Usability Engineering didaktischer Software

Michael Janneck
Universität Hamburg, Fachbereich Informatik

Abstract: Um zu einem sinnvollen und erfolgreichen Einsatz didaktischer Software im Unterricht zu kommen, ist ein methodisches Vorgehen bei der Gestaltung eine notwendige Voraussetzung. Usability Engineering-Vorgehensmodelle müssen dafür angepasst werden, denn sie berücksichtigen die Besonderheiten didaktischer Anwendungskontexte nicht. In diesem Beitrag werden diese Besonderheiten herausgearbeitet und ein Vorgehensmodell für das Usability Engineering didaktischer Software vorgeschlagen, das auf der Idee der Koevolution von Software und Unterrichtskonzepten basiert.

1 Einleitung

Der Einsatz von didaktischer Software[1] im Unterricht[2] hat oft nicht die gewünschten positiven Auswirkungen auf das Lernen. Das liegt zum einen sicher daran, dass die sog. „Neuen Medien“ vielfach als Wundermittel zur Lösung altbekannter pädagogischer, didaktischer und organisatorischer Probleme in der Bildung hochstilisiert werden – ein Anspruch der nicht eingelöst werden kann. Zum anderen sind die Ursachen aber auch darin zu suchen, dass die Entwicklung didaktischer Software nicht mit Methoden durchgeführt wird, die mit einiger Sicherheit zu gebrauchstauglichen Ergebnissen führen.

Für die Entwicklung von E-Learning-Systemen sind zwar bereits einige Vorgehensmodelle vorgeschlagen worden, z. B. von Kerres (Ke01), aber aus Sicht der Informatik ist daran zu kritisieren, dass diese sich nicht am Stand der Kunst des Usability Engineering orientieren, der ein partizipatives und zyklisches Vorgehen favorisiert. Andererseits können Vorgehensmodelle des Usability Engineering nicht unverändert übernommen werden, da sie in betrieblichen Anwendungskontexten entstanden sind und die Besonderheiten von Unterricht nicht berücksichtigen. In diesem Beitrag stelle ich daher ein Vorgehensmodell zur Entwicklung didaktischer Software vor, das auf aktuellen Usability Engineering-Vorgehensmodellen aufbaut und das die Besonderheiten didaktischer Anwendungskontexte berücksichtigt.

Dazu resümiere ich zunächst wichtige Prinzipien des Usability Engineering und gehe dann auf die Besonderheiten didaktischer Anwendungskontexte sowie die Rolle von Medien in der Unterrichtsplanung ein. Darauf aufbauend skizziere ich dann ein Vorgehensmodell zur

[1] Unter *didaktischer Software* verstehe ich Software, die absichtsvoll für die Verwendung in Lern-Lehr-Situationen gestaltet ist.

[2] Unter Unterricht verstehe ich organisierte Lern-Lehr-Situationen, also Schulunterricht, Lehrveranstaltungen an Hochschulen, Weiterbildungsangebote usw.

koevolutionären Entwicklung von didaktischer Software und Unterrichtskonzepten. Ich schließe mit einem Fazit.

2 Usability Engineering

In der Literatur sind eine ganze Reihe von Vorgehensmodellen zum Usability Engineering beschrieben worden. Neben älteren Arbeiten von Norman und Draper (ND86) sowie Nielsen (Ni93) haben vor allem das Contextual Design (BH98), der Usability Engineering Lifecycle (Ma99), das Scenario-based Design (RC02) und der Design-Use-Cycle (Dz97) breite Beachtung gefunden. Mit der Norm ISO 13407 („Human-centered design processes for interactive systems“) (ISO99) liegt auch ein internationaler Standard vor. Die genannten Modelle unterscheiden sich in erster Linie in der Auswahl von Einzelmethoden, die im Entwicklungsprozess verwendet werden. Im prinzipiellen Vorgehen stimmen sie jedoch weitgehend überein: Sie beziehen den Anwendungskontext in die Entwicklung ein, beteiligen die BenutzerInnen als ExpertenInnen für ihre Arbeit und organisieren den Entwicklungsprozess in mehreren Zyklen.

Software im Kontext *Usability (Gebrauchstauglichkeit)* ist keine unabhängige Produkteigenschaft, sondern hängt von Software und Anwendungskontext gleichermaßen ab: Eine Software kann immer nur für bestimmte Menschen (mit ihren individuellen perzeptiven und motorischen, kognitiven, emotionalen und sozialen Fähigkeiten und Zielen), die in einem bestimmten physischen und sozialen Umfeld bestimmte Aufgaben bearbeiten, gebrauchstauglich sein (DIN99; Ob01). Die Anwendungskontexte sind dabei nicht statisch, sondern unterliegen einem ständigen Wandel. Im Usability Engineering wird Softwareentwicklung daher nicht allein als die Anpassung der Software an den bestehenden Kontext gesehen, sondern auch als Anpassung des Kontextes an die entstehende Software. Usability Engineering ist also immer auch Arbeitsgestaltung (Ha94) bzw. Organisationsentwicklung (WR95). Für didaktische Software wäre Usability Engineering entsprechend auch Unterrichtsplanung. Rogers (Ro94) nennt diesen Prozess der wechselseitigen Anpassung *Koevolution* von Software und Anwendungskontexten und hebt noch einen weiteren Aspekt hervor: Software wird nicht nur im Gestaltungsprozess hergestellt, sondern auch in der Benutzung sozial konstruiert. Die BenutzerInnen bestimmen dadurch, dass sie sich die Software auf eine bestimmte Art und Weise nutzbar machen, maßgeblich deren Gestalt mit (Or92).

