

CoEUD - Component-based End User Development

Gunnar Stevens; Volker Wulf
Fraunhofer Institut für Angewandte Informationstechnik FIT
Schloss Birlinghoven
53754 Sankt Augustin

Kurzfassung

Das CoEUD Projekt beschäftigt sich mit der Entwicklung und Umsetzung von durch den Endbenutzer dynamisch anpassbaren Infrastrukturen komponenten-basierter Systeme und Services für qualifizierte Arbeitsbereiche. Es stellt damit die notwendige Ergänzung zur Forschung nach technischer Flexibilisierung von IT-Systemen dar. Durch den Ansatz einer benutzer-zentrierten Forschung und Entwicklung soll das Know-How der Endbenutzer optimal im Entwicklungsprozess berücksichtigt werden.

1. Einleitung

1.1 Allgemeine Situation

Die neu aufkommenden Organisationsformen, die mittels veränderten Arbeitsprozessen flexible Reaktionen auf eine sich ständig ändernde Umwelt erlauben und der in Wirtschaftswissenschaften anzutreffende Trend, die Flexibilisierung von Geschäftsprozessen in den Vordergrund zu stellen, hat auch bei der Gestaltung von IT-Anwendungslandschaften zu einem grundlegenden Wandel geführt. Zentralistische und monolithische IT-Systeme werden zunehmend durch service-basierte Applikationen ergänzt bzw. ersetzt. Dadurch soll es möglich sein, je nach Bedarf die Services zu neuen Applikationen neu zu orchestrieren.

Doch die alleinige technische Fokussierung wird nicht ausreichend sein. So hat die Forschung, die durch das Produktivitätsparadox angeregt wurde unser Verständnis für die wirtschaftlichen Effekte von Informationstechnologien enorm verbessert. So zeigen Brynjolfsson und Hitt, dass die meisten Produktivitätseffekte nicht direkt auf Computer-Anwendungen zurückzuführen sind, sondern auf die spezifische Aneignung der Artefakte in der organisationalen Umwelt [Brynjolfsson 1998]. Aneignung von Technologien wird dabei als kreativer Prozess des Anwenders – sowohl Organisation, als auch Endbenutzers – verstanden, der auch über die zugehörigen intendierten Nutzungsmöglichkeiten der Softwareentwickler hinausgehen kann [Pipek 2005]. Der erfolgreiche Prozess der Technologie-Aneignung führt dabei zu neuen Arbeitsweisen, innovativen Geschäftsprozessen oder veränderten organisatorischen Strukturen, aus denen wiederum neue IT-Anforderungen entstehen [Orlikowsky 1996, Pipek/Wulf 1999].

Deshalb muss ein Ansatz, der eine Flexibilisierung von IT-Systemen zur Erhöhung der Reaktionsfähigkeit von Organisationen zum Ziel hat, sich mit den Dynamiken des Anwendungskontexts auseinandersetzen. Daneben muss die Ausdifferenzierung von Anforderungen aus verschiedenen Anwendungsfeldern berücksichtigt werden. Insbesondere sollte der Ansatz das Potential der Endbenutzer fruchtbar machen, die aufgrund ihrer Praxiskenntnis und ihrer Nutzungserfahrung im kreativen Handeln innovative Lösungen entwickeln [von Hippel 1988]. Entsprechend gilt es, den in den letzten Jahren zunehmenden Trend der Flexibilisierung von

IT-Systemen im Software Engineering aufzugreifen und im Sinne des „easy-to-adapt“-Prinzips des End User Developments weiterzuentwickeln [Lieberman et al. 2006].

1.2 Ziele des Projekts

Das Ziel des Projekts ist es, hierzu ein integriertes Gesamtkonzept zu entwickeln, dass die zunehmende Komplexität von IT-Infrastrukturen für den Endbenutzer handhabbar macht und insbesondere fachlich hoch qualifizierte Endbenutzer in die Lage versetzt die Potentiale, die die zunehmende Flexibilität moderner Software-Architekturen bietet, zur Verbesserung ihrer alltäglichen Arbeitspraxis zugänglich zu machen. Entscheidend für das Projekt wird es sein, McIllroy's Vision eines Marktes frei kombinierbarer Anwendungskomponenten und -services [McIllroy, 1968; Szyperski, 1998] mit Konzepten des End User Development (EUD) [Sutcliffe/Mehandjiev 2004] zu verbinden.

Mit dem Ansatz des Component-based EUD greift das Projekt ein innovatives Thema mit hohem Potential in einer frühzeitigen Phase seiner Entwicklung auf. Durch die Integration von Grundlagen-orientierter Forschung zum End User Development und in die Integration in produktionsreife Technologien, wie Eclipse als universelles Tool und Rich Client Plattform, wird eine Win-Win Situation für Wissenschaft und Praxis geschaffen.

