ENABLING CONTEXT-ADAPTIVE COLLABORATION FOR KNOWLEDGE-INTENSE PROCESSES

Stephan Lukosch  
Faculty of Technology, Policy, and Management, Delft University of Technology  
PO box 5015, 2600 GA Delft, The Netherlands

Dirk Veiel, Jörg M. Haake  
Faculty of Mathematics and Computer Science, FernUniversität in Hagen, 58084 Hagen, Germany

Keywords: Shared workspaces, Adaptation, Group context.

Abstract: Knowledge workers solve complex problems. Their work seems not to be routinisable because of the unique results and constellation of actors involved. For distributed collaboration knowledge workers need many different tools, which leads to knowledge dispersed over different locations, artifacts, and systems. Context-based adaptation can be used to support teams by shared workspace environments best meeting their needs. We propose an ontology representing context in a shared workspace environment, and a conceptual architecture for context sensing, reasoning, and adaptation. We report on first experiences demonstrating the applicability of our approach and give an outlook on directions of future work.

1 INTRODUCTION

The design, development and production of new and innovative products often requires knowledge which is distributed over a whole company. The design team knows about the preferences of the customers regarding the look and feel of the product. The development department knows about the restrictions (e.g., weight, size, budget) and capabilities (e.g., functionality, savings, improvements). Finally, the production department knows how to build and ensure the quality of the product. All steps, from the design to the production of a product, depend on the previously acquired knowledge of all participants and are regarded as knowledge work (Drucker, 1999; Drucker, 1993).

Knowledge workers solve complex problem. According to Sari et al. (Sari et al., 2007) knowledge work seems to be not routinisable because of the uniqueness of the results and the unique constellation of actors involved. When knowledge workers collaborate in distributed teams, they need many different tools (IT systems; e.g., CAD/CAM environments, ERP systems, IDEs). This leads to knowledge and information dispersed over different team members, artifacts and systems. Teamwork also requires communication over different systems and artifacts, and leads to difficult communication and coordination. These problems make distributed collaboration difficult.

Current approaches to support distributed collaboration include multimedia communication systems, repositories for shared documents and shared workspaces systems, notification systems, shared editors, shared calendars, and workflow systems. These systems either leave the organization of collaborative work to the users or ignore the changing needs of users to include different tools and artifacts. While shared workspace systems try to combine repositories with coordination and communication support, they still require team members to manually maintain the organization of social processes, artifacts, and tools. This includes the integration of the tools. As a consequence, teams may fail to adapt their shared workspace to best meet their current needs.

Our idea is to address these problems by enabling context-adaptive collaboration in an integrated shared work environment that integrates the different tools of the team members and adapts the behavior of the shared work environment according to the group context. Though not routinisable, knowledge work shows high similarity across very divergent situations (Sari et al., 2007). We want to identify such recurring situations and possibilities to improve the current interaction by analyzing the context and adapting the behav-
ior of the shared work environment to best meet the current situation. We suggest that the overhead for manual adaptation may be decreased by computer-supported adaptation of shared workspace features. We assume that such adaptation may positively impact team performance (e.g., by changing affordances in a way that improves interaction, process and/or output quality).

We propose that such adaptation can be based on context information (i.e. information captured by the system about individual as well as group use and preferences) and on adaptation rules defining the kind of adaptation. Adaptation includes modification of the set of applications/services for each user, changing their respective UI, and changing their content. The actual adaptation rules need to evolve over time, thus requiring means for users to understand ongoing adaptations and how they can change them. In this way, we see the (evolving) rules as another part of the context information.

Considering the above, it is essential to provide a context model that allows the system to recognize collaboration situations which are suitable for adaptation and which can easily be extended to include further context factors. Furthermore, it is necessary to provide a system architecture that can use this context model and the included adaptation rules to adapt its runtime behavior in a way that the interaction among the knowledge workers is improved.

