J. Ziegler & A. Schmidt (Hrsg.): Mensch & Computer 2010
München: Oldenbourg Verlag, 2010, S. 225-234

# Kontext-basierte Adaption von gemeinsamen Arbeitsbereichen

Dirk Veiel[1], Jörg M. Haake[1], Stephan Lukosch[2]

[1]Fakultät für Mathematik und Informatik, FernUniversität in Hagen

[2]Faculty for Technology, Policy and Management, Delft University of Technology

**Zusammenfassung**

Heutzutage kooperieren viele Teams mittels gemeinsamen Arbeitsbereichen, die Dienste zur Unterstützung von Interaktion und Kooperation in der Gruppe anbieten. Die Anforderungen an effektive Gruppeninteraktion verändern sich über die Zeit in Abhängigkeit von der aktuellen Problemstellung und den Gruppenzielen. Ein idealer gemeinsamer Arbeitsbereich muss deshalb Mittel zur Anpassung seiner Dienste bzw. Interaktionsmöglichkeiten an die aktuellen Anforderungen des Teams bieten. Die Analyse von Adaptionsmöglichkeiten in einer Service-orientierten Architektur für gemeinsame Arbeitsbereiche bietet die Grundlage für ihre Erweiterung um Kontext-basierte Adaption. Mittels eines Frameworks für Kontextmodellierung und kontextbasierte Adaption von gemeinsamen Arbeitsbereichen wird deren kontinuierliche Anpassung unterstützt. Ein Prototyp zeigt die Realisierbarkeit dieses Ansatzes.

## 1 Einleitung

Komplexe Probleme, wie z.B. das Schreiben komplexer Dokumente, Produkt-Design oder Softwareentwicklung, werden heute oftmals in Teams unter Nutzung gemeinsamer Arbeitsbereiche gelöst, die Problem- und Gruppen-spezifische Artefakte und Werkzeuge bereitstellen. Je nach Aufgabe oder Phase der Zusammenarbeit variieren die notwendigen Artefakte und Werkzeuge. Deshalb müssen gemeinsame Arbeitsbereiche eine wechselnde Menge von Ein- und Mehrbenutzer-Werkzeugen sowie deren Integration unterstützen. Wechselnde Anforderungen erfordern die Adaption des gemeinsamen Arbeitsbereichs, die zu einer zusätzlichen Belastung der Benutzer führt. Deshalb sollte die Anpassung so einfach wie möglich sein, wobei der Wechsel zwischen komplexen Kooperationssituationen mit Anpassung multipler Werkzeuge und Artefakte unterstützt werden muss.

Aktuelle Ansätze unterstützen ein manuelles Tailoring gemeinsamer Arbeitsbereiche entweder bzgl. der Artefakte (Typen, Instanzen) und Arbeitsbereichsstruktur oder sie erlauben die Adaptation einzelner Werkzeuge an die Bedürfnisse eines einzelnen Benutzers und in selte-

nen Fällen an die Bedürfnisse einer Gruppe. Unterstützung für die Adaptation eines ganzen Arbeitsbereichs an Gruppenbedürfnisse fehlt bis heute.

Wir schlagen eine Erweiterung gemeinsamer Arbeitsbereiche vor, welche die Kontext-basierte Adaption von gemeinsamen Arbeitsbereichen an die Gruppenbedürfnisse unterstützt. Unser Ansatz besteht aus zwei Teilen: 1) einem Framework für Kontextmodellierung und kontextbasierte Adaption, das die Konversion von Werkzeugen in Kontext-adaptive Varianten unterstützt, welche die notwendigen Schnittstellen für eine geeignete Laufzeitumgebung anbieten; 2) einer Laufzeitumgebung, die das Kontextmodell pflegt und darauf Adaptionsregeln zur Anpassung der Konfiguration von gemeinsamen Arbeitsbereichen und des Werkzeugverhaltens ausführt.

