A. M. Heinecke, H. Paul (Hrsg.): Mensch & Computer 2006: Mensch und Computer im Struktur*Wandel*.
München, Oldenbourg Verlag, 2006, S. 83-92

Partizipation im Nutzungskontext

Gunnar Stevens, Sebastian Draxler

Universität Siegen, Wirtschaftsinformatik

Zusammenfassung

In diesem Beitrag wird das Konzept des *Participatory Design in Use* vorgestellt sowie das Konzept zu existierenden Ansätzen, die sich ebenfalls mit den Fragen auseinandersetzen, wie der unmittelbare Nutzungskontext bei der Einbeziehung der Nutzer in den Design-Prozess ausgenutzt werden kann. Hierbei werden drei Dimensionen – Strukturierung der Design-Tätigkeit, des Gestaltungsspielraum und des Design Prozess identifiziert – die helfen, die verschiedenen Ansätze einzuordnen.

Anschließend stellen wir einen ersten Prototyp vor, der die Kluft zwischen Nutzer und Designer verringert, indem er Partizipationsmöglichkeiten besser in die Applikation integriert. Die vorgestellte Lösung wird als Teil eines Open Source Projekts praktisch genutzt, um die Beteiligung der Nutzer zu erhöhen und die Ideen und Probleme der Nutzer besser festzuhalten und effektiver in den Entwicklungsprozess zu integrieren.

Die realisierte Lösung verfolgt dabei jedoch nicht den Designansatz gängiger Feedback-Mechanismen, bei denen allein das Know-how der Nutzer abgeschöpft werden soll. Vielmehr soll die Transparenz des Entwicklungsprozesses erhöht werden, indem dieser in die Applikation eingebettet wird. Zudem sollen den Nutzern geeignete Mittel an die Hand gegeben werden, um die alltägliche Nutzung und die Reflektion über diese Nutzung als Teil eines partizipativen Entwicklungsprozesses besser miteinander besser zu vereinen.

1 Neue Formen der Benutzerpartizipation

Schon in den 80er Jahren hat unter anderen Christiane Floyd gefordert, dass man Software zyklisch entwickelt (Floyd, Reisin et al. 1989). Seit dieser Zeit erweitern verschiedene Faktoren moderner Softwareproduktion und -distribution neue Möglichkeiten des Evolutionären Design (Nichols and Twidale; Bleek, Jeenicke et al. 2002; Bleek, Jeenicke et al. 2002; Rittenbruch, McEwan et al. 2002; Nichols, McKay et al. 2003). Die neuen Möglichkeiten des Evolutionären Designs mit radikal kurzen Entwicklungszyklen geben Anlass, sich über die Konsequenzen von Participatory Design und über neue Formen der Nutzerpartizipation Gedanken zu machen.

In diesen Beitrag wollen wir das Konzept des *Participatory Design in Use (PaDU)* als eine der neuen möglichen Formen der Benutzerpartizipation vorstellen und genauer untersuchen.

Das Ziel besteht dabei darin, die Möglichkeiten der Partizipation während der Nutzung des Systems zu vergrößern.

Dabei kann bei PaDU innerhalb der PD-Ansätze besonders der zeitliche und räumliche Aspekt charakterisiert werden:

- PaDU ist eine Post-Deployment Methode
- PaDU ist Methode für räumlich verteilte Partizipation, die hierfür Werkzeuge der Computer vermittelten Kommunikation nutzt

Unter Bezug auf die klassische Taxonomie des Software Engineering ist *Participatory Design in Use* eine PD-Methode, die in die späte Phase Softwarelebenszyklus fällt (Muller, Haslwanter et al. 1997). Die Forschung zum gegenseitigen Lernen in einer Nutzer-Entwickler Gemeinschaft zeigt, dass ein konkretes, funktionsfähiges System dem Nutzer dabei unterstützt die Gestaltung zu bewerten und zu kritisieren. Nichtsdestotrotz existieren bis dato nur wenige PD-Methoden, welche speziell für diese Phase konzipiert worden sind. PD adressierte diese Phase hauptsächlich durch die Forderung nach anpassbaren Systemen (Henderson and Kyng 1991; Wulf 1994).

