C. Stary (Hrsg.): Mensch & Computer 2005: Kunst und Wissenschaft – Grenzüberschreitungen der interaktiven ART. München: Oldenbourg Verlag. 2005, S.81-90

Co-adaptive Entwicklungsumgebungen als Grundlage transdisziplinärer Software-Projekte

Alexander Karosseit, Frank Fuchs-Kittowski

Fraunhofer-Institut für Software- und Systemtechnik Berlin

Zusammenfassung

Hochgradig interaktive und reaktive Software-Systeme zur Repräsentation und Vermittlung komplexer fachlicher Zusammenhänge gewinnen auf vielen Anwendungsgebieten zunehmend an Bedeutung. Während der Entwicklung derartiger Software, sollten in der Regel die Fachlichkeit und die mediale Qualität einen hohen Stellenwert haben. Oft überlagern jedoch technologische Aspekte der Software-Entwicklung die inhaltliche und methodische Ausgestaltung. Als entscheidender Grund dafür werden Verständnisschwierigkeiten an den entsprechenden Prozessschnittstellen ausgemacht. Klassische Techniken der Erhebung und Verwaltung von Anforderungen erweisen sich als kaum praktikabel. Die in diesem Beitrag vorgestellte Lösung des Problems bricht bestehende Barrieren auf, indem Methoden und Werkzeuge der technischen Teile der Entwicklung systematisch zur Arbeitsgrundlage auch der fachlich-inhaltlichen Entwicklung werden. Die entstandene adaptierbare Entwicklungsumgebung wird dazu individuell an den jeweiligen Projektkontext angepasst. Die dann noch bestehende Kluft zwischen den Konzepten der Anwendungswelt und denen der technischen Entwicklung wird über die gezielte Motivation und Anleitung der Domänen-Fachleute geschlossen. Die Funktionstüchtigkeit dieses so etablierten co-adaptiven Prozesses zeigt die Arbeit im konkreten Projekt „LeMOLernen“.

1 Einleitung

Hochgradig interaktive und reaktive Software-Systeme spielen für viele Anwendungsbereiche eine zentrale Rolle bei der Repräsentation und der Vermittlung komplexer fachlicher Sachverhalte. Die Entwicklung solcher Systeme stellt eine besondere Herausforderung transdisziplinären Arbeitens dar. Etwa bei der Erstellung hochwertiger, multimedialer Anwendungen zur Unterstützung des Lernens und Lehrens ist in der Regel eine Vielzahl verschie-

dener Disziplinen direkt in den Entstehungskontext involviert. Oft handelt es sich bei den Beiträgen sowohl der Techniker, als auch der verschiedensten Fachexperten, Autoren und Redakteure um Leistungen mit wissenschaftlichem Anspruch. Jeweils aus einem anderen Blickwinkel heraus und auf der Grundlage jeweils verschiedener Vorgehens- und Sichtweisen, werden fachdidaktische und mediendidaktische Konzepte entwickelt, Inhalte verfasst, zusammengetragen und strukturiert. Schließlich wird eine Software-Infrastruktur implementiert, die diese Konzepte materialisiert: Funktionalitäten der Navigation, der Interaktion und der Präsentation werden entwickelt. Die Redaktion und der Verlag bzw. das Marketing runden die Ergebnisse schließlich zum Produkt ab und bilden damit auch die zentrale Schnittstelle zum Anwendungskontext. Dieser ist mit Lehrern, Lernenden und Administratoren multidisziplinär strukturiert und wirkt über Feedback, systematische Evaluation oder auch durch direkte Manipulation der Nutzer am System auf den ebenfalls multidisziplinären Entstehungskontext zurück.

Diese als Transdisziplinarität (Jahn 2005) zu bezeichnende Charakteristik ist bei den hier adressierten Software-Entwicklungsprojekten besonders stark ausgeprägt. Die vielfältigen Schnittstellen innerhalb solcher Entwicklungsprozesse, die meist geprägt sind von den technologiefernen Beteiligten, führen zu hohen Reibungsverlusten. Verschiedene Herangehensweisen, z. T. unterschiedliche Fachsprachen, Modellbildungen etc. führen dabei mehr als bei der klassischen Entwicklung, etwa betrieblicher Informationssysteme, zu nicht reproduzierbaren, kaum zu steuernden und hoch iterativen Vorgehensweisen. Klassische Vorgehensmodelle, die die Software-Entwicklung beherrschbar machen sollten, lassen sich praktisch nicht anwenden. Für die in solchen Prozessen nötigen Abstraktionen, Modellsprachen, aber auch Entwicklungsumgebungen, fehlt einem Großteil der Projektbeteiligten das Verständnis. Häufiges Scheitern solcher Projekte ist schwerpunktmäßig auf diese Tatsache zurückzuführen.

