Integration von Software Engineering und Usability Engineering

Karsten Nebe[1], Sandro Leuchter[2], Astrid Beck[3]

[1]: Universität Paderborn, C-LAB, Fürstenallee 11, 33102 Paderborn,
Karsten.Nebe@c-lab.de
[2]: Fraunhofer-Institut für Informations- und Datenverarbeitung (IITB), 76131 Karlsruhe,
Sandro.Leuchter@iitb.fraunhofer.de
[3]: Hochschule Esslingen, Flandernstr. 101, 73732 Esslingen
Astrid.Beck@hs-esslingen.de

Bei der Vielfalt an existierenden Softwarelösungen spielt heutzutage die Qualität eine entscheidende Rolle für den Wettbewerb und für den Erfolg am Markt. Qualität besitzt in diesem Zusammenhang viele Ausprägungen, wie beispielsweise die Reliabilität und Stabilität der Lösung, der geringe Bedarf an nachträglichen Änderungen etc. Dazu bietet das Software Engineering eine breite Auswahl an Methoden, systematischen Vorgehensweisen (sog. Vorgehensmodelle) und Standards. Neben den genannten Qualitätsaspekten ist aber auch die Gebrauchstauglichkeit (Usability) der Softwarelösung eine weitere, bedeutende Ausprägung der Softwarequalität. Das Kriterium Usability spiegelt sich dabei nicht nur auf Seiten der Nutzer, sondern auch auf Seiten der herstellenden Organisationen wieder. Die Vorteile für die Nutzer sind weitreichend und beinhalten laut Jokela eine erhöhte Produktivität, verbesserte Qualität der Arbeit und eine erhöhte Nutzerzufriedenheit. Auf der Herstellerseite zählen dazu finanzielle Vorteile, wie beispielsweise die Reduzierung der Support- und Trainingskosten [Jo01]. Die Usability zählt im direkten Vergleich mit anderen konkurrierenden Produkten zu einem der wichtigsten Abgrenzungsmerkmale und ist damit essenzieller Teil des Softwarequalitätsmanagements [JWC01].

Gleichermaßen wichtig ist Qualität dabei auch für den Prozess der Erstellung selbst. Ideal wäre, wenn bereits durch den Prozess der Erstellung sichergestellt werden könnte, dass die Lösung eine angemessene Qualität aufweist. Genau dort setzt das Usability Engineering an, dessen Ziel die Sicherstellung der Umsetzung gebrauchstauglicher Lösungen bei der Entwicklung von Software ist.

Das Usability Engineering allein kann aber nicht ohne das Software Engineering existieren. Auch gibt es eine Reihe von Aktivitäten die beide Disziplinen gleichermaßen in ihren Prozessen vorsehen. Daher bedarf es einer Integration der beiden Disziplinen Software Engineering und Usability Engineering. Ziel ist es die Ziele und Vorgehensweisen des Software Engineering mit denen des Usability Engineering so zu vereinen, dass eine systematische und planbare Umsetzung bei der Entwicklung entsteht, welche die Faktoren Kosten, Zeit und Qualität sowohl aus Sicht des Software Engineering als auch aus Sicht des Usability Engineering angemessen berücksichtigt.