Neuere Arbeiten wie eXtreme Participation (Rittenbruch, McEwan et al. 2002), greifen jedoch die veränderten Produktionstechniken auf, welche die Kosten einer späten Änderung dramatisch reduziert haben (Beck 2000). Insbesondere Nicolas et al. schlagen in ihren Arbeiten (Nichols and Twidale; Nichols, McKay et al. 2003) neue Formen vor, die sie mit Post-deployment Partizipation bezeichnen.

Der zweite Aspekt ergibt sich aus der Forderung, Nutzern Partizipationsmöglichkeiten bereitzustellen, die jederzeit im lokalen Nutzungskontext zur Verfügung stehen. Da aber der lokale Nutzungskontext von der Entwicklung räumlich abgetrennt ist, muss diesem verteilten Aspekt Rechnung getragen werden. Dieser zweite Aspekt unterscheidet die co-lokalen PD-Methoden auf der einen Seite von Laborstudien, wo Nutzer ihren normalen Nutzungskontext verlassen müssen, als auch von ethnographischen Studien, in denen Forscher sich zu den Nutzern begeben müssen.

Aufgrund des verteilten Setting entsteht das Problem, wie die verschiedenen Akteure einen gemeinsamen Kontext herstellen und diesen kommunizieren. Information, die implizit in den lokalen Kontext eingebettet ist, muss expliziert und mittels Computer vermittelter Kommunikation (CMC) den andern zugänglich gemacht werden. Insbesondere die Forschung zu Remote Evaluation (Hartson, Castillo et al. 1996; Castillo 1997) hat einen genaueren Blick auf den Aspekt von CMC und Benutzerfeedback für den Fall von benutzerverfassten Critical Incident Berichten gelegt und die Weiterentwicklung dieser Ideen von Nichols, McKay et al. (2003), die erste Ideen vorschlagen, wie sich normale Nutzer stärker in Open Source Entwicklung einbinden lassen.

Ansätze zur Remote Participation finden sich zudem in PD orientieren Projekten für Kooperationssysteme, die ihre eigenen Kommunikations- bzw. Kooperationssysteme benutzen, um hierüber einen Metadiskurs über die System zu führen (Bleek, Jeenicke et al. 2002).

1.1 Ein Klassifikationsschema für Post-deployment und Remote Partizipationsmethoden

Im vorherigen Abschnitt haben wir das Konzept des Participatory Design in Use (PaDU) durch seine besonderen zeitlichen und räumlichen Eigenschaften charakterisiert. Aufgrund dieser Charakterisierung haben wir nach existierenden Konzepten in Literatur und Praxis gesucht, die man aufgrund dieser Charakterisierung im weitesten Sinne unter eine räumlich verteilte Post Deployment Methode subsumieren kann. Anschließend haben wir die Gemeinsamkeiten und Unterschiede dieser Ansätze analysiert und drei Dimension identifiziert, die helfen die unterschiedlichen Konzepte genauer zu klassifizieren.

Im Folgenden sollen die drei Dimension erläutert und anschließend die verschiedenen Arbeiten in das Klassifikationsschema eingeordnet werden.

Strukturierung der Design Tätigkeit

Partizipationsmethoden strukturieren implizit oder explizit die Gestaltungsaktivität des Nutzers. Um dies zu klassifizieren, greifen wir auf Donald Schön zurück, der zwischen den Design Aktivitäten *Problem Framing* und *Problem Solving* unterscheidet (Schön 1983). Er insistiert gegenüber rationalistischen Vorstellungen, dass die Reflektion über den Gestaltungskontext und der Nutzungssituation ein entscheidender Teil des professionellen Gestaltungsprozess ist. Obwohl er es nur für den professionellen Kontext konstatiert, gehen wir davon aus, dass dies für die Gestaltung durch den Nutzer ebenso zutrifft.

Die Ansätze zu anpassbaren Systemen sind ein Beispiel für Problem Solving orientierte Methoden. Sie fokussieren auf die Frage „wie“ eine Umgestaltung des Systems vom Benutzer vollzogen werden kann. Sehr viele Anstrengungen wurden unternommen, um Werkzeuge für Endbenutzer zur Verfügung zu stellen, die ihn bei dieser Aktivität unterstützen (Lieberman, Paternò et al. 2005).