Die Wissenschaft erlangt hierüber die Möglichkeit theoretische Erkenntnisse zum EUD in Real-World Settings zu erproben und ggf. zu falsifizieren. Auf Grundlage der gesammelten Erkenntnisse lassen sich so im Projekt bestehende Ansätze zum EUD weiterentwickeln und ergänzen. Auf der anderen Seite ermöglicht das Projekt der Industrie frühzeitig innovative Lösungsansätze für das zukünftig immer stärker werdende Problem zu entwickeln, wie komplexer werdende IT-Landschaften für den Endbenutzer handhabbar gemacht werden können und wie sich Potentiale flexibler Infrastruktur für die Praxis produktiv umgesetzt werden können

1.3 Relevante Forschungsfelder

1.3.2 Konzepte zur Flexibilisierung aus dem Bereich des Software Engineerings

Als Alternative zu den sogenannten schwergewichtigen Vorgehensweisen als Varianten des Wasserfall-Modells [Boehm 1976; Droschel 1999] oder auch nach dem Unified Process/Rational Unified Process (UP/RUP) werden in der Literatur die leichtgewichtigen oder agilen Methoden vorgeschlagen. Neben dem bis heute bekanntesten Vertreter, dem eXtreme Programming [Beck 2000], gehören dazu z.B. SCRUM oder Chrystal [Martin 2002; Larman 2003].

Konzeptionell ist allen Ansätzen gemein, dass sie ein gebrauchstaugliches Softwaresystem und seine anwendungsorientierte Konstruktion in den Vordergrund stellen und weitestgehend auf zusätzliche Managementdokumente und technische Modelle verzichten. Ein Softwaresystem soll in raschen Iterationen nach den fachlichen Vorgaben seiner Anwender entwickelt werden. Ausgehend von festen Budgets und Zeithorizonten sollen sich Entwickler und Kunden über die Inhalte und Prioritäten der zu realisierenden Systemmerkmale abstimmen, wobei die Kunden den Prozess bestimmen.

Insgesamt lässt sich feststellen, dass agile Vorgehensweisen die Theorie und Praxis der Softwaretechnik in den letzten Jahren wesentlich beeinflusst und verändert haben (siehe auch www.agilealliance.org). Obwohl die Ansätze sehr anwendungs- und benutzerorientiert sind, wird in diesem Kontext eine klare Trennung zwischen Benutzern/Anwendern und Entwick-

lern aufrechterhalten. Zu untersuchen ist, wie agile Softwareentwicklung und benutzergetriebene Softwareanpassung zusammenpassen.

In der Softwaretechnik hat sich die Diskussion über Software-Architekturen [Shaw/Garlan 1996; Clements et al. 2001; Bass et al. 2003] als eine eigenständige konzeptionelle Ebene der Softwareentwicklung etabliert. Den grundlegenden Problemen bei der Weiterentwicklung großer Software, wie mangelnde Änderbarkeit und Strukturverfall, soll eine architekturgetriebene (Weiter-) Entwicklung entgegenwirken [Jacobson et al. 1998; Fowler 2003]. Bei architekturgetriebener Entwicklung werden explizite Architekturmodelle eingesetzt und die Einhaltung der Regeln und Strukturmerkmale kontrolliert. Das sog. Refactoring [Fowler 2001; Kerievsky 2004; Feathers 2004] als Teil der agilen Methoden zielt auf die Wiederherstellung sauberer Architekturkomponenten unter Beibehaltung der bereits realisierten Funktionalität ab.

Bisher wurde eher am Rande diskutiert, wie gut sich unterschiedliche Modellarchitekturen [Evans 2003; Züllighoven et al. 2004] strukturerhaltend ändern lassen. Aktuell werden Modellarchitekturen, z.B. Service-Architekturen, unter dem Aspekt ihrer Änderbarkeit diskutiert. Die Diskussion wird besonders bei agilen Methoden verstärkt relevant, weil sie generell fixierte Design-Entscheidungen auch bei Architekturen ablehnen. Offen ist die Frage, ob die schrittweise Entwicklung einer konkreten Software-Architektur im Rahmen einer frühzeitig festgelegten Modellarchitektur stattfinden sollte.

