SPIELERISCHES KONSTRUIEREN IM VIRTUELLEN MEDIUM:

Digitale Baukästen in interkulturellen Computer Clubs

VON KAI SCHUBERT, MICHAEL VEITH, GUNNAR STEVENS UND VOLKER WULF

1. EINLEITUNG

Der digitale Medienumbruch ist durch das Vordringen vernetzter Computer in eine Vielzahl von Anwendungsbereichen gekennzeichnet. Um das in der medienwissenschaftlichen Diskussion häufig unterstellte emanzipatorische Potenzial dieses Medienumbruchs zu realisieren, besteht eine zentrale Herausforderung darin, Computermedien so zu entwerfen, dass Endnutzer sie neu entwickeln oder existierende verändern können. Die angewandte Informatik ist die Wissenschaftsdisziplin, die sich grundlegend mit der Gestaltung des digitalen Medienumbruchs beschäftigt. Für sie besteht eine zentrale Herausforderung darin, das bisher schon weitgehend ausgearbeitete *easy-to-use-Prinzip* zum *easy-to-develop-Prinzip* weiterzuentwickeln. Bisher ist es jedoch nur in Ansätzen gelungen, diese Vision software-technisch zu konkretisieren und adäquat auszugestalten.

Ferner bieten insbesondere Konzepte, die unter den Sammelbegriffen *Social Software* oder *Web 2.0* zusammengefasst werden, fruchtbaren Raum, um endnutzergetriebene Softwareentwicklung durch soziale Interaktion so zu potenzieren, dass auch komplexere Programmieraufgaben realisiert werden können. Aus einer Reihe von Vorarbeiten wissen wir, dass solch ein technischer Fortschritt stets auch von sozialem Wandel auf Basis der inter-personalen Interaktionen begleitet ist – informelle Gruppen und neue Wissensdomänen mit verlagerten Kapitalstrukturen entstehen. Das Artefakt, hier die gemeinschaftlich, interaktiv entwickelte Software, wird zur geteilten Praxis, zur Gemeinsamkeit. Soziale Unterschiede respektive Ungleichheiten können sich so relativieren und sogar ins fruchtbare Gegenteil kippen – zur Basis von systemischer und sozialer Weiterentwicklung werden.

In der Welt der Spiele findet man interessante Vorläufer des *easy-to-develop*-Prinzips in Form der Konstruktionsbaukästen. Einfache Konstruktionsbaukästen wurden bereits Ende des 19. Jahrhunderts kommerziell angeboten. In den letzten Jahren wurden diese Konstruktionsbaukästen durch die Integration von entsprechender Hard- und Software teilweise digitalisiert. Durch diese Weiterentwicklungen fand ein Umbruch im Konstruktionsmedium statt, der hier genauer untersucht und gestaltet werden soll. Ein paradigmatisches Beispiel für diesen Umbruch bildet *Lego Mindstorms* und die damit verbundene Vision des digitalen *Lego*. *Lego Mindstorms* ist ein Konstruktionsbaukasten, der das Spielen mit den bekannten

Lego-Bausteinen mit dem Programmieren am Computer kombiniert und damit die virtuelle mit der realen Welt verbindet. Bei der Gestaltung dieses Umbruchs hinsichtlich des Konstruktionsmediums besteht die zentrale Herausforderung darin, ein digitales Pendant zum *Lego*-Baustein zu finden. *Scratch*, eine rein digitale Umsetzung der Baustein-Metapher, erscheint als interessanter Kandidat, der im Fortlauf dieses Aufsatzes noch näher dargestellt werden wird.

Deshalb wurde zunächst die kollektive Konstruktionspraxis im Umgang mit *Lego* Mindstorms und *Scratch* in einem Anwendungsfeld untersucht, um so den stattfindenden Medienumbruch besser zu verstehen und zu analysieren. Darauf aufbauend wurden erste Versuche unternommen für diesen Konstruktionsbaukasten eine für Nichtprogrammierer geeignete Entwicklungsumgebung zu implementieren. Diese wurde bereits erprobt, eine Evaluation ob der Zielführung zum kooperativen, komponenten-basierten Konstruieren erfolgt in Kürze. Die in dieser Fallstudie erzielten Ergebnisse werden im Hinblick auf ihren übergeordneten Beitrag zur Ausarbeitung des *easy-to-develop-Prinzips* diskutiert.

2. ELEMENTE DER UNTERSUCHUNG DIGITALER KONSTRUKTIONSBAUKÄSTEN

Computer und Computerprogramme halten seit Jahren Einzug in die Bereiche Erziehung und Unterhaltung von Kindern. Erstaunlicherweise hat die angewandte Informatik den Bereich der Unterhaltung erst in den letzten Jahren entdeckt. Wirtschaftlich interessant ist der Bereich der Computerspiele bereits seit den 1970er Jahren und generiert mittlerweile jährlich Umsätze in Milliardenhöhe. Digitale Konstruktionsbaukästen sind diesbezüglich eine interessante Klasse von Anwendungen.

2.1 DER COMPUTER ALS SPIELMITTEL

Spielen an sich ist durch drei Merkmale gekennzeichnet. Es ist zweckfrei und folgt einer intrinsischen Motivation, es lebt von Wiederholung und Ritualen und es bedeutet eine Realitätstransformation.[1] Obwohl das Spiel an sich zweckfrei ist, werden zugleich übergeordnete, höhere Ziele des Spiels festgestellt. Spielen dient der allseitigen Förderung des Kindes, unterstützt die sozialen Fähigkeiten von Kindern, kompensiert soziale Defizite und dient (psychoanalytisch gesehen) der Angstabwehr, der Identitätsfindung und der Realitätsbewältigung.[2] Spiel ist also ein wesentlicher Bestandteil der kindlichen Entwicklung und stellt einen hohen Anteil an den Freizeitbeschäftigungen von Kindern dar – wobei Kinder mit und ohne Spielmittel (Spielzeug) spielen.

1 Vgl. Riemann: „Spiel, Spieltheorien in der Pädagogik“, S. 491; Oerter: „Spiel“, S. 216.

2 Vgl. Riemann: „Spiel, Spieltheorien in der Pädagogik“, S. 492.

> Spielerisch eignen sich Kinder (und Jugendliche) so Ausschnitte aus der gesellschaftlich-historisch entstandenen Welt an und erweitern dabei zugleich ihr Verständnis dieser Welt und ihre eigenen Wahrnehmungs- und Handlungskompetenzen.[3]

Die Spielwelten von Kindern sind ein Wechselspiel der gegenständlichen Welt und der psychischen Welt. Spielzeuge stellen für Kinder dabei gegenständliche Stellvertreter einer medial vermittelten, fiktionalen Welt dar. Diese andere Welt wird durch Spielzeug in der realen Welt sinnlich erfahrbar.[4]

Der Computer als ein besonderes Spielmittel macht die medial vermittelte Welt nicht nur für Sinneswahrnehmung, sondern auch für Spielhandlungen zugänglich.[5] „Die Spielwelt entsteht nicht als Überbrückung zwischen der inneren und äußeren Welt […], sondern sie wird medial vermittelt."[6] Die Interaktion und die damit verbundenen Spielmöglichkeiten sind für das kindliche Spiel besonders reizvoll.