BenutzerInnenbeteiligung Die Einbeziehung der (zukünftigen) BenutzerInnen in den Entwicklungsprozess ist ein Schlüsselelement im Usability Engineering (RSS$^+$94; SN93). Als *ExpertInnen des Anwendungskontextes* verfügen die BenutzerInnen über das Wissen über ihre tagtäglichen Arbeitsabläufe und deren Zusammenhang mit anderen Tätigkeiten in ihrem Umfeld. Die tatsächliche Praxis weicht oftmals erheblich von formalen Arbeitsplatzbeschreibungen und Organigrammen, den Vorstellungen des Managements und erst recht dem naiven Verständnis der SoftwareentwicklerInnen ab, so dass nur durch Benut-

zerInnenbeteiligung die Sicherheit erlangt werden kann, dass die Aufgaben der BenutzerInnen auf angemessene Weise unterstützt werden.

Bei der Form kann man prinzipiell zwischen der Beteiligung der BenutzerInnen als *gleichberechtigte PartnerInnen* im Gestaltungsprozess („Cooperative Design“) (BEK89; GK91) und einer *evaluativen Beteiligung* unterscheiden, bei der die BenutzerInnen zwar für die Anforderungsermittlung und die Evaluation beobachtet oder befragt werden, aber nicht (systematisch) an Gestaltungsentscheidungen beteiligt sind. In der Regel ist es in Softwareentwicklungsprojekten nicht möglich, *alle* BenutzerInnen *direkt* im Entwicklungsprozess zu beteiligen, weil die BenutzerInnengruppe dafür zu groß ist. Eine Beteiligung ist in diesen Fällen nur *indirekt* über *BenutzervertreterInnen* möglich. Da dabei immer die Gefahr besteht, dass Anforderungen selektiv und verzerrt in die Entwicklung einfließen, sollten zumindest auf einem niedrigen Niveau, z. B. durch eine evaluative Beteiligung, weitere BenutzerInnen einbezogen werden.

Zyklisches Vorgehen Anwendungskontexte unterliegen einem ständigen Wandel: Durch den Gestaltungsprozess selbst, durch die Einführung einer neuen Software und die damit verbundenen neuen Arbeitspraktiken, durch die Aneignung der Software durch die BenutzerInnen, aber auch allein dadurch, dass Menschen sich beständig weiter entwickeln (lernen!). Deshalb und weil viele Anforderungen an eine Software sich erst im Umgang damit erkennen lassen, können die Anforderungen niemals vollständig spezifiziert werden. Aus diesem Grund sind lineare Vorgehensmodelle, die davon ausgehen, dass Anforderungen an eine Software einmal erhoben und dann systematisch umgesetzt werden, für die Entwicklung gebrauchstauglicher Software ungeeignet (Fl92).

Im Usability Engineering wird ein *zyklisches Vorgehen* favorisiert, das es ermöglicht, Gestaltungsentscheidungen im Anwendungskontext zu überprüfen und die Ergebnisse einer formativen Evaluation wieder in den Entwicklungsprozess einfließen zu lassen. Softwareentwicklung ist so verstanden ein Lernprozess, bei dem die SoftwareentwicklerInnen und die BenutzerInnen gemeinsam mit jedem neuen Zyklus ein tieferes Verständnis der Software und des Anwendungskontextes gewinnen und in eine neue Version der Software und in Veränderungen des Kontextes umsetzen.

3 Besonderheiten bei der Entwicklung didaktischer Software

Die vorstehend genannten Usability Engineering-Vorgehensmodelle können für die Entwicklung didaktischer Software nicht unverändert übernommen werden, denn sie sind ursprünglich in der Entwicklung von Software entstanden, die an Erwerbsarbeitsplätzen in Unternehmen zum Einsatz kommt. Unterricht (als didaktischer Anwendungskontext) unterscheidet sich aber in drei wesentlichen Aspekten von diesen betrieblichen Anwendungskontexten.