Als Kriterium für die Güte änderungsfreundlicher Architekturen gilt die Separation of Concerns, also die Trennung von Themen und Zuständigkeiten. Als wesentliche Zuständigkeitsbereiche oder Dimensionen werden genannt: Die eingesetzten Technologien, die Fachlogik und die Interaktion mit dem System (auch Benutzungsmodell genannt). Hier wird z.B. vorgeschlagen, dass die Elemente einer Architektur jeweils nur Entwurfsentscheidungen einer Dimension realisieren sollen, um nach dem Geheimnisprinzip Änderungen möglichst lokal zu halten [Züllighoven et al. 2004].

Neben diesen strukturellen Fragen, hat insbesondere Szyperski die organisationalen und marktökonomischen Aspekte des Komponenten Paradigma beleuchtet [Szyperski 1998].

1.3.1 End User Development

Die Forschung zu Systemen, die Endnutzern Möglichkeiten zur Anpassung an veränderte Anforderung nach der eigentlichen Entwicklung bieten [Teege 1998; Kahler et al. 1999; Wulf 2001, Lieberman et al. 2006], ist bisher im Wesentlichen aus drei Richtungen betrieben worden: Zum einen wurden Domänen spezifische Programmiersprachen entwickelt, die von Nicht-Programmierern benutzt werden können [Nardi 1993; Eisenberg 1995; Fischer 1997]. Eine zweite Forschungsrichtung hat den Ansatz ‚Programming by Example‘ entwickelt. Dabei zeichnet ein Software-Agent die Interaktionen des Endbenutzers auf und kreiert ein Programm, das diesen Interaktionen entspricht [Myers 1990; Repenning 1993; Lieberman 2001]. Die dritte hier relevante Forschungsrichtung ist die Adaptierung von Anwendungen durch Parametrisierung ihres Verhaltens.

Empirische Untersuchungen haben Anfang der 90er Jahre bereits Bedingungen identifiziert, unter denen Endbenutzer Anwendungen anpassen [Mackay 1990, Nardi/Miller 1990, Oppermann 1994]. Darüber hinaus haben Studien auf die Bedeutung kooperativer Netzwerke bei der Anpassungen und der Aneignung von Systemen hingewiesen [Mackay 1991; Wulf 1999]

Der Ansatz der komponenten-basierten Anpassbarkeit [Stiemerling 2000; Won/Wulf 2003] verbindet das Gebiet des End User Development mit Ansätzen aus der Forschung zu Software-Architekturen [Shaw/Garlan 1996; Szyperski 1998; Clements et al. 2001; Bass et al.

2003]. Ziel ist es, dem Endbenutzer das zugrunde liegende Komponentennetzwerk so zugänglich zu machen, dass er es interaktiv an die speziellen Bedürfnisse des Anwendungskontexts anpassen kann.

Aus der Sicht des EUD kommt dem Übergang von der Nutzung zur Anpassung der technischen Infrastruktur eine entscheidende Rolle hinzu [de Souza, 2001]. Entsprechend besitzt die Verstehbarkeit der Anpassungsoptionen und des zugrunde liegenden Softwaresystems einen hohen Stellenwert. Deshalb sollte schon die Komponenten- und Service-Infrastrukturen EUD-orientiert gestaltet werden [Stevens/Wulf 2002; Mørch et al. 2004]. Ansätze aus benutzer- und anwendungszentrierter Modellierung [Züllighoven et al. 2004] können dabei helfen, die Kluft zwischen Endbenutzer und den anzupassenden Softwarekomponenten zu verringern

Bei der Anpassung und Zusammensetzung von Komponenten- und Service-Infrastrukturen durch den Endbenutzer besteht jedoch noch erheblicher Forschungsbedarf [Mørch et al. 2004; Klann 2004; Lieberman et al. 2006]. Speziell die Aspekte der Human Factors werden in der Forschung zu Software-Architekturen meist zu wenig berücksichtigt. Umgekehrt gilt für das End User Development, dass eine integrierte Sichtweise auf die Thematik, die sowohl die Anwender- als auch die Herstellerperspektive berücksichtigt, noch in den Anfängen steckt.

2. Projektstatus

Das Projekt versteht die Flexibilisierung von IT-Systemen als einen Teil einer integrierten Technik und Organisationsentwicklung [Wulf/Rohde 1995], der sich in die drei Aspekte Software Engineering flexibler IT-Systeme, Gestaltung von Benutzer zentrierten Anpassungswerkzeugen und Infrastrukturen zur Unterstützung von Aneignungsprozessen unterteilen lässt.



