„Deconstructive Software Design“ und praxis-basiertes Lernen

Roswitha HOFMANN und Doris ALLHUTTER

Anknüpfend an prozessorientierte TA-Ansätze wie ‚Constructive Technology Assessment’ (CTA) und ‚Innovationsorientierte Technikfolgenabschätzung’ (ITA) stellen wir unseren Software Design Ansatz ‚Deconstructive Design’ vor. Der Ansatz verbindet eine Perspektive auf Softwareentwicklung als soziale multidimensionale Aushandlungsprozesse mit praxis-basierten, situativen und kontextbezogenen Lernkonzepten. Im Sinne der Einbeziehung sozialer Aspekte in informationstechnologische Entwicklungsprozesse unterstützt „Deconstructive Design“ praktische Softwareentwicklungsprojekte methodisch und integriert durch den Einsatz einer dekonstruktiven Methode (des sog. ‚Mind Scriptings’) eine Metaebene im Entwicklungsprozess, durch die eine wertesensible Reflexion eingebracht werden kann. Der diskursive Ansatz der Dekonstruktion zielt in diesem Zusammenhang auf das Sichtbarmachen implizit gesellschaftsrelevanter Designentscheidungen und die Antizipation daraus resultierender gesellschaftlicher Folgewirkungen ab.

Wie CTA versteht ITA Unternehmen selbst als Hauptakteure von und für TA und beide Ansätze versuchen Antizipation, Reflexivität und Lernen (vgl. Steinmüller/Tacke/Tschiedel 1999; Sundermann 1999; Schot/Rip 1996) in den wirtschaftlichen Planungsprozess zu integrieren. Technikentwicklung wird im Rahmen dieser Ansätze als iterativer und häufig konfliktueller Prozess verstanden, der in ein Netzwerk verflochtener Interessenskonstellationen eingebunden ist. Diskursive und beteiligungsorientierte Verfahren rücken damit in den Vordergrund und die Frage nach den gesellschaftlichen Machtkonstellationen, in denen Technikentwicklung stattfindet, wird aufgeworfen. Das Wissenschaftsverständnis der Technoscience begreift Technikforschung und -entwicklung noch weitreichender als soziokulturelle Aktivitäten, die durch ihre bedeutungsgebenden Praktiken soziale Verhältnisse mitgestalten. Wissenschafts- und TechnikforscherInnen wie McKenzie und Wajcman (1999), Cockburn und Omrod (1997) oder Norbjerg und Kraft (2002) beschreiben Technikentwicklung als sozialen und vergeschlechtlichten Prozess, in dem grundlegende Paradigmen der Informatik als soziale Strukturen und Machtverhältnisse wirksam werden.

Auch im Rahmen der Forschung über Software-Entwicklungsprozesse rückt die Frage nach sozialen Aspekten immer mehr in den Vordergrund und es zeigt sich vermehrt der Bedarf an (quantitative Zugänge ergänzenden) qualitativen Forschungszugängen und kritischen Design-Ansätzen. Qualitative Herangehensweisen sollen vor allem dazu dienen, die sozialen und kooperativen Dimensionen der Softwareentwicklung zu verstehen, um in einem weiteren Schritt zur Verbesserung von Methoden und Prozessen zu führen (vgl. Dittrich u.a. 2008). Aus dem Diskurs zu Human-Computer Interaction (HCI) entwickelten sich bereits seit den 1970er Jahren eine Reihe kritischer Design-Ansätze, die die Notwendigkeit, bedeutungsgebende Praktiken in der Softwareentwicklung zu reflektieren, betonen (vgl. Mathiassen 1998; Agre 1997; Schön 1983; Gaver u.a. 2004; Friedman u.a. 2006; Sengers u.a. 2005).

Soziale Dimensionen, wie etwa geschlechter- und diversitätsrelevante Aspekte, werden in der Produktentwicklung oft weitgehend ausgeblendet und können somit nicht verhandelt oder diskutiert werden. Stattdessen wird in den verschiedenen Phasen des Systemdesigns oft unbewusst auf Verfahren zurückgegriffen, die sich sowohl negativ auf die Produktqualität auswirken als auch Ansprüchen sozialer Nachhaltigkeit nicht genügen. Im Rahmen geschlechter- und diversitätssensibler Auseinandersetzungen mit Informations- und Kommunikationstechnologien wurde beispielsweise immer wieder kritisiert, dass

EntwicklerInnen ihre eigenen Präferenzen und Kompetenzen unbewusst als repräsentativ für jene der zukünftigen NutzerInnen sehen, wodurch der Kreis der potentiellen AnwenderInnen eingeschränkt wird. Durch diese sogenannte „I-Methodology“ (Akrich 1995) sowie durch das Zurückgreifen auf imaginäre NutzerInnenbilder werden soziale Vorstellungen und Stereotype aktiviert und unbewusst und unreflektiert in Produkte eingeschrieben. Eine an einem sozial nachhaltigen Qualitätsverständnis ausgerichtete Softwareentwicklung muss wertesensibel gestaltet werden, wodurch neben technikzentrierten Qualitätskriterien auch soziale Aspekte grundsätzlich einzubeziehen sind (vgl. Allhutter/Hanappi-Egger 2006).