Neueste Untersuchungen haben ergeben, dass Computerspiele im Freizeitverhalten von Kindern eher Lückenfüller sind.[7] Als zweitwichtigster Grund für das Benutzen von Computerspielen wird angegeben, dass man mit anderen zusammen spielen kann. „Es geht also nicht ums Spiel an sich, sondern darum, es mit anderen zusammen tun zu können."[8] Die Untersuchungen ergaben, dass Kinder als eigene Kompetenzanforderungen Herausforderungen im kognitiven Bereich (logisches Denken, Überlegen, Nachdenken und Phantasie für Lösungen haben) reizvoll finden. Erst danach werden sensomotorische und emotionale Kompetenzen gerne angewandt.[9]

2.2 DIGITALES KONSTRUKTIONSSPIELZEUG

Die gängigen Computerspiele haben im Allgemeinen jedoch das Defizit, dass sie nur in geringem Maße von den Spielern umgestaltet werden können. Die Spielregeln sind durch das System fest vorgegeben und können nicht durch die beteiligten Personen ausgehandelt werden. Die grundlegende Spielstruktur ist von den Entwicklern bereits vorgegeben und kann nur in Maßen verändert werden. Auf der anderen Seite wurden die Möglichkeiten, die der Computer als universelle Maschine bietet, nur von einer kleinen Anwendergruppe genutzt. Die so genannten *Hacker* bzw. *Nerds* betrachten den PC an sich als ein Spielzeug, mit dem man

3 Fromme u.a.: Computerspiele in der Kinderkultur, S. 11.

4 Vgl. Meder/Fromme: „Computerspiele und Bildung", S. 19.

5 Vgl. ebd., S. 20.

6 Ebd.

7 Fromme u.a.: Computerspiele in der Kinderkultur, S. 49.

8 Ebd.

9 Vgl. Fromme u.a.: Computerspiele in der Kinderkultur, S. 100ff.

herumexperimentieren kann. Sie können dabei als der Typus des *homo ludens* angesehen werden, der im zweckfreien Umgang die Möglichkeiten des Universal-Computers auskundschaftet.

Der hohe Lernaufwand und der eingeschränkte Nutzen führen dazu, dass dies nur für wenige Endbenutzer interessant ist. Deshalb besteht die Herausforderung in der bewussten Gestaltung von flexiblen Computersystemen, die den ‚normalen' Computerbesitzern das Potenzial des Universal-Computers zugänglich machen, ohne sie dabei zu unter- oder überfordern. In der angewandten Informatik werden aktuell unter dem Begriff des *end user development* (EUD) verschiedene Forschungsrichtungen vereinigt, die eine integrierte, wissenschaftliche Sichtweise auf diese Thematik entwickeln.[10] Konstruktionsbaukästen bieten in der Welt der Spiele Möglichkeiten zur Gestaltung des Spielgegenstands durch die Spieler. Insofern findet man hier Vorreiter des *easy-to-develop*-Prinzips, die es bereits seit gut 200 Jahren gibt. Seit einigen Jahren wird Konstruktionsspielzeug wie *Lego* oder *Fischertechnik* durch Computer erweitert. Hier werden also traditionelles (*Lego, Fischertechnik*) und digitales Konstruktionsspielzeug (Computer) miteinander vereinigt. Das Besondere an den digitalen Baukästen besteht zum einen darin, dass mit diesen Produkten das Programmieren von Computern Einzug in die Kinderkultur erhalten hat. Es eröffnet den Kindern die Möglichkeit, das Spiel konstruktiv zu gestalten, und es befähigt die Kinder, in Grenzbegriffen zu denken und organisatorisch zu handeln. Zum anderen gelingt diesen Baukästen aufgrund ihrer hybriden Stellung als sowohl materielles als auch digitales Konstruktionsmedium eine Verknüpfung von materieller und digitaler Welt.[11]

Für die grundlegende Frage nach den Gestaltungsmöglichkeiten eines Konstruktionsmediums – wie es der Computer als universelle Maschine darstellt – und dessen Nutzung im Bereich des spielerischen Konstruierens steht mit dem weit verbreiteten, digitalen Konstruktionsbaukasten ein ideales Untersuchungsobjekt zur Verfügung. Die Anfänge digitaler Baukästen findet man in Mitte der 1980er Jahre. Die beiden Wissenschaftler Stephen Ocko und Mitchel Resnick aus der Arbeitsgruppe von Seymour Papert begannen am MediaLab des MIT Robotersteuerungen für Kinder zu entwickeln. Sie experimentierten mit elektronischen Schnittstellen, die es Kindern erlauben sollten, Motoren und Sensoren an Computer anzuschließen und mittels der Programmiersprache LOGO anzusteuern.[12] Papert und seine Kollegen hatten schon früher versucht, Roboter mittels LOGO zu programmieren. Eine wichtige Weiterentwicklung von Ocko und Resnick im Vergleich zu früheren Arbeiten war es, den Kindern nicht nur das Programmieren zu erlauben, sondern die angesteuerten elektromechanischen Geräte auch selber bauen zu lassen. Sie führten diese Arbeiten in einem Forschungsprojekt weiter,

10 Lieberman u.a.: End User Development.

11 Vgl. auch Martin u.a.: „To Mindstorms and Beyond: Evolution of a Construction Kit for Magical Machines".

12 Vgl. Resnick/Ocko: „LEGO-Logo: Learning Through and about Design"; Resnick: „Behavior Construction Kits".

das 1998 von der *Lego Company* unter der Produktbezeichnung *Lego Mindstorms* kommerzialisiert wurde.[13]

Bei *Lego Mindstorms* wird ein hybrider Ansatz verfolgt, weil hier ein Konstruktionsbaukasten besteht, der das Spielen mit den bekannten *Lego*-Bausteinen mit dem Programmieren am Computer kombiniert. Der spielerische Aspekt bei *Lego* und die gute Dokumentation der zugrunde liegenden Technik und insbesondere die darauf aufbauenden Open-Source-Projekte, sprechen dafür, *Lego Mindstorms* als Untersuchungsobjekt zu wählen. Ein anderes, aktuelleres Untersuchungsobjekt stellt die ebenfalls am MIT entwickelte Softwareumgebung *Scratch* dar. *Scratch* ist sowohl der Name einer neuen, leicht zu erlernenden Programmiersprache als auch der Name der dazugehörigen Programmierumgebung. *Scratch* basiert auf *Smalltalk*/*Squeak* und ist selbst in *Squeak* implementiert. *Scratch* ist (pre)interpretiert durch vordefinierte Codeblöcke, visuell puzzleteilartig mit eindeutigen Formen dargestellt, die durch den User via *drag and drop* manipuliert und ähnlich wie bei *Lego* oder einem Puzzle nur mit bestimmten, passend geformten, anderen Elementen zusammengefügt werden können. *Scratch* wurde für Anwender ohne weitere Programmierkenntnisse entwickelt und eignet sich daher besonders für Kinder. Allerdings kann, durch die offene und freie Gestaltung der Umgebung Scratch auch für komplexere und ernsthaftere Programmieraufgaben genutzt werden. Ein potenzieller Zugang für erwachsene Nutzern ist also auch gegeben.