Es muss gelingen, systematische, nachvollziehbare Entwicklungsprozesse zu etablieren, kommunikative Reibungsverluste durch eine nahtlose Integration aller Beteiligten zu vermeiden und damit auch alle personellen Ressourcen möglichst effektiv in den gesamten Prozess einbinden zu können. Der dazu verfolgte Ansatz soll die Kluft zwischen den Vertretern der Fachdomäne, den Endanwendern und der technischen Umsetzung schließen, indem Vorgehensweisen und Entwicklungswerkzeuge so adaptiert werden, dass sie dem Verständnis und den individuellen Bedürfnissen aller Beteiligten möglichst weit entgegenkommen. Das heißt, die Begriffswelt, die Metaphorik und die Interaktionsschemata der Methoden und Werkzeuge der Software-Entwicklung sollen so gestaltet sein, dass sie den Vorgehensweisen und Hilfsmitteln der Anwendungsdomäne nachempfunden sind. So wird es möglich, Hemmschwellen, die eine echte Partizipation von technisch nicht versierten Fachleuten am technologischen Prozess bisher verhinderten, abzubauen. Domänensachverständige können so direkt am Zielartefakt mitwirken, von der direkten Rückkopplung des prototypischen Entwickelns profitieren. Das setzt einen Prozess der Individualisierung der Entwicklungsprozesse und der entsprechend unterstützenden Werkzeuge voraus. Diese Prozesse werden co-adaptiv sein in dem Sinne, dass die Entwicklungsumgebung an die Nutzerbedürfnisse angepasst wird, und im Gegenzug dazu motiviert, die Arbeitsweisen der Nutzer zu beeinflussen - einen Lernprozess im Umgang mit der Technologie anzustoßen (Fischer et al. 2004).

Die vorliegenden Ergebnisse wurden als Lösung konkreter Probleme im Projektablauf der Entwicklung an Lernsoftware erarbeitet. Dazu wurden bestehende Verfahren des Requirements Engineering und der Systementwicklung betrachtet und auf Schwachstellen bzgl. dieses Projektkontexts untersucht. Die wesentlichen Ansätze werden im Anschluss an diese Einleitung beschrieben. Aus diesen Erkenntnissen wurde die in Abschnitt 2 beschriebene Konzeption einer co-adaptiven Entwicklungsumgebung abgeleitet.

Der gefundene Lösungsansatz zum Schließen der methodischen Lücken wurde im Projekt direkt experimentell umgesetzt. Die Ergebnisse dessen werden in Abschnitt 3 vorgestellt und die in der Anwendung gemachten Erfahrungen wiedergegeben. Der Beitrag der Lösung zum erfolgreichen Verlauf der Entwicklung illustriert dabei die Machbarkeit der nötigen Infrastruktur und die Tragfähigkeit der Lösung insgesamt. In Abschnitt 4 schließlich wird das Ergebnis der Arbeit zusammengefasst.

2 Integration von Software Engineering und Anwendungsdomäne

Das Problem der "Brandmauer der Informatik" (Dijkstra 1989) ist sicher so alt, wie die Informatik selbst. Die Konflikte, die sich aus dem Wunsch nach einer klaren Trennung zwischen der abzubildenden Lebenswelt und der formalisierten "korrekten" Welt der Algorithmen ergeben, werden bis heute auf wissenschaftstheoretischem Niveau diskutiert (Jahn 2005).

Die alltägliche Software-Entwicklungspraxis bedient sich bei der Arbeit an der Schnittstelle weitgehend erfolgreich der Ergebnisse des "klassischen" Requirements Engineering. Diese setzen im Wesentlichen auf die widerspruchsfreie, vollständig und korrekte Erfassung, Formalisierung und Verwaltung von Anforderungen in Form mehr oder minder genormter Spezifikationensdokumente (Rupp 2002; Schienmann 2002).