In Abschnitt 2 betrachten wir kurz verwandte Arbeiten. In Abschnitt 3 analysieren wir die Adaptionsmöglichkeiten in Service-orientierten gemeinsamen Arbeitsbereichen. Abschnitt 4 präsentiert dann sowohl das Framework für Kontextmodellierung und kontextbasierte Adaption als auch die Architektur der Laufzeitumgebung. Abschnitt 5 beschreibt unsere Erfahrungen, Schlussfolgerungen und offene Fragen.

# 2 Verwandte Arbeiten

Zuerst diskutieren wir Umgebungen für gemeinsame Arbeitsbereiche bzgl. möglicher Adaptionen. Danach betrachten wir Kontext-adaptive Systeme bzgl. ihrer Eignung für die Unterstützung Kontext-adaptive Kooperation in gemeinsamen Arbeitsbereichen.

BSCW (Appelt & Mambrey 1999) und CURE (Haake et al. 2004) sind Vertreter web-basierter gemeinsamer Arbeitsbereiche. Sie bieten eine Menge von Kooperationsdiensten, z.B. für Kommunikation und Teilen von Dokumenten. Diese Systeme fokussieren auf eine spezielle Anwendungsdomäne und bieten eine feste Menge von Kooperationsdiensten und Interaktionsmöglichkeiten. Einige Systeme, z.B. CURE, unterstützen Tailoring, jedoch ohne automatische Adaption der Funktionalität an Gruppen- bzw. Kooperations-Bedürfnisse.

Prominente Beispiele für Kontext-basierte Adaption betrachten hauptsächlich Einzel-Anwender und messen dem Ortsbezug die größte Relevanz zu (z.B. Schilit et al. 1994, Abowd et al. 1997, Kindberg et al. 2002) oder fokussieren auf Lernerprofile (ITS). Diese Systeme können Situationen erkennen, die Adaptionen für Einzelanwender notwendig machen. *COLER* (Constantino-González & Suthers 2003) bietet einen Software-Coach an, um die Kollaboration zu verbessern. Während der kollaborativen Softwareentwicklung kann diese Komponente mögliche Adaptionen erkennen. Der *Semantic Workspace Organizer* (*SWO*) (Prinz & Zaman 2005) ist eine BSCW-Erweiterung, die die Nutzeraktivitäten und die Textdokumente analysiert, um geeignete Ablageorte für neue Dokumente im System vorzuschlagen und den Anwender bei der Suche nach Dokumenten zu unterstützen. Das ECOSPACE Projekt stellt einen integrierten, kollaborativen Shared Workspace zur Verfügung (Martínez-Carreras et al. 2007). ECOSPACE liegt eine Service-orientierte Architektur zugrunde und bietet eine Menge von Kollaborationsdiensten an, die auf Basis einer Ontologie orchestriert oder choreographiert werden können. Die hierfür benötigte Ontologie ist derzeit allerdings

noch nicht veröffentlicht (Vonrueden & Prinz 2007). *CoBrA* (Chen et al. 2004a) ist eine Agenten-basierte Architektur, die einen gemeinsamen Kontext nutzt, um die Service-Agenten an den entsprechenden Anwenderkontext anzupassen. Das zugrunde liegende Kontextmodell basiert auf *SOUPA* (Chen et al. 2004b). Gross und Prinz (Gross & Prinz 2003) stellen ein Kontextmodell und ein Kooperationssystem vor, das Kontext-adaptive Awareness zur Verfügung stellt. (Fuchs 1999) beschreibt einen integrierten synchronen und asynchronen Notifikationsservice für Awareness-Informationen namens *AREA*, der den Kontext nur für Awareness-Informationen nutzt. Diese Beispiele zeigen, das UI Adaptionen üblicherweise auf Einbenutzer-Anwendungen abzielen, oder die UIs nur einem Nutzer angezeigt werden.

Systeme wie *Nexus* (Lehmann et al. 2004) und *CoBrA* (Chen et al. 2004a) fokussieren auf das Ermitteln exogener Informationsquellen, unterstützen allerdings nicht die Ermittlung von Informationen aus den angebotenen Anwendungen heraus.