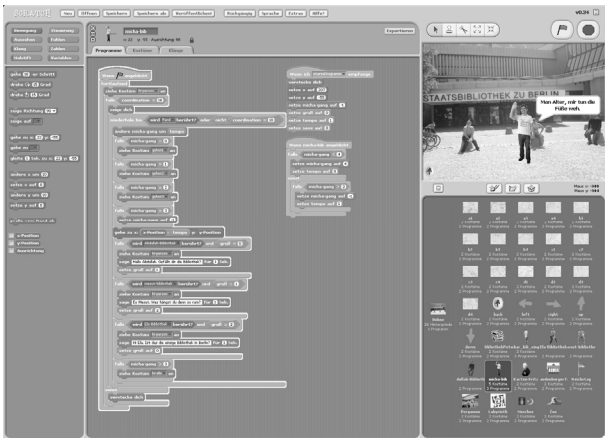

Abb. 1: Der digitale Konstruktionsbaukasten Scratch. Hier ist das Splatch-Framework auf Ebene 3 zu sehen.

13 Zur Zeit wird am MIT an der Verbesserung der Hardware gearbeitet. Die neue Generation von programmierbaren Bricks, auch Crickets genannt, ist deutlich kleiner. Crickets haben in etwa die Größe von Matchbox-Autos. Sie besitzen eine bidirektionale Infrarot-Schnittstelle, und ihnen liegt eine Bus-Architektur zugrunde (vgl. Martin u.a.: „Meta-Cricket"). Aus den Arbeiten am MIT sind weitere experimentelle Computerspielzeuge entwickelt worden, die aber nicht kommerziell verfügbar sind (vgl. Resnick u.a.: „Digital Manipulatives").

2.3 GESTALTUNG ANPASSBARER SYSTEME

Ein weiterer Zugang zum Thema des e*nd user development* findet sich im dem Bereich, der sich mit der Gestaltung anpassbarer Software-Systeme auseinandersetzt. Unter Anpassbarkeit (*tailorability*) versteht man dabei die Entwicklung von Systemen, die Endnutzern Möglichkeiten zur Anpassung an veränderte Anforderungen nach der eigentlichen Entwicklung bieten[14].

Empirische Untersuchungen in diesen Bereich haben Anfang der 1990er Jahre bereits Bedingungen identifiziert, unter denen Benutzer Anwendungen adaptieren.[15] Viele marktgängige Anwendungen versuchen den Benutzern Adaptierungsmöglichkeiten unterschiedlicher Komplexität bereitzustellen. Auf Grund ungenügender Gestaltung der Benutzerschnittstelle und unzureichender technischer Flexibilität bleibt die Nutzung dieser Möglichkeiten zur Adaption aber meist hinter den Erwartungen zurück.

Im Bereich der *Computer Supported Cooperative Work* (CSCW) sind einige Ansätze entwickelt worden, die eine größere Anpassbarkeit von Systemen erlauben. Das *Button*-System, entwickelt unter der Leitung von Alan McLean (*Xerox EuroParc*), bietet u.E. die interessantesten Konzepte des nahtlosen (*seamless*) Übergangs zwischen verschiedenen Komplexitätsniveaus der Adaptierung.[16] OVAL[17] ist ein anpassbares Werkzeug für kooperative Arbeit, das aus vier Typen von Bausteinen (*objects*, *views*, *agents* und *links*) besteht, mit denen sich *groupware*-Anwendungen zusammensetzen lassen. Ein Ansatz von Teege[18] beruht auf der Idee, dass Benutzer ihr System anpassen können, indem sie das Verhalten von Komponenten ihrer *groupware* durch das Auswählen von Features verändern können. *Groove* ist ein kommerzielles *peer to peer groupware*-System, das diesen Ansatz umgesetzt hat.[19] Der hier verwendete Ansatz entstammt dem *groupware*-Bereich und basiert auf zur Laufzeit zusammensteckbarer Komponenten.

14 Vgl. Bentley/Dourish: „Medium Versus Mechanism: Supporting Collaboration Through Customisation"; Kahler u.a.: „Tailorable Systems and Cooperative Work"; Lieberman, Henry u.a. (Hrsg.): End User Development; sowie Wulf u.a.: „Component-based Tailorability: Enabling Highly Flexible Software Applications".

15 Vgl. Mackay: „Users and customizable Software"; Nardi/Miller: „Twinkling Lights and Nested Loops: Distributed Problem Solving and Spreadsheet Development"; Oppermann/Simm: „Adaptability: User-Initiated Individualization".

16 MacLean u.a.: „User-tailorable Systems: Pressing the Issue with Buttons".

17 Malone u.a.: „Experiments with Oval: A Radically Tailorable Tool for Cooperative Work".

18 Teege: Individuelle Groupware: Gestaltung durch Endbenutzer.

19 Slagter/Ter Hofte: End-User Composition of Groupware Behaviour.

2.4 GESTALTUNGSMETAPHERN BEIM SYSTEMDESIGN

Bei der informationstechnischen Begleitung eines Medienumbruchs steht man vor der Herausforderung, das Neue mit dem Alten in Verbindung zu setzen. Eine Möglichkeit besteht in der Verwendung von Analogien und Metaphern.[20] Im Entwurf von Anwendungssystemen, insbesondere bei Benutzeroberflächen, sind Metaphern ein allgemein anerkanntes Gestaltungsmittel. So schreiben Lippert u.a. in Anlehnung an Kent Beck:

> Jedes Softwareprojekt sollte von Metaphern geleitet sein. Dies können ganz einfache bildliche Vorstellungen davon sein, wie ein System funktioniert, welche grundsätzlichen Annahmen gemacht wurden. […] Eine Metapher gibt immer auch technische Anleitung im Sinne einer Architekturvorgabe.[21]

Die Frage, welche Rolle Metaphern bei der Vermittlung der Informatik und Gestaltung von Programmierumgebung haben, ist jedoch umstritten. Weite Beachtung hat Dijsktras Position gefunden:

> By means of metaphors and analogies, we try to link the new to the old, the novel to the familiar. Under sufficiently slow and gradual change, it works reasonably well; in the case of a sharp discontinuity, however, the method breaks down [...]. Coping with radical novelty requires [...] [that] one must consider one's own past, the experiences collected, and the habits formed in it as an unfortunate accident of history, and one has to approach the radical novelty with a blank mind, consciously refusing to try to link history with what is already familiar, because the familiar is hopelessly inadequate.[22]

Dagegen hat Mayer in empirischen Studien gezeigt, dass Anfänger besser lernen, wenn ihnen ein metaphorisches Modell der Computersprache gegeben wird, als wenn ihnen nur eine wörtliche, technische Darstellung gegeben wird.[23] Blackwell kommt aufgrund seiner Experimente für den Fall von metaphorischen Bildern beim visuellen Programmieren hingegen zu dem Schluss: „The conclusion of the

---

20 Den Unterschied zwischen einer Analogie und einer Metapher definiert Travers: „Analogy usually refers to the construction of explicit mappings between two well-established domains, whereas metaphor is more often implicit." (Travers: Programming with Agents, S. 31).