Insbesondere auf dem Gebiet interaktiver Software zur Repräsentation und Vermittlung komplexer fachlicher Zusammenhänge ist es erfahrungsgemäß kaum möglich, konzeptionelle Phasen der Entwicklung abschließend in Anforderungsmodellen (etwa use-cases der UML) zu fixieren. Neben den zwischen Anwendungsdomäne und Entwicklung oft divergenten Vorstellungen von der Form der Abstraktion ist es hier nötig, experimentell zu arbeiten. Man ist auf zeitnahe Rückkopplungen zwischen fachlichem Konzept und der Wirkungsweise der technischen Realisierung, also den schnellen Wechsel zwischen Entwurf und Simulation angewiesen (Preim 1999).

Um diesen Wechsel möglichst nahtlos vollziehen zu können, ist ein gängiger Ansatz, potentielle Nutzer bzw. Domänenfachleute in diesen Prozess des prototypischen evolutionären Arbeitens eng einzubeziehen. Der in Wood et al. (1995) verfolgte Ansatz versucht in strukturierten Workshops, Nutzer auch an technisch geprägten Teilen des Entwicklungsprozesses partizipieren zu lassen. Dabei kommen die tatsächliche „haptische“ Erfahrung mit dem zu

erwartenden Entwicklungsergebnis und vor allem die Einflussmöglichkeit der Anwender oft zu kurz.

Direkt Eingriffsmöglichkeiten in die Gestaltung der Nutzerschnittstelle über so genannte interaktive Prototypen bietet Bødker et al. (1991). Dabei wird ein erster Aufschlag für die Visualisierung und die verwendeten Interaktionsschemata in einen horizontalen Prototypen umgesetzt und den Anwendern zur Nutzung zur Verfügung gestellt. Neben der Beobachtung des Verhaltens der Nutzer besteht die Möglichkeit der direkten Manipulation der Vorlage durch den Nutzer. Die gewonnen Erkenntnisse fließen in den abgekoppelten Prozess der Implementierung. Ein kurzzyklisches Experimentieren und die Erfahrung des konstruktiven Mitwirkens am eigentlichen Artefakt sind für Domänenexperten so nur begrenzt gegeben.

Einen Schritt weiter gehen Konzepte wie „Hyper Media Storyboards“ (Geukes 2000). Dabei wird eine domänespezifische Script-Sprache entwickelt, die Fachautoren in die Lage versetzen soll, ihre Ideen selbst in ein lauffähiges System umzusetzen. Nach Einschätzung des Verfassers selbst ist die Sprache in der Praxis jedoch nicht in der Lage, Autoren zur selbständigen Arbeit an einem Software-Artefakt zu befähigen bzw. zu bewegen (Geukes 2000).

Auch das „End-User Development“ setzt auf die Mitwirkung des „Nichtinformatikers“ beim Software-Lebenszyklus. Im Mittelpunkt steht dabei die Idee, den Reifeprozess von Software-Systemen und damit zu großen Teilen das, was man herkömmlich als Wartungsphase bezeichnet mit den Prozessen der Nutzung zu verbinden. In der Anwendung kann der Nutzer das System, etwa über einfache Mechanismen der Konfiguration, auf seine individuellen Bedürfnisse ausrichten, erweitern und vervollkommnen. Darüber hinaus soll erreicht werden, dass über das Arbeiten am individualisierten System der Umgang mit der Technologie co-adaptiv, iterativ erlernt wird (Fischer et al. 2004). Eine explizite während der Entwicklungszeit produktiv einsetzbare Infrastrukturunterstützung für diese Konzepte existiert nicht.

Bei der Entwicklung hochgradig interaktiver und reaktiver Software-Systeme zur Repräsentation komplexer fachlicher Inhalte ergibt sich die Notwendigkeit, eine Lösung zu finden, die sowohl die geografischen Distanzen als auch die fachlichen und methodischen Differenzen innerhalb des heterogenen Teams überbrücken muss. Die enge Integration aller Projektbeteiligten sollte sowohl organisatorisch durch die klare Definition von Prozessen als auch durch eine kontext- und fachspezifische sowie vernetzte Infrastruktur forciert werden.