Abbildung 1 Aufbau des Projekts CoEUD

Entsprechend hat sich das Projekt eine drei Säulen Architektur gegeben, dass die verschiedene Aspekte von CoEUD behandelt (vgl. Abbildung 1). Die drei Säulen von CoEUD werden dabei auf der Grundlage komplexer, realistischer Anwendungsszenarien fundiert und im weiteren Projektverlauf durch prototypische Implementierungen auf eine empirische Basis gestellt. Darüber hinaus wird ein integriertes Referenzmodell entwickelt, dass die verschiedenen Perspektiven zusammengeführt und Querverweise und Abhängigkeiten zwischen den Perspektiven darlegt. Insbesondere wird dargelegt, wie ein EUD-orientierter Entwicklungsprozess verwirklicht werden kann.

2.1 Projektverlauf

Das Projekt läuft vom 3/2006 – 2/2009 und befindet sich in der initialen Phase in der Anwendungsszenarien und erste Umsetzungskonzepte entwickelt werden. Dabei geht es im ersten Schritt darum, Chancen, Strategien und Barrieren von CoEUD für die Praxis zu ermitteln. Dabei untersucht das CoEUD Projekt die Frage einmal aus der Sicht des Endbenutzers und ergänzend hierzu aus der Sicht des Herstellers.

Aus der Sicht des Endbenutzers wird untersucht, wie ein diversifiziertes und heterogenes Netz von Komponenten und Services in die jeweils konkrete, aber dynamische Arbeitspraxis integriert werden kann. Als Forschungsmethoden kommen dabei die (Retro-)analyse laufender und durchgeführter IT-Projekte, sowie ethnographische Untersuchungen von Softwareentwicklern und ihres Umgangs mit Eclipse und ihrer Strategien zur Eclipse Anpassung zur Anwendung. Dabei gehen wir davon aus, dass Softwareentwickler als Lead User fungieren und sich in ihrer Arbeitspraxis heute schon verallgemeinerungsfähige Anpassungsstrategien für Komponenten basierte Systeme finden lassen.

Aus der Sicht des Herstellers geht es um Identifikation der neuen Anforderungen an Komponenten und Services für diversifizierte und heterogene Anwendungsdomänen, bei denen die Endanwender Komponenten und Services in ihren jeweiligen Anwendungskontext integrieren müssen. Als Methoden wird dabei eine (Retro-)analyse laufender und durchgeführter IT-Projekte unter dieser Perspektive vorgenommen und es wird eine Bewertung aktueller Software-Engineering Trends durchgeführt.

In der Konzeptionsphase geht es darum, erste Entwürfe für eine Realisierung von Co-EUD zu erstellen. In der Umsetzungsphase werden die zuvor entwickelten Konzepte prototypisch umgesetzt. Eine zentrale Rolle auf der Frontend-Seite spielt dabei das Eclipse-Framework. Das Framework wurde von Anfang an als offene, flexibel erweiterbare Plattform für Software-Entwickler konzipiert [Gamma/Beck 2003; D'Anjou et al. 2005]. Aktuell wird es zu einer universellen, Komponenten basierten Plattform für Arbeitsplatzanwendungen weiterentwickelt [Grama et al. 2004]. Bei den Backend-Strukturen bieten sich service-orientierte Architekturen an [Dostal et al. 2004]. Erste Schritte hin zu einer Domänen spezifischen service-orientierten Architektur wurden z.B. von OrbiTeam für das Groupware-System BSCW in Angriff genommen [Koch 1999; Fraunhofer 2004]. Hierüber lassen sich Basisdienste einer Groupware in andere Applikationen integrieren. Durch Einbindung in einen Eclipse-basierten Rich-Client lassen sich bei geeigneter EUD-Unterstützung prinzipiell die Anpassungsmöglichkeiten des Systems für den Endbenutzer vergrößern [Stevens et al. 2004].

Die Konzepte und prototypischen Implementierungen werden bei den Industriepartnern in ihrer Arbeitspraxis bzw. am Usability-Center des Fraunhofer FIT praktisch evaluiert. Bei der Evaluation ist eine Methoden-Triangulation vorgesehen. Hierbei werden kontrollierte Experimente und quantitative Logfile-Auswertungen mit ethnographischen Workplace-Studies verbunden. Die Laborstudien dienen der systematischen und reproduzierbaren Erhebung von Usability-Problemen bei der Nutzung und Anpassung von EUD-orientierten Applikationen. Durch die ethnographischen Workplace-Studies wird die externe Validität der Tests abgesichert. Über die Logfile-Auswertungen sollen repräsentative Aussagen über Anpassung und Nutzung von Komponenten- und Servicenetzwerken erzielt werden. Daneben werden im Projekt Evaluationsmethoden entwickelt, die es Benutzer erlauben, die Flexibilität von Softwaresystemen aus ihren Nutzungskontext heraus zu bewerten. Hierzu werden Methoden des Software-Engineering mit Methoden der Software-Ergonomie zu verbunden.