21 Lippert u.a.: Software entwickeln mit eXtreme Programming: Erfahrungen aus der Praxis.

22 Dijkstra: „On the Cruelty of the Really Teaching Computing Science", S. 1398.

23 Vgl. Mayer: „Different Problem-Solving Competencies Established in Learning Computer Programming with and without Meaningful Models".

research described in this dissertation will be that the case for the importance of metaphor is greatly over-stated."[24]

Travers wiederum hat bestehende Programmiersprachen-Ansätze auf ihren metaphorischen Gehalt hin untersucht.[25] Er bezieht sich dabei auf neuere Theorien zu Metaphern und den Begriff der *theory-constitutive metaphor*[26]. Travers hat fünf verschiedene metaphorische Modelle ausgemacht, die den verschiedenen Programmierparadigmen zugrunde liegen. Aufgrund von Überlegungen, die Anleihen aus der Entwicklungspsychologie nehmen, plädiert er dafür, die Metapher des *animacy and agents* als Grundlage für Programmierumgebungen für Kinder zu wählen. Er selbst hat dazu die Bedeutung der Metapher des *animacy and agents* exemplifiziert und darauf aufbauend sein System *LiveWorld* entwickelt. Cottmann dagegen argumentiert in ihrer Dissertation gegen die Verwendung einer animistischen Systemgestaltung, da hierdurch die Entwicklung des Computerverständnisses bei Kindern, die sich in der egozentristischen Phase befinden, gehemmt wird.[27]

Das Konzept der Software-Komponenten stellt einen neuen, viel versprechenden Ansatz aus der professionellen Software-Entwicklung dar, der wegen seiner Nähe zur traditionellen *Lego*-Bausteinmetapher für die Gestaltung von digitalen Konstruktionsbaukästen besonders gut geeignet erscheint.

2.5 UNTERSTÜTZUNG FÜR KOOPERATIVES ENTWICKELN UND ANPASSEN

Die Arbeiten von Mackay, aber auch von anderen[28] haben gezeigt, dass Anpassen häufig in kooperativer Art geschieht und dabei die Etablierung einer Anpassungskultur eine wichtige Herausforderung darstellt. Aufbauend auf diesen Arbeiten hat Kahler[29] gemeinsame Repositories zum Austausch von angepassten Systemversionen konzipiert.

Im Bereich des Software-Engineering wird dem kooperativen Programmieren, das über eine gemeinsame Versionsverwaltung hinausgeht, große Bedeutung zugesprochen. Vivacqua und Lieberman beschreiben *Expert Finder*, ein System zur Unterstützung von Kooperation zwischen *Java*-Programmierern.[30] *Expert Finder* hilft dabei, unter den Teilnehmern am System geeignete Ansprechpartner zu

24 Vgl. Blackwell: Metaphor in Diagrams.

25 Vgl. Travers: Programming with Agents.

26 Vgl. Boyd: „Metaphor and theory change"; vgl. Lakoff/Johnson: Metaphors We Live By; Sweetser: From Etymology to Pragmatics; Lakoff: „The Contemporary Theory of Metaphor"; Reddy: „The Conduit Metaphor".

27 Vgl. Cottmann: Wie verstehen Kinder Maschinen und Computer?

28 Vgl. Mackay: Users and Customizable Software; Gantt/Nardi: „Gardeners and Gurus: Patterns of Cooperation Among CAD Users"; Trigg/Bødker: „From Implementation to Design".

29 Vgl. Kahler: Supporting Collaborative Tailoring.

30 Vgl. Vivacque/Lieberman: „Agents to Assist in Finding Help".

Problemen im Umgang mit *Java*-Konstrukten jeder Art aufzufinden.[31] Während der *Expert Finder* und verwandte Systeme Unterstützung durch geeignete menschliche Akteure empfehlen, weist eine andere Form die Kooperationsunterstützung Programmierer auf die Wiederverwendung geeigneter Codemodule hin. Der *Code Broker*[32] zeigt proaktiv Entwicklern in ihrer konkreten Situation nutzbare Komponenten aus einem *repository* an. Darauf aufbauend haben Fischer u.a. eine allgemeine Konzeption kooperativer Software-Entwicklung entwickelt.[33] Dieser Ansatz basiert auf dem SER-Modell (*seeding, evolutionary growth, reseeding*)[34], das drei gleichnamige Phasen definiert. In der ersten Phase (*seeding*) wird eine grundlegende Software- und Wissensbasis für das System gelegt, der potenziellen Teilnehmern einen Anreiz zur Teilnahme bzw. Mitwirkung bietet. Die Weiterentwicklung geschieht in der zweiten Phase (*evolutionary growth*) durch die engagierte Nutzung des Systems. Die dritte Phase (*reseeding*) definiert eine Neuordnung oder Neustrukturierung des Bestands, wenn dieser unübersichtlich geworden ist.

3. DER EINSATZ DES DIGITALEN KONSTRUKTIONSBAUKASTENS SCRATCH IN EINEM INTEGRATIVEN COMPUTERCLUB

Anwendungsfeld unserer Untersuchungen ist ein interkultureller und generationenübergreifender Computerclub (come_IN). Der Computerclub wird gemeinsam mit einer Grundschule in einem Stadtteil von Bonn getragen. In diesem etwa einen halben Quadratkilometer großen Stadtviertel leben derzeit etwa 8.700 Menschen aus 109 verschiedenen Nationen. Neben der deutschen Majorität von ca. 78% bilden die Türken mit etwa 10% die zweitstärkste und die Polen mit 1,2% die drittgrößte Nationalität in der Gesamtbevölkerung. Der Anteil von 27,8% an Zuwanderern (wobei hier deutsche Spätaussiedler mitgerechnet sind) übertrifft den Durchschnitt innerhalb der gesamten Stadt (22,4%) deutlich. Eine überdurchschnittlich hohe Bevölkerungsdichte und ein niedriges Durchschnittsalter von 34,9 Jahren zeichnen ebenso das Bild wie auch eine leicht erhöhte Arbeitslosigkeit (11,5%) und – im Gegensatz zur Gesamtentwicklung der Stadt – leichte Abwanderungstendenzen aus dem Viertel.[35]

Etwa 220 Kinder besuchen die als offene Ganztagsschule geführte Grundschule, wobei die Verteilung der Nationalitäten in etwa der des Stadtteiles entspricht (siehe oben). Die Schule hat eine lange Tradition und arbeitet nach den reformpädagogischen Ideen Maria Montessoris. Jahrgangsmischung, Freiarbeit,

31 Vgl. auch Reichling u.a.: „Expert Recommender".

32 Vgl. Ye: Supporting Component-Based Software Development with Active Component Repository Systems.

33 Vgl. Fischer u.a.: „Fostering Social Creativity by Increasing Social Capital".

34 Vgl. Fischer u.a.: „Seeding, Evolutionary Growth and Reseeding".

35 Sämtliche statistischen Angaben aus den Jahren 2005/06 stammen von der Statistikstelle der Stadt Bonn.

Praxisbezug und ein schülerorientierter Unterricht sind realisiert. Gemeinsames Lernen von Schülern unterschiedlichen Alters, erreicht durch Jahrgangsmischung sowie Team- und Projektarbeit bildet einen der Grundsteine der Schule. Aus einer Kooperation von Wissenschaftlern der Universität und der Schulleitung entstand in den Jahren 2002/03 die Idee und das Konzept eines interkulturellen Computerclubs.[36] Im März 2004 konnte der Club dann offiziell eröffnet werden.