1 Integrationsansatz der gemeinsamen Aktivitäten

In der Wissenschaft und Praxis existiert heute bereits eine Vielzahl von Integrationsansätzen, die z. T. deutliche Unterschiede in der Schwerpunktsetzung aufzeigen. Einige der Ansätze beziehen sich auf die konkrete Durchführung und sind auf die Definition von Aktivitäten und Ergebnissen und deren Verknüpfung mit vorhandenen Aktivitäten des Entwicklungsprozesses gerichtet. Sie zielen auf die „sanfte Einführung“ von Usability Engineering-Aspekten auf Ebene einer gemeinsamen Basis, wie beispielsweise das Verzahnen von Ergebnissen. Insgesamt fokussieren diese Ansätze darauf, möglichst wenig organisationale und strukturelle Veränderungen herbeizuführen. So stellt Schaffer eine Methode zur Integration von Usability Engineering Aktivitäten vor, die auf der Evaluation und dem Test einer existierenden Lösung basieren [Sc04]. Die Ergebnisse fließen dabei sukzessive in den Prozess der Entwicklung ein. Ferre ermittelte eine Auswahl von generischen Usability Engineering Aktivitäten, die in Abstimmung mit Experten am geeignetsten zur Integration sind [Fe03]. Dabei wurde zusätzlich bewertet, wie fremdartig („less aliened“) diese in Bezug zum Software Engineering sind, ob sie sich mit geringem Aufwand integrieren lassen („low integration costs“) und inwieweit die allgemeine Anwendbarkeit gegeben ist („higher general applicability“). Ein weiteres Beispiel, das sich in diese Art von Integrationsansätzen einreiht, ist das Usage-Centred Design [CBN03], welches auf Ebene der Softwaremodellierung ansetzt. Fokus ist nicht per se der Nutzer, sondern die Nutzung (d. h. die intendierten Aufgaben der Nutzer und wie diese durchgeführt werden). Das Vorgehen bezieht sich auf Maßnahmen und Methoden, durch die Software, insbesondere das User-Interface, erstellt werden soll, und zielt damit auf die Parallelen zwischen Software Engineering und Usability Engineering.

Denen sehr ähnlich sind die Ansätze der gemeinsamen Spezifikation. Sie setzen auf die Kommunikation und die Sicherstellung des beiderseitigen Informationsflusses und definieren dafür gemeinsame Notationen und Artefakte des Entwicklungsprozesses. Beispiele sind die Annotierung von Use Cases mit Aufgabenbeschreibungen für die Erweiterung von Software Engineering Artefakten zur User-Interface-Spezifikation ([CL99], [Ro99]), die Ergänzung objektorientierter Software Engineering Notationen und Modelle ([NC00], [DP01], [Va01]) oder patternbasierte Vorgehensweisen, die auf wiederkehrende Muster und entsprechende nutzerzentrierte und optimierte Lösungen der Entwicklung fokussieren ([DLH02], [Ju03], [Ti05]).

Diese beiden Arten von Ansätzen (also die Verknüpfung gemeinsamer Aktivitäten sowie die gemeinsame Spezifikation) lassen sich als eine Gruppe von Ansätzen zusammenfassen, die sich auf die konkrete Anwendung, also den operativen Prozess in einer Organisation beziehen.

2 Integrationsansatz Prozessdefinition und Modellbildung

Eine weitere Gruppe von Ansätzen nimmt auf die Ebene der Prozessdefinition und Modellbeschreibung Bezug.

Diese zielen auf eine Vorgabe für die Entwicklung und beinhalten sowohl konkretere Ansätze, bezogen auf die Integration von Usability Engineering-Aktivitäten in bereits existierende Software Engineering-Modelle, als auch prinzipielle Modellaspekte, losgelöst von konkreten Software Engineering-Modellen.

Zu dieser Gruppe zählen unter anderem vollständige, eigenständige Modelle, die mit entsprechendem Fokus aus Sicht des Usability Engineering entstanden sind, oder aber auf die Integration von Usability Engineering Aktivitäten in bereits existierende Software Engineering Modelle zielen. Zu den eigenständigen Modellen des Usability Engineering zählen beispielsweise der „Star Lifecycle“ von Hix und Hartson [HH93], das „Task-Centered User Interface Design“ von Lewis und Rieman [LR94], das „Contextual Design“ von Beyer und Holtzblatt [BH98], der „Usability Engineering Lifecycle“ von Mayhew [Ma99], das „Usage Centered Design“ von Constantine [CL99] oder der „Scenario-based Approach“ von Caroll [Ca00]. Einige Beispiele für jene Ansätze, die an die an bestehende, etablierte Software Engineering Modelle anknüpfen, um durch Verzahnung der Aktivitäten eine Integration zu erreichen sind: Burmester et al. [BMS05] führen die konzeptuellen Unterschiede zwischen Usability Engineering und dem Vorgehensmodell des „Rational Unified Process“ (RUP) auf und liefern Beispiele für die Integration. Dies geschieht durch die Einbringung von Usability Engineering Aspekten in die verschiedenen Phasen und Aktivitäten des Vorgehensmodells. Meinhardt & Beck [MB05] zeigen auf, wie durch den „Vorgehensbaustein Benutzbarkeit und Ergonomie“ im V-Modell-XT [Hö08] die Usability umfassend in die Projekte einbezogen werden kann und was bei einer erfolgreichen Umsetzung zu beachten ist. Düchting et al. [DZN07] analysieren die agilen Vorgehensmodelle „Extreme Programming“ und „Scrum“ in Bezug auf die Umsetzung von Usability Engineering Aktivitäten auf Basis der DIN ISO 13407 und leiten entsprechende Empfehlungen für die nutzerzentrierte Umsetzung in der Praxis ab. Paelke & Nebe greifen diese Empfehlungen auf, erweitern sie, leiten daraus ein adaptiertes Vorgehensmodell für Scrum ab und zeigen die Anwendung anhand eines Beispiels im Bereich der „Mixed-Reality“ [PN08].