3 Die Co-adaptive Entwicklungsumgebung

Aus der Adaption, der Synthese und der Erweiterung, der in den vorangegangenen Abschnitten vorgestellten Ansätze, wurde eine Systematik abgeleitet, die es erlaubt, Fachleute und potentielle Endanwender in den Software-Entwicklungsprozess direkt einzubinden. Es wurde eine Möglichkeit gefunden, die Beiträge technologieferner Rollen des Projekts direkt in das zentrale Artefakt – das evolutionär entwickelte Zielsystem – fließen zu lassen. Kern des Ansatzes ist eine Entwicklungsinfrastruktur, die es erlaubt fachabhängige Sichten auf das Software-Artefakt zu etablieren. Das heißt, die Nutzerschnittstellen zur Entwicklung am

Zielsystem können bez. der verwendeten Metaphorik, Fachterminologie, des Screen Designs, der Aufbau-, und der Ablaufstrukturierung an die Erfordernisse - die Arbeitsweise und das Verständnis - der Fachdomäne angepasst werden.

Die entwickelte Infrastruktur trägt die wesentlichen Züge eines Domänenframeworks (Mattsson 1996). Dabei wird ein Architekturskelett, das alle „vitalen" Teile (z. B. elementare Persistenz- und Kommunikationskomponenten) der Zielanwendung bereits enthält, im Zuge der Anwendung systematisch zum konkret angestrebten Software-Produkt ausgeprägt. Entscheidender Unterschied beim Einsatz der hier vorgestellten Infrastruktur ist die enge Integration adaptierbarer Entwicklungszugänge mit dem Rahmen für die Zielanwendung und eine klare Trennung in zwei Konzept-Ebenen. Auf der einen Seite erfolgt die Konkretisierung des Frameworks für den Einsatzkontext. IT-Spezialisten erarbeiten dabei in enger Zusammenarbeit mit allen beteiligten Disziplinen – also innerhalb eines eigenen iterativen Entwicklungszyklus' – das Entwicklungswerkzeug als Grundlage für die Erstellung des eigentlichen Zielsystems im Rahmen der zweiten inhaltlich fachlich geprägten Ebene. Diese ist ebenfalls als vollständiger Entwicklungsprozess mit eigener fachbezogener Anforderungsanalyse zu charakterisieren.

Der implizierte Entwicklungsprozess gestaltet sich entsprechend zweistufig: In einer initialen Phase (in Abbildung 1 als technische Entwicklung bezeichnet), werden die Entwicklungswerkzeuge an die Arbeitsweise und das Verständnis der Domänenexperten angepasst. Das geschieht im Wesentlichen über die Umgestaltung und Umstrukturierung der entsprechenden Nutzeroberflächen und die Vorauswahl, ggf. Ergänzung und die Konfiguration der zur Verfügung gestellten Basisfunktionalitäten. Ferner wird die grundsätzliche Abbildung der Oberflächen-Konfiguration in das Erscheinungsbild sowie die Navigations- und Interaktionsmuster der Zielanwendung festgelegt.

Dazu werden aus den in der Abbildung 1 als „Frame" bezeichneten „Baukästen" die benötigten Basis-Komponenten konkretisiert. Für den „Functional Frame" etwa bedeutet das, dass die dort enthaltenen vorgefertigten Elemente zur Erstellung einfacher Werkzeuge zur Analyse digitaler Bilder (zoomen, markieren, beschriften, etc.) so aufbereitet werden, im nicht trivialen Fall nach dem „white box"-Prinzip am Code, dass die Domänenexperten und Fachdidaktiker in die Lage versetzt werden, Bildinhalte etwa im Rahmen von Methodentrainings so zu inszenieren, dass sie ad hoc mit einem konkreten Zielartefakt arbeiten können. Die Konkretisierung des „Screen design Frames" besteht im Wesentlichen aus der Übertragung der projektabhängigen Gestaltungsrichtlinien auf die Komponenten, aus denen sich das Zielsystem aufbauen soll. Dieses „skinning" ist mit der stylesheet-basierten Gestaltung von Frontends bei Conten-Management-Systemen zu vergleichen. Mit der Ausprägung des „Navigational Frame" werden Navigationspfade, Interaktionsmuster und Rahmenfunktionalitäten, wie Inhaltsverzeichnisse, Suchmechanismen etc. festgelegt. Die in Abbildung 1 wiedergegebenen „Content Grabbers" umfassen eine Sammlung grundlegender Werkzeuge zur Erfassung und Strukturierung der elementaren medialen Inhalte. Auch diese werden im ersten Schritt an die Bedürfnisse, beispielsweise der Autoren, angepasst. Das bedeutet beispielsweise, dass die Metaphorik der Bedienelemente von Eingabemasken an die mentalen Modelle und Fachtermini der Domänenexperten angeglichen werden.