3.1 Die Rolle von Medien im Unterricht

Unter Unterrichtsmedien verstehe ich mit Schulz (Sc81) allgemein „gegenständliche Mittler“ bzw. „Verständigungenmittel“ im Lern-Lehr-Prozess. Inwiefern Medien eigenständige Strukturmerkmale von Unterricht sind, ist dabei in der Literatur umstritten (Me94). Aus pragmatischen Gründen kommt der Frage nach den Medien bei der Unterrichtsplanung aber in jedem Fall eine große Bedeutung zu, denn die Beziehung der Medien zu den anderen Handlungsmomenten muss in der Unterrichtsplanung berücksichtigt werden (Sc81, 124f.): Medien können das *Thema* des Unterrichts auf verschiedene Weise repräsentieren, etwa als Exemplar, als Abbildung oder als Gestaltungsmittel. Sie können die *Ziele* unterschiedlich fördern: es gibt monovalente Medien, die nur eine pädagogische Intention unterstützen, und polyvalente Medien, die verschiedene Intentionen unterstützen können. Jedes Medium setzt auch eine bestimmte *Ausgangslage* bei Lernenden und Lehrenden voraus, etwa Kenntnisse im Umgang mit Computern und Internet, und Medien sind für unterschiedliche *Methoden* geeignet, etwa für Einzelunterricht oder Gruppenarbeit. Nicht zuletzt hängt der Einsatz von Medien auch immer von *institutionellen Rahmenbedingungen* ab. Nicht alle wünschenswerten Medien sind immer verfügbar und umgekehrt ist die Verwendung bestimmter Medien manchmal vorgeschrieben.

Fazit: Medien enthalten immer gewisse Vorfestlegungen hinsichtlich der anderen Aspekte von Unterricht und sind nur dann geeignet, wenn diese Vorfestlegungen zum Unterricht passen: „Unterrichtsmedien sind »tiefgefrorene« Ziel-, Inhalts-, und Methodenentscheidungen. Sie müssen im Unterricht durch das methodische Handeln [...] wieder »aufgetaut« werden“ (Me94, 150).

3.2 Die besondere Rolle der Lehrenden

Lehrende haben als Mitglieder von Lern-Lehr-Gruppen eine besondere Rolle inne, denn einerseits haben sie durch ihren amtlichem Auftrag die Gesamtverantwortung für den Unterricht, andererseits verfügen sie in der Regel auch über einen Entwicklungs- und Informationsvorsprung gegenüber den Lernenden. Sie haben damit sowohl die *Legitimation*, den Nutzungskontext zu gestalten, als auch die *Position*, entscheidend auf den Unterrichtsverlauf einzuwirken. Das macht die Lehrenden zu idealen Mitgliedern des Entwicklungsteams im Sinne einer kooperativen BenutzerInnenbeteiligung, denn sie bringen (hoffentlich) nicht nur die benötigte didaktische Kompetenz mit, sondern sie können auf Grund ihrer besonderen Rolle auch die Verwendung der entstehenden Software im Unterricht verfügen und den Anwendungskontext nach den Vorstellungen des Entwicklungsteams gestalten. Realistisch betrachtet ist das Usability Engineering didaktischer Software ohne Lehrende nicht möglich.

Das darf aber nicht darüber hinweg täuschen, dass die Zielgruppe sowohl der Softwareentwicklung wie der Unterrrichtsplanung nicht die Lehrenden, sondern die Lernenden sind. Die Lernenden sind allerdings als Partizipationspartner in einer sehr schlechten Position, weil sie in der Praxis typischerweise nicht einmal gleichberechtigt an der Unterrichts-

planung beteiligt sind. Für persönlich bedeutsame Lernprozesse ist die Beteiligung der Lernenden an der Unterrichtsplanung aber zwingend erforderlich (Sc81; CF84).

Fazit: In die Entwicklung didaktischer Software müssen *sowohl Lehrende als auch Lernende* angemessen einbezogen werden. Das setzt eine Beteiligung der Lernenden an der Unterrichtsplanung voraus.

3.3 Die zeitliche Begrenzheit und hohe Dynamik von Unterricht

Einem zyklischen Vorgehen liegt implizit die Annahme zu Grunde, dass der Anwendungskontext, mit dem eine Software entwickelt wird, zwar einerseits einem ständigen Wandel unterliegt, andererseits aber auch über einen längeren Zeitraum hinweg relativ stabil bleibt (Ma87). Nur unter dieser Prämisse ist es überhaupt sinnvoll, Software für diesen Anwendungskontext zu entwickeln, denn würde er sich permanent radikal verändern, gäbe es keine Strukturen oder Prozesse, die man objektivieren könnte.

Unterricht ist anders, denn nur für vergleichsweise kurze Zeit, etwa ein Semester oder ein Schuljahr kommen Menschen zusammen, die sich gemeinsam mit einem Thema beschäftigen. Und selbst in gewohnten Unterrichtsformen werden Thema und Formen der Zusammenarbeit verhandelt, bevor sich allmählich gemeinsame Arbeitsweisen etablieren (Tu65). Es ist daher fast schon trivial festzustellen, dass selbst bei sehr kurzen Entwicklungszyklen bestenfalls kleine Anpassungen einer Software vorgenommen werden können. Eine Entwicklung im Sinne einer wechselseitigen Anpassung von Software und Anwendungskontext ist also scheinbar nicht möglich.