Im Wesentlichen konzentrieren sich diese Ansätze auf die Verknüpfung mit Phasen, Aktivitäten oder Ergebnissen bestehender (oder erzielter) Strukturen, welche schließlich die Grundlage für die Integration darstellen. Es geht also um die gezielte Verzahnung mit existierenden, etablierten Vorgehensweisen auf Ebene der Modelle.

3 Abstrakter Integrationsansatz

Neben der Gruppe von Ansätzen auf Ebene des operativen Prozesses und der Gruppe von Ansätzen auf Ebene der Modelle existieren noch weitere, meist abstrakte Integrationsansätze.

Sie sind losgelöst von spezifischen Modellen oder Vorgehensweisen und beschreiben organisatorische Rahmenbedingungen, Prinzipien und Paradigmen sowie Metamodelle. So untersucht Pawar die Frage nach prinzipiellen Gemeinsamkeiten zwischen Aktivitäten des Software Engineering und Usability Engineering bezüglich des Informationsaustauschs [Pa04]. Ergebnis ist ein Framework mit der Definition des Informationsaustauschs zwischen Aktivitäten des Software Engineering und des Usability Engineering sowie einer zugehörigen Koordinationsstrategie. Ähnlich dem Abstraktionsgrad eines generischen Frameworks stellen Granollers et al. ein allgemeines Prozessmodell (Usability Engineering Process Model) für die nutzerzentrierte Entwicklung vor, das traditionelle Vorgehensweisen des Software Engineering mit Prototyping und Evaluation verbindet [GLP02]. Ferre präsentiert ein Set von handhabbaren Usability-Techniken, die inkrementell und ohne großen organisatorischen Aufwand in einen Entwicklungsprozess einfließen können [Fe03]. Weitere generische Prozessmodelle beschreiben Sousa et al. mit dem „UPi: a software development process aiming at usability, productivity and integration“[SFM05], Jokela mit dem „Method-Independent Process Model of User-Centred Design“[Jo02] oder Pawar mit dem „Common Software Development Framework For Coordinating Usability Engineering and Software Engineering Activities“[Pa04].

Diese Gruppe von Ansätzen entsprechen der Vorstellung nach Standardvorgehensweisen für die Umsetzung, vergleichbar zu Standards des Software Engineering und Usability Engineering. Die Strategien zur Umsetzung sind dementsprechend abstrakt und müssen für die konkrete Anwendung auf die jeweilige Situation übertragen werden.

4 Ganzheitlicher Integrationsansatz

Alle Ansätze haben gemeinsam, zum systematischen Vorgehen der Entwicklung gebrauchstauglicher Produkte beizutragen. Wichtig dabei ist die Berücksichtigung der Sichtweisen beider Disziplinen. Je nach Schwerpunkt adressieren existierende Ansätze drei unterschiedliche Abstraktionsebenen [NZP08], [Ne09]:

- die abstrakte, übergeordnete Ebene von Standards, die als Rahmen dienen zur Sicherstellung und Wahrung von Konsistenz und Qualität, sowohl innerhalb und über die Grenzen einer Organisation hinweg,

- die Ebene von Modellen, also in der Definition des Vorgehens, welche als Vorlage für die Durchführung dient, und

- die operative Ebene (den Prozess), also der praktischen Durchführung von Aktivitäten und der Verarbeitung von Ergebnissen innerhalb der Organisation.