Die Ubiquitous Computing Community legen häufig ihr Augenmerk auf das Ändern von Konfigurationsparametern (z.B. Bildschirmgröße, Layout), um die Geräte des Nutzers bestmöglich an die aktuellen Gegebenheiten/Bedürfnisse anzupassen. In diesen Szenarios werden lediglich einzelne Nutzer betrachtet, d.h. Änderungen an der Konfiguration eines Geräts beeinflussen die anderen Clients/Nutzer nicht.

Die zuvor genannten Ansätze zielen bzgl. Kontextrepräsentation und Adaptionen auf spezielle Domänen ab (z.B. Einbenutzer-Systeme, ITS und CSCW-Subdomänen). Nur *ECOSPACE* bietet die Möglichkeit, Kollaborationsdienste in das bestehende System zu integrieren. Ebenso intendiert nur *ECOSPACE* Adaptionen auf Basis des Gruppenkontexts und für mehrere Nutzer eines Kooperationssystems, wobei das zugrundeliegende Kontextmodell noch nicht veröffentlicht ist. Aktuelle Ansätze unterstützen Kontext-basierte Adaption von gemeinsamen Arbeitsbereichen deshalb nur in ungenügender Art und Weise.

# 3 Adaptionsmöglichkeiten in Service-orientierten, gemeinsamen Arbeitsbereichen

Eine kollaborative Anwendung nutzt üblicherweise das Model-View-Controller Paradigma (Krasner & Pope 1988) und kann auf einer Client-Server-Architektur basieren. Heutzutage werden Service-orientierte Client-Server-Architekturen verwendet, um Kollaborationsdienste für kollaborative Anwendungen zu entwickeln. Wir teilen diese Art von Architekturen in vier Schichten auf: *UI*, *Logic*, *Services* und *Model Layer* (siehe Abbildung 3). Die View- und Controller-Bestandteile der Anwendungen ($Application_X$ *UI*) werden üblicherweise auf der Client-Seite (*UI Layer*) ausgeführt. Der *Logic Layer* besteht aus der anwendungsspezifischen Business-Logic ($Application_X$ *Logic*), die auf Basis diverser Services aus dem *Service Layer* aufgebaut sein kann. Der *Service Layer* enthält sowohl *Application Services* als auch *Collaboration Services*. Er wird von der höher liegenden Schicht genutzt, um auf die Artefakte ($Artifact_X$) aus dem *Model Layer* zuzugreifen. Sobald mehr als eine Anwendung vorhanden ist, sprechen wir von einer gemeinsamen Arbeitsumgebung.

Legt man diese Schichten zugrunde, kann man bei Adaptionen eines gemeinsamen Arbeitsbereichs vier Bereiche unterscheiden. (1) *Modifikation der Anwendungsoberflächen:* Üblicherweise lässt sich das UI als Hierarchie von visuellen Komponenten modellieren, woraus sich folgende Adaptionsmöglichkeiten ergeben: (i) hinzufügen, entfernen und ersetzen kompletter visueller Komponenten in der Hierarchie; (ii) ersetzen ausgewählter Views und Controllers. (2) *Modifikation der Anwendungslogik*: Hier könnte man entweder die interne Struktur der Anwendungslogik ändern, indem man Dienste, die von der Anwendung genutzt werden, hinzufügt, entfernt oder durch andere Dienste ersetzt, oder die Ausführungsstruktur der Dienste ändern. (3) *Modifikation der Dienste*: Hier können sowohl Anwendungsdienste als auch Kollaborationsdienste hinzugefügt, entfernt oder ersetzt werden. Dies beeinflusst die Art und Weise, wie die Nutzer miteinander kommunizieren, sich koordinieren und Objekte gemeinsam nutzen. (4) *Modifikation des gemeinsamen Modells*: Adaptionen können hier z.B. Artefakte und Sessions anpassen. Artefakte bzw. Dokumente können erzeugt oder gelöscht und deren Attribute verändert werden. Sessions können erzeugt und gelöscht, Teilnehmer, Anwendungen oder Artefakte hinzugefügt oder entfernt werden.

Wir nutzen nun obige Adaptionsmöglichkeiten um eine Architektur zur Spezifikation und Implementierung solcher Adaptionen zu definieren und führen ein Framework ein, das Entwickler bei der Erweiterung von Anwendungen um Adaptionsfunktionalitäten unterstützt.