Im Gegensatz dazu verfolgen z.B. Remote Evaluation Methoden ein ganz anderes Konzept darüber, welche Aktivitäten des Nutzers unterstützt werden sollen. So soll sich der Nutzer beim Verfassen eines Bereichtes nicht über die Umsetzungsdetails Gedanken machen. Vielmehr besteht seine Aufgabe darin, jene Stellen im System auszumachen, bei denen die gewählte Implementierung unzureichend ist und verbessert werden sollte. Idealerweise soll der Nutzer über die konkrete Situation reflektieren und hierbei das Nutzungsproblem so genau explizieren und den Entwicklern verständlich machen, dass diese hierfür eine adäquate Lösung finden können.

Weiterhin kann man bei der Strukturierung danach unterscheiden, ob die Aktivität des Gestaltens als eine individuelle oder als eine kollektive Aktivität aufgefasst wird und wie die Kooperation zwischen den einzelnen Akteuren strukturiert wird. Klassisches Tailoring versteht das Anpassen als eine individuelle Tätigkeit. Open Source Projekte verfolgen dagegen meist das Ideal kollektiver Diskurse. Dabei zeigt sich in den Diskussionsverläufen in Entwicklerforen, dass hier nicht allein die Lösung eines Problems diskutiert wird. Vielmehr ist die Reflektion darüber, was das zu lösende Problem ist, ein integraler Teil der Diskussion.

Die Strukturierung des Gestaltungsraums

Die Methoden unterscheiden sich auch dahingehend, wie stark der Gestaltungsraum auf spezielle, von den Entwicklern antizipierte Vorfälle eingeschränkt wird bzw. wie viel Raum der Eigeninitiative und Kreativität dem Nutzer gegeben wird.

Innerhalb der Forschung zur Anpassbarkeit wird so zwischen Systemen unterschieden die nur flache Anpassungen des Systems erlauben und Methoden, die darauf abzielen auch radikale Änderungen des Systemdesigns vorzunehmen (Henderson and Kyng 1991; Morch 1997). Der Hauptgrund in diesem Bereich Gestaltungsspielräume klein zu halten, ist Komplexität zu reduzieren.

Feedback-Mechanismen von Massenprodukten gehören ebenfalls zu den Methoden, bei denen der Gestaltungsraum stark vorstrukturiert und nur auf einen kleinen Bereich beschränkt ist, z.B. einen Fehlerbericht abzusenden wenn eine Anwendung abstürzt. Eine besondere Form stellen Werkzeuge dar, die in Applikationen eingebaut sind, um das Verhalten einer großen Klasse von Nutzern zu erfassen und statistisch auszuwerten. Insgesamt liegt bei diesen Methoden die Deutungshoheit des Feedbacks auf Seiten der Entwickler, so dass man bei diesen Ansätzen nur von einer marginalen bis gar keiner Beteiligung seitens der Nutzer sprechen kann (vgl. nächsten Abschnitt).

Auf der anderen Seite stehen Community Systeme, welche häufig in Open Source Projekten eingesetzt werden. Diese Systeme bieten Entwicklern wie Nutzern je nach Konfiguration die gleichen Möglichkeiten an um Gestaltungsideen in den Softwareprozess einzubringen. Sie adressieren jedoch nicht die Frage, was geeignete Werkzeuge und Ausdrucksmittel für Nutzer sind, um Gestaltungsideen ihre festzuhalten. Auch lassen sie die Frage außer Acht, wie die Kreativität der Nutzer angeregt werden kann.

Strukturierung des Design Prozess

Die letzte Dimension ergibt sich, wenn man z.B. die üblichen Feedback Systeme der kommerziellen Produkte von Microsoft mit z.B. den Partizipationswerkzeugen für Open Source Projekte auf Sourceforge[1] vergleicht. Es ist durchaus denkbar, dass die professionellen Entwickler in beiden Fällen die gleichen Werkzeuge benutzen, um das Feedback des Nutzers zu verwalten. Der entscheidende Unterschied ist, ob auch der Benutzer Zugang zu diesen Systemen hat und verfolgen kann wie mit seinen Berichten verfahren wird.