Um aber den Zielen und Sichtweisen beider Disziplinen gleichermaßen gerecht zu werden, bedarf es vollständiger und umfassender Integrationsansätze, die Aspekte aus allen drei Ebenen enthalten, angepasst an die Situation, den organisatorischen Kontext und an die Komplexität der Organisation.

Ziel des Workshops sind die Erarbeitung und Formalisierung von Integrationskonzepten sowie die Methoden beider Disziplinen auf den genannten Abstraktionsebenen voran zu treiben. Die ausgewählten Beiträge sollen als Arbeits- und Diskussionsgrundlage dienen, die sowohl theoretische Ansätze als auch praktische Erfahrungen schildern. Ziel ist es, langfristig ein besseres Verständnis beider Disziplinen bzw. deren Vertreter füreinander zu entwickeln und zu einem integriertem, systematischen Vorgehen zu kommen, dass auf Gemeinsamkeiten, nicht auf Trennendes setzt. Letztlich soll somit die Erreichung des Qualitätsziels Gebrauchstauglichkeit sichergestellt werden.

Der Workshop stellt die Fortführung des, im März 2008 durchgeführten, Workshops "Integration von Usability-Engineering und Software-Engineering" in Esslingen, dar, der von den GI-Fachgruppen Software-Ergonomie (FB Mensch-Computer-Interaktion) und Requirements-Engineering (FB Softwaretechnik) veranstaltet wurde. Geleitet wird der Workshop von Karsten Nebe, Sandro Leuchter und Friedrich Strauß, unterstützt durch das Programmkommittee, bestehend aus Andreas M. Heinecke, Astrid Beck, Markus Dahm, Kai-Christoph Hamborg, Rainer Heers, Andrea Hermann und Barbara Paech.

Literaturverzeichnis

[BH98] Beyer, H. & Holtzblatt, K. (1998): Contextual Design: defining customer-centered systems. Morgan Kaufmann, San Francisco

[BMS05] Burmester, M., Machate, J. & Sandweg, N. (2005): Integration benutzerzentrierter Methoden in die Software-Entwicklung (Integrating User-Centered Design Methods with Software Engineering). In: Ziegler, J., et al. (eds.) i-com Zeitschrift für interaktive und kooperative Medien, vol. 4, pp. 31-40, Oldenburg Verlag, München; Wien

[Ca00] Caroll, J. M. (2000): Making use: Scenario-based design of human-computer interactions. MIT Press, Cambridge

[CL99] Constantine, L. & Lockwood, L. (1999): Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design. Addison-Wesley (ACM Press), NY

[CBN03] Constantine, L. L., Biddle, R. & Noble, J. (2003): Usage-centered design and software engineering. Models for integration. In: IFIP Working Group 2.7/13.4, ICSE 2003 Workshop on Bridging the Gap Between Software Engineering and Human-Computer Interaction. Portland

[DP01] Da Silva, P. P. & Paton, N. W. (2001): A UML-based Design Environment for Interactive Applications. In: Proceedings of the second international Workshop on User Interfaces to Data Intensive Systems UIDIS ’01, p. 60. IEEE Computer Society Press, Los Alamitos

[DZN07] Düchting, M., Zimmermann, D., Nebe, K. (2007): Incorporating User Centered Requirement Engineering into Agile Software Development. In: Proceedings of the HCII 2007, vol. 4550. Springer, Berlin

[DLH02] Duyne van, D., K., Landay, J. A., Hong, J. I. (2002): The Design of Sites: Patterns, Principles, and Processes for Crafting a Customer-centered Web Experience. Addison-Wesley, Boston

[Fe03] Ferre, X. (2003): Integration of Usability Techniques into Software Development Process. Bridging The Gaps Between Software Engineering and Human-Computer Interaction. In: Proceedings of ICSE’03 International Conference on Software Engineering, pp. 28-35, ACM Press, Portland

