e-Prototyping

Wolf-Gideon Bleek, Martti Jeenicke, Ralf Klischewski

Universität Hamburg, Fachbereich Informatik/Softwaretechnik
Vogt-Kölln-Str. 30, D-22527 Hamburg
{bleek, jeenicke, klischewski}@informatik.uni-hamburg.de

1. Einleitung

Prototyping hat mittlerweile einen festen Platz im Repertoire der anwendungsorientierten Softwareentwicklung. Es gilt in vielen Projekten als Mittel der Wahl, um iterativ den Anforderungen eines spezifischen und einer vorausschauenden Anforderungsanalyse schwer zugänglichen Anwendungskontextes zu begegnen. Jedoch: „traditionelles" Prototyping geht von Voraussetzungen aus (u.a. hinsichtlich Zeitrahmen, Beteiligte, Kommunikation, Kontrollierbarkeit, Relevanz der Anwendung), die bei der Entwicklung von Lösungen für e-Business, e-Commerce und e-Government nicht oder nur eingeschränkt gegeben sind. Ist damit das Ende des Prototyping eingeläutet? Wird diese Vorgehensweise in der Welt der vernetzten Systeme Bestand haben (können)? Und falls ja, was sind die erforderlichen Anpassungen beim Projektdesign? Brauchen wir ein „e-Prototyping"?

Die Frage nach der Rolle des Prototyping im e-Zeitalter bettet sich ein in die Diskussion um Vorgehensweisen, die mittlerweile häufig mit dem Stichwort Web-Engineering verbunden wird. Die Promotoren des Web-Engineering verstehen darunter eine neue Disziplin, die für einen Prozess und eine systematische Herangehensweise an die Entwicklung hochwertiger web-basierter Systeme eintritt (vgl. Murugesan und Deshpande 2001). In Abgrenzung zum „klassischen" Software Engineering wird z.B. immer wieder darauf hingewiesen, dass die meisten web-basierten Systeme dokumentenorientiert (statisch oder dynamisch) sind mit der Betonung von visueller Kreativität und Präsentation an der Benutzungsschnittstelle, die einer Vielzahl von Benutzerprofilen gerecht werden soll. Die Entwicklungszeiten sind einerseits vergleichsweise kurz und erlauben kaum gründliches Planen und Testen, andererseits werden Web-Projekte in der Regel ohne zeitliches Ende betrieben. Systematisches Entwickeln wird noch erschwert durch Mangel an sicherem Wissen über die Systemkonfiguration beim Web-Anwender sowie die Heterogenität der Qualifikationen und der Sichten auf Qualität von den an der Entwicklung beteiligten Individuen.

In diesem Beitrag rufen wir uns zunächst die Grundannahmen des in den 80er und 90er Jahren geprägten Prototyping hinsichtlich Motivation, Anwendungsbereiche, Vorgehensweisen in Erinnerung (siehe Abschnitt 2). Kontrastiv dazu untersuchen wir die neuen Bedingungen, Schwierigkeiten und Herausforderungen bei der Einbeziehung von und Kommunikation mit den relevanten Akteuren, mit deren Beteiligung man Klarheit über Anforderungen an die jeweilige anwendungsorientierte Softwareentwicklung für e-Lösungen zu gewinnen sucht (Abschnitt 3). Darauf aufbauend skizzieren wir einen Ansatz für das e-Prototyping (Abschnitt 4): Da sich mit dem Internet keine soziale Laborsituation für das Prototyping herbeiführen lässt, läuft jedes Release bzw. jede veröffentlichte Version als Produktivsystem – beim Management von Web-Engineering Prozessen sollten deshalb kurze Entwicklungszyklen mit häufigen, jeweils auch innovativen Versions-Releases angestrebt und Rückmeldungen von deren Benutzern (bzw. anderen relevanten Akteuren) über verschieden Kommunikationskanäle eingeholt werden. Diese Kommunikationskanäle bedürfen der besonderen Aufmerksamkeit, um aus dem

Strom der Rückmeldungen systematisch Anforderungen für die evolutionäre Softwareentwicklung und das Release-Management gewinnen zu können.