Im Fall von kommerziellen Produkten wie z.B. Apples Safari oder Microsoft Office, ist für einen Benutzer, der einen Verbesserungsvorschlag sendet, der dahinter liegende Entwicklungsprozess nicht sichtbar. Im Gegensatz dazu hat in verteilten Open Source Projekten der Benutzer Zugang zu den gleichen Informationen wie die Entwickler des Systems (z.B. Quellcode, Diskussionen). Wie Hartson, Castillo et al. (1996) auf Grund ihrer Nutzerbefragung anmerken, wünschen sich Nutzer aber ein Feedback darüber, was mit ihren Berichten geschieht. Dies deckt sich auch mit unseren Erfahrungen.

Die beiden Arten, den Benutzer am dahinter liegenden Entwicklungsprozess teilzuhaben zu lassen unterscheiden wir in einen transparenten gegenüber einen opaken Entwicklungspro-

[1] Sourceforge ist ein Portal zur Unterstützung von Open-Source Anwendungen. Diverse Dienste wie die Verwaltung von Programmcode, Foren und Fehlerdatenbanken können kostenlos genutzt werden.

zess. Im transparenten Fall wird versucht, den Entwicklungsprozess nachvollziehbar, verständlich und weitestgehend öffentlich zu gestalten. Im Gegensatz dazu bedeutet ein opaker Entwicklungsprozess, dass dem Nutzer kein Einblick in Prozess gewährt wird und insbesondere nicht, wie mit seinen Beitrag weiter verfahren wird.

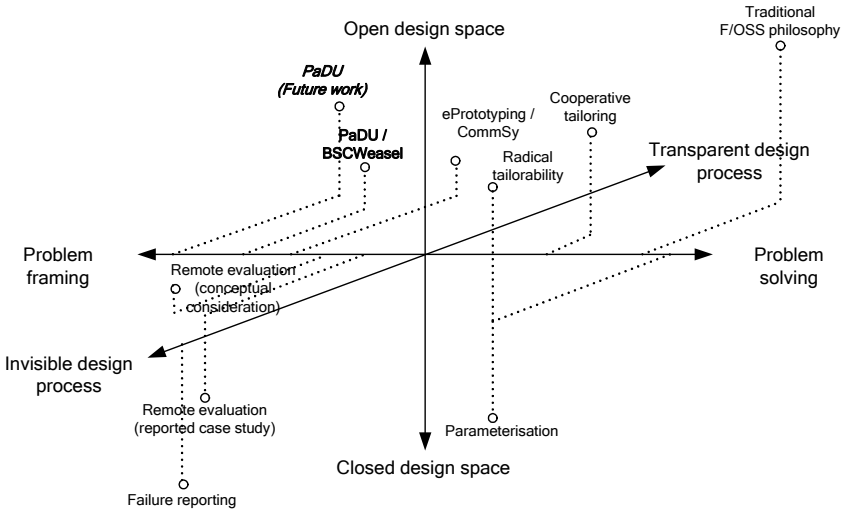

Abbildung 1: Taxonomie über die verschiedenen Methoden, die räumlich verteilte Post-Deployment Partizipation zu fördern

Mit der Hilfe dieser drei Dimensionen lässt sich ein genauerer Überblick über die verschiedenen Methoden erlangen, die eine Form der Post Deployment und Remote Partizipation ermöglichen. Abbildung 1 zeigt eine erste Landkarte dieser verschiedenen Methoden. Die weiter unten beschriebene PaDU Realisierung und deren Einbettung in die Entwicklung des Groupware Client BSCWeasel gehört dabei zu Methoden, die sich an der Tätigkeit des Problem Framing und eines großen Gestaltungsspielraum orientieren, sowie Benutzerpartizipation in einen transparenten Entwicklungsprozess integrieren (vgl. Abb. 1). In Abbildung 1 ist auch dargestellt, in welche Richtung die Lösung weiterentwickelt werden soll.[2]

[2] Zwar ist der unten dargestellte Entwicklungsprozess insofern transparent, als dass sowohl das System zur Verwaltung der Designideen, als auch der Source Code öffentlich zugänglich ist. Jedoch ist der Prozess, nach welchen Kriterien über die Umsetzung eines Vorschlags entschieden wird nicht festgelegt und somit für die Nutzer nicht transparent. Eine Weiterentwicklung des Ansatzes bedeutet deshalb, auch hier geeignete Verfahren zu finden und idealer weise in die softwaretechnische Unterstützung aufzunehmen.