Betrachtet man das Unterrichtsgeschehen aus der Perspektive der Bildungseinrichtung, dann hat es allerdings sehr wohl langfristige Aspekte. Lehrende bieten oft Unterricht mit ähnlichen Themen immer wieder an und sie entwickeln Konzepte, mit deren Hilfe sie ihren Unterricht planen. Genauso sammeln auch die Lernenden Erfahrungen und entwickeln Strategien, wie sie am besten ihre Ziele erreichen können. So etablieren sich Strukturen und Prozesse, die für das Usability Engineering didaktischer Software genutzt werden können. Der erste Eindruck, dass z. B. eine Vorlesung *Praktische Informatik I* jedes Jahr wieder angeboten wird, sollte aber nicht glauben machen, dies sei immer wieder der gleiche Unterricht, denn schließlich sind in jedem Jahr andere Menschen beteiligt.

Fazit: Die Entwicklung didaktischer Software kann nicht mit einem einzigen Anwendungskontext (Unterricht) erfolgen, sondern muss sich über mehrere Unterrichte erstrecken, die gewisse Struktur- und Prozessähnlichkeiten aufweisen.

4 Die koevolutionäre Entwicklung von Software und Unterrichtskonzepten

Aufbauend auf diesen Vorüberlegungen werde ich nun ein zyklisches und partizipatives Usability Engineering-Vorgehensmodell für die Entwicklung didaktischer Software skiz-

zieren, das die beschriebenen Besonderheiten des Anwendungskontextes „Unterricht“ berücksichtigt. Das Modell ist eine Anpassung des Design-Use-Cycle (Dz97), der wiederum auf dem STEPS-Projektmodell aus der Softwaretechnik (FRS89) basiert. Einen Überblick bietet die Abbildung 1, auf die ich später genauer eingehe.

4.1 Das Unterrichtskonzept als abstrakter Nutzungskontext

Medien liegen immer Entscheidungen ihrer Gestalter hinsichtlich Zielen, Inhalten und Methoden des adressierten Unterrichts zu Grunde. In den Medien werden diese Entscheidungen „tiefgefroren“ und im unterrichtlichen Handeln wieder „aufgetaut“. Darin liegt begründet, ob sie zu dem mit ihnen durchgeführten Unterricht passen oder nicht. Die *explizite Darstellung* von Ziel-, Inhalts- und Methodenentscheidungen, die einer didaktischen Software zu Grunde liegen, soll in dem hier beschriebenen Vorgehen als *Unterrichtskonzept* erfolgen. Unterrichtskonzepte sind „*Gesamtorientierungen didaktisch-methodischen Handelns*, in denen ein begründeter Zusammenhang von Ziel-, Inhalts- und Methodenentscheidungen hergestellt wird. In ihnen werden explizit ausgewiesene oder implizit als gültig unterstellte Unterrichtsprinzipien, Annahmen über die organisatorisch-institutionellen Rahmenbedingungen des Unterrichts sowie bestimmte Erwartungen an das Verhalten [von Lehrenden und Lernenden] miteinander verknüpft“ (JM94, 290). Unterrichtskonzepte sind absichtsvoll normativ, d. h. sie beschreiben, wie sich ihre ErfinderInnen guten Unterricht unter alltäglichen Bedingungen vorstellen.

Das Unterrichtskonzept hat dabei für das Entwicklungsteam die Funktion eines „abstrakten Anwendungskontextes“. Da die Entwicklung didaktischer Software nicht mit einem (einzigen) konkreten Kontext erfolgen kann, ist ein anderer Orientierungsrahmen notwendig. Durch Unterrichtskonzepte können die aus Sicht des Entwicklungsteams relevanten Strukturmerkmale von Unterricht mit der didaktischen Software benannt und damit sowohl innerhalb des Entwicklungsteams als auch nach außen kommunizierbar gemacht werden.

Einer Lern-Lehr-Gruppe wird so die Möglichkeit gegeben, die grundlegenden Ziel-, Inhalts- und Methodenfestlegungen einer didaktischen Software mit dem von ihr geplanten Unterricht zu vergleichen. Sie kann dann informiert entscheiden, ob sie die Software als Unterrichtsmedium verwenden will oder nicht. Das Unterrichtskonzept determiniert dabei in keiner Weise den Unterricht, auch wenn es natürlich wünschenswert ist, wenn sich die Lern-Lehr-Gruppe von dem Konzept inspirieren lässt – insbesondere wenn sie noch keine Erfahrung mit der Software hat.

Eine weitere Verwendung findet das Unterrichtskonzept in der Evaluation der Softwarenutzung als Referenzrahmen. Da kein Unterricht eine idealtypische Umsetzung des Konzeptes sein kann, ist in der Evaluation nicht nur zu fragen, inwieweit Software und Unterricht zueinander passen, sondern auch, inwieweit der Unterricht dem der Softwareentwicklung zu Grunde liegenden Unterrichtskonzept entspricht. Werden Änderungsbedarfe festgestellt, dann muss geprüft werden, ob sie ggf. auf eine Abweichung vom Unterrichtskonzept zurückzuführen sind.

