Von System zu Nexus - Welche methodischen Veränderungen bei menschenzentrierter Entwicklung erfordert der technologische Wandel?

Christiane Floyd

floyd@informatik.uni-hamburg.de

Abstract: Meine Heimatdisziplin Software Engineering hat zwei problematische Abgrenzungen vollzogen: zum menschlich-sozialen Kontext ebenso wie zur technischen Einbettung von Software. Dadurch sind Fragen der Software-Nutzung sowie Fragen von Hardware-Software-Wechselwirkungen innerhalb der Disziplin nicht thematisierbar. Stattdessen ist eine Familie von einander überlagernden Disziplinen der Informatik entstanden, in deren Schnittmenge das Anliegen des Workshops „Soziotechnisches Systemdesign im Zeitalter des Ubiquitous Computing" verortet werden kann. In meiner Forschung habe ich mich, ausgehend vom softwaretechnischen Paradigma, mit Prozessen von Softwareentwicklung und -nutzung beschäftigt. Da ich dabei die Anliegen der beteiligten Menschen in den Vordergrund stelle, deren Arbeits- und Kommunikationsmöglichkeiten oder andere Interessen auf unterschiedliche Weise von Software tangiert werden, spreche ich allgemein von menschenzentrierter Entwicklung. Die entsprechenden Methoden sind nicht nur von der Softwaretechnik geprägt, sondern auch durch Ansätze, die heute der Softwareergonomie, den Informationssystemen und dem Requirements Engineering zugeordnet werden. Diese Erfahrungen sind der Hintergrund meines Vortrags, die Übertragung auf den Bereich Ubiquitous Computing kann ich nur ansatzweise leisten.

Die Orientierung auf „soziotechnische Systeme" stammt aus den 40er Jahren, wurde in Norwegen schon um 1950 in die Informatik übernommen und diente dazu, das sinnvolle Zusammenwirken von Menschen und Maschinen zu betrachten. Auch heute ist der Begriff „soziotechnisches Systemdesign" hilfreich, wenn er als Mantelbegriff verstanden wird, der auf eine Gesamtheit unterschiedlicher Ansätze verweist. Im engeren Sinn ist der Begriff aber vielfach kritisiert worden und hat sich als nicht ausreichend für die menschenzentrierte Entwicklung erwiesen: zum einen, weil der Begriff „System" problematisch ist: schon auf technischer Ebene ist das System nicht eindeutig abzugrenzen und bezogen auf den sozialen Kontext stellt sich die Frage, ob dieser überhaupt als System aufgefasst werden sollte; zum anderen, weil dieser Begriff menschliche und technische Elemente integriert, aber ihr Verhältnis zueinander nicht beschreibt und so in den Augen vieler eine Gleichartigkeit und Gleichrangigkeit unterstellt; und letztlich weil die Art der Nutzung des technischen Systems im menschlichen Kontext offen bleibt.

Methoden im Bereich menschenzentrierter Entwicklung gehen über allgemeine softwaretechnische Einsichten vor allem in folgenden Punkten hinaus:

- Visionsbildung für das zu entwickelnde System im Einsatz
- Nutzung von Leitbildern und Metaphern zur Orientierung im Design
- Partizipation = Zusammenarbeit mit den an der Entwicklung Interessierten, insbesondere mit den Benutzerinnen und Benutzern
- Iterative Vorgehensweisen, wie Prototyping oder inkrementelle Systementwicklung

- Etablierung, Monitoring und Reflexion des Entwicklungsprozesses
- Berücksichtigung der rechtlichen Grundlagen, bezogen auf alle Betroffenengruppen.

Grundlegende Ansätze in diesen Bereichen wurden seit circa 1980 entwickelt, als interaktive Systeme und Personal Computing in die Praxis eindrangen, das Arbeitsleben revolutionierten, und die Metaphern vom Computer als Werkzeug und Medium motivierten. Charakteristisch für die Systeme dieser Zeit ist, dass ihre Nutzung entweder bezogen auf eine Art von Organisation bzw. Arbeitstätigkeit gedacht war, oder dass sie sich an einzelne Nutzer wandten, die Systeme für allgemeine Tätigkeiten (wie etwa Textbearbeitung) einsetzten. Inzwischen gibt es zu allen genannten Punkten eine Vielfalt von Vorgehensweisen, die zu Methoden ausgearbeitet und in Pilotprojekten erprobt wurden. Nur wenige Methoden haben Breitenwirkung entfaltet, umso mehr aber die ihnen zugrunde liegenden Einsichten - so sehr, dass deren Ursprung nicht immer bis zu den Wurzeln zurück verfolgt werden kann.