2 Realisierung einer wieder verwendbaren PaDU Lösung

Im Folgenden soll die im Rahmen des BSCWeasel[3] Projekt entwickelte Lösung vorgestellt werden, die es Nutzern des BSCWeasel vereinfachen soll, sich am Gestaltungsprozess zu beteiligen.

Bei den Realisierungen solcher Systeme kann man grob zwischen Werkzeugen unterscheiden, die in die eigentliche Applikation integriert sind, wie die „Fehlerbericht senden" Funktionalität bei Microsoft Windows und externen Werkzeugen, wie die Fehlerdatenbank und das Diskussionsforum bei Sourceforge. Während die eingebauten Werkzeuge dem Benutzer einen leichtern Zugang aus dem Nutzungskontext erlauben und die Möglichkeit bieten bestimmte wichtige Daten über den lokalen Nutzungskontext automatisch zu sammeln, sind externe Werkzeuge meist günstiger in der Anschaffung, da hier häufig schon existierende Systeme benutzt werden, die nur per Nutzungskonvention zu Partizipationstools umgewandelt werden.

Aufgrund unserer Untersuchungen haben wir einen hybriden Ansatz gewählt. Zur Verwaltung der Beiträge setzen wir auf das kommerzielle Issue Tracking System JIRA. Clientseitig entwickelten wir ein Plugin, das in die Anwendung integriert wird und somit sehr nah am Nutzungskontext ist. Dieses Plugin ermöglicht den Nutzern des BSCWeasel Systems nun einen speziellen Zugang aus ihren Nutzungskontext heraus.

2.1 JIRA

JIRA ist ein kommerzielles Produkt, dass sich ursprünglich zur Unterstützung des professionellen Softwareentwicklungsprozesses gedacht war. Die Hersteller charakterisieren das System selbst als: "*an issue tracking and project management application developed to make this process easier for your team. JIRA has been designed with a focus on task achievement is instantly usable and is flexible to work with.*"

In JIRA werden die verwalteten Vorgänge als Texte gespeichert, denen beliebige Anhänge hinzufügt werden können (z.B. Logfiles oder Screenshots). Das System erlaubt es den Entwicklern die Vorgänge zu diskutieren, jemanden mit der Abarbeitung zu beauftragen und den Zustand des Vorgangs zu verändern, wobei der Workflow anpassbar ist. Üblicherweise wird das System über ein Webinterface bedient, JIRA besitzt jedoch auch eine Web Service Schnittstelle auf SOAP Basis, die es erlaubt, dass System per Remote Procedure Call zu bedienen.

Wir haben uns für JIRA aus verschiedenen Gründen entschieden: Um die Transparenz des Prozesses zu verbessern, sollten professionelle Entwickler und Nutzer das gleiche System benutzen, damit sie auf die gleichen Informationen zugreifen können. Außerdem sollte der Verwaltungsaufwand minimiert werden, die Beiträge der Nutzer und der Diskurs hierüber in den Entwicklungsprozess zu integrieren. Zum anderen war eine Web Service Schnittstelle

[3] Unter http://www.bscweasel.de kann die Software herunter geladen, ausprobiert und bei Bedarf eigenständig angepasst werden. Eine frühe Version des PaDU Plugin ist dort schon integriert.

eine obligatorische Anforderung, damit man das System nahtlos in den Anwendungskontext integrieren kann.

2.2 PaDU Plugin

BSCWeasel ist ein so genannter Rich Client für das BSCW Groupware System auf Basis der Eclipse Rich Client Plattform (RCP), wie sie z.B. bei IBM Workplace/Lotus Notes Verwendung findet.