[GLP02] Granollers, T., Lorès, J. & Perdrix, F. (2002): Usability Engineering Process Model. Integration with Software Engineering. In: Proceedings of the Tenth International Conference on Human-Computer Interaction. pp. 965-969. Lawrence Erlbaum Associates, New Jersey

[HH93] Hix, D. & Hartson, H. R. (1993): Developing User Interfaces: Esuring Usability Through Product and Process. John Wiley & Sons, New York

[Hö08] Höhn, R., Höppner, S., Rausch, A., Broy, M. (2008): Das V-Modell XT: Grundlagen, Methodik und Anwendungen, Springer

[Jo01] Jokela, T. (2001): An Assessment Approach for User-Centred Design Processes. In: Proceedings of EuroSPI 2001, Limerick Institute of Technology Press, Limerick

[Jo02] Jokela, T. (2002): A Method-Independent Process Model of User-Centred Design. In: Kluver, B. V. (ed.) Proceedings of the IFIP 17th World Computer Congress. vol. 226 pp. 23-38. ACM Press, Deventer

[JWC01] Juristo, N., Windl, H. & Constantaine, L. (2001): Special Issue on Usability Engineering in Software Development. In: IEEE Software, vol. 18. IEEE Computer Society Press, Los Alamitos

[Ju03] Juristo, N., Lopez, M., Moreno, A. M. & Sánchez, M. I. (2003): Improving software usability through architectural patterns. In: Proceedings of the International Conference on Software Engineering. pp.12-19. IEEE Computer Society Press, Los Alamitos

[LR94] Lewis, C. & Rieman, J. (1994): Task-Centered User Interface Design. A Practical Introduction. Retrieved from http://hcibib.org/tcuid/index.html, on: 13.08.08

[Ma99] Mayhew, D. J. (1999): The Usability Engineering Lifecycle. Morgan Kaufmann, San Francisco

[MB05] Meinhardt, H. J. & Beck, A. (2005): Usability im neuen V-Modell XT. In: Stary, C. (eds.) Mensch & Computer 2005: Kunst und Wissenschaft – Grenzüberschreitungen der interaktiven ART. Oldenbourg, München

[NZP08] Nebe, K., Zimmermann & D., Paelke, V. (2008): Integrating Software Engineering and Usability Engineering. In: Pinder, S. (ed.) Advances in Human-Computer Interaction, chapter 20, pp. 331-350. I-Tech Education and Publishing, Wien

[Ne09] Nebe, K. (2009): Integration von Usability Engineering und Software Engineering: Konformitäts- und Rahmenanforderungen zur Bewertung und Definition von Softwareentwicklungsprozessen. Aachen, Germany, Shaker Verlag

[NC00] Nunes, N. J. & Cunha, J. F. (2000): Towards a UML profile for interaction design: the Wisdom approach. In: Proceedings of UML2000, LNCS, vol. 1939, pp. 101–116. Springer, Berlin

[PN08] Paelke, V. & Nebe, K. (2008): Integrating agile methods for mixed reality design space exploration. In: Proceedings of the 7th ACM conference on Designing interactive systems DIS '08. ACM Press, New York

[Pa04] Pawar, S. A.(2004): A Common Software Development Framework For Coordinating Usability Engineering and Software Engineering Activities. Master Thesis, Blacksburg, Virginia

[Ro99] Rosson, M. B. (1999): Integrating development of task and object models. In: Communication of the ACM, vol. 42(1), pp. 49-56. ACM Press, New York

[Sc04] Schaffer, E. (2004): Institutionalization of usability: a step-by-step guide. Addison-Wesley; Pearson Education, Inc., Boston

[SFM05] Sousa, K., Furtado, E. & Mendoca, H. (2005): UPi: a software development process aiming at usability, productivity and integration. In: Proceedings of the 2005 Latin American conference on Human-computer interaction CLIHC '05. ACM Press, NY

[Ti05] Tidwell, J. (2005): Designing Interfaces: Patterns for Effective Interface Design. O'reilly, Sebastopol

[Va01] Van Harmelen, M. (2001): Object Modeling and User Interface Design: Designing Interactive Systems. Addison-Wesley, Boston