This paper is structured as follows. First, we review existing shared work environments and context-adaptive systems. Then, we identify the basic elements of our context model along an example of two collaborating knowledge workers. In a second step, we introduce a conceptual architecture that uses our context model to enable context-adaptive collaboration. In a third step, we validate our context model by presenting an example for a possible adaptation and adaptation rule. This example shows how our context model and conceptual architecture can be used to recognize situations in which adaptation can improve the team interaction and how adaptation rules can be triggered and executed to actually improve the interaction within the team. Finally, we report on first experiences before concluding the paper and raising questions for future work.

2 RELATED WORK

In the following, we first review relevant shared work environments and discuss how these systems deal with possible adaptations. Then, we take a look at context-adaptive systems in general. We review whether the taken approaches are suitable to support context-adaptive collaboration in knowledge-intense processes. We discuss the used context models and whether these models are suitable to model collaboration situations.

BSCW (Appelt and Mambrey, 1999) and CURE (Haake et al., 2004b; Haake et al., 2004a) are web-based shared work environments offering a variety of collaboration services, e.g. document sharing and communication. CHIPS (Wang and Haake, 2000) is a cooperative hypermedia system with integrated process support. TeamSpace (Geyer et al., 2001) offers support for virtual meetings and integrates synchronous and asynchronous team interaction into a task-oriented collaboration space. BRIDGE (Farooq et al., 2005) is a collaboratory that focuses on supporting creative processes and as such integrates a variety of collaboration services. All of the above examples focus on a specific application domain. Though they all offer a variety of services, the systems are independent of each other and do not allow to integrate additional services. Some of them, e.g. CHIPS or CURE are highly tailorable, but they do not adapt their offered functionality to improve the collaboration within a team.

The most prominent examples for context-based adaptation focus on single users and consider location as most relevant context information (e.g. (Schilit et al., 1994; Abowd et al., 1997; Kindberg et al., 2002) or focus on learner profiles (cf. ITS). Compared to single-user ITS, COLER (de los Angeles Constantino-González and Suthers, 2003) provides a software coach for improving collaboration. CoBrA (Chen et al., 2003; Chen et al., 2004) is an agent-based architecture that uses shared context knowledge represented as an ontology to adapt service agents according to a users context. Gross and Prinz (Gross and Prinz, 2004) introduce a context model and a collaborative system that supports context-adaptive awareness. The context model consists of events, artifacts, locations, etc. The main restrictions of their approach are that the context representation is only used to update and visualize awareness information and that only one cooperative application can be used. Edwards (Edwards, 2005) explores the space between two different context understandings: in CSCW research, people are assumed to be the consumer of context information; the ubiquitous computing community has the opinion that systems are the consumers of context information. Intermezzo (Edwards, 2005) tries to fill this gap through the creation of new higher-level services, but in the end collaborative applications are missing. Rittenbruch describes an approach to the representation of context of aware-
iness information but real world examples are missing (Rittenbruch, 1999). Fuchs (Fuchs, 1999) describes an integrated synchronous and asynchronous notification service for awareness information called AREA, but again AREA uses the context representation only for awareness information. Ahn et al. (Ahn et al., 2005) introduce a knowledge context model. Based on this context model they implement the virtual workplace support system (VWSS). One drawback of their solution is that their knowledge context model has to be extended for other application domains. The Semantic Workspace Organizer (SWO) (Prinz and Zaman, 2005) is an extension of BSCW. It analyzes user activities and textual documents inside the shared workspace to suggest appropriate locations for new document upload and for document search. The ECOSPACE project aims at providing an integrated collaborative work environment (Prinz et al., 2006; Martinez-Carreras et al., 2007). For that purpose, ECOSPACE uses a service-oriented architecture and provides a series of collaboration services for orchestration and choreography. The orchestration and choreography is based on an ontology which still has to be described (Martinez-Carreras et al., 2007; Vöndrueden and Prinz, 2007).

The above approaches focus on context representations and adaptations which are used in specific domains, e.g., single-user systems or ITS, or on subdomains in the field of CSCW, e.g. awareness or knowledge management. Adaptation based on group context and for multiple users of a cooperative system is intended only by ECOSPACE, but the required context model is still an open issue. Similarly, only ECOSPACE supports the integration of different collaboration services within the same shared work environment. In summary, current approaches do not sufficiently support a context-based adaptation of shared work environments.