2. Prototyping in der anwendungsorientierten Softwareentwicklung

Der Terminus Prototyp kommt aus dem Griechischem und kann wörtlich übersetzt werden als „der erste seiner Art". Nach (Brockhaus 1992) ist ein Prototyp die „erste betriebsfertige Ausfertigung eines Geräts, einer Maschine, z.B. Rennwagen im Automobilsport o.ä., der die Nullserie folgt". Diese Definition wird dem Umstand gerecht, dass Prototypen lange Zeit vor allem in elektrischen und mechanischen Disziplinen verwendet wurden. Hier bezeichnet der Begriff die erste Version eines Produktes in Originalgröße, z.B. ein Auto oder Flugzeug, an dem die Eigenschaften und die Qualitäten des späteren Produkts getestet und Erfahrungen für die Serienproduktion gesammelt werden können. Der Prototyp ist folglich Teil eines Lernprozesses. Prototyping, die Tätigkeit der Erstellung eines Prototyps, stellt in diesem Zusammenhang eine wohldefinierte Phase des Entwicklungsprozesses dar.

Im Kontext der Softwaretechnik hat der Begriff Prototyp bzw. Prototyping allerdings nicht strikt dieselbe Bedeutung, da es einige Unterschiede zwischen den Disziplinen gibt. So dienen Prototypen in dieser Fachrichtung niemals der Vorbereitung der Serienproduktion, da es diese im eigentlichen Sinne nicht gibt. Vielmehr ist i.d.R. das Ergebnis der Softwareentwicklung, die häufig zusätzlich in eine übergeordnete Systementwicklung eingebettet ist, ein einzelnes Produkt. Diese Tatsache relativiert die ursprüngliche Bedeutung von „erster seiner Art". Außerdem ist nach (Floyd 1984) in der Softwareentwicklung nicht klar, in welchem Zusammenhang der Prototyp zum späteren Produkt steht.

Nach (Sommerville 1996, S. 138-153) ist Prototyping im Software Engineering hauptsächlich eine Methode zur Anforderungsermittlung und –verifikation. Es unterstützt den Kommunikationsprozess zwischen Entwicklern und Benutzern. Mit Hilfe von Prototyping kann den Benutzern eines Systems dessen zukünftige Arbeitsweise demonstriert werden. Gleichzeitig ermöglicht es ein „Experimentieren" mit den möglichen Anforderungen (ebd. S.138). Sommerville unterscheidet zwei Arten von Prototyping: evolutionäres und „Wegwerf" Prototyping. Ziel der evolutionären Vorgehensweise ist es, am Ende aller Entwicklungstätigkeiten ein lauffähiges System zu haben. Charakteristisch dafür ist kontinuierliche Veränderung die durch einen iterativer Prozess erreicht wird, in dem zunächst die von den Entwicklern die am besten verstandenen Anforderungen implementiert werden, bevor die weniger verständlichen Benutzerwünsche angegangen werden. Um dies erfolgreich durchführen zu können bedarf es Techniken, die schnelle System-Iterationen erlauben (ebd. S.142). Allerdings wird evolutionäres Prototyping nach Sommervilles Sicht nur in einigen Fällen der Softwareentwicklung verwendet, in denen es am Anfang der Entwicklung nicht möglich ist eine komplette Systemspezifikation zu erstellen.