3.1 DER CLUBBETRIEB

Einmal wöchentlich (zurzeit jeden Montag) findet der Club von 17 bis 19 Uhr statt. In dessen Rahmen ist den Schülern zusammen mit ihren Eltern die Möglichkeit gegeben, praxisrelevante Projekte und Ideen mit Hilfe des Computers und weiterer Informations- und Kommunikationstechnologie umzusetzen. Dabei werden sie von Lehrern, Tutoren und den Wissenschaftlern der Universität betreut. Die Teilnahme am Club ist freiwillig. Der Raum des Clubs ist mit derzeit zwölf Arbeitsplatzrechnern ausgestattet und befindet sich direkt in der Schule. Neben den normalen Computern steht weitere Technik wie Scanner, Drucker, digitale Film- und Fotokameras, Datenprojektor und ein speziell ausgerüsteter Multimedia-PC zur Verfügung. Sämtliche Computer verfügen zudem über Zugang zum Internet.

Ist die Teilnahme am Club freiwillig, unterliegt sie doch gewissen Regeln. Die wichtigste und seit Beginn geltende ist, dass jedes Kind von mindestens einem Elternteil begleitet werden muss. Neben dem Schreiben von Texten haben sich die Aktivitäten im Club auf ein sehr weites Feld von Arbeiten mit verschiedenster Informations- und Kommunikationstechnologie ausgeweitet. Tabellen-, Bild- und Videobearbeitung, Zeichenprogramme und die Nutzung des Internets spielen verstärkt eine Rolle. Jeder Teilnehmer bringt sich und seine Fertigkeiten ein, lernt von und mit den anderen und steuert so etwas zur Bildung eines gestalterischen Spielraumes des Clubs bei. Rollenbildungen wie die des Experten für Bildbearbeitung sind infolgedessen zu beobachten.[37]

Die einzelnen Projekte sind lokalen und aktuellen Themen gewidmet und orientieren sich im zeitlichen Ablauf zumeist an den Schulhalbjahren. An konkreten Ergebnissen sind bisher beispielsweise mehrere Hefte erschienen. Sie beinhalten Berichte über die aktuellen Aktivitäten des Clubs wie beispielsweise Interviews mit Altstadtbewohnern, Lebens- und Häusergeschichten, Reiseberichte oder auch Fotos schöner und unschöner Orte im Stadtviertel.

36 Stevens u.a.: „Bridging among Ethnic Communities by Cross-Cultural Communities of Practice“.

37 Detailliert dazu Veith u.a.: „Working in an Inter-Cultural Computer Club“.

3.2 DIE GEBURT VON *SPLATCH*

Zu den jüngsten Aktivitäten des Computerclubs gehörte außerdem eine gemeinschaftliche Fahrt nach Berlin, welche von den Mitgliedern des Computerclubs mit Hilfe von Stadtkarten und Onlineanwendungen, wie beispielsweise *Google Earth*, geplant und von den Teilnehmern der Fahrt durch Kameras, Camcorder etc. festgehalten wurde.[38] Die unzähligen dabei entstanden digitalen Foto-, Video- und Audiomaterialien sollten auf Wunsch der Clubmitglieder in eine permanent archivierbare und gemeinsam nutz- und manipulierbare Form gebracht werden. Von Seiten der wissenschaftlichen Betreuung des Clubs wurde ein dazu geeignetes digitales Framework in Form eines *Scratch*-Skripts vorgestellt (siehe Abb. 2).

Ebene 1: Orientierungslevel *Ebene 2: Ikonenlevel*

Abb. 2: Ebenen 1 und 2 des Splatch-Frameworks/Berlinkarte.

Das Framework, das mittels *Scratch* realisiert wurde, besteht im Wesentlichen aus einer digitalen Stadtkarte von Berlin, ausgeführt in drei Ebenen, versehen mit diversen Navigationsmöglichkeiten. Das Setting *Berlinkarte* wurde bewusst ausgewählt, da alle Teilnehmer sich im Vorfeld der Exkursion ausgiebig mit solchen Karten beschäftigt hatten und so eine gewisse Vorkenntnis und Vertrautheit mit der Darstellung gegeben war. Die User verknüpfen die virtuellen Orte im Framework leicht mit den gemeinsamen Erlebnissen und bekommen durch den ‚Wiedererkennungseffekt' einen Anreiz zur Mitarbeit. Die erste Ebene stellt eine Gesamtansicht des Stadtplans von Berlin dar. Der User kann durch Klick auf ein bestimmtes, bei Mauskontakt namentlich ausgewiesenes Gebiet auf diesem Plan zu Ebene zwei gelangen. Die zweite Ebene gliedert sich in 16 Teilgebiete, welche jeweils einer maßstäblich größeren Ansicht eines Berliner Stadtteiles entsprechen. Auf diesen Karten kennzeichnen kleine grafische Icons die für die Teilnehmer relevanten Orte. Diese können ihrerseits angeklickt werden, um in die unterste, dritte Ebene zu erreichen. Diese dritte Ebene stellt den Raum für die selbst kreierten Projekte der User dar. Im Falle der Berlinfahrt waren relevante Orte

38 Siehe dazu auch Veith/von Rekowski: „Splatch: Scaffolding User Created Content".

beispielsweise der Reichstag oder der Zoo. Gemeinsame Interessengruppen hatten sich schnell gefunden, um zusammen an den Inhalten für die dritte Ebene zu arbeiten. Nachdem alle Dateien gesammelt, ausgelesen, thematisch geordnet und auf einem Server im clubinternen Netzwerk zugänglich gemacht waren, konnten die Teilnehmer unter Zuhilfenahme von schriftlich ausgegebenen Tutorien, das Bildbearbeitungsprogramm *The Gimp* dazu nutzen, aus eigenen digitalen Photos, Charaktersprites für ihre *Scratch*projekte auszuschneiden. Diese wurden im Anschluss meistens in andere Photos eingefügt, mit Sprechblasen und Soundeffekten versehen, mit Hilfe mehrerer Einzelbilder zu bewegten Animationen umgewandelt, und vielen anderen, in *Scratch* implementierten, Manipulationsmöglichkeiten unterzogen. *Scratch* bietet jedoch nicht nur die Möglichkeit, eigene Inhalte in ein bestehendes Framework einzupflegen, sondern bietet dem User auch stets die Möglichkeit zu interagieren und das ursprüngliche Framework auf aktuelle Bedürfnisse anzupassen, um es so für zukünftige Aktivitäten weiterhin nutzen zu können. Bislang ist es allerdings problematisch, größere *Scratch*projekte zu entwickeln, da die Möglichkeiten, mehrere Projekte zu einem zu verbinden, noch sehr eingeschränkt sind. Mit zukünftiger Offenlegung des Quellcodes geht natürlich auch eine bessere Erweiterbarkeit von *Scratch* und damit perspektivisch der *Scratch*projekte einher.