4.2 Das Entwicklungsteam und die Lern-Lehr-Gruppen

Da sich der Usability Engineering-Prozess über mehrere Unterrichte erstreckt und eine Anpassung an einen einzigen davon nicht möglich ist, ist die kooperative Beteiligung von BenutzerInnen für diese nicht sonderlich attraktiv und würde darüber hinaus auch zu einer starken Fluktuation im Entwicklungsteam führen. Dennoch ist es unverzichtbar, Lernende und Lehrende als Mitglieder im Entwicklungsteam zu haben, allerdings nicht als RepräsentantInnen einer bestimmten Lern-Lehr-Gruppe, sondern als PraktikerInnen, die ihre Expertise unabhängig von einem bestimmten Unterricht einbringen und dafür auch angemessen entlohnt werden. Damit ist verbunden, dass das Entwicklungsteam nicht die Legitimation und Möglichkeit hat, Unterricht zu gestalten. Diese Aufgabe verbleibt einzelnen *Lern-Lehr-Gruppen* im Rahmen der Unterrichtsplanung.

Der Unterschied zur kooperativen BenutzerInnenbeteiligung besteht also darin, dass die BenutzervertreterInnen nicht aus dem Nutzungskontext kommen und ggf. demokratisch legitimiert sind, sondern sich aktiv Nutzungskontexte suchen oder neu schaffen, indem sie entweder als Lernende an geeignet erscheinendem Unterricht teilnehmen oder als Lehrende Unterricht im Sinne des Entwicklungsteams anbieten. Eine mit diesem Ansatz verbundene Gefahr ist allerdings die Instrumentalisierung von Unterricht zu Zwecken der Evaluation der Software insbesondere durch Lehrende. Dieser Gefahr kann nur durch hohe Professionalität und Reflexion des eigenen Tuns begegnet werden.

4.3 Der Entwicklungsprozess

Der Entwicklungsprozess didaktischer Software ist zyklisch angelegt. Wie in Abbildung 1 zu sehen ist, werden in jedem Zyklus zwei Phasen durchlaufen: Gestaltung und Nutzung, in denen jeweils die dargestellten Aktivitäten stattfinden. An der Entwicklung sind ein Entwicklungsteam und mehrere Lern-Lehr-Gruppen (grau unterlegt) beteiligt.

4.3.1 Gestaltungsphase

Im ersten Schritt entwirft das Entwicklungsteam eine Vision der Software und eines realistischen, mit Softwareunterstützung stattfindenden, Unterrichts. Dabei dient ein Unterrichtskonzept als abstrakter Anwendungskontext, der vom Entwicklungsteam gestaltet wird. Es ist zweckmäßig, etablierte Konzepte oder Veranstaltungsformen als Ausgangspunkt zu wählen. Liegen bereits Evaluationsergebnisse aus vorangehenden Nutzungsphasen vor, dann werden diese natürlich in den Entwurf einbezogen. Die Vision umfasst dann einerseits eine Spezifikation der Software als auch ein grobes Unterrichtskonzept.

Auf der Grundlage dieser Vorarbeiten werden dann im nächsten Schritt Software und Unterrichtskonzept ausgearbeitet. Das heißt, die Software wird nach allen Regeln der Kunst implementiert und das Unterrichtskonzept wird so verallgemeinert und dokumentiert, dass für Außenstehende verständlich wird, welche Annahmen des Entwicklungsteams über Ziele, Inhalte und Methoden des Unterrichts der Software zu Grunde liegen. Die so entste-