Aufgrund der Evaluation und der systematischen Analyse wird ein erstes Referenzmodells von CoEUD entwickelt. Die Erstellung des Referenzmodells schließt den ersten Zyklus des Projekts ab. In der zweiten Hälfte des Projekts, wird analog zum ersten Zyklus, das Refe-

renzmodells verfeinert und anhand von prototypischer Umsetzung validiert, wobei in diesem Zyklus der Fokus stärker auf einer Verzahnung der einzelnen Ansätze liegt.

2.2 EUD Maßnahmen

Im CoEUD Projekt werden Konzepte entlang der drei Säulen entwickelt und praktisch erprobt.

- **Entwicklung Software-Engineering-Methoden**

EUD-orientierte Komponenten- und Service-Infrastrukturen stehen zwei Herausforderungen gegenüber: Zum einen müssen die Komponenten interoperabel sein und die Integration in fremde Infrastrukturen ermöglichen. Da aber bei der Integration von Komponenten zu einer optimal angepassten Einheit häufig ein menschlicher Eingriff nötig ist, müssen die zugrunde liegenden Software-Infrastrukturen auch hierauf ausgelegt sein und entsprechende Human Factors schon bei der Gestaltung mit berücksichtigt werden. Deshalb sollen im CoEUD neben der Untersuchung geeigneter Softwarezerlegung, Konzepte selbstbeschreibungsfähiger und semi-automatisch integrierbarer Komponenten und Services entwickelt werden.

Doch selbst die Gestaltung hoch anpassbarer Systeme kann nicht sämtliche Nutzungssituationen und die daraus abgeleiteten Flexibilitäts-Anforderungen vorhersehen und deshalb kann es immer wieder vorkommen, dass die bereitgestellten Anpassungsmöglichkeiten der Systeme nicht ausreichen. Daher sollen im CoEUD Projekt Konzepte entwickelt werden, die den Entwicklungsprozess auf nicht antizipierte Anforderungen vorbereitet und die Reaktionsfähigkeit des Softwareprozesses auf User Driven Innovation erhöht. Insbesondere gilt es eine Infrastruktur zu etablieren, die hilft solche nicht antizipierten Situationen im Nutzungskontext sichtbar zu machen.

- **Entwicklung von Technologien dezentralen Managements und kooperativen Appropriations-Infrastrukturen**

Ähnlich wie bei der Konzeption der Gestaltung von Serviceinfrastrukturen geht es hier darum, bestehende Ansätze weiterzuentwickeln und systematisch mit den Möglichkeiten Service orientierter Architekturen und des Eclipse-Frameworks abzugleichen. Hier liegt der Schwerpunkt jedoch in der Konzeption von Techniken zur Unterstützung von kooperativen EUD innerhalb von verteilten Arbeitsprozessen. Insbesondere geht es um die Gestaltung von verteilten Repositories und Integration in den Nutzungskontext, die Visualisierung ähnlicher Nutzer bzw. Nutzungen und Etablierung von Peer & Expert Recommendation sowie die Bereitstellung von verteilten Annotations- und Bewertungsmöglichkeiten von Komponenten und Services.

- **Entwicklung von Anpassungswerkzeugen**

In diesem Bereich geht es insbesondere darum, das Flexibilisierungspotential des Eclipse-Framework besser aus dem Nutzungskontext zugänglich zu machen. Dabei steht man vor der Herausforderung, dass Eclipse zwei – nicht unmittelbar verzahnte – Komponenten kennt. Einmal auf der software-technischen Ebene der Plugins und auf der Ebene des User Interface durch so genannte Views.

Eclipse verfolgt mit seinem Plugin-Konzept einen Ansatz, der sich als Flexibilität durch Erweiterbarkeit bezeichnen lässt. Ein Plugin ist im technischen Sinne *“the smallest application unit of the Eclipse Platform function that can be developed and delivered separately”* [IBM 2005], wobei Eclipse hier auf den Industriestandard OSGi aufsetzt.

Neben dieser Art der Anpassbarkeit auf der Plugin-Ebene hat der Benutzer noch die Möglichkeit, das User Interface durch Rekomponieren von Interface-Elementen - den so genannten Views - an seine Bedürfnisse anzupassen.