# 4 Adaptionen in Service-orientierten, kollaborativen Anwendungen

Nachfolgend zeigen wir das zugrundeliegende Kontextmodell, auf Basis dessen die Bedingungsteile der Adaptionen in unserem *Context and Adaptation Framework (CAF)* (Lukosch et al. 2009) definiert und (zur Laufzeit) erkannt werden können. Das Kontextmodell berücksichtigt Szenarien, in denen mehrere Akteure kollaborieren, um ein gemeinsames Ziel zu erreichen. Es reflektiert grundlegende Konzepte für kollokierte und verteilte Kollaboration.

Abbildung 1 zeigt die grundlegenden Konzepte und Relationen unseres Kontextmodells. Eine *Application* implementiert das Model-View-Controller (MVC) Paradigma (Krasner & Pope 1988) und besteht aus *View* und *Controller* Komponenten. *Views* und *Controller* nutzen *Services*, um auf *Artifacts* zuzugreifen. *Artifacts* nutzen *Services*, um *Views* und *Controller* über Änderungen zu benachrichtigen. Jede *Application* ist Bestandteil eines *User Workspace* und wird durch eine *Application Factory* erzeugt. Die *Application Factory* spezifiziert, welche *Applications* im Arbeitsbereich zur Verfügung stehen und wie diese initialisiert werden können. Jeder *Actor* besitzt einen *User Workspace* und gehört mindestens zu einem *Team*. Der *User Workspace* definiert die *Roles* eines *Actors* innerhalb eines *Teams*. Jede *Role* erlaubt es dem *Actor*, spezifische *Actions* innerhalb einer *Application* auszuführen. Die innerhalb einer *Application* verfügbaren *Actions* werden durch die zugehörige *Application Functionality* definiert. *Actors* interagieren mit der *Application*, indem sie durch die *Role* erlaubte *Actions* ausführen. Diese *Actions* werden von der entsprechenden *Controller* Komponente einer *Application* empfangen. Die Klasse *Application Functionality* hat diverse Un-

terklassen, wie z.B. *Communication*, *Shared Editing (SE)*, *Awareness*, *Management* oder *Workflow Management*. Diese sind aus Gründen der Übersichtlichkeit nicht in Abbildung 1 zu sehen. Diese Klassen werden eingesetzt, um sowohl Funktionalitäten (z.B. *Chat*, *Text Editor*, *Remote Field of Vision*, *Concurrency Control Management*) als auch unterstützte Aktionstypen (beim Chat beispielsweise *OpenChat u*nd *SendMsg*) zu spezifizieren.