Im Gegensatz dazu verstehen (Budde et al.1992), ebenso wie wir, unter Prototyping in der Softwareentwicklung einen (ganzheitlichen) Ansatz, der auf einer evolutionären Sichtweise von Softwareentwicklung basiert und Auswirkungen auf den gesamten Entwicklungsprozess hat. Es umfasst sowohl die Erstellung, als auch das Experimentieren mit frühen Versionen (den Prototypen) eines zukünftigen Anwendungssystems. So ermöglicht es eine erfahrungsbasierte Konstruktion von Software. Prototyping dient als eine Kommunikationsbasis für alle am Entwicklungsprozess beteiligten Personen. Dabei ist ein Prototyp ein wichtiges Artefakt und eine Quelle der Erkenntnis innerhalb eines kontinuierlichen Lernprozesses. Aus diesem Grunde glauben wir, dass der Terminus „Wegwerf-Prototyp", wie er von (Sommerville 1996) verwendet wird, ungeeignet ist, auch wenn er so in vielen Projekten verwendet wird (z.B. für kleine Prototypen, die zur Verifikation von Anforderungen verwendet werden). Wir orientieren uns daher im folgenden an dem eher europäisch geprägtem Verständnis von Prototyping (siehe etwa Budde et al.1992 und Floyd 1984).

Der Prozess der Erstellung eines Prototyps besteht nach (Floyd 1984) aus vier Schritten:

- **Auswahl der Funktionen***:* Im Rahmen der Funktionsauswahl muss die Entscheidung getroffen werden, ob der Prototyp eher viele Funktionen, die allerdings nicht detailliert ausgestaltet werden, („horizontales Prototyping“) oder aber wenige, detaillierte Funktionen („vertikales Prototyping“) enthalten soll.

- **Konstruktion***:* Der Schritt Konstruktion umfasst (ebd.) alle Anstrengungen, um den Prototypen verfügbar zu machen. Da der Prototyp den Lernprozess der an der Entwicklung Beteiligten unterstützen soll, liegt der Fokus bei der Konstruktion nicht auf allen Qualitätsanforderungen, die an das spätere System gestellt werden.

- **Evaluation***:* Nachdem der Prototyp konstruiert ist, wird er einer Evaluation unterzogen. Dies ist der entscheidende Schritt beim Prototyping. Hierbei werden Informationen gesammelt und Erfahrungen gemacht, die für das spätere Produkt von großem Nutzen sind. Daher sollte er sorgfältig geplant und durchgeführt werden.

- **Weitere Verwendung***:* An die Evaluation schließt sich die Entscheidung über die spätere Verwendung an. Es ergeben sich dabei zwei, von den gemachten Erfahrungen abhängige, Möglichkeiten. Zum einen kann der entstandene Prototyp als wichtiges Lernmedium angesehen werden und anschließend nicht weiter verwendet werden. Häufig werden diese Prototypen als Wegwerfprototypen bezeichnet. Als zweite Möglichkeit kann der Prototyp komplett oder auch nur in Teilen in das spätere Produkt einfließen.

Prototyping lässt sich nach den Zielen, mit denen es im Entwicklungsprozess eingesetzt wird, klassifizieren (ebd.). **Exploratives Prototyping** wird immer dann verwendet, wenn die Problemstellung unklar ist. Die Anforderungen der Benutzer und des Managements werden durch Prototyping geklärt, wobei die Entwickler die Anwendungsdomäne sowie die Aufgaben der Benutzer kennen. **Experimentelles Prototyping** hingegen wird dann eingesetzt, wenn die technische Umsetzung der Anforderungen geklärt werden soll. Dabei probieren die Entwickler einige ihrer Ideen aus, um so einen Eindruck über die technische Umsetzbarkeit und die Zweckmäßigkeit des Anwendungssystems zu erhalten. **Evolutionäres Prototyping** ist ein kontinuierliches Verfahren, in dem die Software evolutionär entwickelt wird. Dabei können die Entwickler das Anwendungssystem schnell an sich ändernde Randbedingungen anpassen.

Ebenfalls nach den mit ihrem Einsatz verfolgten Zielen und den Zielgruppen lassen sich auch Prototypenarten klassifizieren (Kieback et al. 1992):

- **Demonstrationsprototyp**: Er soll insbesondere die Projektinitiierung unterstützen, indem er dem Auftragsgeber zeigt, wie ein Anwendungssystem prinzipiell aussehen kann. Zusätzlich sollen diese Art von Prototypen eine Vorstellung der Handhabung und des Umfangs eines Anwendungssystems im Einsatzkontext geben.