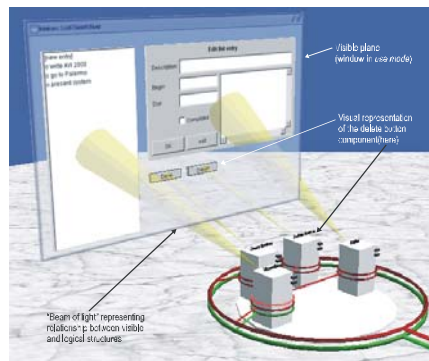


Abbildung 2 Ko-Referenzialität [de Souza et al. 2001] zwischen User Interface und Komponentenstruktur im FreEvolve-Ansatz mittels 3D-Ansicht [Stiemerling 2000, Hallenberger 2000]

In Eclipse hat man entsprechend zwei Arten von *Application Units*, die angepasst werden können – Plugins und Views. Im Gegensatz z.B. zu dem FreEvolve Ansatz [Stiemerling 2000] besteht jedoch keine Ko-Referenzialität zwischen beiden Ebenen [de Souza et al. 2001], so dass es Benutzern schwer fällt, zwischen beiden Ebenen zu wechseln. Hinzu kommt, dass auch andere für das EUD relevante Elemente (Individualisieren der Komponenten, das Hilfesystem, etc.) nur unzureichend die Komponenten-basiertheit des Systems berücksichtigen. Dadurch ergeben sich so für den Benutzer *hidden dependencies* [Green 1989], d.h. die Abhängigkeiten sind aufgrund der Nutzungserfahrung nur schwer zu erkennen.

Aus diesem Grunde wird im Projekt untersucht, inwieweit sich bestehende, in der EUD Forschung entwickelte Ansätze mittels des Eclipse Framework umsetzen lassen und sich hierdurch verbesserte Anpassungswerkzeuge und User-Interfaces entworfen werden können. Insbesondere ist das Disclosure Prinzip [Di Giano, 1996] zu nennen, dass auf die Visualisierung zugrundeliegender Ablaufmechanismen abzielt. Eine Anwendung des Disclosure Prinzip auf Komponenten basierte Anpassbarkeit ist in Abbildung 2 zu sehen. Durch die 3D Darstellung ist erkennbar, welche Software-Komponente für welche User Interface Element verantwortlich ist. Eine andere interessante Technik aus der EUD Forschung ist das Prinzip der Direct Activation [Wulf/Golembek 2001]. Hierbei geht es um die Integration von Anpassungswerkzeugen in den unmittelbaren Nutzungskontext und die direkte Aktivierbarkeit der Werkzeugs aus dem Nutzungskontext heraus.

3. Ausblick

Das Projekt befindet sich noch in einer frühen Phase, weshalb an dieser Stelle nur ein sehr vager Ausblick gegeben werden kann. Insgesamt lässt sich aber feststellen, dass in der Diskussion über Flexibilisierung von Organisation und einhergehender IT-Konzepte, wie Componentware und Service Oriented Architecture, häufig der eigentliche Nutzer aus dem Blickfeld gerät. Um die Vorteile flexibler Software Architekturen aber auch für den Nutzer zur Verfügung zu stellen und das Anwendungswissen des Nutzer besser im Software Engineering zu integrieren, sollte der Nutzer wieder stärker in das Zentrum der Betrachtung gerückt werden. Aus diesem Grund besitzt das Projekt ein hohes Potential für Praxis.

Zudem zeigen erste Arbeiten aus dem Vorfeld des Projekts, die die Konzepte der kooperativen EUD- und der Appropriationsunterstützung auf Eclipse übertragen haben, wie die im Projekt anvisierten Ziele sich konkret umsetzen lassen. Das hierbei entstandene Konzept der Community Help in Context (ChiC) [Stevens/Wiedenhöfer, 2006], sowie das Konzept des Participatory Design in Use (PaDU) [Stevens/Draxler, 2006] stellen viel versprechende erste

Ansätze dar, die es gilt in den weitem Projektverlauf zu elaborieren und eingehender zu evaluieren.