Die Teilnehmer haben diese neue Art und Weise der Schaffung gemeinsamer Artefakte sehr positiv angenommen und im Kontext ihrer Arbeit an den Teilprojekten durch die Anwendung von *Scratch* viele Programmiergrundlagen (selbst erstellte Variablen, Kontrollstrukturen, Objektbegriff etc.) erlernt, sowie ihre Kenntnisse im Umgang mit Bildbearbeitungs- und anderen Programmen erweitert. Dabei standen ihnen wissenschaftliche und studentische Tutoren zur Seite.

4. DISKUSSION

Aus den Erfahrungen mit der Projektarbeit in *Scratch* – bei dem das prototypische Framework *Splatch* entstanden ist – und vorhergegangenen Arbeiten und Projekten mit *Lego Mindstorms* im Computerclub *come_IN* können grob drei analytische Ebenen abgeleitet werden, die sich auf die eine oder andere Art mit den Begrifflichkeiten der Interaktion auseinander setzen: die Ebenen der Inhalte, der Gestaltung und der Metaphern.

Interaktion im Kontext unseres Forschungsfeldes wiederum kann in drei spezifischen Qualitäten beobachtet und beeinflusst werden. Erstens ist hier die interpersonale Qualität von Interaktion zu nennen: die Kooperation zwischen Menschen bei der Bearbeitung von *Scratch*-Projekten. Diese wird besonders durch die Lehrer und Tutoren im Club beobachtet, diskutiert und irritiert. Eine technische Unterstützung findet in diesem Bereich bisher nicht gesondert statt und scheint nach unseren bisherigen Erkenntnissen für die typischen Gruppengröße in unserem Feld nicht erforderlich. Bei einer weiteren Öffnung des Clubs hin zum Stadtteil und einer Vernetzung der *come_IN*-Clubs über verschiedene Standorte

kann jedoch eine gezielte Kommunikations- und Interaktionsunterstützung notwendig werden. In der softwareergonomischen Forschung würde man dann von einer Mensch-Maschine-Mensch-Interaktion sprechen. Eine zweite Qualitätsdimension besteht in der klassischen Interaktion von Mensch und Maschine. Hier sind insbesondere gestaltungspsychologische und lerntheoretische Fragestellungen von gesondertem Interesse. Als dritte Qualitätsdimension kann in unserem speziellen Feld noch so etwas wie eine Gruppen-Maschinen-Interaktion beobachtet werden, welche über den klassischen Interaktionsbegriff aus der HCI hinausgeht (klassisch sind die Betrachtungsweisen Mensch-Maschine-Interaktion und Mensch-Maschine-Mensch-Interaktion, wobei jeweils von Einzelpersonen ausgegangen wird, die mit Technik interagieren und/oder als Mediatoren zur inter-personalen Interaktion nutzen). Konsumption wie auch das Teilen von medialen Artefakten geschehen hier, auch wird die Entwicklung der Artefakte in dieser Form der Interaktion ausgehandelt (während die eigentliche Konstruktion von Artefakten eher bei der Mensch-Maschinen-Interaktion zu finden ist).

4.1 SPIELERISCHE (DE-)KONSTRUKTION

Neben der intendierten spielerischen Aneignung von rudimentären Programmierkenntnissen (z.B. dem generellen Wissen ob der Veränderbarkeit von Programmabläufen) durch *Lego Mindstorms* und dem *Splatch*-Projekt kommt es im Computerclub regelmäßig zu Spielsessions mit *Trackmania*[39]. Die Kinder, vorwiegend Jungen, versuchen sich gegenseitig bei diesem Autorennspiel bei den gefahrenen Bestzeiten zu übertreffen. Wirkt dieses Spiel für die Eltern und Betreuer des Clubs eher wie ein Pausenfüller, so gehen die Kinder regelrecht auf in dieser Tätigkeit gegen andere und den Rechner gewinnen zu wollen. Im Gegensatz dazu mutet die Möglichkeit den *Lego*-Roboter, der mit Freuden zusammengesteckt wurde, am Rechner zu programmieren wie ein langweiliger Aufwand an, Eltern und Betreuer zufrieden zu stellen. Splatch überwindet dieses Problem, indem es den Kindern (und hier sind die Mädchen genauso angesprochen wie die Jungen) neben der Programmierbarkeit der Elemente eben auch die spielerische Irritation, ja, Dekonstruktion aus dem bestehenden sozialen Kontext heraus, ermöglicht. So ist es mit einem Mal möglich, Fallschirmspringer durch den Reichstag schweben zu lassen und einen Avatar, ein Abbild seiner selbst, das Brandenburger Tor erklimmen zu lassen. Neben sozialen Konventionen werden dort spielerisch auch Naturgesetze ausgehebelt, passive Konsumption paart sich mit aktiver Gestaltung zur Entwicklung durch den Endnutzer.

39 *Trackmania* (Nations) ist ein französisches, im Basispacket kostenloses 3D-Autorennspiel, welches auch eine Gestaltung von eigenen Rennstrecken zulässt. Siehe Homepage des Projekts: http://www.trackmania-the-game.de:8080/tmu/, 04.02.2008.

4.2 VISUELLE PROGRAMMIERUNG FÜR NOVIZEN

Das im oberen Abschnitt erwähnte *Trackmania* verfügt über die Möglichkeit, eigene Strecken zu bauen, auf denen man dann Rennen gegen andere Spieler oder den Computer fahren kann. Diese Funktion wird allerdings in *come_IN* nicht (mehr) genutzt, die Kinder stempeln es als zu zeitaufwendig und langweilig ab. Während der Phase des Konstruierens kann nicht gespielt werden, eine Veränderung der Kurse zur Laufzeit ist nicht möglich. Auch können bestehende Kurse nicht dekonstruiert werden, um dann an individuelle Wünsche angepasst zu werden; ein Kurs muss von Grund auf neu konzipiert werden.

Bei *Lego Mindstorms* ist der Sachverhalt ähnlich gelagert: auch hier empfinden die Kinder einen zu starken Bruch zwischen der spielerischen Phase (hier dem physikalischen Zusammenstecken von Legoteilen) und der Phase der Programmierung, die auf virtueller Ebene stattfindet (der Programmcode wird dann über eine Schnittstelle auf den Rechner des Roboters gespielt). Auch hier ist also keine Veränderbarkeit zur Laufzeit (die auch hier die Spielzeit ist) möglich. Ferner fehlt den Grundschülern in den meisten Fällen noch das Verständnis für komplexere Programmierspezifika wie Schleifen und Variablen, die für eine erfolgreiche Steuerung des Roboters jedoch sehr wichtig sind.

Scratch/*Splatch* hingegen bietet den Mitgliedern von *come_IN* die Möglichkeit während der aktiven Nutzung des Programms Elemente davon zu verändern oder gar den Programmablauf zu beeinflussen. Das hat den Vorteil, dass a) die Veränderungen in der Entwicklung unmittelbar erfahrbar werden, b) gleichzeitig so Konstruieren aber auch Teil des Spiels wird. Weiterhin bestehen bleibt jedoch das Problem, dass komplexere Programmieraufgaben von den Kindern alleine nicht bewerkstelligt werden können, da das formal-logische Denken in dieser Altersklasse noch nicht hinreichend ausgeprägt ist. Mit Hilfe der Eltern und Betreuer jedoch gelingt es im Club immer wieder auch komplexere Programmabläufe visuell zusammen zu stellen. Hier ist zu beobachten, dass sich die Eltern – im Gegensatz *zu Lego Mindstorms* und *Trackmania* – stärker abgeholt fühlen, d.h. ihre eigenen Interessen vertreten sehen, wenn ihre Kinder mit Inhalten spielen, die sie selbst mit gesammelt und bearbeitet haben.