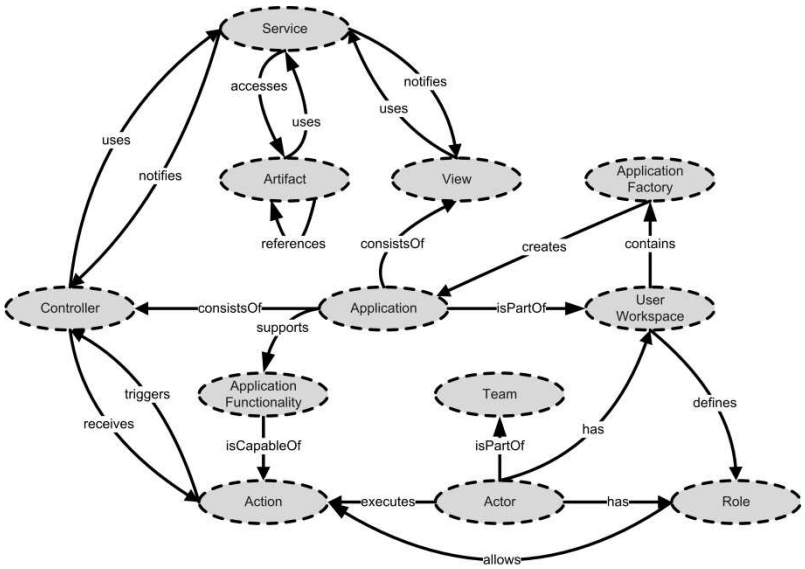


*Abbildung 1 Kontextmodell für Kollaboration*

Die zuvor genannten Klassen sind hilfreich, um die Anwendungen und Konfigurationen eines gemeinsamen Arbeitsbereichs zu modellieren und den aktuellen Zustand zur Laufzeit zu erfassen. Als Beispielszenario nehmen wir an, dass ein verteiltes Autorenteam, bestehend aus Alice, Bob und Dave, gemeinsam einen Artikel schreibt. In unserem Szenario treffen sich die Autoren informell, um den Gegenstand des Artikels, die Grobstruktur und die Arten der Zusammenarbeit auf Basis von der Organisation vordefinierten Adaptionsrichtlinien festzulegen. Es herrscht Konsens, dass Alice der Erstautor ist, der sowohl die Koordination als auch das Einrichten der Kollaborationsumgebung übernimmt.

Eine Adaptionsrichtlinie modelliert gewünschtes Kooperationsverhalten und impliziert passendes Verhalten von Benutzern und System. Sie besteht aus einer Menge von Adaptionsregeln zur Anpassung der Systemkonfiguration und dazugehörigen Gruppen- bzw. Teampräferenzen bzgl. darin verwendeter Anwendungen, Dienste und deren Konfiguration. Mit dem System werden vordefinierte Adaptionsrichtlinien ausgeliefert, die von den Entwicklern erstellt wurden. Während der Nutzung des Systems können Situationen auftreten, die Änderungen dieser Richtlinien notwendig machen. Adaptionsregeln können nur von Experten geändert werden, welche die Domäne, den Prozess und die Möglichkeiten des Systems ken-

nen. Gruppenpräferenzen bzgl. der Anwendungen, Dienste und deren Konfiguration können von den Teammitgliedern selbst angepasst werden.

In unserem Beispiel wurden die ausgewählten Richtlinien von der zugehörigen Organisation auf Basis der Erfahrungen von vorhergehenden Autorenteams vordefiniert. Diese Erfahrungen können z.B. mittels Feedback über den genutzten Schreibprozesses und der eingesetzten Adaptionsrichtlinien gesammelt werden. Ein Expertenteam kann auf dieser Basis potenzielle Verbesserungsmöglichkeiten finden (z.B. unpassende oder nicht vorhandene Anwendungen, Dienste oder Awareness Widgets identifizieren). Mittels eines Regeleditors können solche Änderungen spezifiziert und zur Laufzeit durchgeführt werden.

Während die Teilnehmer in unserem Beispiel ihre Gruppenpräferenzen festlegen, diskutieren sie mögliche Einstellungen. Sie sind sich darüber einig, dass der gemeinsame zeitgleiche Zugriff (*Co-Access*, d.h. mindestens zwei von ihnen greifen zur gleichen Zeit auf dasselbe Artefakt zu) eine Situation darstellt, über die sie sich immer bewusst sein wollen. Aus diesem Grund legen sie fest, dass beim *Co-Access* auf den Arbeitsbereich des Artikels ein Chat (Anwendungsklasse *SynchronousCommunication*) genutzt werden soll.

Wir gehen nachfolgend davon aus, dass Alice die Kollaborationsumgebung eingerichtet, den Schreibplan verteilt hat und dieser nun umgesetzt wird. Sie öffnet den Arbeitsbereich, um mit ihrer Arbeit zu beginnen. Bob öffnet kurz darauf ebenfalls den Arbeitsbereich. Das System repräsentiert den aktuellen Zustand im Kontextmodell. Um in dieser Situation (*co-access*) einen Chat zu etablieren, ist folgende Adaptionsregel notwendig:

```
rule "open synchronous communication channel"
 when
  artifacts: getArtifactsInContext("OpenWorkspace:bob")
  actors: getActorsInContext(artifacts)
  comms: getApplicationsInContext(actors, "SynchronousCommunication")
  selectedApplication: selectFirst(comms)
 then
  openForAll(selectedApplication, actors)
end
```