4. Literatur

- Bass, L.; Clements, P.; Kazman, R. (2003): *Software Architecture in Practice*, Addison-Wesley, 2003. Mellon
- Beck, K. (2000): *eXtreme programming explained – embrace change*. Addison-Wesley, 2000.
- Boehm, B. (1976): Boehm, B.: *Software engineering. IEEE Transactions on Computers* 25(1976) 1226-1241.
- Brynjolfsson, E., Hitt, L.M.(1998): Beyond the productivity paradox. *Communications of the ACM*, 1998. 41(8): S. 49-55
- Clements, P.;Kazman, R.; Klein, M. (2001): *Evaluating Software Architectures*. Boston u.a.: Addison-Wesley, 2001. Mellon
- D'Anjou, J.; Fairbrother, S.; Kehn, D.; Kellerman, J.; McCarthy, P. (2005): *The Java Developer's Guide to Eclipse*, Addison-Wesley, 2005.
- de Souza, C.S., S.D.J. Barbosa, Silva, S.R.P. (2001): Semiotic engineering principles for evaluating end-user programming environments. In: *Interacting with Computers*, 13(4), 2001, S. 467-495.
- DiGiano, C. (1996): *Self-Disclosing Design Tools: An Incremental Approach Toward End-User Programming*. PhD-Thesis, University of Colorado, 1996.
- Dostal, W.; Jeckle, M.; Melzer, I.; Zengler, B. (2004): *Service-orientierte Architekturen mit Web Services : Konzepte – Standards – Praxis*; Spektrum Akademischer, 2004.
- Droeschel, W .; Wiemers, M. (1999): *Das V-Modell 97*, Muenchen, Wien: Oldenbourg, 1999
- Eisenberg, M.: *Programmable Applications: Interpreter meets Interface*. In: *SIGCHI Bulletin*, 27(2), 1995, S. 68-93.
- Evans E. (2003): *Domain Driven Design*, Addison-Wesley, 2003, Mellon
- Feathers , M.C. (2004): *Working Effectively with Legacy Code*, Pearson Education, Upper Saddle River, NJ.
- Fischer, G. (1997): *Domain-Oriented Design Environments: Knowledge-Based Systems for the Real World*, in: *International Journal "Failure & Lessons Learned in Information Technology Management"*, 1(2), 1997, S. 123-133
- Fowler M. (2003): *Patterns of Enterprise Application Architecture*. Boston, MA, Pearson Education, 2003
- Fowler, M., Beck, K., Bryant, J., Opdyke, W. und Roberts, D. (2001): *Refactoring - Improving the Design of Existing Code.*, Addison-Wesley, Boston.
- Fraunhofer (2004): *X-BSCW, XML-RPC Application Programming Interface to BSCW*, Technical Paper, Fraunhofer FIT, 2004; <http://bscw.fit.fraunhofer.de/api/X-BSCW.pdf> (7.3.2005).
- Gamma, E.; Beck, K. (2003): *Contributing to Eclipse*, Addison-Wesley Professional, 2003.
- Green, T. R. G. (1989): Cognitive dimensions of notations. In: *People and Computers*, Cambridge University Press, 1989, S. 443-460.
- Grama H.; Attenborough, K.; Banks-Binici, J.; Marsden, J.; Kraenzel C.; Calow, J.; Ramaswamy, S.; Zurko, M.E. (2004) *IBM Workplace Client Technology (Rich Client*

- Edition) Technical Overview*, Redbooks Paper 2004; URL: <http://www.redbooks.ibm.com/redpapers/pdfs/redp3884.pdf> (2.3.2005).
- Hallenberger, M. (2000): Eine 3D Benutzerschnittstelle für komponentenbasierte Anpassbarkeit. Diplomarbeit, Universität Bonn, Institut für Informatik III, 2000.
- IBM (2005): Draft: Eclipse Platform Technical Overview. Whitepaper, IBM Corporation and The Eclipse Foundation, 2005.
- Jacobson, I., Booch, G. und Rumbaugh, J. (1998): *The Unified Software Development Process*, Addison-Wesley, Reading, Massachusetts.
- Kahler, H.; Stiernerling, O.; Wulf, V.; Hoepfner, J. (1999): *Gemeinsame Anpassung von Einzelplatzanwendungen*, in: *Proceedings der Fachtagung Software-Ergonomie 1999*, S. 183–194.
- Kerievsky, J. (2004): *Refactoring to Patterns*, Addison-Wesley, Boston.
- Klann, M. (2004): *EUD-Net's Roadmap to End-User Development*; Deliverable 1.1 of the EUD-Network of Excellence, IST-2002-8.1.2.
- Koch, T. (1999): *XML in practice: the groupware case*, in: *Proceedings of IEEE WET ICE Workshop 1999*, Stanford University.
- Larman, C. (2003): *Agile and Iterative Development: A Manager's Guide*. Addison-Wesley Professional; 1st edition (August 15, 2003).
- Lieberman, H. (Hrsg.) (2001). *Your wish is my command: Programming by example*, Morgan Kaufmann, 2001.
- Lieberman, H.; Paternò, F.; Wulf, V. (Hrsg.): *Empowering people to flexibly employ advanced information and communication technology*, Kluwer, in Druck.
- Mackay, W.E. (1990): *Patterns of Sharing Customizable Software*, in: *Proceedings of ACM CSCW '90*, ACM Press, 1990, S. 209 – 221.
- Mackay, W.E. (1991): *Triggers and barriers to customizing software*, in: *Proceedings of ACM CHI '91*, ACM Press, 1991, S. 153 – 160.
- Martin, R.C. (2002): *Agile Software Development, Principles, Patterns, and Practices*. Prentice Hall; 1st edition (October 15, 2002).
- Mørch, A, Stevens, G., Won, M., Klann, M., Dittrich Y., Wulf, V. (2004): *Component-based technologies for end-user development*, in: *Comm. of the ACM*,. 47, S. 59-62, 2004.
- McIlroy, M. (1968): *Mass produced software components*, in: *Report on a conference of the NATO Science Committee*, 1968, S. 138–150.
- Myers, B.A. (1990): Creating user interfaces using programming by example, visual programming, and constraints, in: *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 12(2), 1990, S. 143 – 177
- Nardi, B. A. (1993): *A Small Matter of Programming - Perspectives on end user computing*, MIT-Press, 1993.
- Nardi, B.; Miller, J. (1990): *The spreadsheet interface: A basis for end user programming*, In: *Proceedings Interact'90* ACM Press, 1990, S. 977-983.
- Oppermann, R.(Hrsg.) (1994): *Adaptive User Support*. Lawrence Earlbaum Associates, 1994.

