Facilitating Group-based Adaptation of Shared Workspaces using a Multi-Agent System

Dirk Veiel
FernUniversität in Hagen, Department of Mathematics and Computer Science, 58084 Hagen, Germany
dirk.veiel@fernuni-hagen.de

Stephan Lukosch
Delft University of Technology, Faculty of Technology, Policy, and Management, Jaffalaan 5, 2628 BX Delft, The Netherlands
s.g.lukosch@tudelft.nl

Martijn Warnier
Delft University of Technology, Faculty of Technology, Policy, and Management, Jaffalaan 5, 2628 BX Delft, The Netherlands
m.e.warnier@tudelft.nl

Michel Oey
Delft University of Technology, Faculty of Technology, Policy, and Management, Jaffalaan 5, 2628 BX Delft, The Netherlands
m.a.oey@tudelft.nl

Jörg M. Haake
FernUniversität in Hagen, Department of Mathematics and Computer Science, 58084 Hagen, Germany
joerg.haake@fernuni-hagen.de

ABSTRACT
In this paper we present an agent-based approach to facilitate context-based adaptations using context information of a group of users. To minimize the effort to collect all relevant information of the current collaboration situation, and the amount of data to be kept, we propose a decentralized, agent-based approach that keeps track of the contextualized state.

Author Keywords
Shared workspaces, adaptation, group context, context-based adaptation, agent, multi-agent system

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: Miscellaneous

INTRODUCTION
In modern organizations work is to a large extent collaborative. Especially knowledge work is increasingly performed by distributed teams cooperating across large, often global distances. While collaboration has become ubiquitous, new challenges occurred that need to be addressed for supporting it effectively. This is especially true when looking at cooperation support systems. These challenges arise from a variety of features that are characteristic for collaboration. Among other aspects, collaborative tasks are often ill-structured, emerge in the course of the collaborative process, and need to respond flexibly to changing goals or situations. Users participating in a collaborative project may find themselves in different physical environments or settings and may use a variety of different devices. Also, users are often involved in more than one project at a time, raising the need for frequent task or tool switches and for rapid cognitive adjustments to the subject at hand.

A shared work environment dealing with the required flexibility, offering the different tools in an integrated fashion, and supporting context-based adaptation that does not only take a single user's context into account but the context of a group or team, can be the solution for these changing needs. Using a service-oriented architecture enables developers and providers of shared work environments to extend existing systems by adding, replacing, or (re-)configuring services and the corresponding UIs offered by different service providers. Usually these actions require manual changes and they can not be applied to a running system without a (re-)initialization. Context-adaptive systems can bridge this gap by supporting (re-)configuration of the system at runtime, by starting and stopping required services and the corresponding UIs, by creating (e.g., based on templates) and opening artifacts to be produced in the current situation. Each of these cases take the current context of the system and the users into account. E.g., as soon as the system recognizes that too many people try to edit a document synchronously, it may change the concurrency control strategy from optimistic to pessimistic.

To be able to recognize these different collaboration situations, the system has to store relevant context information. Relevant means that the system needs at least this information to be able to detect the occurrence of a specified collabo-

Submitted for review to CAICOLL 2010.

ration situation at runtime. When the system has to be able to detect many of these situations at the same time, it has to keep track of almost every interaction of the users with the system. This tracking process can either be implemented on the client or on the server-side. In either way, mapping these interactions to the context (the state layer in [5]) leads to an tremendous amount of data to be managed. In a distributed setting (i.e. when clients track the interactions) the collected data is spread over all clients that participate in the global collaboration. A problem may arise as soon as a group context representation is needed that requires an integration of different user context representations. Usually, this step is necessary to create the current contextualized state of the group. Such an integration leads to enormous communication activities on the underlying network and may lead to conflicts while merging the different context representations.

In the following, we first describe our approach to address the above issues, then discuss related approaches, before we conclude.

APPROACH

We use the four-layered framework for context-based adaptations and the collaboration domain model for describing collaboration environments and collaborative situations as described in [5] to manage the context and handle context-based adaptations. The domain model and its usage for context-based adaptations is described in [13]. To minimize the effort to collect all relevant information of the current collaboration situation, and the amount of data to be kept, we propose a decentralized, agent-based approach (cf. Figure 1) that only keeps track of the contextualized state. We use AgentScape [9] – a multi-agent platform – to implement two different categories of agents: user and group agents.