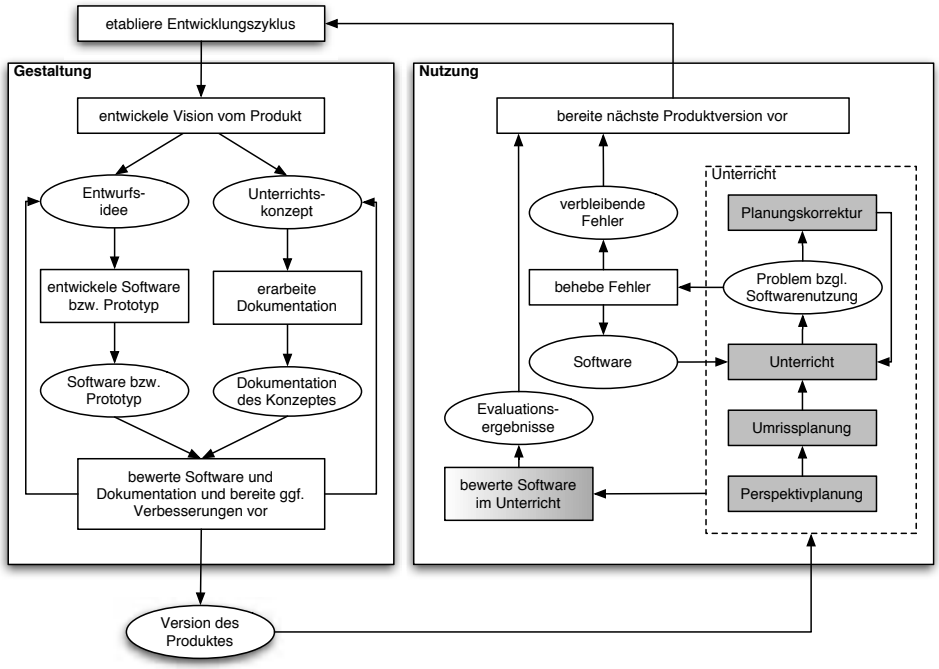

Abbildung 1: Koevolutionäre Entwicklung von Software und Unterrichtskonzepten

hende Software und die Dokumentation des Unterrichtskonzeptes werden vom Entwicklungsteam noch einmal außerhalb eines Nutzungskontextes bewertet und auf Stimmigkeit geprüft. Fallen an dieser Stelle gravierende Probleme auf, dann wird ggf. der Entwurf überarbeitet. Andernfalls bilden die Software und das Unterrichtskonzept zusammen eine neue Produktversion, die von mehreren Lern-Lehr-Gruppen genutzt werden kann.

4.3.2 Nutzungsphase

In der Nutzungsphase planen die Lern-Lehr-Gruppen ihren Unterricht und führen ihn durch. Das Entwicklungsteam evaluiert in Zusammenarbeit mit ausgewählten Lern-Lehr-Gruppen die Nutzung des Produktes.

Unterrichtsplanung Da Unterrichtsplanung bereits ein Teil des Unterrichts ist (Sc81), ist sie – anders als vielleicht auf den ersten Blick nahe liegt – als Teil der Nutzungs- und nicht der Gestaltungsphase anzusehen. Das ist auch deshalb plausibel, weil eine Version des Produktes (Software und Dokumentation des Unterrichtskonzeptes) bereits früh in der Planung verwendet werden, wenn die Auswahl von bestimmten Medien für den Unterricht von der Lern-Lehr-Gruppe diskutiert wird. Selbst dann, wenn sich eine Lern-Lehr-Gruppe *gegen* die Verwendung der Software als Unterrichtsmedium ausspricht, können also wertvolle Nutzungserfahrungen gesammelt werden.

Wird die Software als Unterrichtsmedium verwendet, dann werden sehr wahrscheinlich in der Durchführung des Unterrichts verschiedenartige Probleme hinsichtlich der Softwarenutzung auftreten. Diese lassen sich in technische Fehler der Software (etwas funktioniert nicht so, wie es soll) und Probleme in der Abstimmung mit dem Unterricht unterscheiden. Technische Fehler können oft kurzfristig vom Entwicklungsteam behoben werden, Abstimmungsproblemen muss auf jeden Fall durch eine Korrektur der Unterrichtsplanung begegnet werden, denn eine Änderung der Software ist im Rahmen eines Unterrichts nicht vorgesehen.

Die Abweichung von Plänen ist immer notwendig, wenn Menschen situiert handeln (Su87) und daher nicht negativ, sondern als Anregung zu Lern- und Veränderungsprozessen positiv zu bewerten. Die Abstimmungsprobleme können in solche unterschieden werden, bei denen im Rahmen der Planungskorrektur eine wenigstens ebenso gebrauchstaugliche Handlungsalternative in Bezug auf die Ziele der Lern-Lehr-Gruppe gefunden wird, und solche, bei denen das nicht der Fall ist. Insbesondere die letzteren Fälle geben Hinweise auf Verbesserungsmöglichkeiten der Software, in beiden Fälle können Verbesserungsmöglichkeiten für das Unterrichtskonzept erkennbar werden.

Evaluation Das Entwicklungsteam befasst sich in der Nutzungsphase mit der Evaluation der Nutzung. Es ist dabei auf die Zusammenarbeit mit Lern-Lehr-Gruppen angewiesen. Bei der Planung und Durchführung der Evaluation und besonders bei der Bewertung der Evaluationsergebnisse muss berücksichtigt werden, inwieweit die reale Unterrichtsplanung mit dem handlungsorientierenden Unterrichtskonzept übereinstimmen. Die folgenden Fälle sind denkbar:

- Der Unterricht wurde in Anlehnung an das Unterrichtskonzept geplant: Die Ergebnisse können direkt für die Weiterentwicklung von Software und Unterrichtskonzept verwendet werden. Das gilt auch dann, wenn im Verlauf des Unterrichts von der ursprünglichen Planung drastisch abgewichen wird.
- Der Unterricht war anders geplant als das Unterrichtskonzept: Die Evaluationsergebnisse können verwendet werden, um die Rahmenbedingungen des erfolgreichen Einsatzes der Software genauer zu bestimmen und das Unterrichtskonzept entsprechend zu überarbeiten. In Bezug auf die Software können eher Änderungen in der Handhabung als der grundsätzlichen Konzeption abgeleitet werden. Gibt es nur wenige oder überhaupt keine Lern-Lehr-Gruppen, die das handlungsorientierende Unterrichtskonzept umsetzen wollen, dann stellt sich die Frage nach seiner prinzipiellen Praxistauglichkeit.