3 A CONTEXT MODEL FOR KNOWLEDGE-INTENSE PROCESSES

Dey et al. define context as any information used to characterize a situation of an entity where entities may be any object, person or place providing information about the interaction between a user and an application (Dey et al., 2001). With this definition, any information may help characterizing the situation of the interaction’s participants because it is part of the context itself. For our purposes, we can narrow this definition so that context includes all information which is necessary or helpful to adapt a cooperative workspace to better fit the needs of a collaborating team. This implies that the context contains information about the team as well as about the current collaborative situation.

There exist different approaches to model context. These approaches range from simple key-value models over graphical models up to sophisticated ontology based models, which support validation and reasoning (Strang and Linnhoff-Popien, 2004). Nelson et al. as well as Levitt et al. hypothesized that routines are key determinants of organizational performance (Nelson and Winter, 1982; Levitt and March, 1988). To discover such routines, to identify adaptations, and to ensure a consistent context model, we need reasoning mechanisms and model our context as an ontology using OWL.

In the following, we describe and explore the basic concepts and relations within a global collaboration space using an example of two collaborating knowledge workers. We use the following convention for the names of concepts: <name of the concept> : <name of the instance> (e.g. Actor: alice). In some cases we use an index to describe the relationship of one instance to another more precisely (e.g. Role: Author_{alice}).

![Figure 1: Basic interactions within a sample global collaboration space.](http://www.w3.org/2004/OWL/)

Figure 1 illustrates the basic interactions within a sample global collaboration space. A global collaboration space contains all actors, user workspaces, collaborative applications, services and artifacts needed to carry out a collaborative project between the involved actors. In our sample environment Actor:alice and Actor: bob are members of a team represented as Team:Project1 (cf. Figure 1) that uses several collaborative applications to design and describe a new product. The concept Role defines a set of possible actions that actors can execute within a collaborative application. Each collaborative application defines a set of supported actions. A subset or the whole set of these possible action is assigned to a role. Thereby, different roles with different access rights can be de-
fined. Each actor can have multiple roles. The role for a collaborative application is assigned to a user by the user’s workspace. As Figure 1 shows, Actor:alice has the role Role:Designer(alice) while using Collaborative Application:Image Editor(alice), and the role Role:Editor(alice) while using Collaborative Application:Text Editor(alice). We distinguish between these two roles to illustrate that each role usually implies different activities and uses specific (i.e. domain specific) applications to operate on artifacts. Actor:bob uses the Collaborative Application:Text Editor(bob) and has the role Role:Reviewer(bob). This role implies a restricted action set that is available to operate on artifacts. Usually, reviewers add comments to the text but are not allowed to edit or delete the text. While using the Collaborative Application:Rich Text Editor(bob), Actor:bob has the role Role:Author(bob).

Figure 2 illustrates that each actor has its own user workspace (User Workspace:UW_alice and User Workspace:UW_bob). As each user’s workspace can be configured differently, a user workspace defines a set of available applications. This set of applications is an attribute of each user workspace. Each of the two actors (Actor:alice and Actor:bob) interacts with two collaborative applications that reside in the corresponding user workspace. Actions, e.g. AddShape:alice or OpenText:bob, are used to describe an interaction between actors and the collaborative applications. When interacting with a collaborative application, actors perform actions which the applications are capable of and are allowed by their role.

A collaborative application uses services, e.g. Service:LineTo or Service:SetContent, to operate on artifacts, e.g. Artifact:Image or Artifact:Text Document. Artifacts may be extended with data and services for coordination of collaborative work, e.g. locking information. All actors and artifacts are potentially located in space and time.

A collaborative application implements the model-view-controller (MVC) paradigm (Krasner and Pope, 1988) (cf. Figure 3). Actors interact with the collaboration application by performing actions allowed by their roles. These actions are received by the corresponding controller components of the application. In Figure 3, Actor:alice performs an EditText:alice. This action is received by the Controller:Text Document_alice. The controller then uses the Service:setContent to modify the Artifact:Text Document. As in MVC, the shared model then uses the Service:Notifier to notify registered view components, e.g. View:Text Document_alice, about modification in the model. By receiving such a notification the registered view component can update the displayed information.