Conceptual Architecture

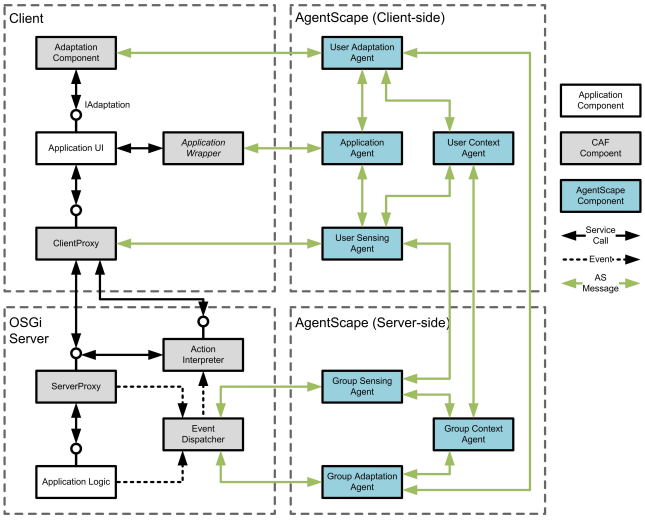

Figure 1. Conceptual architecture

Using this agent-based approach we can minimize the amount of data we have to track and keep as the contextualized state. Tracking or sensing the relevant context information can be done on that side of the system (either at the client or the server) where the information is available. Therefore we split the agents into user and group agents. While creating the group's contextualized state the group agents only acquire information relevant in the current situation, i.e. only information that is required for monitoring adaptation rules.

We can integrate applications using *Application Agents* that run outside our main system (CONTact). Such applications can communicate with the application agent to provide relevant context information. This approach supports the required flexibility of an open collaborative and context-adaptive workspace system.

Furthermore, we are able to create a virtual presence of each user. This can be used, e.g., in negotiations about schedules that affect the corresponding user, to support immediate response to urgent tasks, or to manage privacy issues in relation of the context information that needs to be shared.

Group agents

Group agents support means of communication to corresponding user agents, to collect and integrate the user's contextualized state into one of the group (*Group Context Agent*), retrieve and monitor adaptation rules using the current contextualized state of its group, and send adaptation actions (*Group Adaptation Agent*) to the corresponding user agents (*User Adaptation Agent*). They use a repository to share and retrieve possible adaptation rules. When necessary (i.e. an adaptation requires additional context information from the server-side), the *Group Sensing Agent* subscribes to relevant events the *Event Dispatcher* supports, and gets notified as soon as these events occur. Thereby, we can collect information that affects the group context immediately and independent of the user's context. Adaptations that need to be handled by the *OSGi Server* itself (like (re-)configuration of services, dynamic service orchestration or choreography) are encapsulated in events and sent through the *Event Dispatcher*) to the (*Action Interpreter*) that receives, extracts the adaptation actions from the event, and executes them.

User agents

A user agent collects relevant context information (*User Sensing Agent*), integrates them into their current contextualized state, handles the user's profile / preferences, and deals with specific privacy issues of the user (*User Context Agent*). After the (*User Adaptation Agent*) received adaptation actions from the *Group Adaptation Agent* it deals with possibly conflicting adaptations, and applies the determined adaptation actions to the client by using the *Adaptation Component*. The *Application Agent* offers context information about the application itself. This enables other agents to collect context information directly from it without asking the corresponding application itself. Useful information that affects possible adaptations are the context concept it belongs to (e.g., *Chat*, *RemoteFieldOfVision*), the adaptation actions that can be applied, and the context concepts of awareness functionality the application may offer. After the *Application Wrapper* has started/injected the *Application Agent*, both can check

whether the context information the *Application Agent* offers, match the current setting of the application itself.

AgentScape

The multi-agent platform AgentScape supports agents as autonomous processes. A uniform middleware layer provides an agent run-time that is available at numerous heterogeneous platforms. Agents in AgentScape can communicate with other agents, can access external services and can be mobile, i.e. are able to migrate between hosts.

The guiding principle in the design of the AgentScape middleware has been to develop a minimal but sufficient open agent platform that can be extended to incorporate new functionality or adopt (new) standards into the platform. This design principle has resulted in a multi-layered architecture with (i) a small *middleware kernel*, called the AgentScape Operating System (AOS), that implements basic mechanisms such as communication, (ii) high-level *middleware services* that implement agent platform specific functionality and policies and (iii) external directory services. This approach simplified the design of the kernel and has made it less vulnerable to errors or improper functioning. The current set of middleware services includes agent servers, host managers, location managers, a look-up service and a web service gateway.

DISCUSSION

In this section we briefly review similar approaches, and point out similarities and differences with our approach.