Splatch ist dabei als anpassbares System konzipiert, dass auf drei spezifischen, voneinander unterscheidbaren Layern wesentlich differenzierbare Anpassungen durch den Endnutzer zulässt. Auf Level 3, der spielerisch narrativen Ebene geschieht ein freies Gestalten durch den Nutzer, übernommen wurden bisher stets die Konzepte *Hintergrund* und *Held*, hier geschieht eine Anpassung bisher nur auf dem Niveau des Austauschs von Kostümen; Hintergrund und Held verändern also nur ihr Erscheinungsbild. Auf Level 2 ist bisher durch Endnutzer in come_IN noch keine Veränderung selbst vorgenommen worden, jedoch wurden Ikonen von anderer Seite erstellt, die dann von Betreuern verändert, hinzugefügt oder entfernt wurden. Die Einstiegs-Kartenmetapher mit ihrem *MouseOver*-Effekt auf Level 1 von *Splatch* wurde bisher noch gar nicht angepasst sondern dient stets als Orientierungsanker und Startpunkt für die Interaktion mit dem Programm. Diese Ebene

dient als Einstiegsbildschirm in ein konkretes Projekt und so ist für ein neues Bonn-Projekt im Club geplant, eine Karte von der Bonner Altstadt zu entwickeln, die dort eingefügt wird.

4.3 DIE BAUSTEIN-METAPHER IM KOLLABORATIVEN EINSATZ

Wie bereits oben erwähnt dienen zwei ganz grundlegende Metaphern in unserer Arbeit dazu, Software assoziierbar und damit anpassbar zu machen. Zum einen ist da die Baustein-Metapher zu nennen, die im Wesentlichen unsere Konstruktionsmetapher ist. Elemente werden durch Bausteine konstruiert, anpassbare Elemente werden so auch als bausteinartig wahrgenommen. Karte und in seiner konkreten Ausprägung als Raum wiederum dienen uns als Rahmungsmetaphern, die als Orientierung genutzt werden, um eigene Inhalte zu lokalisieren und einzubetten. Eine weitere, bisher unerwähnt gebliebene Metapher, ist das Nutzerkonzept des *Ichs*, des Selbst. In *Splatch* wird dieses Ich durch die Kinder sehr häufig in der Realisation eines Avatars ausgedrückt. Dies können konkrete Fotoausschnitte ihrer Person sein oder auch Grafiken, die sich entweder selbst erstellt oder im Internet entdeckt haben. Dieser Avatar wird auch stets sehr selbstbewusst als *Ich* referenziert. Treten mehrere Kinder in *Splatch* als Avatar auf, sprechen sie von *wir* und kommunizieren auch entsprechen darüber. Dies ist insbesondere von größtem Interesse, wenn es um die Gestaltung der Anpassung der Software geht: Dort werden *Ichs* und *Wirs* verändert und vor einem Hintergrund bewegt und automatisiert. *Splatch* liefert eine Qualität der Identifizierung mit dem (Lern-)Gegenstand, den *Lego Mindstorms* (in Form des Roboters) oder *Trackmania* (in Form eines allmächtigen Streckenbauers) nicht leisten können[40].

5. ZUSAMMENFASSUNG

Der aktuelle Medienumbruch ist durch die Ergänzung des *easy-to-use* durch das *easy-to-design*-Prinzip gekennzeichnet. Während Nutzer bisher eher mit einer fest implementierten Anwendung interagiert haben, wird diese Interaktion durch eine Metainteraktion im Sinne der Gestaltung von Anwendungsbaukästen erweitert. Nutzer werden befähigt, die technischen Artefakte zu modifizieren, mit denen sie während der normalen Nutzung interagieren.

Sowohl Interaktion als auch Metainteraktion sind kollektive Phänomen, insofern als Nutzer dabei immer sozial eingebettet agieren. Im Rahmen des Computer-Clubs *come_IN* untersuchen wir, welche Effekte von Interaktion und Metainteraktion auf die interkulturellen Beziehungen von Nutzer ausgehen können. Wir haben dazu die Anwendungsbaukästen *Lego Mindstorm* und *Scratch* als techni-

40 Papert hatte schon in LOGO das Design-Prinzip der ‚Autochthonie' eingeführt. Die Kinder sollten sich in die Schildkröte hineinversetzen können, um sowohl die Distanz als auch die Furcht vor dem Lerngegenstand abzubauen (vgl. Papert: Gedankenblitze, S. 14).

sche Basis zur interkulturellen Projektarbeit erprobt. Dabei haben wir Erkenntnisse sowohl bezüglich der Ausgestaltung von Anwendungsbaukästen gewonnen als auch untersuchen können, welche Effekte Interaktion und Metainteraktion auf die sozialen Beziehungen und Identitäten deutsch- und türkischstämmiger Familien haben.

DANKSAGUNG

Wir danken allen Beteiligten des Projektes, insbesondere den Teilnehmern und Teilnehmerinnen des Clubs. Der Computerclub *come_IN* wird begleitet von einem Forschungsprojekt, gefördert vom deutschen Bundesministerium für Bildung und Forschung (BMBF, Fkz: KB00905). *Die Gestaltung von Computersystemen durch den Nutzer – Exemplarische Analyse einer medientechnologischen Herausforderung* ist ein Teilprojekt des Kulturwissenschaftlichen Forschungskollegs *Medienumbrüche – Medienkulturen und Medienästhetik zu Beginn des 20. Jahrhunderts und im Übergang zum 21. Jahrhundert* der Universität Siegen und wird gefördert von der Deutschen Forschungsgesellschaft (DFG).

LITERATURVERZEICHNIS

Bentley, Richard/Dourish, Paul: „Medium Versus Mechanism: Supporting Collaboration through Customisation“, in: Proceedings of the Fourth European Conference on Computer-Supported Cooperative Work (ECSCW) 1995, S. 133-148.

Blackwell, Alan Frank: Metaphor in Diagrams, University of Cambridge 1998 (Diss.).

Boyd, Richard: „Metaphor and Theory Change: What is ‚Metaphor‘ a Metaphor for?“, in: Or- tony, Andrew (Hrsg.): Metaphor and Thought, Cambridge 1993, S. 481-532.

Cottmann, Kathrin: Wie verstehen Kinder Maschinen und Computer? Eine empirische Studie mit Konsequenzen für Pädagogik und Softwareentwicklung. München 1998.

Dijkstra, Edsger Wybe: „On the Cruelty of the Really Teaching Computing Science“, in: Communication of the ACM, Bd. 32, Nr. 12, 1989, S. 1398-1404.

Fischer, Gerhard u.a.: „Seeding, Evolutionary Growth and Reseeding: The Incremental Development of Collaborative Design Environments“, in: Olson, Gary M. u.a. (Hrsg.): Coordination Theory and Collaboration Technology, Mahwah, NJ 2001, S. 447-472.

