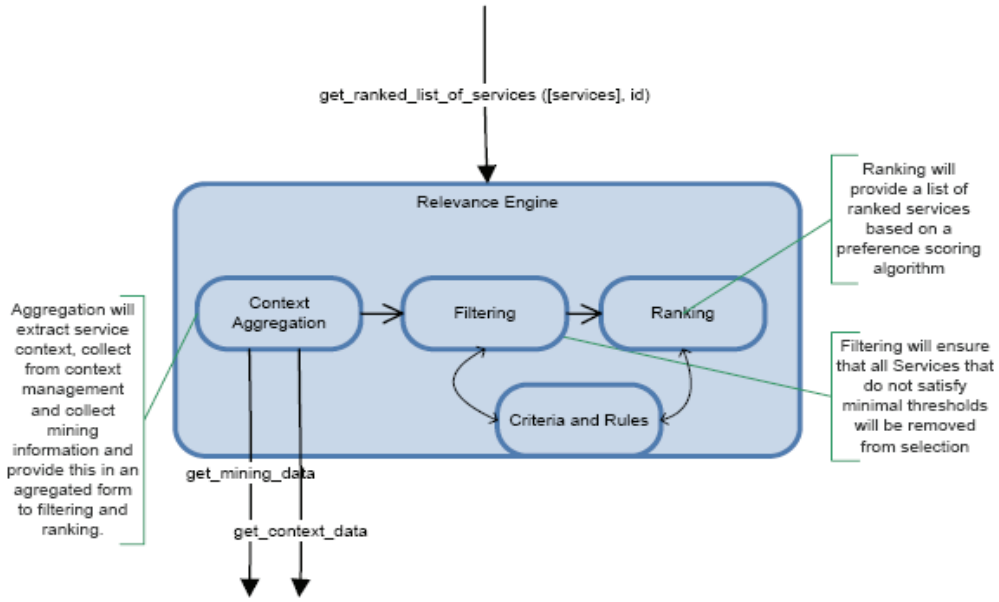
The decision is not made by the context itself; rather the collaboration platform selects the most relevant service based on context information. For this purpose, all services are registered in a service registry such as UDDI. Figure 4 displays the overview of the service management showing the mechanisms for lookup of a service. Figure 5 refines the details of ranking a set of services featuring context-based metadata as executed by the relevance engine to create a ranking of these services. To do so, the relevance engine retrieves relevant context information.

In the PCSA, there are many collaboration services readily available. Services can complement or compete with each other; for example, two providers can provide two services with the same function. However, each particular collaboration instance might require different kinds of services, depending on the context. The key of adaptation is centered on how to use context and interaction information and service information to select suitable service instances for the collaboration. The *Service Management* is not only for managing collaboration services but also for selecting the right service based on the context. To this end, three sources of information are used by *Service Management*: context information, interaction information, and service meta-information.

While context and interaction information can be obtained from corresponding components, the service meta-information has to be managed by the *Context Management*. In doing so, we have to integrate different kinds of meta-information associated with services. We developed a service meta-information model used to relate different types of information associated with services, based on what service selection is performed. In this model, we first define a service category to indicate the type of services, such as SMS and DocumentSharing. Then operations offered by services are mapped into one or more categories.

For each service operation, a set of criteria will be used to represent the meta-information about service operation.

Figure 5. Details of the ranking process



A criterion is represented as a quadruple (name, type, value, weight), indicating the name of the criterion, the data type, value of the criterion, and weighted factor, respectively. For example, an SMS service provides an operation named sendSMS, which can be associated with the following criteria:

name	Type	value	weight
cost	double	1.3 EUR	0.25
reliability	double	1.0 %	0.75

Based on context information, interaction information, and service meta-information, the *ServiceManagement* performs the selection and ranking of services. This involves multiple steps. First, using context reasoning, the *Service Management* picks up the right service categories. Next, based on service meta-information and interaction metrics, the services are ranked. Then the best service is selected based on its rank. The reasoning step is performed by sending a request to the *Context Management*. For ranking services, we have developed a modified Logic Scoring Preference (LSP) algorithm (Reiff-Marganiec, Yu & Tilly, 2007).