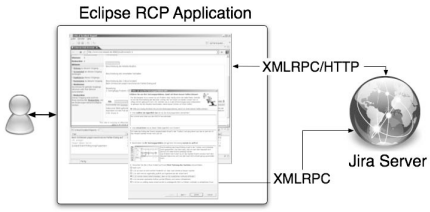

Abbildung 2: Kommunikation zwischen einer mit PaDU erweiterten Applikation und dem JIRA Server

Obwohl das PaDU Plugin[4] im Rahmen des BSCWeasel Projekts entstanden ist, hängt das Plugin nur von einigen Eclipse Grundkomponenten ab. Dies bedeutet, dass das Plugin in jede Applikation integriert werden kann, die auf dem Eclipse RCP Framework basiert.

Die Funktionsweise des Plugin ist wie folgt: PaDU kapselt die von JIRA zur Verfügung gestellten Kommunikationsdienste und stellt stattdessen der Eclipse RCP Anwendung eine Schnittstelle zur Verfügung um Nutzerberichte lokal zu verwalten, an das JIRA System zu senden und die Beiträge zu diskutieren. Dies geschieht unter Verwendung von Webservices für das Versenden und einfachem HTTP für die Anzeige. So werden zum Beispiel detaillierte Informationen über einen Beitrag im integrierten Webbrowser angezeigt.

Schauen wir uns das realisierte Benutzer-Interface genauer an: Über den Standarderweiterungsmechanismus von Eclipse integriert das PaDU Plugin eine permanent sichtbare Schaltfläche (zeigt einen traurigen smiley) in die RCP, die dem Benutzer als Eintrittspunkt dient, um sich am Entwicklungsprozess zu beteiligen (vgl. Abb. 3).

Die Schaltfläche öffnet ein Dialogfenster, welches ein Formular zur Beschreibung des Nutzungsproblems enthält (vgl. Abb. 3 (rechts). Der Dialog ist dabei eine Abwandlung des ursprünglichen Critical Incident Dialog (Castillo), so wie er in der Studie von (Hartson, Castillo et al. 1996) benutzt wurde.

[4] Das Plugin stellt im Eclipse Framework eine (auslieferbare) Komponente dar, die bestimmte Funktionalität anbietet. Plugins können nachträglich in eine ausgelieferte Anwendung installiert werden. Die gesamte Funktionalität der BSCWeasel Software wurde mittels solcher Plugins realisiert.

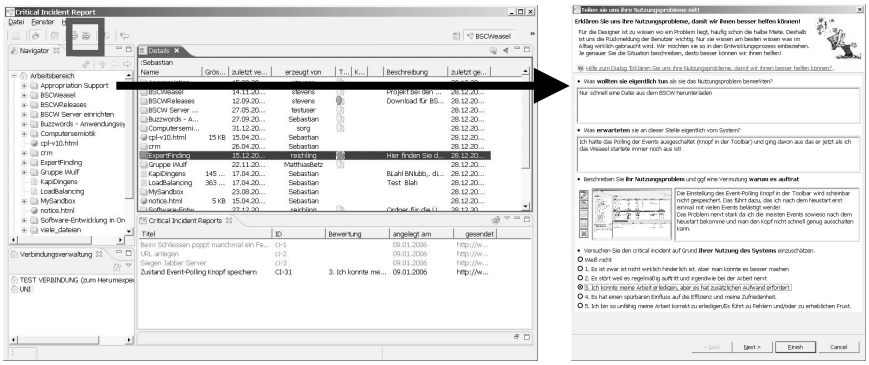

Abbildung 3: Von PaDU in die Oberfläche integrierter Button als Einstiegspunkt,(links). Dieser öffnet einen Dialog, der Nutzer unterstützt, über das Problem zu reflektieren und an die Entwicklungscommunity zu schicken (rechts)

Der Dialog hilft, über den Nutzungskontext zu reflektieren und ihn in Relation zum Nutzungsproblem zu beschreiben. Zusätzlich soll der Nutzer das Ausmaß des Problems aus seiner Sicht einschätzen, d.h. handelt es sich nur um ein kosmetisches Problem, oder konnte der Benutzer auf Grund von schlechtem Anwendungsdesign seine Arbeit nicht durchführen.