Diese initiale Phase der Ausprägung der Entwicklungsumgebung benötigt technischen Sachverstand. Ziel der Weiterentwicklung des Ansatzes ist es jedoch, den dafür benötigten Aufwand weit geringer halten zu können, als den der zentralen zweiten Phase. Das Entwicklungssystem, bestehend aus der adaptierbaren Werkzeugumgebung, ergänzt um die Leistung der technischen Vorarbeiten, ist aus der Sicht der Anwender tatsächlich als adaptiv zu bezeichnen, da es sich von innen heraus proaktiv an die Gegebenheiten der Entwicklungspraxis anpasst. Das entspricht, wenn auch in einem leicht modifizierten Sinne, der Begriffsbildung in (Schneider-Hufschmied et al. 1993).

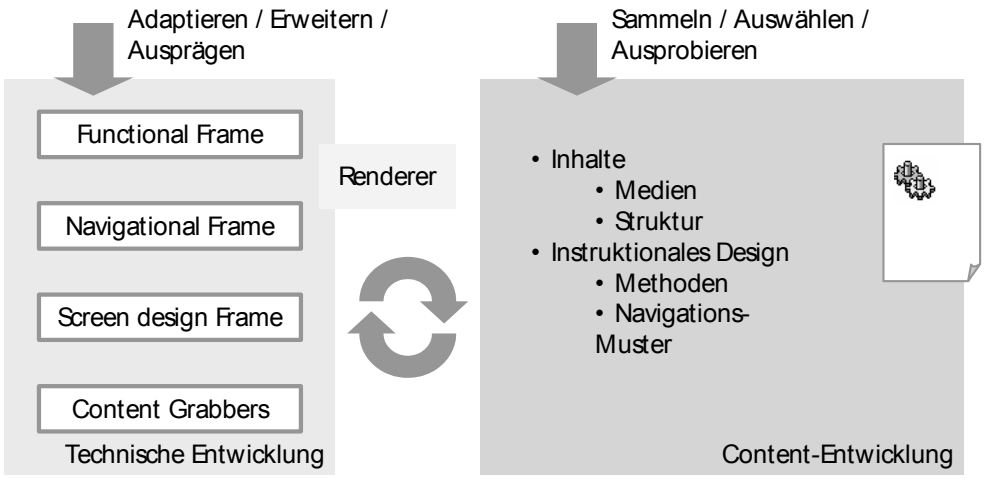

Abbildung 1: Co-adaptives, framework-basiertes Vorgehen

Die zweite zentrale Phase der Entwicklung ist die der fachlichen und inhaltlichen Ausgestaltung des Systems. Dabei muss die fachliche Expertise den Entwicklungsprozess bestimmen. Sie darf bez. des Gestaltungsspielraums möglichst wenig durch infrastrukturelle Aspekte eingeschränkt werden. Neben der grundsätzlich an die Fachlichkeit angepassten Entwicklungsumgebung ist es dazu erforderlich, alle Entwicklerschnittstellen so zu gestalten, dass sie leicht erlernbar sind. Das ermöglicht den tatsächlichen co-adaptiven Prozess der gegenseitigen Annäherung. Auch hier ist es in der Regel unrealistisch zu fordern, dass die Software-Infrastruktur aus sich heraus selbsterklärend genug ist, um eine tatsächlich motivierende, Frustration vermeidende Einarbeitung zu erlauben. Dieser Vermittlungsaspekt ist vielmehr als integraler Bestandteil des iterativen Prozesses zu sehen, der in der Rückkopplung zwischen den beiden Phasen nicht nur die Entwicklungssoftware selbst evolutionär optimiert und die Domänenkenntnisse der Techniker weiterentwickelt, sondern auch die systematische Entwicklung von technologischer Kompetenz bei IT-fernen Fachexperten und Endnutzern unter Anleitung der Software-Entwickler zum Ziel hat.

Erst in diesem Zusammenspiel und durch die bewusste Gestaltung dieses soziotechnologischen Prozesses wird es möglich, die Gestaltungsspielräume für die Entwicklung zu erkennen und auszuschöpfen. Die gemeinsame Arbeit an einem zentralen Artefakt forciert die

nötige enge Kooperation zwischen den Disziplinen, ohne durch methodische Restriktionen die Entfaltung emergenter, kreativer Prozesse zu sehr einzuschränken. Neben der prinzipiellen Domänenunabhängigkeit des Ansatzes wird hier auch die Abgrenzung zu etablierten fachspezifischen Entwicklungsparadigmen deutlich, die jeweils dezidiert für den spezialisierten Einsatz entwickelt sind. Die Flexibilität und der diskursive Charakter des hier vorgestellten Ansatzes erweist ist eine wichtige Voraussetzung für Innovation.