- **Funktioneller Prototyp**: Dieser präsentiert Teile der Benutzungsschnittstelle sowie einen Ausschnitt der Funktionalität, wobei er meistens bereits den technischen Aufbau, also die Architektur, des zu entwickelnden Systems besitzt. Funktionelle Prototypen werden i.d.R. von allen an der Entwicklung beteiligten Gruppen bewertet, um die Problemstellung zu klären oder aber Fragen des Entwurfs zu beantworten.

- **Oberflächenprototyp**: Mit diesem Prototyp soll nur die Handhabung des späteren Systems bewertet und getestet werden, wobei keinerlei Funktionalität implementiert wird. Für die Klärung von softwaretechnischen Fragen werden von den Entwicklern bewertete Prototypen, sogenannte Labormuster verwendet.

- **Pilotsystem**: Diese zeichnen sich dadurch aus, dass sie im Anwendungsbereich eingesetzt werden. Dadurch bilden sie das Kernsystem der späteren Anwendung, das sicher, sprich fehlerfrei, funktioniert und evolutionär erweitert wird.

In der Praxis finden sich vielfältige Mischformen der oben genannten Konzepte, und vielfach würden Analytiker Prototyping beobachten können, wenngleich sich manche Akteure dessen nicht unbedingt bewusst sind.

Allen Prototypen gemeinsam ist ihr Wert als Diskussionsgegenstand. Der Softwareentwicklungsprozess gewinnt die Möglichkeit, Grenzen, Erwartungen und fachliche sowie technische Inhalte am Gegenstand zu diskutieren. Dies unterstützt einen Lernprozess zwischen den Beteiligten und hilft, Missverständnisse früh auszuräumen. Letztlich sind es die vielfachen Erfahrungen der Verbesserung von Qualität und der Reduktion von Aufwand insgesamt, die dem Prototyping zu einen anerkannten Platz in der Methodik der anwendungsorientierten Softwareentwicklung verholfen haben.

Neue Herausforderungen für Softwareentwicklung sind nun durch die Bedingungen gegeben, die Entwickler in Web-Projekten vorfinden. Die in der Praxis deutlich vorherrschende ad-hoc Vorgehensweise wirft neue Fragen über die Rolle von Prototyping als ein methodisches Konzept innerhalb des Web-Engineerings auf.

3. Beteiligung der relevanten Akteure

Prototyping ist eine Möglichkeit, andere Personen als die Software-Entwickler in den Entwicklungsprozess einzubeziehen, mit dem Ziel die Produktentwicklung zu verbessern. In diesem Abschnitt werden wir die Bedingungen, Schwierigkeiten und Herausforderungen bei der Einbeziehung von Akteuren und der damit verbundenen Kommunikation beschreiben. Dies dient der Anforderungsermittlung im Web-Engineering. Zuerst werten wir unsere Erfahrungen aus Praxisprojekten aus, an denen wir beteiligt waren:

- **Fallbeispiel 1: Das „CommSy"-Projekt**
  CommSy ist ein web-basiertes *Community System* zur Unterstützung der Kommunikation und Koordination in Lern- und Arbeitsgruppen (Bleek et al. 2000, www.commsy.de). In den Projekten, in denen das System eingesetzt wurde, haben die Anwender sehr schnell das bereitgestellte Medium selbst für Feedback während des Einsatzes benutzt. Dies war notwendig, da eine direkte Kommunikation zu den Entwicklern wegen der vielen Einsatzkontexte nicht möglich war. Dazu wurden elektronische Diskussionsforen angeboten, die sehr stark für technisches und fachliches Feedback genutzt wurden (einige Nutzer machten dort auch die Notwendigkeit eines Workshops deutlich). Motivierte und sehr aktive Anwender haben sich selbst organisiert und erreichten so maßgeblichen Einfluss auf den Entwicklungsprozess. Wir haben bei der Auswertung der unterschiedlichen Kommunikationskanäle Schwierigkeiten festgestellt. Da die Nutzer entweder ihre Mitteilungen in eines der Diskussionsforen schrieben oder per E-Mail direkt an die Entwickler sendeten, war es für diese schwierig, den Überblick zu behalten (alle Entwickler hatten ihre eigene E-Mail Adresse und es gab bis zu 10 Foren). Ein zusätzliche Schwierigkeit war es, aus dem Feedback Design-Entscheidungen abzuleiten und diese nach Wichtigkeit einzuordnen. Wir haben uns deshalb dazu entschlossen, „story cards" zu verwenden, die aus dem Extreme Programming (s.u.) bekannt sind. Diese Karten wurden in der „Architekturgruppe" diskutiert. Eine andere Schwierigkeit bestand in der einseitigen Richtung der Feedback Kommunikation: Die Nutzer bekamen meist keine Mitteilung darüber, ob die Anmerkungen wahrgenommen wurden und zu einer Entwicklungsentscheidung geführt haben. Darüber hinaus wurde nicht bekannt wann die Anpassungen realisiert wurden und in welcher Version diese nutzbar werden würden. Diese offenen Punkte benötigen in der Zukunft mehr Aufmerksamkeit.

- **Fallbeispiel 2: Das Stadtinformationssystem**
  Für die Realisierung eines städtischen Informationssystems (Bleek 2001) wurde ein „Big-Bang" Vorgehen gewählt, d.h. es wurde etwa 1¼ Jahre Entwicklung ohne ein Release betrieben. Dieses Vorgehen scheiterte noch bevor Ergebnisse der Öffentlichkeit präsentiert werden konnten, weil eine ganze Reihe von Problemen (Aneignung von Technologie, Performance, Verbindung und Zusammenarbeit mit anderen Systemen, Missverständnisse, Interessenkonflikte) nicht bewältigt werden konnte. In einem zweiten Anlauf konnten nur drastische Änderungen das Projekt retten: kurze Entwicklungszyklen und eine Entwicklung im kleinen Team gepaart mit einer abgeschlossenen kleinen Funktionalität machte eine kurze „Time to Release/Time to Market" Periode möglich. Allerdings gab es trotzdem keine systematische Sammlung und Auswertung von Kommunikation mit den Anwendern (z.B. Beschwerden über E-Mail oder über das Call Center). Gleichzeitig ist die Planung von Folgeversionen ausschließlich durch Management Entscheidungen vorangetrieben worden und wird nicht nach außen kommuniziert. Dies und der fehlende Kommunikationskanal für die Anwender (die einzige Möglichkeit Anmerkungen zu machen, besteht in einem elektronischen Gästebuch) macht die unterbewertete Rolle der Anwender in diesem Entwicklungsprozess deutlich und führt erwartungsgemäß zu wiederholten Enttäuschungen im Bereich von Stadtinformationssystemen.

Probleme im Bereich Web-Engineering

Unsere Auswertung der Fallbeispiele zeigt, dass die gleichen Fragen wiederholt auftreten und zu wesentlichen Problemstellungen fokussiert werden können:

- Wie können die (initialen) Anforderungen für netzbasierte Web-Applikationen definiert werden?
- Wie können Anforderungen systematisch gesammelt werden, wenn die Anwendergruppe (die Web-Nutzer) unbekannt und in ihren Charakteristika schwer zu beschreiben sind?
- Welche Akteure sollten in den Prozess eingebunden sein und auf welche Weise?
- Was sind die Konsequenzen für die Anforderungsermittlung und den Entwicklungsprozess bei einer stetigen Weiterentwicklung des technischen Systems, das die Grundlage für die Applikation darstellt?