Figure 4 summarizes the above model of a global collaboration space and shows the basic ontology concepts and their relations. We modeled this ontology using OWL. In addition to the above, we introduced additional concepts, like Communication, Shared Editing, or Awareness, to classify the func-
tionality a Collaborative Application offers. For overview reasons, we only show here three high-level concepts, but we distinguish further high-level as well as additional sub-level concepts.

4 CONCEPTUAL ARCHITECTURE

Due to the openness of the real world, we may either have to deal with evolving context dimensions or live with a closed set of context dimensions. In both cases new questions arise. How would users be able to deal with evolving context dimensions? How can we enable users to build social solutions around a system with fixed context dimensions? Though our current prototype uses a closed set of context dimensions, our conceptual architecture allows integrating services for extending context dimensions.

![Conceptual architecture](image)

Our conceptual architecture consists of four layers: Application User Interfaces (UIs), Adaptation Server, Collaboration Services, and Shared Model (cf. Figure 5).

A flexible adaptive system executes a cycle of
1. User interaction
2. Sensing user activities
3. Adapting system behavior
4. Modifying adaptation knowledge (e.g., if users want to change adaptation rules)

We use the sensing functionality to illustrate the interactions between the different components of our architecture. The UI-part of a collaborative application (Application UIs) is used to collect relevant information about the interaction between the user and the application. This information (including the service calls) is forwarded to the Service Call & Interaction Sensing component of the Adaptation Server, which extracts relevant information and updates Context Knowledge via the Context Knowledge Service Provider. This information is passed to the Collaborative Application Manager (CAM), which triggers the services of the corresponding Collaborative Application. Applications can use several basic services from different Service Providers to implement the application logic. Thereby, we can integrate different services into an application and adapt application behavior across different service providers.

Next, we describe the adaptation functionality. Based on the event log, the Service Call & Interaction Sensing component prioritizes the events and adds all information to a queue. If a high priority event is present, the number of events exceeds a predefined threshold value, or a timeout occurs (i.e., no further events came in for a certain time) the Reasoner is triggered. The Reasoner uses the current Context Knowledge, which is represented by instances of our ontology concepts and relations between those, via the Context Knowledge Service Provider and the current Adaptation Knowledge, which e.g. includes adaptation rules or axioms, via the Adaptation Knowledge Service Provider to find relevant adaptation rules. Executing these adaptation rules will potentially modify the set of applications/services for each user, change their respective UI, and/or change their content in order to improve team interaction. The information about the modifications/changes is passed to the CAM to update the current service configuration. The Notifier sends the adaptation information to the subscribed UI-parts of a Collaborative Application. The UI-parts then refresh their view according to the current service configuration in the CAM and/or use some Adaptation Functions through the Adaptation Functionality Service Provider to modify the content of the UI.

Adaptation rules are initially defined by users or developers using the Collaborative Adaptation Knowledge Editor (CAKE). Rules need to match the needs of the team, thus requiring means for users to understand ongoing adaptations (traceability, reflection, understandability of changes) and how they can change them.

The CAKE is used to support the modification of adaptation knowledge. It allows users to access, review, edit, and create new user-defined adaptation rules. CAKE is a collaborative application using our architecture meaning that it can be adapted as well. CAKE uses the Adaptation and Context Knowledge Service Providers to access and modify the corresponding adaptation information.
5 ADAPTATION EXAMPLE

As a scenario consider that a team consisting of Alice and Bob synchronously collaborates on a shared text document. We assume, that Alice and Bob created local workspaces. Alice then created a shared text document and opened a shared text editor to work on a design document. Bob later opened the same shared text document to review the current state of the design. The team members have different roles highlighting their task within the team.