Fischer, Gerhard u.a.: „Fostering Social Creativity by Increasing Social Capital“, in: Wulf, Volker/Huysman, Marleen (Hrsg.): Social Capital and Information Technology, Cambridge, MA 2004, S. 355-400.

Fromme, Johannes u.a.: Computerspiele in der Kinderkultur, Opladen 2000.

Gantt, Michelle/Nardi, Bonnie A.: „Gardeners and Gurus: Patterns of Cooperation Among CAD Users“, in: Proceedings of the Conference on Human Factors in Computing Systems (CHI) 1992, S. 107-117.

Kahler, Helge: Supporting Collaborative Tailoring, University of Roskilde 2001 (Diss.).

Kahler, Helge u.a.: „Tailorable Systems and Cooperative Work – Introduction“, in: Special Issue of Computer Supported Cooperative Work: The Journal of Collaborative Computing 2000, S. 1-4.

Lakoff, George: „The Contemporary Theory of Metaphor“, in: Ortony, Andrew (Hrsg.): Metaphor and Thought, Cambridge 1993, S. 202-251.

Lakoff, George/Johnson, Mark: Metaphors We Live By, Chicago 1980.

Lieberman, Henry (Hrsg.): Your Wish is My Command: Programming by Example, San Francisco 2001.

Lieberman, Henry u.a. (Hrsg.): End User Development, Springer, London 2006.

Lippert, Martin u.a.: Software entwickeln mit eXtreme Programming: Erfahrungen aus der Praxis, Heidelberg 2001.

Mackay, Wendy: Users and Customizable Software: A Co-Adaptive Phenomenon, Massachusetts Institute of Technology, Boston 1990 (Diss.).

MacLean, Allan u.a.: „User-Tailorable Systems: Pressing the Issue with Buttons“, in: Proceedings of the Conference on Computer Human Interaction (CHI) 1990, S. 175-182.

Malone, Thomas W. u.a.: „Experiments with Oval: A Radically Tailorable Tool for Cooperative Work“, in: ACM Transactions on Information Systems, 13, 2 (1996), S. 175-205.

Martin, Fred u.a.: „To Mindstorms and Beyond: Evolution of a Construction Kit for Magical Machines“, in: Druin, Allison (Hrsg.): Robots for Kids. Exploring New Technologies for Learning Experiences, San Francisco 2000.

Martin, Fred u.a.: „MetaCricket: A Designer's Kit for Making Computational Devices“, in: IBM Systems Journal Bd. 39, Nr. 3-4, 2000, S. 795-815.

Mayer, Richard E.: „Different Problem-Solving Competencies Established in Learning Computer Programming With and Without Meaningful Models“, in: Journal of Educational Psychology, Bd. 67, Nr. 6, 1975, S. 725-734.

Meder, Norbert/Fromme, Johannes: „Computerspiele und Bildung. Zur theoretischen Einführung“, in: Fromme, Johannes/Meder, Norbert (Hrsg.): Bildung und Computerspiele, Opladen 2001, S. 11-28.

Nardi, Bonnie A.: A Small Matter of Programming – Perspectives on End User Computing, Cambridge u.a. 1993.

Nardi, Bonnie A./Miller, James R.: „Twinkling Lights and Nested Loops: Distributed Problem Solving and Spreadsheet Development", in: International Journal of Man Machine Studies, Bd. 34, Nr. 2, 1991, S. 161-184.

Oerter, Rolf: „Spiel", in: Lexikon der Psychologie in fünf Bänden, Heidelberg/Berlin 2001, S. 216-218.

Oppermann, Reinhard/Simm, Helmut: „Adaptability: User-Initiated Individualization", in: Oppermann, Reinhard (Hrsg.): Adaptive User Support – Ergonomic Design of Manually and Automatically Adaptable Software, Hillsdale, NJ 1994, S. 14-66.

Papert, Seymour: Gedankenblitze. Kinder, Computer und Neues Lernen, Hamburg 1985.

Reddy, Michael: „The Conduit Metaphor: A Case of Frame Conflict in Our Language About Language", in: Ortony, Andrew (Hrsg.): Metaphor and Thought, Cambridge 1993, S. 164-201.

Reichling, Timm u.a.: „Expert Recommender: Designing for a Network Organization", in: Computer Supported Cooperative Work: The Journal of Collaborative Computing (JCSCW), Bd. 16, Nr. 4-5, 2007, S. 431-465.

Resnick, Mitchel: „Behavior Construction Kits", in: Communications of the ACM, Bd. 36, Nr. 7, 1993, S. 64-71.

Resnick, Mitchel u.a.: „Digital Manipulatives: New Toys to think With", in: Proceedings of the Conference on Human Factors in Computing Systems (CHI) 1998, S. 281-287.

Resnick, Mitchel/Ocko, Stephen: „LEGO-Logo: Learning Through and about Design", in: Papert, Seymour (Hrsg.): Constructionism, Norwood 1991, S. 141-150.

Riemann, Sabine: „Spiel, Spieltheorien in der Pädagogik", in: Reinhold, Gerd (Hrsg.): Pädagogik-Lexikon, München/Wien 1999, S. 490-493.

Slagter, Robert J./Ter Hofte, G. Henri: End-User Composition of Groupware Behaviour: The CoCoWare. NET Architecture (Technical Report), Enschede 2002.

Stevens, S. u.a.: „Bridging among Ethnic Communities by Cross-Cultural Communities of Practice", in: Proceedings of the Second International Conference on Communities and Technologies (C&T 2005), Dordrecht 2005, S. 377-396.

Sweetser, E.: From Etymology to Pragmatics: Metaphorical and Cultural Aspects of Semantic Structure, Cambrigde 1990.

Teege, Gunnar: Individuelle Groupware: Gestaltung durch Endbenutzer, Wiesbaden 1998.

Travers, Michael David: Programming with Agents: New Metaphors for Thinking about Computation, Massachusetts Institute of Technology, Cambridge 1996 (Diss.).

Trigg, Randall H./Bødker, Susanne: „From Implementation to Design: Tailoring and the Emergence of Systematization in CSCW“, in: Proceedings of the Conference on Computer Supported Cooperative Work (CSCW) 1994, S. 45-54.

Vivacque, Adriana/Lieberman, Henry: „Agents to Assist in Finding Help“, in: Proceedings in the Conference on Human Computer Interaction (CHI 2000), New York 2000, S. 65-72.

Veith, Michael/von Rekowski, Thomas.: „Splatch: Scaffolding User Created Content“, in: IADIS International Conference e-Society 2008, Algarve, Portugal 2008 (im Druck).

Veith, Michael u.a.: „Working in an Inter-Cultural Computer Club: Effects on Identity and Role Affilation“, in: IADIS International Journal on WWW / Internet, Jg. 5, Nr. 2, 2007, S. 100-112.

Wulf, Volker u.a.: „Component-Based Tailorability: Enabling Highly Flexible Software Applications“, in: International Journal on Human-Computer Studies (IJHCS), Bd. 66. Nr. 1, 2008, S. 1-22.

Ye, Yunwen: Supporting Component-Based Software Development with Active Component Repository Systems, University of Colorado, Boulder, 2001 (Diss.).