Die Evaluationsergebnisse und eventuell nicht behobene technische Fehler bilden die Grundlage für die nächste Gestaltungsphase.

5 Fazit und Ausblick

In diesem Beitrag habe ich ein Vorgehensmodell für das Usability Engineering didaktischer Software vorgestellt, dass auf der Idee der Koevolution von Software und Unter-

richtskonzepten beruht. Das beschriebene Vorgehensmodell ist nicht am grünen Tisch entstanden, sondern hat sich insbesondere in der Entwicklung der webbasierten Kooperationsplattform CommSy (JJS04) als geeignet erwiesen. Auf Details des konkreten Entwicklungsprozesses kann ich hier aus Platzgründen allerdings nicht eingehen (vgl. aber JaIV). Ein Beispiel für die Verknüpfung von projektorientierter Lehre mit CommSy haben wir bereits veröffentlicht (JJP02; JJK$^+$03). Es wäre interessant, das Vorgehen auch anderer erfolgreicher E-Learning-Projekte genauer zu untersuchen.

Der nächste Schritt meiner Arbeit ist ein detaillierter Vergleich mit anderen Vorgehensmodellen, z. B. der im deutschsprachigen Raum verbreiteten gestaltungsorientierten Mediendidaktik von Kerres (Ke01) oder dem Instruktionsdesign (Ni01). LeserInnen, die mit diesen Ansätzen vertraut sind, werden einige Unterschiede und Gemeinsamkeiten bereits erkannt haben, etwa das hier favorisierte zyklische Vorgehen und die Beteiligung von Lernenden am Entwicklungsprozess.

Einen Schwerpunkt in meinem Beitrag habe ich auf die Besonderheiten von Unterricht als Anwendungskontext von Software gelegt und die Unterschiede herausgearbeitet, die zu der im Usability Engineering bislang vorwiegend betrachteten Entwicklung von Software zur langfristigen Unterstützung von Erwerbsarbeit bestehen. Dabei fällt auf, dass ähnliche Unterschiede auch zu projektorientierten Formen der Arbeitsorganisationen in Unternehmensnetzwerken bestehen – möglicherweise ließe sich das von mir vorgeschlagene Vorgehensmodell auf derartige Anwendungskontexte übertragen.

Danksagung

Ich danke allen KollegInnen aus dem WissPro-Projekt und dem CommSy-Team, ohne deren Arbeit es keine Grundlage für die hier vorgestellten Ideen gäbe, und insbesondere Monique Janneck, Bernd Pape und Matthias Finck für konstruktive Kritik an einer früheren Version dieses Beitrags.

Literatur

[BEK89] Bjerknes, G., Ehn, P., und Kyng, M. (Hrsg.): *Computers and Democracy : A Scandinavian Challenge*. Avebury. Aldershot u. a. 1989.

[BH98] Beyer, H. und Holtzblatt, K.: *Contextual Design : Defining Customer-Centered Systems*. Morgan Kaufmann. San Francisco u. a. 1998.

[CF84] Cohn, R. C. und Farau, A.: *Gelebte Geschichte der Psychotherapie: Zwei Perspektiven*. Klett-Cotta. Stuttgart. 1984.

[DIN99] Deutsches Institut für Normung. *DIN EN ISO 9241: Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten, Teil 11: Anforderungen an die Gebrauchstauglichkeit: Leitsätze*. Berlin. 1999.

[Dz97] Dzida, W.: International user-interface standardization. In: Tucker Jr., A. B. (Hrsg.), *The Computer Science and Engineering Handbook*. S. 1474–1493. CRC Press. 1997.

[Fl92] Floyd, C.: Software development as reality construction. In: Floyd, C., Züllighoven, H., Budde, R., und Keil-Slawik, R. (Hrsg.), *Software Development and Reality Construction*. S. 86–100. Springer. Berlin u. a. 1992.

[FRS89] Floyd, C., Reisin, F.-M., und Schmidt, G.: Steps to software development with users. In: *Proceedings of ESEC 1989*. S. 48–64. Springer. Berlin u. a. 1989.

[GK91] Greenbaum, J. und Kyng, M. (Hrsg.): *Design at Work : Cooperative Design of Computer Systems*. Erlbaum. Hillsdale u. a. 1991.

[Ha94] Hacker, W.: Arbeits- und organisationspsychologische Grundlagen der Software-Ergonomie. In: Eberleh, E., Oberquelle, H., und Oppermann, R. (Hrsg.), *Einführung in die Software-Ergonomie*. S. 53–93. de Gruyter. Berlin. 2. Auflage. 1994.