The most prominent examples for context-based adaptation focus on single users and consider location as most relevant context information (e.g., [12, 1, 6, 7]) or focus on learner profiles (cf. ITS). All of these systems are able to identify situations requiring adaptation for a single user. *COLER* [4] provides a software coach for improving collaboration, thus being able to identify adaptation needs during collaborative software development. The *Semantic Workspace Organizer (SWO)* [11] 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.

CoBrA [2, 3] is an agent-based architecture that helps agents to acquire, reason about and share context knowledge. This architecture uses a broker agent that maintains and manages the shared context model. While our approach focusses on collecting, creating, and using the contextualized state of a group of users, they use one broker agent for all computing entities (e.g., PCs, PDAs) in a specified space.

ECOSPACE [10, 8, 14] uses a context model of a group that represents the characteristics related to the users' environment that may affect the way a collaborative activity of any type is executed. The system uses the context model in its dynamic service composition phase at runtime to select appropriate services for service composition. But the context model has not been published yet.

CONCLUSION

In this paper we presented our agent-based approach to facilitate context-based adaptations using context information of a group of users. We use AgentScape – a multi-agent platform – to implement agents that create the contextualized state, and monitor and apply adaptations.

Currently, we are focusing on creating a language which allows end-users to write adaptation rules and introduce new context concepts.

REFERENCES

1. G. D. Abowd, C. G. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton. Cyberguide: a mobile context-aware tour guide. *Wireless Networks*, 3(5):421–433, 1997.
2. H. Chen, T. W. Finin, and A. Joshi. Using owl in a pervasive computing broker. In S. Cranefield, T. W. Finin, V. A. M. Tamma, and S. Willmott, editors, *Proceedings of the Workshop on Ontologies in Agent Systems (OAS 2003)*, pages 9–16, 2003.
3. H. Chen, T. W. Finin, and A. Joshi. Semantic web in the context broker architecture. In *Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom 2004)*, pages 277–286. IEEE Computer Society, 2004.
4. M. de los Angeles Constantino-González and D. D. Suthers. Automated coaching of collaboration based on workspace analysis: Evaluation and implications for future learning environments. In *Proceedings of the 36th Hawai'i International Conference on the System Sciences (HICSS-36)*. IEEE Press, 2003.
5. J. M. Haake, T. Hussein, B. Joop, S. Lukosch, D. Veiel, and J. Ziegler. Context modeling for adaptive collaboration. Technical Report 2/2009, Universitt Duisburg-Essen, jul 2009.
6. T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, B. Serra, and M. Spasojevic. People, places, things: web presence for the real world. *Mobile Network Applications*, 7(5):365–376, 2002.
7. O. Lehmann, M. Bauer, C. Becker, and D. Nicklas. From home to world - supporting context-aware applications through world models. In *PerCom*, pages 297–308, 2004.
8. M. A. Martnez-Carreras, A. Ruiz-Martnez, F. Gmez-Skarmeta, and W. Prinz. Designing a generic collaborative working environment. In *IEEE International Conference on Web Services (ICWS 2007)*, pages 1080–1087, 2007.
9. B. J. Overeinder and F. M. T. Brazier. Scalable Middleware Environment for Agent-Based Internet Applications. In *Proceedings of the Workshop on State-of-the-Art in Scientific Computing (PARA'04)*, volume 3732 of *LNCS*, pages 675–679. Springer, June 2004.

10. W. Prinz, H. Loh, M. Pallot, H. Schaffers, A. Skarmeta, and S. Decker. ECOSPACE – towards an integrated collaboration space for eprofessionals. In *International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pages 39–45, 2006.

11. W. Prinz and B. Zaman. Proactive support for the organization of shared workspaces using activity patterns and content analysis. In *GROUP '05: Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work*, pages 246–255. ACM, New York, NY, USA, 2005.

12. B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *First Annual Workshop on Mobile Computing Systems and Applications (WMCSA)*, Dec. 1994.

13. D. Veiel, J. M. Haake, and S. Lukosch. Extending a shared workspace environment with context-based adaptations. In *Groupware: Design, Implementation, and Use: 15th International Workshop, CRIWG 2009, Peso da Rgua, Douro, Portugal, September 13-17, 2009*, LNCS 5784, pages 174–181. Springer-Verlag Berlin Heidelberg, sep 2009.

14. M. Vonrueden and W. Prinz. Distributed document contexts in cooperation systems. In B. N. Kokinov, D. C. Richardson, T. Roth-Berghofer, and L. Vieu, editors, *Modeling and Using Context, 6th International and Interdisciplinary Conference, CONTEXT 2007*, LNCS 4635, pages 507–516. Springer-Verlag Berlin Heidelberg, 2007.