Die obige Adaptionsregel besteht aus einem Bedingungs- und einem Aktionsteil, der wiederum mehrere Adaptionsaktionen enthalten kann. Die Auswertung des obigen Bedingungsteils wird durch Bob ausgelöst, sobald er den Arbeitsbereich öffnet. `getArtifactsInContext` liefert eine Menge von Artefakten zurück, die sich im Kontext der Aktion *OpenWorkspace: bob* befinden. Die Funktion `getActorsInContext` ermittelt auf Basis des Kontexts alle Nutzer, die ebenfalls dieses Artefakt geöffnet haben. Die synchronen Kommunikationsmöglichkeiten der Nutzer werden über die Funktion `getApplicationsInContext` ermittelt. Hierbei gehen die vom Team im Kontext festgelegten Präferenzen (Chat) mit ein, so dass die Funktion `selectFirst` die Anwendung Chat zurückliefert. Falls der Bedingungsteil keine leere Menge ergibt, wird der Aktionsteil ausgeführt. Im Aktionsteil der Adaptionsregel wird anschließend bei allen beteiligten Nutzern der Chat geöffnet (siehe Abbildung 2).


*Abbildung 2 Arbeitsumgebung von Bob; der neu erzeugte Chat ist unten rechts zu sehen.*

Wie können nun existierende bzw. neue Anwendungen einer gemeinsamen Arbeitsumgebung für Adaptionen vorbereitet werden? Unser Ansatz erweitert herkömmliche Client-Server-Anwendungen, basierend auf der Architektur aus Abschnitt 3, indem er Komponenten unseres *CAFs* (in Abbildung 3 grau unterlegt) hinzufügt. Die *UI Layer* umfasst die *Application UIs* und die *Adaptation Component* von *CAF*. Wir nutzen die *Adaptation Component*, um *Application UIs* zu starten und zu stoppen, oder eine von der Anwendung angebotene Adaptionsschnittstelle anzusprechen, über die folgende Adaptionsaktionen ausgeführt werden können: GUI-Komponenten anzeigen, verbergen, darauf den Fokus setzen, deren Inhalt verändern, sie hervorheben, Views maximieren/minimieren, oder den Read-Only-Modus zu aktivieren.

Der *Logic Layer* enthält sowohl die *Application Logic* als auch den *Adaptation Server*. Das *CAF* fügt die Komponenten *Basic Services* und *Notifier* zur *Application Logic* hinzu. Die *Application Logic* kann *Basic Services* nutzen, um Anwendungen in das *CAF* zu integrieren, oder den *Notifier* nutzen, um Notifikationen an Clients zu verschicken. Der Entwickler kann *Basic Services* nutzen, um z.B. Sensing-Funktionalität, Verwaltung von Zugriffsrechten und Nutzern sowie Datenbankzugriffe in seine Anwendung zu integrieren und dadurch eine Brücke zum *Adaption Server* zu bauen. Außerdem können *Basic Services* z.B. die Konfiguration der *Application Logic* ändern. Der *Adaptation Server* basiert ebenfalls auf einer Service-orientierten Architektur und beinhaltet Komponenten wie die *Sensing Engine* und die *Adaptation Engine*. Die *Sensing Engine* nutzt Informationen über die Serviceaufrufe und Nutzerinteraktionen um den Kontextzustand zu aktualisieren. Die *Adaptation Engine* nutzt diesen um adäquate Adaptionsregeln zu selektieren und ausführen zu lassen. Adaptionsaktionen

können sich auf *Application UI*, *Application Logic*, *Services* und das *Shared Model* beziehen. Der *Adaptation Server* nutzt *Context Services* (*CAF Services*), um auf das *Context Model* (*CAF Model*) zuzugreifen.

Das *Adaptation Runtime Environment* (rechts in Abbildung 3) erlaubt es Adaptionsregeln auszuführen, welche die Konfiguration des gemeinsamen Arbeitsbereichs und das Verhalten der Anwendungen über entsprechende Adaptionsschnittstellen anzupassen. Es unterteilt sich dabei in die Schichten *Adaptation Server*, *Context Services* und *Context Model*.