Die Wechselwirkungen mit Hardware und Systemsoftware wurden insofern relevant, als sie bestimmten, welche Entscheidungen an der Benutzungsoberfläche möglich und sinnvoll waren. Usability (Gebrauchstauglichkeit oder Benutzbarkeit) wurde nur durch die Eigenschaften der Benutzungsschnittstelle bestimmt, obwohl technische Eigenschaften - wie Sicherheit, Effizienz oder Anpassbarkeit (z. B. Portabilität, Erweiterbarkeit auf große Nutzergruppen, und Mehrkanalfähigkeit) ebenfalls erhebliche Auswirkungen darauf haben, ob sich ein technisches System im Einsatz als nützlich erweist.

Neben der Methodik zur Softwarekonstruktion im engeren Sinne liefert die Softwaretechnik zwei wesentliche Grundlagen für die Methodik menschengerechter Systementwicklung:

1. Ein Verständnis von Softwareprojekten, wobei die im Wasserfallmodell aufgezeigten „Phasen" als logische Ebenen aufgefasst werden, die zeitlich flexibel miteinander verzahnt werden können
2. Ein Verständnis von Softwarearchitektur, das die Beherrschung von Komplexität, die Lokalisierung von Änderungen und die Entkopplung verschiedener Entwurfsanliegen gestattet

Nach meiner Überzeugung liegt dabei der Schwerpunkt heute auf der Architektur. Ich sehe „architekturzentrierte Softwaretechnik" als Grundlage als Voraussetzung für die Durchführbarkeit von menschenzentrierter Systemgestaltung.

Seit den 90er Jahren gab es einen Entwicklungssprung durch die weltweite Vernetzung und die damit verbundene Entwicklung von Netzsoftware und -applikationen. Durch diesen technischen Wandel entstanden neue Herausforderungen an Methoden: Systementwicklung für Communities; systeme, die nicht mehr festen Anforderungen gehorchen, sondern offene Handlungsräume bieten sollten; verteilte Architekturformen mit Wechselwirkungen zwischen lokal und remote durchgeführten Leistungen; Globalisierung der Softwareentwicklung - und so weiter. Partizipation und iterative Vorgehensweisen mussten neu durchdacht werden, rechtliche Grundlagen gab es zum Teil noch nicht und mussten sinnvoll vorausgedacht werden, und so weiter.

Zur gleichen Zeit wurden in Organisationen einzelne Systeme in Softwarelandschaften integriert, die als ganzes stimmig sein und die Interoperabilität zwischen Organisationen gestatten sollte. Wegen der allgemeinen Verfügbarkeit und der selbstverständlichen Voraussetzung von Software in vielfältigen Kontexten, wurde es üblich, von Softwareinfrastruktur zu sprechen.

Im Zeitalter des Ubiquitous Computing schließlich dient Software in vielen Fällen wesentlich dazu, eine Verbindung zwischen verschiedenen technischen Geräten zu ermöglichen. Das will die Metapher „Nexus" ausdrücken. Im Gegensatz zum nach wie vor gültigen Begriff System stellt sie die Verknüpfung in den Vordergrund, wobei

beim Design nicht klar ist, welche technische Umgebung beim Einsatz des zukünftigen Systems vorliegen wird, weil sich diese laufend weiter entwickelt und Systeme in verschiedenen Kontexten eingesetzt werden können. Angesichts der Vielfalt an Systemen und Methoden ist nicht Ratlosigkeit das Gebot der Stunde, sondern Achtsamkeit. Es gilt, das eigene Entwicklungsszenario zu verstehen und mit verfügbaren Methoden in Verbindung zu bringen. Dabei geht es nicht um Methoden als ein- für allemal festgefügte Gebilde. Nach meiner Überzeugung gibt es Methoden in diesem Sinne nicht. Vielmehr geht es um situierte Prozesse im jeweiligen Projekt, in denen Methoden ausgewählt, angepasst, genutzt und (weiter-)entwickelt werden.