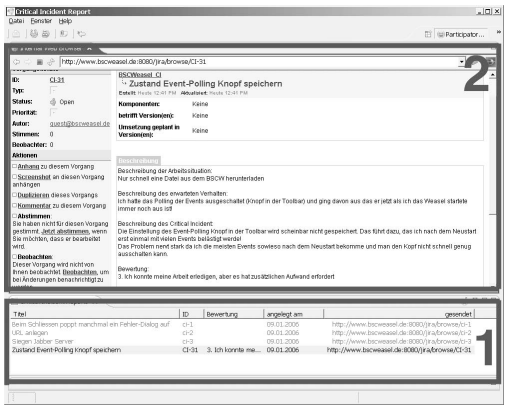

Abbildung 4: 1) Liste der selbstverfassten Beiträge 2) Das JIRA Web Interface um den Verlauf gesendeten Beitrags zu verfolgen und zu kommentieren

Einer der Unterschiede zu dem Ansatz von (Hartson, Castillo et al. 1996) und der hier vorgestellten PaDU Lösung ist der Prozess nachdem der Benutzer den Senden-Button gedrückt hat. Um der Forderung nach einem transparenten Prozess gerecht zu werden, wurde die PaDU Lösung so gestaltet, dass der Benutzer an die Stelle geführt wird, wo sein Beitrag innerhalb des JIRA Systems geführt wird. Hier werden die gesamten Anforderungen der Weiterentwicklung verwaltet. Dieses Feature erlaubt es dem Nutzer den Fortgang seines Beitrags weiterzuverfolgen, die Kommentare von anderen Nutzern/Entwicklern zu lesen bzw. selber neue Kommentare zu verfassen.

Darüber hinaus werden seine Beiträge zusätzlich lokal abgespeichert. Dies erlaubt es dem Nutzer einen Überblick über seine Beiträge zu liefern[5]. Weiterhin ist es so möglich, den Zeitpunkt des Verfassens eines Beitrags und das Veröffentlichen auf der Entwickler Community zu entkoppeln, und dem Nutzer die Möglichkeit zu geben, vor dem Veröffentlichen noch mal zu überarbeiten bzw. als nicht mehr relevant zu löschen. Realisiert wurde die lokale Verwaltung der Beiträge als so genannte Eclipse Perspektive, auf die nach dem Versenden eines Beitrags gewechselt wird. Ein Bildschirmfoto dieser Darstellung ist in Abbildung 4 zu sehen.

3 Zusammenfassung und Ausblick

Wir haben uns Aufgrund der neuen Möglichkeiten agiler Softwareentwicklung gefragt, was dies für Partizipatives Design bedeuten kann. Nach unserer Auffassung ergeben sich neue Möglichkeiten einer Partizipation im Nutzungskontext. Diese Partzipationsform kann durch räumlich verteilte Post Deployment Partizipation charakterisiert werden. Durch die Analyse von existieren Methoden in diesen Bereich haben wir eine Taxonomie abgeleitet, die eine bessere Klassifizierung der einzelnen Methoden erlaubt. Um Erfahrung in diesen Bereich zu sammeln und die Partizipation der Nutzer an der Entwicklung des BSCWeasel Groupware Client zu erhöhen, haben wir ein Komponente entwickelt und in einen konkreten Entwicklungskontext integriert, so dass die prinzipiellen Möglichkeiten der Partizipation im Nutzungskontext konkret umgesetzt werden. Das System ist zurzeit praktisch im Einsatz und wird von einem Teil der BSCWeasel Nutzer aktiv genutzt. Über das Plugin wurden bisher 71 (Stand: 5.2.2006) Berichte aus der BSCWeasel Nutzung heraus geschrieben und in JIRA verwaltet. Die Möglichkeit der Diskussion wurde bisher nicht so stark genutzt wie erhofft, jedoch stärker als vor der Einführung des Tools. Es lassen sich zu diesem Zeitpunkt auch erste Phänomene erkennen, die jedoch noch genauer analysiert werden müssen:

- PaDU vereinfacht Nutzern den Zugang zu und die Beteiligung an Designdiskursen
- Um die Partizipation zu fördern, ist ein transparenter Prozess wichtig
- Auch professionelle Entwickler können von einem PaDU Plugin profitieren, da es sie unterstützt, Gestaltungsideen aus einem Nutzungskontext zu festzuhalten.