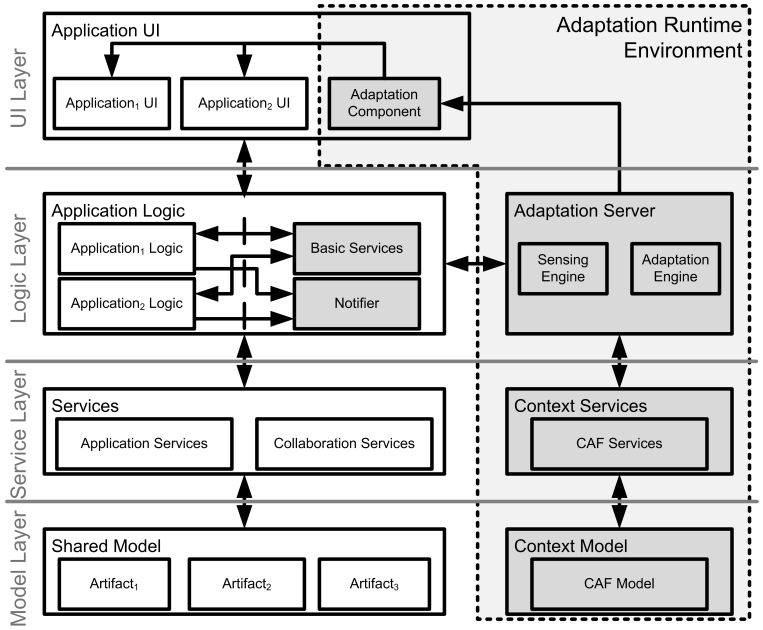


*Abbildung 3 Um das* Context and Adaptation Framework (CAF) *erweiterte Client-Server Architektur*

Ein flexibles, adaptives System durchläuft eine Schleife von (1) Nutzerinteraktion, (2) Ermittlung der Nutzeraktivitäten (*Sensing*), (3) Adaption des Systemverhaltens und (4) Anpassung des Adaptionswissens (z.B. Änderung einer Adaptionsregel). Der Nutzer interagiert mit den Anwendungen des gemeinsamen Arbeitsbereichs. Üblicherweise führt dies zu Serviceaufrufen, die genutzt werden können, um die Kontextrepräsentation zu aktualisieren. Die *Adaptation Engine* nutzt diese Kontextrepräsentation um adäquate Adaptionsregeln zu selektieren und auszuführen.

# 5 Zusammenfassung

In diesem Beitrag präsentierten wir eine Service-orientierte Architektur für gemeinsame Arbeitsbereiche und analysierten prinzipielle Adaptationsmöglichkeiten auf vier Ebenen (UI,

Logik, Service, und Modell). Wir führten ein Kontextmodell und eine Syntax für Adaptionsregeln zur Definition Kontext-basierter Adaption gemeinsamer Arbeitsbereiche ein. Mittels des vorgestellten Frameworks unterstützen wir die Integration Service-orientierter Werkzeuge in unsere Adaptions-Laufzeitumgebung.

Unser Ansatz geht über aktuelle Ansätze hinaus (vgl. Abschnitt 2): (1) das Kontextmodell und das Sensing unterstützt die Repräsentation des aktuellen Kooperationszustands für eine Gruppe von Nutzern des gemeinsamen Arbeitsbereichs; (2) der Ansatz nutzt diese Kontextinformation zur Anpassung der Interaktionsmöglichkeiten zwischen den Nutzern sowie zwischen Nutzern und ihren Werkzeugen und beeinflusst so das mögliche Kooperationsverhalten; (3) der Ansatz ist offen für Erweiterung um neue Services, Werkzeuge und Adaptionsregeln. Durch Erweiterung des Kontextmodells sind neue Regeln und Berücksichtigung neuer Kooperationsaspekte möglich, ohne alte Regeln zu beeinflussen.