‚Deconstructive Design’ knüpft an die Tradition kritischer Design-Ansätze an und stellt die Aushandlung von Designentscheidungen in Organisationen und Teams in den Mittelpunkt. Diese Aushandlungsprozesse finden vor dem Hintergrund gesellschaftlicher Diskurse statt, sie sind durch formale Strukturen wie auch Wissenshierarchien geprägt und zeichnen sich in der Regel durch ein stark technikzentriertes professionelles Selbstverständnis der EntwicklerInnen aus (vgl. Allhutter/Hanappi-Egger/John 2007). Zwar ist die Rolle von individuellen und organisationalen Lernprozessen bereits seit längerem Thema im Bereich der Organisations- und Innovationsforschung (vgl. Flood/Romm 2006). Kritische Software-Designansätze, die einen geeigneten Ausgangspunkt für eine wertesensible Softwareentwicklung bieten, vernachlässigen allerdings häufig das Potenzial praxis-basierten, situativen und kontextsensiblen Lernens (Lave/Wenger 1991), das zu veränderungsorientierten Entwicklungsprozessen beiträgt. ‚Deconstructive Design’ integriert reflexives Handeln als Teil von individuellen und organisationalen Lernprozessen und ermöglicht durch die Implementierung von „Triple-Loop-Learning“ Prozessen (vgl. Flood/Romm 1996) die Entwicklung nachhaltiger Lernfähigkeit in Projektroutinen.

Referenzen

Agre, P., 1997, Computation and Human Experience, Cambridge, University Press.

Akrich, M., 1995, User Representations: Practices, Methods and Sociology. In: Rip, A., Misa, T.J., Schot, J. (Hg.): Managing Technology in Society. The Approach of Constructive Technology Assessment, Pinter Publishers, London/New York, 167-184.

Allhutter, D., Hanappi-Egger, E., 2006, The Hidden Social Dimensions of Technologically Centred Quality Standards: Triple-Loop-Learning as Process Centred Quality Approach. In: Dawson, R./Georgiadou, E./Linecar, P./Ross, M./Staples, G. (Hg.): Perspectives in Software Quality, The British Computer Society, 179-195.

Allhutter, D., Hanappi-Egger, E., John, S., 2007, Gendered Software Design: Zur Sichtbarmachung von Gender Scripts in technologischen Artefakten, Forschungsbericht der Abteilung Gender and Diversity in Organizations.

Cockburn, C., Omrod, S., 1997, Wie Geschlecht und Technologie in der sozialen Praxis „gemacht“ werden. In: Döllinger, I./Krais, B. (Hg.): Ein alltägliches Spiel. Geschlechterkonstruktionen in der sozialen Praxis, edition suhrkamp, Frankfurt am Main, 17-47.

Dittrich, Y., Rönkkö, K., Eriksson, J., Hansson, C. and Lindeberg, O., 2008, Cooperative method development. Combining qualitative empirical research with method, technique and process improvement, Empirical Software Engineering 13(3), 231-260.

Flood, R, Romm, N., 1996, Triple Loop Learning, John Wiley and Sons, London, UK.

Friedman, B., Kahn, P. H., Jr., Borning, A., 2006, Value Sensitive Design and information systems. In: P. Zhang & D. Galletta (Hg.): *Human-computer interaction in management information systems: Foundations,* 348-372, Armonk, New York; London, England.

Gaver, W. W., Bowers, J., Boucher, A., Gellerson, H., Pennington, S., Schmidt, A., Steed, A., Villars, N., and Walker, B., 2004, The drift table: designing for ludic engagement. In CHI 2004 Extended Abstracts, Vienna, Austria, ACM Press, 885-900.

Lave, J., Wenger, E., 1991, Situated Learning. Legitimate Peripheral Participation. Cambridge University Press, Cambridge, U.K.

Mathiassen, L., 1998, Reflective Systems Development, In: Scandinavian Journal of Information Systems, 10(1&2), 67-118.

McKenzie, D., Wajcman, J., 1999, The Social Shaping of Technology, 2. Auflage, Open University press, Buckingham.

Norbjerg, J., Kraft, P., 2002, Software Practice is Social Practice. In: Dittrich, Y., Floyd, C., Klischewski, R. (Hg.): Social Thinking, Software Practice, MIT Press, Cambridge, Massachusetts/London, 205-222.

Schön, D., 1983, The Reflective Practitioner: How Professionals Think in Action, Basic Books.

Schot, J., Rip, A., 1996, The Past and Future of Constructive Technology Assessment, Technological Forecasting and Social Change 54, 251-268.

Sengers, P., Boehner, K., David, S., Kaye, J.J., 2005, Reflective Design. In: Proceedings of the 4th Decennial Conference on Critical Computing: Between Sense and Sensibility, Dänemark, 49-58.

Steinmüller, K., Tacke, K., Tschiedel, R., 1999, Innovationsorientierte Technikfolgenabschätzung. In: Bröchler, S., Simonis, G., Sundermann, K. (Hg.): Handbuch Technikfolgenabschätzung, edition sigma, Berlin, 129-144.

Sundermann, K., 1999, Constructive Technology Assessment. In: Bröchler, S., Simonis, G., Sundermann, K. (Hg.): Handbuch Technikfolgenabschätzung, edition sigma, Berlin, 119-128.