[ISO99] International Organization for Standardization. *ISO 13407: Human-centered Design Processes for Interactive Systems*. Genf. 1999.

[JaIV] Janneck, M.: *Softwareunterstützung für Lernprojekte : Zur Gestaltung gebrauchstauglicher didaktischer Software (Arbeitstitel)*. Dissertation. Universität Hamburg, Fachbereich Informatik. i.V.

[JJK+03] Jackewitz, I., Janneck, M., Krause, D., Pape, B., und Strauss, M.: Teaching social informatics as a knowledge project. In: van Weert, T. J. und Munro, R. K. (Hrsg.), *Informatics and the Digital Society*. S. 261–268. Kluwer Academic. Boston u. a. 2003.

[JJP02] Jackewitz, I., Janneck, M., und Pape, B.: Vernetzte Projektarbeit mit CommSy. In: Herzceg, M., Prinz, W., und Oberquelle, H. (Hrsg.), *Mensch & Computer 2002 : Vom interaktiven Werkzeug zu kooperativen Arbeits- und Lernwelten*. S. 35–44. Teubner. Stuttgart u. a. 2002.

[JJS04] Jackewitz, I., Janneck, M., und Strauss, M.: Commsy: Softwareunterstützung für wissensprojekte. In: Pape, B., Krause, D., und Oberquelle, H. (Hrsg.), *Wissensprojekte – Gemeinschaftliches Lernen aus didaktischer, softwaretechnischer und organisatorischer Sicht*. S. 186–202. Waxmann. Münster u. a. 2004.

[JM94] Jank, W. und Meyer, H.: *Didaktische Modelle*. Cornelsen Scriptor. Berlin. 3. Auflage. 1994.

[Ke01] Kerres, M.: *Multimediale und telemediale Lernumgebungen : Konzeption und Entwicklung*. Oldenbourg. München u. a. 2. Auflage. 2001.

[Ma87] Mathiassen, L.: Systems, processes, and structures. In: Docherty, P., Fuch-Kittowsky, K., Kolm, P., und Mathiassen, L. (Hrsg.), *System Design for Human Development and Productivity: Participation and Beyond*. S. 49–61. Elsevier Science. 1987.

[Ma99] Mayhew, D. J.: *The Usability Engineering Lifecycle : A Practitioner's Handbook for User Interface Design*. Morgan Kaufmann. San Francisco u. a. 1999.

[Me94] Meyer, H.: *Unterrichtsmethoden : 1. Theorieband*. Cornelsen Scriptor. Frankfurt am Main. 6. Auflage. 1994.

[ND86] Norman, D. A. und Draper, S. W. (Hrsg.): *User Centered System Design : New Perspectives on Human-Computer Interaction*. Erlbaum. Hillsdale, N. J. u. a. 1986.

[Ni93] Nielsen, J.: *Usability Engineering*. Academic Press. Boston u. a. 1993.

[Ni01] Niegemann, H. M.: *Neue Lernmedien : konzipieren, entwickeln, einsetzen*. Huber. Bern u. a. 2001.

[Ob01] Oberquelle, H.: Softwareergonomie. In: Schwabe, G., Streitz, N., und Unland, R. (Hrsg.), *CSCW Kompendium : Lehr- und Handbuch zum computerunterstützten kooperativen Arbeiten*. S. 87–97. Springer. Berlin u. a. 2001.

[Or92] Orlikowski, W. J.: The duality of technology: Rethinking the concept of technology in organizations. *Organization Science*. 3(3):398–424. 1992.

[RC02] Rosson, M. B. und Carroll, J. M.: *Usability Engineering : Scenario-Based Development of Human-Computer Interaction*. Morgan Kaufmann. San Francisco u. a. 2002.

[Ro94] Rogers, Y.: Exploring obstacles: Integrating CSCW in evolving organisations. In: *Proceedings of the CSCW 1994 Conference*. S. 67–77. ACM. 1994.

[RSS+94] Rauterberg, M., Spinas, P., Strohm, O., Ulich, E., und Waeber, D.: *Benutzerorientierte Software-Entwicklung*. vdf. Zürich. 1994.

[Sc81] Schulz, W.: *Unterrichtsplanung : Mit Materialien aus Unterrichtsfächern*. Urban und Schwarzenberg. München u. a. 3. Auflage. 1981.

[SN93] Schuler, D. und Namioka, A. (Hrsg.): *Participatory Design : Principles and Practice*. Erlbaum. Hillsdale u. a. 1993.

[Su87] Suchman, L.: *Plans and Situated Actions : The Problem of Human-Machine Communication*. Cambridge University Press. Cambridge. 1987.

[Tu65] Tuckman, B.: Developmental sequence in small groups. *Psychological Bulletin*. 63(6):384–399. 1965.

[WR95] Wulf, V. und Rohde, M.: Toward an integrated organization and technology development. In: *Proceedings of the Symposium on Designing Interactive Systems*. S. 55–64. ACM-Press. New York. 1995.