CONCLUSION AND FUTURE DIRECTIONS

This chapter describes the inContext pervasive and collaborative working environment. Motivated by the lack of suitable CWEs for emerging team forms, the inContext project has introduced novel techniques to integrate existing collaboration services, context, and interaction-based collaboration to support advanced features supporting different collaborative teams, ranging from mobile and nomadic to ad hoc ones. In this chapter, we presented the main components of the inContext environment in more detail.

More specifically, we have presented achievements in the areas of team forms and interaction, service management and context gathering, and reasoning.

Computer-supported or mediated human-to-human collaboration offers the great flexibility of working on joint activities, together with other

team members across space and time. People work on numerous activities in different projects/teams at the same time. Context helps people to focus their attentions on relevant pieces of information. Interruptions can be minimized by promoting important requests. Various contextual information such as location, presence, and so forth help to establish team awareness in disparate teams. By using human interaction patterns, we can reveal hidden interactions and make collaboration more effective.

An SOA-based approach to challenges in CWE allows us to support changing needs of distributed teams. These requirements become increasingly dynamic as teams are dispatched in a multitude of collaborative environments in respect to team coupling, time of existence, location dynamics, and so forth. A Web services-based architecture allows collaboration services to be discovered, assembled, aggregated, and adapted according to users' context.

Still there is space for improvements. One aspect is to study the nature of human collaboration itself and to use this knowledge to design software that supports the user and yet is as transparent as possible. We also see a growing need for collaborative devices to automatically sense the user's context and to provide information, notification, and support based on that context.

We believe that our human-centered approach of service-interaction mining will be a significant contribution to useful service provisioning and, during service lookup, for ranking of discovered services.

ACKNOWLEDGMENT

This research is partially supported by the EU STREP project inContext (FP6-034718). We thank all members of the inContext consortium for their contributions to the development of the inContext environment; in particular, thanks are due to Axel Polleres (DERI, Ireland), Dino Baggio (Electrolux, Italy), Sarit Moretzki (Comverse, Israel) and Pete Kendal (WMLGA, UK).

REFERENCES

Abowd, G.D., Dey, A.K., Brown, P.J., Davies, N., Smith, M., & Steggles, P. (1999). Towards a better understanding of context and context awareness. *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing*, 304–307.

Baldauf, M., Dustdar, S., & Rosenberg, F. (2006). A survey on context aware systems. *International Journal of Ad Hoc and Ubiquitous Computing* (forthcoming).

Bardram, J.E., & Hansen, T.R. (2004). The AWARE architecture: Supporting context-mediated social awareness in mobile cooperation. *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, 192–201.

Bentley, R., Horstmann, T., Sikkel, K., & Trevor, J. (1995). Supporting collaborative information sharing with the World-Wide Web: The BSCW shared workspace system. *Proceedings of the 4th International WWW Conference*, 63–74.

Biegel, G., & Cahill, V. (2004). A framework for developing mobile, context-aware applications. *Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications*, 361–364.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). *Pattern-oriented software architecture: A system of patterns*. West Sussex, England: John Wiley & Sons Ltd.

Chen, H., Finin, T., & Joshi, A. (2003). An ontology for context-aware pervasive computing environments. *Knowledge Engineering Review*, *18*(3), 197–207.

Cuddy, S., Katchabaw, M., & Lutfiyya, H. (2005). Context-aware service selection based on dynamic and static service attributes. *Proceedings of the IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, 13–20.

Dey, A.K., & Abowd, G.D. (1999). *Towards a better understanding of context and contextawareness* [GVU Technical Report GITGVU-99-22]. Georgia Institute of Technology.

Dorn, C., & Dustdar, S. (2006). Sharing hierarchical context for mobile Web services. *Distributed and Parallel Databases*, Special Issue on Context-Aware Web Services, forthcoming.

Dustdar, S. (2004). Caramba—A process-aware collaboration system supporting ad hoc and collaborative processes in virtual teams. *Distributed and Parallel Databases*, *15*(1), 45–66.

Dustdar, S., & Hoffmann, T. (2006). Interaction pattern detection in process oriented information systems. *Data and Knowledge Engineering,* forthcoming.