Das Ergebnis der Arbeit in der zweiten Phase der Entwicklung ist aus softwaretechnologischer Sicht ein Repostitorium mit darstellungsunabhängigen Repräsentationen von Inhalten, assoziiert mit entsprechenden Funktionalitäten der Interaktion und der Manipulation. Dies ist in Abbildung 1 durch das Dokumenten-Icon ganz rechts im Bild symbolisiert. Der konfigurierte „Renderer“ als physisches Ergebnis der technischen Entwicklungsphase schließlich fungiert als eigentliche Ablaufumgebung für die entstehende Software. Er setzt die Darstellungsunabhängigen Inhalte und Funktionen in die dem Nutzer präsentierten Software-Oberflächen um.

4 Der Prototyp: LeMOLernen

Die im vorangegangenen Abschnitt vorgestellten Konzepte sind Ergebnisse der Arbeit am vom BMBF geförderten Projekt „LeMOLernen“. Die Charakteristik des Softwareentwicklungsprojekts – entsprechend der Verallgemeinerungen zu interaktiven, direkt reaktiven Systemen in der Einleitung – machte es erforderlich bzgl. des Vorgehens neue Wege zu beschreiten. Die folgenden Ausführungen illustrieren, wie die Konzepte in der Praxis konkret umgesetzt werden konnten.

Aufgabe war es, eine Lernsoftware für den Einsatz im Geschichtsunterricht zu entwickeln. Sie sollte mit einer enormen Fülle an redaktionell aufbereitetem Quellenmaterial Inhalte zur Geschichte des zwanzigsten Jahrhunderts in Deutschland vermitteln helfen. Zentrales Ziel war jedoch die Unterstützung der Herausbildung historischer Methodenkompetenz auf der Basis neuer Medien bei Schülern und Lehrern gleichermaßen. Das heißt die Quellmaterialien waren so in geführte, interaktive Übungen einzubetten, dass Nutzer in die Lage versetzt werden, sowohl das richtige Vorgehen zu wählen, beispielsweise bei der Filmanalyse, als auch das richtige Werkzeug einzusetzen. Dabei kam es darauf an, die Wahl der Mittel zwar zu unterstützen, nicht jedoch vorzugeben.

Die Aufgabe der Inszenierung von Inhalten in Verbindung mit Methoden, Werkzeugen und entsprechenden Zielen der Kompetenzvermittlung, hat zu viele nuancierte Facetten, als dass sie unmissverständlich an einen Domänenunkundigen Techniker vermittelt oder gar formal abschließend spezifiziert werden könnte. Nur experimentelle Prozesse lassen Fachautoren und fachkundige Endnutzer, wie Lehrer, zu Ergebnissen, wie dem in Abbildung 2 dargestellten kommen. Der im vorangegangenen Abschnitt beschriebene „Functional Frame“ stellt dabei abstrakte Komponenten für das Betrachten von Videosequenzen bereit, die Möglichkeit Texteingabefelder zu positionieren sowie Drag&Drop-Elemente zu spezifizieren. Die Anpassung im Sinne der Co-Adaption bestand darin, diese Mittel in den Rahmen der Aus-

gestaltung digitaler Arbeitsblätter zu übertragen und damit für diese Domäne zu konkretisieren. Dazu wurde beispielsweise das Bedienelement der Werkzeugleiste als Teil des Interaktionsschemas konfiguriert und gestaltet. Sie ist damit Teil aller Arbeitsblätter und spezialisiert für den historischen Kontext. Das Verhalten und die Gestaltung der Medien-Player und der einzelnen Elemente sind ebenfalls auf den Einsatzkontext abgestimmt. So müssen sich Redakteure in Bezug auf das Beispiel nur noch mit Kategorien, wie „Aufgabenstellung“, „Video-Präsentation“, mit oder ohne „Standbild-Funktion“ und der groben Strukturierung der Screens befassen.