- Orlikowski, W.J.: Evolving with Notes: Organizational change around groupware technology. In *Groupware & Teamwork*, J. Wiley, 1996, S. 23 – 60.
- Pipek, V. (2005): *From Tailoring to Appropriation Support: Negotiating Groupware Usage*. PhD Thesis, University of Oulu: Oulu, Finland, 2005.
- Pipek, V., Wulf, V. (1999): A Groupware's Life. In: *Proc. of ECSCW '99*, Kluwer, 1999, S. 199 – 219.
- Repenning, A. (1993): *Agentsheets: A Tool for Building Domain-Oriented Dynamic, Visual Environments*, University of Colorado at Boulder, Ph.D. Thesis, 1993.
- Shaw, M.; Garlan, D. (1996): *Software-Architecture: Perspectives of an Emerging Discipline*, Prentice Hall, 1996.
- Stevens, G.; Budweg, S.; Pipek, V. (2004): The "BSCWeasel" and Eclipse-powered Cooperative End User Development, in: *Proc. Workshop "Eclipse as a Vehicle for CSCW Research" at the Int. Conf. on CSCW 2004*, 2004.
- Stevens, G. Draxler, S.: Partizipation im Nutzungskontext. In: *Konferenzband Mensch & Computer 2006*, (in Druck).
- Stevens, G., Wiedenhöfer, T.: CHIC - A pluggable solution for community help in context. In: *Proc. of the NordCHI 2006*, (in Druck).
- Stevens, G.; Wulf, V. (2002): *A new dimension in access control: studying maintenance engineering across organizational boundaries*, in: *Proceedings of the ACM Conference CSCW, 2002*, S. 196-205.
- Stiemerling, O. (2000): *Component-based Tailorability*, Dissertation, Bonn, 2000.
- Sutcliffe, A.; Mehandjiev, N. (Hrsg.) (2004): *End-user development*, Special Issue of the *Communications of the ACM 47(9)*, ACM Press, 2004.
- Szyperski, C (1998): *Component Software Beyond Object-Oriented Programming*, Addison-Wesley, 1998.
- Teege, G. (1998): *Individuelle Groupware: Gestaltung durch Endbenutzer*, Wiesbaden 1998.
- von Hippel, E. (1988): *The sources of innovation*. Oxford University Press 1988.
- Won, M.; Wulf, V. (2003): *Anpassungsumgebung für komponentenbasierte Software: Kooperativ und lernförderlich*, in: *i-com 2(1)*, 2003, S. 28 -34
- Wulf, V. (1999): „Let's see your Search-Tool!“ – *Collaborative use of Tailored Artifacts Groupware*, in: *Proceedings of GROUP 1999*, S. 50-60.
- Wulf, V. (2001): *Zur anpassbaren Gestaltung von Groupware: Anforderungen, Konzepte, Implementierungen und Evaluationen*, GMD Research Series, No. 10/2001, 2001
- Wulf, V., Golombek, B.(2001): Exploration environments: concept and empirical evaluation. In: *Proc. of the GROUP. 2001*, S. 107-116.
- Wulf, V., Rohde, M.(1995): Towards an Integrated Organization and Technology Development. In: *Proc. of Designing Interactive Systems. 1995*, S. 55-64.
- Züllighoven, H., et. al. (2004):, *Object-Oriented Construction Handbook*, dpunkt.verlag/Copublication with Morgan-Kaufmann Oktober 2004