Die weitere Forschung soll dabei die Gründe für die Nutzung bzw. Nichtnutzung bestimmen. Hierzu soll eine systematische Befragung der Nutzer, die diese Möglichkeit der Partizipation verwenden, als auch derer die es nicht verwenden durchgeführt werden. Auch soll versucht werden, PaDU in einen größeren Rahmen einzusetzen und zu evaluieren.

Daneben soll die PaDU Realisierung in Richtung kreatives, reflektionsunterstützendes Gestaltungswerkzeug für Endbenutzer weiterentwickelt werden, (vgl. PaDU Future Work in Abbildung 1). Auf Grund unserer Erfahrung mit der Benutzung des PaDU Plugin gilt es Nutzern nicht nur textliche Ausdrucksmittel, sondern insbesondere auch graphische Ausdrucksmittel zur Verfügung zu stellen. Mittels einer Nutzungskamera soll der Benutzer in die

[5] Innerhalb von JIRA ist dies nur möglich, wenn der Benutzer seine Beiträge nicht anonym absendet.

Lage versetzt werden ein Bildschirmfoto des aktuellen Systems anzufertigen und mittels eingebauten Graphikwerkzeugs umzugestalten und zu annotieren. Unsere Erfahrung hat auch gezeigt, dass PaDU als Teil einer Community orientierten, Kontextbezogenen Infrastruktur zur Aneignungsunterstützung betrachtet und entsprechend gestaltet werden sollte. Diese zu bewerkstelligen steht jedoch noch aus.

Literaturverzeichnis

Beck, K. (2000): Extreme Programming Explained, Embrace Change, Addison-Wesley.

Bleek, W.-G.; Jeenicke, M. et al. (2002): Developing web-Based Appications through e-Prototyping. Annual International Computer Software and Applications Conference, 2002.

Bleek, W.-G.; M. Jeenicke et al. (2002): Framing Participatory Design Through e-Prototyping. Proceedings of the Participatory Design Conference PDC 2002.

Castillo, J. (1997): The User-Reported Critical Method for Remote Usability Evaluation.

Floyd, C.; Reisin, F. et al. (1989): STEPS to Software Development with Users. ESEC (48-64).

Hartson, H. R.; J. C. Castillo et al. (1996): Remote Evaluation: The Network as an Extension of the Usability Laboratory. Proc. of CHI'96 Human Factors in Computing Systems.

Henderson, A.; Kyng, M. (1991): There's no place like home – continuing design Design at work.

Lieberman, H.; Paternò, F. et al., Eds. (2005): End-User Development. Springer.

Morch, A. (1997): Three Levels of End-User Tailoring: Customization, Integration, and Extension. Computers and Design in Context, The MIT Press: 51-76.

Muller, M. J.; Haslwanter, J. et al. (1997): Participatory Practices in the Software Lifecycle. Handbook of Human-Computer Interaction, Elsevier: 255-313.

Nichols, D.; McKay, D. et al. (2003): Participatory Usability: supporting proactive users. Proc. of the 4th Annual Conference of the ACM Special Interest Group on Computer Human Interaction – New Zealand Chapter (CHINZ'03).

Nichols, D.; Twidale, M.: The Usability of Open Source Software. First Monday 8(1).

Rittenbruch, M.; McEwan, G. et al. (2002): Extreme Participation – Moving extreme programming towards participatory design. Proc. of the Seventh Bienniel Participatory Design Conference.

Schön, D. A. (1983): The Reflective Practitioner. How Professionals think in Action, Basic Book.

Wulf, V. (1994): Anpaßbarbarkeit im Prozeß evolutionärer Systementwicklung. GMD-Spiegel: 41-46.