Fast keine abstrakten technischen Konzepte enthält schließlich der für dieses Projekt konfigurierte „Content-Grabber“. Die Nutzer-Oberflächen sind so strukturiert und gestaltet, dass ausschließlich die im Projekt verabredeten Fachtermini, entsprechende Ikonographie sowie einfachste Strukturierungsmechanismen zum tragen kommen. Die Einfachheit der Eingabeabläufe versucht dabei der Äußerung eines der beteiligten Historiker entgegenzukommen, wonach am besten mit Bleistift und Papier gearbeitet werden sollte - jedoch ohne den Anspruch tatsächlich Mittel zur Handschriftenerkennung einzusetzen. Der akzeptierte Umgang mit den elementarsten Bedienelementen aktueller Informationstechnik, wie Texteingabefeldern – damit Tastatur und Maus – ist das damit Ergebnis der Adaption des Autors gegenüber dem System.

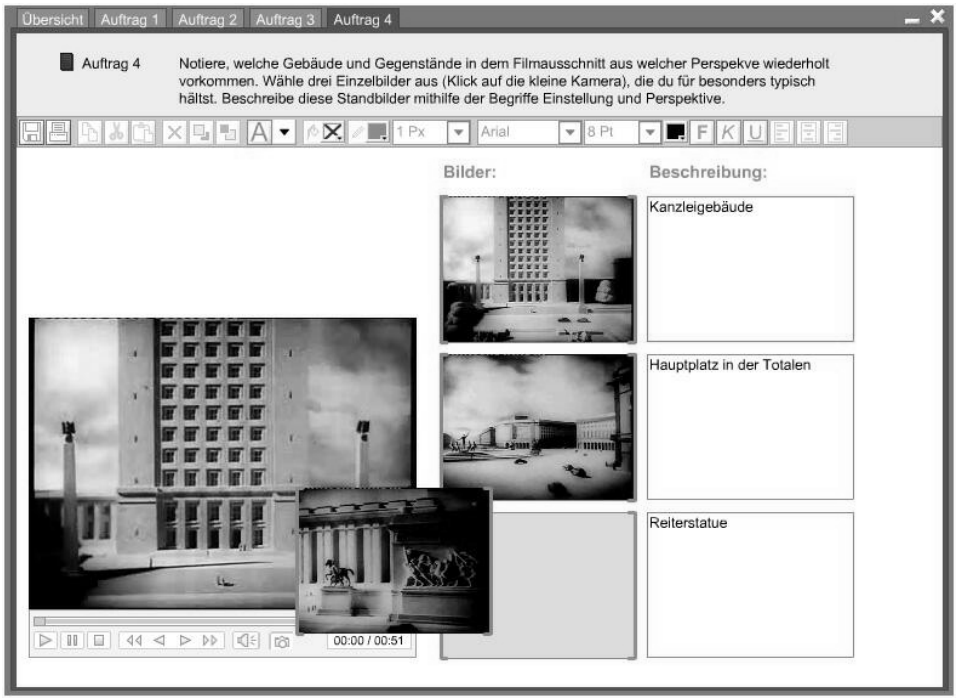

Abbildung 2 Ergebnis co-adaptiven Designs (digitales Arbeitsblatt in LeMOLernen)

Diese Herangehensweise ermöglichte nach anfänglichen Schwierigkeiten in der Zusammenarbeit, im Sinne der Problemstellung dieses Beitrags, den Konsens zwischen den Entwicklungsbeteiligten. Die Erarbeitung der konsensfähigen Arbeitsabläufe und Sichten war dabei die Hauptaufgabe. Mit dem Einsetzen der Erkenntnis über die eingeräumten Gestaltungsspielräume und die eröffneten Möglichkeiten des selbstbestimmten und explorativen Arbeitens stieg die Bereitschaft der Fachexperten, direkt am Zielartefakt zu arbeiten, sehr schnell. Selbst Unzulänglichkeiten des während des Projektes lediglich prototypisch erarbeiteten Entwicklungs-Frameworks wurden akzeptiert und zum Teil sehr kreativ umgangen. Der co-adaptive Prozess wurde dabei von Seiten der Fachredaktion etwa soweit betrieben, dass eine Auseinandersetzung mit den sehr technischen Aspekten, beispielsweise der Datenhaltung, erfolgte.