Die vorliegende Implementierung des Frameworks und der Laufzeitumgebung in Eclipse[38] unter Verwendung von R-OSGi[39] und RMI und die Integration mehrerer Werkzeuge (Chat, Skype, Workspace- und Document-Management, Shared Editing) zeigt die Realisierbarkeit unseres Ansatzes. Funktionale Tests mit typischen Kooperationsszenarien zeigen die korrekte Funktion der Adaptionsfunktionalität. Offene Fragen umfassen Werkzeuge für die Nachvollziehbarkeit von Adaptionen, das Entwerfen von Adaptionsregeln, die Verhandlung bei unerwünschten Adaptionen, das Testen von Adaptionsregeln in realen Arbeitssituationen, um daraus evtl. Best Adaptation Practice ermitteln zu können und die Optimierung der Regelausführung. Als aktuellen Ansatz (Kolfschoten et. al 2010) verfolgen wir hier die Möglichkeit, Kollaborationsprozesse zu modellieren und zu simulieren und so Aussagen über die Effektivität von Adaptionen zu machen.

# 6 Literaturverzeichnis

Abowd, G. D., Atkeson, C. G., Hong, J., Long, S., Kooper, R., & Pinkerton, M. (1997). Cyberguide: a mobile context-aware tour guide. *Wireless Networks*, 3(5):421–433.

Appelt, W. & Mambrey, P. (1999). Experiences with the BSCW shared workspace system as the backbone of a virtual learning environment for students. In *Proc. of ED-MEDIA99*.

Chen, H., Finin, T. W., & Joshi, A. (2004a). Semantic web in the context broker architecture. In *Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom 2004)*, 277–286. IEEE Computer Society.

Chen, H., Perich, F., Finin, T., & Joshi, A. (2004b). Soupa: Standard ontology for ubiquitous and pervasive applications. In *Mobile and Ubiquitous Systems: Networking and Services, 2004*. 258–267.

[38] http://www.eclipse.org

[39] http://r-osgi.sourceforge.net

Constantino-González, M. & Suthers, D. D. (2003). 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.

Fuchs, L. (1999). AREA: a cross-application notification service for groupware. In *ECSCW'99: Proceedings of the sixth conference on European Conference on Computer Supported Cooperative Work*, 61–80. Kluwer Academic Publishers.

Gross, T. & Prinz, W. (2003). Awareness in context: A light-weight approach. In *Proceedings of the Eights European Conference on Computer-Supported Cooperative Work - ECSCW 2003*, 295–314. Kluwer Academic Publishers, NL.

Haake, J. M., Schümmer, T., Haake, A., Bourimi, M., & Landgraf, B. (2004). Supporting flexible collaborative distance learning in the CURE platform. In *Proceedings of the Hawaii International Conference On System Sciences (HICSS-37)*. IEEE Press.

Kindberg, T., Barton, J., Morgan, J., Becker, G., Caswell, D., Debaty, P., Gopal, G., Frid, M., Krishnan, V., Morris, H., Schettino, J., Serra, B., & Spasojevic, M. (2002). People, places, things: web presence for the real world. *Mobile Network Applications*, 7(5):365–376.

Kolfschoten, G.; Lukosch, S. & Seck, M. (2010). Modeling Collaboration Processes to Understand and Predict Group Performance. *Proceedings of the IUI workshop on Semantic Models for Adaptive Interactive Systems (SEMAIS)*

Krasner, G. E. & Pope, S. T. (1988). A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80. *Journal of Object-Oriented Programming*, 1(3):26–49.

Lehmann, O., Bauer, M., Becker, C., & Nicklas, D. (2004). From home to world - supporting context-aware applications through world models. In *PerCom*, 297–308.

Lukosch, S., Veiel, D., & Haake, J. M. (2009). Enabling context-adaptive collaboration for knowledge-intense processes. *Proceedings of the 11th International Conference on Enterprise Information Systems (ICEIS 2009)*, volume HCI, 34–41. INSTICC, Portugal.

Martínez-Carreras, M. A., Ruiz-Martínez, A., Gómez-Skarmeta, F., & Prinz, W. (2007). Designing a generic collaborative working environment. In *IEEE International Conference on Web Services (ICWS 2007)*, 1080–1087.

Prinz, W. & Zaman, B. (2005). 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*, 246–255. ACM, USA.

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

Vonrueden, M. & Prinz, W. (2007). Distributed document contexts in cooperation systems. In *Modeling and Using Context, 6th International and Interdisciplinary Conference, CONTEXT 2007*, LNCS 4635, 507–516. Springer.