Fahy, P., & Clarke, S. (2004). CASS—A middleware for mobile context-aware applications. *Proceedings of the Workshop on Context Awareness, MobiSys 2004*. Retrieved January 1, 2007, from http://sigmobile.org/mobisys/2004/context_awareness/papers/cass12f.pdf

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design patterns—Elements of reusable object-oriented software*. Boston, MA: Addison-Wesley.

Gu, T., Pung, H.K., & Zhang, D.Q. (2004). A middleware for building context-aware mobile services. *Proceedings of the IEEE Vehicular Technology Conference (VTC 2004)*, 2656–2660.

Hofer, T., Schwinger, W., Pichler, M., Leonhartsberger, G., & Altmann, J. (2002). Context awareness on mobile devices—The hydrogen approach. *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, 292–302.

Jørstad, I., Dustdar, S., & van Do, T. (2005). A service oriented architecture framework for collaborative services. *Proceedings of the 3rd International Workshop on Distributed and Mobile Collaboration (DMC) IEEE WETICE'05.*

Kephart, J.O., & Chess, D.M. (2003). The vision of autonomic computing. *Computer*, *36*(1), 41–50.

Pokraev, S., et al. (2005). Service platform for rapid development and deployment of context aware, mobile applications. *Proceedings of the International Conference on Web Services*, 639–646.

Powell, A., Picolli, G., & Ives, B. (2004). Virtual teams: A review of current literature and directions for future research. *ACM SIGMIS Database, 35*(1), 6–36.

Raento, M., Oulasvirta, A., Petit, R., & Toivonen, H. (2005). Contextphone: A prototyping platform for context-aware mobile applications. *IEEE Pervasive Computing*, *4*(2), 51–59.

Reiff-Marganiec, S., & Turner, K.J. (2002). Use of logic to describe enhanced communication services. In D.A. Peled, & M.Y. Vardi (Eds.), *LNCS2529: Formal techniques for networked and distributed systems—FORTE2002*. Berlin: Springer.

Reiff-Marganiec, S., Yu, H.Q, & Tilly, M. (2007). Service selection based on non-functional properties. *Proceedings of NFPSLASOC 2007.*

Roman, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R.H., & Nahrstedt, K. (2002). A middle-ware infrastructure for active spaces. *IEEE Pervasive Computing*, *1*(4), 74–83.

Rosenberg, F., Platzer, C., & Dustdar, S. (2006). Bootstrapping performance and dependability attributes of Web services. *Proceedings of the IEEE International Conference on Web Services (ICWS'06).*

Solarski, M., Strick, L., Motonaga, K., Noda, C., & Kellerer, W. (2004). Flexible middleware support for future mobile services and their context-aware adaptation. In Aagesen, Anutariya, & Wuwongse (Eds.), *INTELLCOMM*, *Lecture Notes in Computer Science* (volume 3283) (pp. 281–292. Springer.

Tang, J.C., Yankelovich, N., Begole, J., Van Kleek, M., Li, F., & Bhalodia, J. (2001). ConNexus to awarenex: Extending awareness to mobile users. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 221–228.

van Do, T., Jørstad, I., & Dustdar, S. (2006). Mobile multimedia collaborative services. In *Handbook of research on mobile multimedia* (pp. 414–429). Hershey, PA: Idea Group Publishing.

Voida, S., Mynatt, E.D., MacIntyre, B., & Corso, G.M. (2002). Integrating virtual and physical context to support knowledge workers. *IEEE Pervasive Computing*, *1*(3), 73–79.

KEY TERMS

Context: The environment in which a system is executed, or for collaborative systems, the environment of the collaborative activity, including information on locations, activities, people, and their relations. Generally, the context data are volatile.

Context-Aware System: A computer system that adapts its behavior with respect to changes in its operating context.

Collaborative Work Environment: A computer system that provides support to conducting tasks in a collaborative manner.

Service-Oriented Architecture: A way of building computer systems from components (called services) that are loosely coupled and dynamically selected at runtime to fulfill user requirements. Often also used for implementations of such systems, with Web services being the most predominant implementation.

Service Selection: The act of dynamically chooses (at system runtime) services to fulfill a certain role in a system following the service-oriented architecture paradigm.