Wir stellen fest, dass diese Probleme in vielen anderen Web-Projekten ebenfalls auftreten. Allerdings müssen wir – um eine konkretere Aussagen machen zu können – erst klären, was wir unter Web-Projekten verstehen. So teilen z.B. Sherrell und Chen (2001) Internet-basierte Applikationen in vier Kategorien ein:

1. Intranets, firmeninterne Netzwerke
2. Web-Präsenzen oder konventionelle Webauftritte, öffentliche Firmen-Webauftritte als Marketinginstrument, PR und andere Zwecke
3. Electronic Commerce Systeme, Transaktionsorientierte „Business To Consumer" Webauftritte
4. Extranets, Dedizierte Business To Business Netzwerke

Wir konzentrieren uns hier auf die zweite Kategorie, welche die Entwicklung von Web-Portalen einschließt. Diese System bieten eine Sammlung von thematisch zusammengestellten Dienstleistungen. Beispiele dafür sind im Bereich e-Government neben anderen städtische, Kreis- oder Landes Websites, die eine Reihe von Dienstleistungen und Informationen über sich selbst und in Verbindung mit Partnern zur Verfügung stellen (z.B. www.hamburg.de, www.ukonline.gov.uk, www.oasis.gov.ie,

www.ecitizen.gov.sg). Darüber hinaus gibt es eine Reihe von Beispielen aus dem kommerziellen Bereich, bei denen einzelne Dienstleistungen, wie z.B. Suchmaschinen oder Web-Mail, zu einem vollwertigen Portal aufgebaut wurden (z.B. Hotmail, Yahoo, Altavista, Netscape). Wir werden hier die erste und vierte Kategorie, in der Entwickler sich auf eine klar ausdefinierte Benutzergruppe und andere Akteure verlassen können, nicht näher betrachten (zum Thema Intranet Entwicklung siehe u.a. Damsgaard und Scheepers 1999 und Scheepers 1999). Die meisten unserer Erkenntnisse lassen sich ebenso auf Projekte der dritten Kategorie anwenden (was klassische Shop-Anwendungen und Auktionssysteme beinhaltet: Amazon, Mp3.com, Dell, eBay), jedoch werden wir nicht speziell auf Workflow- und Transaktionsmanagement eingehen. Die oben genannte Kategorisierung beinhaltet allerdings keine Projekte wie „CommSy", bei denen kostenlose Dienstleistungen an ausgewählte aber verteilte Nutzergruppen angeboten werden.

Die von uns beobachteten Probleme im Bereich Web-Engineering können zusätzlich mit den von Nambisan and Wang (1999) angebotenen Kategorien betrachtet werden. Sie argumentieren, dass webbasierte Applikationsentwicklung drei Stufen der Web-Technologie Aneignung durchlaufen, bei der jede Stufe drei potentiellen Wissensbarrieren bei der Integration von Web-Technologie gegenübersteht:

- Das Fehlen von Wissen über Technologien, da Web-Technologien nie vollständig entwickelt sind und sich ständig verändern (Technologie bezogene Barriere),

- Das Wissen über Prozesse und Methoden ist unzureichend, da e-Anwendungen andere Herangehensweisen und spezifische Kombinationen aus unterschiedlichen Ressourcen erfordert, als klassischer Systementwurf (Projekt bezogene Barriere), und

- Domänenwissen ist schwer zu erhalten, da e-Business Applikationen die Grenzen von einer Organisation überschreiten (Anwendungsbezogene Barriere).