This is expressed in Figure 6 by Role concepts\(^2\) (e.g., Author) connected to possible actions (e.g., OpenText, EditText, Annotate) related to an application concept (e.g., TextEditor). We assume that the assignment of roles (instances of Role concepts) to concrete Actors (e.g., Bob, Alice) was done initially when creating application instances (i.e., when opening a shared text editor instance on a concrete document). Team members can now synchronously access and edit document parts, which can lead to conflicts or potential collaboration situations. To improve the interaction within team, each user’s workspace could be adapted by providing additional awareness information or establishing a communication possibility.

In Figure 6 two actors access the same Artifact: Text Document. These actors have different roles which allow the actors to perform different actions. This is expressed by the relation of roles to actions, which are supported by the collaborative applications within the user’s workspace. As an example, consider Actor: alice in the Role: Author. The Role: Author is related to all actions supported by the application TextEditor.

\[^2\]Please note, concepts are shown as ovals whereas instances of ontology concepts are shown as rectangles.

Figure 6: Context graph.

In this adaptation rule, availableAppl returns a set of Collaborative Applications which support Communication and are connected to all members of the team in their current context (cf. Figure 4 for correspond-
ing concepts and relations). \texttt{sharedArtifact} returns a set of artifacts for which the whole team belongs to the context of each artifact and \texttt{selectOneFrom} selects from a set of context elements the one which has been used most by the collaborating team members. The corresponding information is stored as activation value within the context graph. The activation value is calculated for each concept in the graph and is updated by a special learning algorithm. Here a \texttt{Chat} application will be opened for Alice and Bob and a chat session will be established between them, as a \texttt{Chat} application the only application available for \texttt{Communication} within both users’ workspaces.

6 EXPERIENCES

The current prototype manages our context model using the semantic web framework Jena \(^3\). The \texttt{Adaptation Server} is based on Equinox \(^4\) and realizes all components as so-called bundles in OSGi \(^5\). In the prototype, we integrated two Collaborative Applications. Firstly, we extended CURE (Haake et al., 2004b; Haake et al., 2004a) to provide service classes for access right, document, user, and workspace management as well as asynchronous awareness and communication. Secondly, we use ActiveMQ \(^6\) to develop and integrate support for the synchronous awareness and communication. The UI-parts for these service classes are implemented as plug-ins for Eclipse \(^7\).

We split up the evaluation in three phases. In the first phase, we tested our approach by implementing the conceptual architecture, integrating two collaborative applications, and conducting functional tests. Currently, we are integrating further applications, like Microsoft Exchange or the Java Content Repository, to extend the number of available application types in our context model.

In the second phase, we have setup a test environment and conducted expert walkthroughs in different work scenarios, e.g. collaborative planning or writing. Feedback from these experts indicates both that adaptation is in general accepted as useful but often users also want to understand why a specific adaptation was performed. Some users also reported that they dislike automatic adaptation in general, as they want to stay in complete control of their workspace. However, even these users would appreciate suggestions for adaptation, i.e. semi-automatic adaptations.

In the third phase, we will evaluate our architecture with the integrated collaborative applications in real-world settings that are based on knowledge-intensive processes as they can be found in collaborative modeling or design. We will also collect more feedback on the existing adaptation rules and identify best practices for collaboration leading to further adaptation rules.

7 CONCLUSIONS

In this paper, we introduced an explicit representation of context and a conceptual architecture for context sensing, reasoning, and adaptation in an integrated shared work environment. We integrated first collaborative applications to show the feasibility of our approach.

Our approach exceeds current approaches by defining an ontology-based context model for shared workspaces and an architecture which allows integrating a variety of services to adapt the shared workspace to best meet the current needs of collaborating users. First experiences show the usefulness as well as problems caused by the adaptation.

The first experiences especially highlight the importance of traceability support for the users. Such a support has to allow users understand why an adaptation was performed. For that purpose, we will as a first step include an Activity Log as well as Activity Indicator (Schümer and Lukosch, 2007). But as in addition some users reported that they want to keep the full control of their workspace, we will also focus on enabling users to define their own adaptation rules as well as supporting a process to negotiate adaptations and adaptation rules.

REFERENCES