Der iterative Charakter der fortwährenden Ausprägung und Anpassung der Entwicklungsumgebung erwies sich im Projekt als tragfähig. Das bedeutet in erster Linie, dass nach der umfangreichen ersten Iteration lediglich noch kleinere Rückkopplungen nötig wurden. Die waren zu einem großen Teil der Tatsache geschuldet, dass das Entwicklungsprinzip selbst parallel zum Projekt weiterentwickelt wurde.

5 Zusammenfassung und Ausblick

Es wurden eine Methodik und die zugrunde liegende Werkzeugumgebung entwickelt, die es ermöglichen transdisziplinär geprägte inhaltlich und fachlich anspruchsvolle, qualitativ hochwertige Entwicklungen interaktiver Software systematisch durchzuführen. Das Vorgehen eröffnet dabei völlig neue Gestaltungsspielräume, indem Domänenexperten und End-User gleichermaßen in die Lage versetzt werden, unabhängig von Transferleistungen von Technologen, Ideen in Software-Repräsentationen umzusetzen und unkompliziert experimentell daran zu arbeiten. Während dieser Entwicklung werden die bestehenden Mittel fortwährend an die sich im Prozess weiterentwickelnden Bedürfnisse angepasst. Die Weiterentwicklung der Bedürfnisse resultiert dabei im Wesentlichen aus der sukzessiven Anpassung der technischen Kompetenzen der Entwicklungsbeteiligten. Mit dem Projektendergebnis „LeMOLernen“, einem markttauglichen Software-Produkt, konnte die Tragfähigkeit des Ansatzes in der Entwicklungspraxis eingehend und erfolgreich erprobt werden. Die nachhaltige Anwendbarkeit des Entwicklungs-Frameworks ist ferner durch die strikte Nutzung von Standard-Architekturen und -Entwurfsparadigmen und eine daraus resultierende sauber trennende Modularität sowie die konsequente Erstellung entsprechender Dokumentation, untermauert.

Die Übertragbarkeit der Entwicklungsinfrastruktur in andere Projektkontexte wird in weiteren Projekten untersucht und weiterentwickelt. Insbesondere die Unabhängigkeit vom bisher unterstützten Fach sowie dem Lehr- und Lernfokus muss in der zukünftigen Arbeit konsolidiert werden.

Literaturverzeichnis

Bødker, S. und Grønbæk, K. (1991): Design in Action: From Prototyping by Demonstartion to Cooperative Prototyping. In Greenbaum, J.; M. Kyng (Hrsg.): Design at Work: Cooperative Design of Computer Systems. Hillsdale, New Jersey: S. 197-218.

Dijkstra, E. W. (1989): On the Cruelty of Really Teaching Computing Science. In: Comm.of the ACM, Nr. 12, S. 1398-1404.

Fischer, G. und Giaccardi, E. (2004): Meta-Design: A Framework for the Future of End-User Development. In: Kluwer Academic Publishers.

Geukes, A. 2000. Design und Produktion Digitaler Interaktiver Lektionen. Design und Produktion Digitaler Interaktiver Lektionen, Wirtschaftswissenschaftliche Fakultät der Universität Bielefeld.

Jahn, T. (2005): Sozial-ökologische Forschung. Ein neuer Forschungstyp in der Nachhaltigkeitsforschung. In Linne, G.; Schwarz M. (Hrsg.): Handbuch Nachhaltige Entwicklung. Wie ist nachhaltiges Wirtschaften machbar? Opladen: Leske + Budrich, S. 545-555.

Mattsson, M. 1996. Object-Oriented Frameworks. Object-Oriented Frameworks, University College of Karlskrona/Ronneby.

Preim, B. (1999): Entwicklung interaktiver Systeme., Berlin: Springer.

Rupp, C. (2002): Requirements-Engineering und -Management. Professionelle iterative Anforderungsanalyse für IT-Systeme. 2. Auflage Carl Hanser Verlag.

Schienmann, B. (2002): Kontinuierliches Anforderungsmanagement. Addison-Wesley.

Schneider-Hufschmied, M.; Kühne, T. und Malinkowski, U. (1993): Adaptive User Interfaces - Principles and Practice., Amsterdam.

Wood, J. und Silver, d. (1995): Joint Application Development., New York: John Wiley & Sons Inc.

Kontaktinformation

Alexander Karosseit
Fraunhofer ISST
Mollstr. 1
10178 Berlin
Telefon: 030/24306-461
Telefax: 030/24306-599
E-Mai: alexander.karosseit@isst.fraunhofer.de