Es gibt bereits Vorgehensmodelle, die auf einige der Barrieren und identifizierten Probleme eingehen; zum Beispiel diskutieren Sherrell und Chen (2001) die Integration von Prototyping in das Web-Engineering in Zusammenhang mit ihrem „W Life Cycle" Modell. Sie versuchen dabei die Beiträge der Nutzer in den Entwicklungsprozess einzubeziehen, indem sie das Produktivsystem nutzen. Allerdings nehmen Sherrell und Chen an, eine bestimmbare und verfügbare Nutzergruppe des Anwendungssystems zu haben. Die ist allerhöchstens bei einigen Web-Projekten der Fall, wie z.B. bei Intranet Entwicklung in kleinen Organisationen. Im folgenden versuchen wir die Schwierigkeiten der Anwender-Entwickler-Beziehung in Web-Projekten mit offenen Nutzergruppen stärker zu analysieren. Wir werden sie mit typischen Problemen der Software-Entwicklung vergleichen und werden einige Basisannahmen des traditionellen Prototyping kritisch hinterfragen.

Den Softwareentwicklungsprozess managen

Anforderungen werden in Software Projekten häufig mit Meilensteinen in Verbindung gebracht, die den gesamten Entwicklungsprozess gliedern. In einem solchen Entwicklungsprozess werden Prototypen gebaut, um neue Erkenntnisse zu gewinnen und die Entscheidungsfindung zu unterstützen. Dies wird eingebettet in Iterationen bestehend aus Anforderungsermittlung, Prototyping, Realisierung und Freigabe des Produktes sowie Revisionsetablierung. E-Projekte haben diese Form von Freiheit nicht:

- Jegliche Software, die zur Nutzung ins Web gestellt wurde, ist ohne Schutz: Öffentlich verfügbare Web-Prototypen sind immer auch öffentlicher Kritik ausgesetzt – es gibt kein „Rumspielen" mit einem Entwicklungssystem.

- Jede Feedback-Runde mit den Anwendern braucht Zeit zur Vorbereitung, zur Präsentation, Kommunikation und Evaluation. Allerdings erlauben der Marktdruck und die hohen Erwartungen es den Web-Projekten meist nicht darauf zu warten.

- Web-Anwender erwarten regelmäßig neue Versionen, besonders wenn sie auf angeforderte Funktionalität warten.

- Dies führt zu deutlich kürzeren Entwicklungszyklen und dementsprechend auch zu höherem Druck auf die Entwickler, ihre Arbeitspakete auf kürzere Entwicklungsperioden abzustimmen.

- e-Applikationen sind alle „early adopters" (Scheepers 1999) – erste Anwender von Technologie – in ihrer Domäne, d.h. sie bieten neue Dienste im Internet an. Die Entwicklung muss deshalb darauf fokussieren, dass die Anwendung hohe Qualität gepaart mit Innovation liefert.

Relevante Akteure einbeziehen

Traditionelles Prototyping steuert bei der Anforderungsermittlung Möglichkeiten zur Kooperation zwischen Entwicklern und (zukünftigen) Anwendern bei. In den meisten Fällen arbeiten die Anwender für eine Organisation, die das Produkt bestellt oder später besitzen wird. Es gibt üblicherweise unterschiedliche Perspektiven und Interessen (z.B. Management vs. Anwender), die aber zumindest erkennbar sind. In Web-Projekten, die Organisationsgrenzen überschreiten, ist die Anforderungsermittlung für netzbasierte Anwendungen deutlich anders:

- Die initialen Anforderungen für eine potentielle Applikation werden durch die Anbieter-Perspektive definiert (die Wünsche und Anforderungen der aktuellen Nutzergruppe werden erst durch die erste lauffähige Version offenkundig);

- Die Nutzergruppe ist nicht klar ausdifferenziert – anders als in Firmen, in denen Nutzer durch ihre Arbeit oder Funktion charakterisiert werden können, sind Web-Nutzer in einer (un)begrenzen Zahl von Klassen strukturiert;

- Die relevanten Akteure können nicht mit einem einfachen Akteurs-Modell repräsentiert werden (z.B. Entwickler, Nutzer und Management) – da die Akteure, die sich im Anwendungssystem engagieren, neue Rollen wie z.B. „technology champion" (Damsgaard and Schepers 1999), (Sub-)Service Provider, und andere übernehmen;

