Beteiligung von Mitarbeitern an der Softwaregestaltung

In den meisten Fällen wird die Einführung einer neuen Software in betrieblichen Organisationseinheiten Top-Down angeordnet. Wer kennt das nicht, ein Softwareupdate wird angekündigt und die MitarbeiterInnen müssen sich mit den neuen Funktionalitäten und Schaltflächen selbstlernend auseinandersetzen. Die Unzufriedenheit seitens der Kollegen ist meist groß und wird in den meisten Fällen neben den sonstigen Arbeitsroutinen als eine erhebliche Belastung empfunden. Routinen müssen neu erlernt werden, häufig werden sogar die Arbeitsprozesse verändert, was eine Veränderung des gewohnten Workflows und damit auch eine Veränderung der eigenen Arbeit zur Folge hat. Über all diese Effekte, sind sich die Programmierer von Software meist nicht im Klaren. Bei der Auftragsvergabe und den entsprechenden Definition werden in der Regel nicht die Anwender befragt, sondern nur die Geldgeber. Dabei wird Software häufig als ein Instrument der Arbeitsbewältigung und weniger als eine Werkzeug der Arbeitsorganisation verstanden. Vielleicht wegen ihrer Stofflosigkeit wird Software häufig unterschätzt. Fehler im Umgang mit Software wird häufig beim Nutzer selbst und viel zu selten im Softwaredesign gesucht. Und die Gestaltung von Software wird häufig viel zu technisch und viel zu wenig als eine soziale Auseinandersetzung verstanden. Dabei bezieht sich Software heute in den seltensten Fällen auf einen lokalen Rechner, sondern ist ganz häufig eine im Netzwerk laufende Anwendung, die von vielen Nutzern gleichzeitig genutzt wird und damit auf einer Reihe an Vereinbarungen angewiesen ist. Neben einem schon älteren Trend, dem kooperativen Prototyping hat sich auch die sogenannte Open-Source-Entwicklung und das sogenannte Enterprise 2.0 durchgesetzt um heute beteiligungsorientiert Software zu gestalten, zu designen und zu nutzen. Ich möchte im folgenden auf alle diese drei Ansätze im Einzelnen eingehen.
Das kooperativen Prototyping (CASE-Tool-Engineering) ist die wohl älteste Methode, um Software in der Gruppe zu gestalten. Dabei werden ausgehend von der graphischen Oberfläche sehr detailliert und und kleinteilig die einzelnen Funktionalitäten implementiert. Dabei ist die Funktionsfähigkeit noch nicht so wichtig, wie die Auseinandersetzung der Kollegen mit der Oberfläche, der Gestaltung und ihrer Nutzung und der Konfrontation mit der sich dadurch verändernden Arbeitsabläufe. Beim kooperativen Prototyping steht ein Moderator - es muss nicht unbedingt ein Programmierer sein - an einem Rechner, der entsprechende Bildschirm wird per Beamer der gesamten Gruppe zugänglich gemacht. Durch schrittweises Vortasten, welche Funktionalität durch welchen Klick ausgelöst wird, werden Schritt für Schritt die neuen Arbeitsvorgänge abgebildet und sind Produkt der Gestaltung durch die anwendenden Kollegen. Die so entstehende Software ist ein jedem Fall ein Produkt der Gruppe und ist als solches in der Regel auch anerkannter, was wiederum zur Folge hat das der Umgang mit Softwarefehlern viel produktiver ist, da sie als Folge eines Verbesserungsprozesses des selbst zu schaffenden Produktes verstanden werden. Im Zuge der immer komplexer werdenden Programmstrukturen hat sich das kooperative Prototyping auf Dauer allerdings nicht durchsetzen können. Konnte dies früher auf ganz Programmstrukturen angewendet werden, so sind heute mit dem kooperativen Prototyping Verfahren nur einzelne Formulare, Ansichten und abgrenzbare Workflows zu gestalten.
Bei der sogenannten Open Source Entwicklung geht man grundsätzlich erst einmal davon aus, das Software nie fertig ist. Sie ist immer wieder neu den Bedingungen der sich verändernden Arbeitswelt anzupassen, dabei fällt es in der Open Source Entwicklung meist schwer komplett neue Konzeptionen zu entwickeln, was dazu führt, dass sich die Software zwar flexibel anpassen lässt, in der Regel aber mit komplett neu zu organisierenden Arbeitsprozessen nicht zu vereinbaren ist. Open Source steht dabei für „Offene Quelle“. Das bedeutet, dass jeder, der programmieren kann, diese Software auch verändern kann. Im Prozess der beteiligungsorientierten Softwaregestaltung ist die Open Source Entwicklung eine, die versucht auch möglichst viele Anwender zu beteiligen.
Grundsätzlich teilt sich Softwareentwicklung in drei Phasen auf:

1. Erstellung eines Pflichtenheft
2. Programmierung
3. Bugfixing

Im sogenannten Pflichtenheft werden alle Funktionalitäten und Wünsche der Anwender aufgenommen. Über ein Kommunikationsmedium z.B. eine Mailingliste, ein Wiki oder auch ein Webforum werden Wünsche der Anwender gesammelt aber auch untereinander diskutiert. Software, die so entsteht ist häufig Teil einer sehr konkreten Problemlösung und bezieht sich weniger auf das Bereitstellen von Standardanwendungen, wie z.B. eine Textverarbeitung o.ä. Auch wenn aus dem Bereich die wohl bekanntesten Open Source Entwicklungen hervorgegangen sind, wie z.B. das Betriebssystem Linux oder auch das Office-Paket OpenOfficeorg, das sich vor allem in der öffentlichen Verwaltung immer größerer Beliebtheit erfreut. Solche Open-Source-Entwicklungen können durch die hohe Beteiligung von Anwendern zu sehr komplexen Systemen wachsen, sie sind in den meisten Fällen aber sehr unkomplexe, kleine Softwarelösungen. Die Aufgabe der Softwareentwickler im Open Source Bereich ist häufig die konkrete Umsetzung der im Pflichtenheft verlangten Funktionalitäten. Im letzten Schritt, dem Bugfixing wird jeder potentielle Fehler an die Programmierer gemeldet und über ein System Punkt für Punkt abgearbeitet. Die entsprechende Fehlerbehebung wird anschließend über das System dem Fehlermeldenden zurückgegeben, so das er eine Bestätigung erhält, dass seine Meldung bearbeitet wurde. Ist das Pflichtenheft abgearbeitet und sind alle Auftauchenden Fehler beseitigt, wird die Software versioniert, das heißt sie hat einen mehr oder weniger stabilen Zustand erreicht und kann weiterentwickelt werden. Damit wiederholt sich der Prozess mit der Erstellung eines neuen Pflichtenhefts. Open Source Software ist meistens so gestaltet, dass sie nicht nur Beteiligungsmöglichkeiten für den Programmierer zur Verfügung stellt, sondern auch für engagierte Anwender (z.B. Hilfetexte schreiben, übersetzen, testen der Software, ).
Der Begriff „Enterprise 2.0“ bezieht sich weniger auf die partizipative Gestaltung von Software selbst, als vielmehr auf eine Unternehmenskultur weg von der hierarchischen, zentralen Steuerung und hin zur autonomen Selbststeuerung von Teams, die von Managern eher moderiert als geführt werden. Software selbst bekommt den Status bottom-up Prozesse zu organisieren. Dahinter steckt die Idee Teams in lockeren Netzwerken zu organisieren. Die Software hat möglichst offen zu sein und sich jedem Arbeitsprozess flexibel anzupassen, ohne das ein professioneller Programmierer erforderlich ist. Die Softwaregestaltung liegt vollkommen in den Händen der Nutzer. Die Halbwertzeit solcher Software ist meist deutlich geringer. Sie ist wertlos, sobald das Projekt abgeschlossen und das Team damit auseinander fällt. Software wird damit zu einer Ressource, auf dessen Basis Produkte entstehen.
Grundlegend für all diese Ansätze sind zwei Annahmen:

1. Sich selbst und die Kolleginnen als Wissensarbeiter zu verstehen. „Ein Wissensarbeiter ist jemand, der mehr über seine Tätigkeit weiß als jeder andere in der Organisation“, so Peter F. Drucker, einer der großen Managementtheoretiker der 50er Jahre. Ausgehend von dieser Definition ist jeder im Betrieb ein Wissensarbeiter. D.h. Die Verantwortung liegt nicht mehr in den Hierarchien, sondern bei jedem einzelnen Kollegen.
2. Einsatz von Software definiert immer auch den Arbeitsprozess. Nicht nur den eigenen, sondern auch den der Kolleginnen. Denn Software ist heute immer auf Netzwerke angewiesen. Selbst die Textverarbeitung greift auf Dokumente zu die auf entfernten Laufwerken zu finden sind.

Die Konsequenz ist, dass es die One-Best-Way Lösung nicht mehr gibt. Es gibt höchstens optimale Lösungen und das sind letztlich die mit der höchsten Akzeptanz bei den BenutzerInnen. Die allerdings ist Resultat der Beteiligung der ArbeitnehmerInnen am Prozess der Systementwicklung. Den KollegInnen muss klar sein, dass Software gestaltbar ist und kein starres Werkzeug. Man muss nicht mit Fehlern und Unzulänglichkeiten leben, sondern sich dafür einsetzen, dass sie an die Arbeitswirklichkeit angepasst wird. Werden die ArbeitnehmerInnen ernsthaft bei der Einführung neuer Software einbezogen, könnte ihre Einführung mit der gemeinsamen Erstellung des Pflichtenhefts beginnen und mit Sicherheit optimaleArbeitsprozesse hervorbringen, da jeder Wissensarbeiter Experte seiner eigenen Tätigkeit ist.