- Akteure mit unterschiedlichen Perspektiven und Interessen sind üblicherweise nicht Teil derselben Organisation, was eine direkte und persönliche Diskussion (z.B. Nutzer und Entscheidungsträger können nicht einfach eingeladen oder zu Einzel- oder Gruppen-Interviews herangezogen werden) oder selbst simple Benutzerbeobachtungen ausschließt. Zusätzlich dazu ist Motivation schwieriger zur erreichen, da die Vorteile für die Anwender nicht in einer Organisation eingebettet sind – Internet-Nutzer bleiben einfach weg von Applikationen, die sie nicht mögen, während Nutzer von Anwendungen in Organisationen meist keine Wahl haben.

Perspektiven einbringen

Je mehr Gruppen von Akteuren in einem Projekt involviert sind, umso wichtiger wird es für die Anforderungsermittlung, deren Perspektiven zu erkennen und ihnen gerecht zu werden. Hierbei hilft es, dass bekannte Rollen wie z.B. Auftraggeber (der Finanzierer des Projektes), Nutzer, Entwickler und Kunde, identifiziert erden können; allerdings gibt es bemerkenswerte Verschiebungen der Interessenlage:

- Auftraggeber erwarten – zumindest in der Theorie –, dass sich ihre Investitionen auszahlen. Allerdings sind Web-Projekte häufig nicht verantwortlich, einen Rationalisierungseffekt herbeizuführen. Das Ergebnis kann auch eine Verbesserung des Image, ein gesteigertes Marktpotential oder eine Erweiterung des Service-Portfolios sein. In vielen Fällen sind die Investitionen in diese Art Projekte als „strategisch“ zu bezeichnen, d.h. es gibt nur schwer irgendwelche Bedingungen und Ziele, die im Rahmen des Projektes eingehalten oder erreicht werden sollen.

- Bei Web-Applikationen sind die Anwender häufig auch die Kunden (oder Klienten) der Organisation, die die Entwicklung in Auftrag gegeben hat (im Gegensatz zu traditionellen Projekten, in denen Anwender für eine Organisation arbeiten, um Werte für externe Kunden zu schaffen). Dies verändert die Beziehung zwischen dem Anwender und dem Auftraggeber.

- Bei großen und schlecht definierbaren Nutzergruppen haben Web-Projekte die Notwendigkeit, eine große Anzahl von Nutzungsperspektiven ständig auszuhandeln. Spezielle Rollen können/sollten identifiziert werden, z.B. Beschwerdeführer, die Fehler oder fehlende Funktionen regelmäßig ankreiden, Freiwillige, die versuchen eine aktive Rolle im Entwicklungsprozess der Web-Applikation einzunehmen indem sie eine große Menge an Zeit investieren, Auswertungen durchzuführen und Vorschläge für Verbesserungen zu machen (deren Motivation ist häufig nicht klar; nach Hars (2001) ist ein Schlüsselfaktor für Freiwillige in Open Source Projekten das persönliche Bedürfnis für eine Software-Lösung).

- Die Perspektive der Entwickler unterscheidet sich maßgeblich von denen der anderen. Ihre Aktivitäten sind geleitet von dem Interesse, die Software fehlerfrei zu halten, die neuesten Backend Technologien zu verwenden und eine State-of-the-Art Anwendung zu entwickeln. Ihre Perspektive ist der eines „Power-Users” gleichzusetzen. Andererseits gibt es Randbedingungen, die durch die Technik bedingt sind, und Grenzen, die durch anderen Akteure gesetzt werden. Die Aufgabe der Entwickler ist es nun, ein System in eine gegebenen Umgebung zu integrieren, und mit existierenden und vordefinierten Technologien zurechtzukommen, um es zuverlässig zum Laufen zu bringen.

Es sind neue Herausforderungen, die relevanten Perspektiven in den Entwicklungsprozess eines Web-Projektes einzubringen. Dies beinhaltet Bestätigung der neuen Perspektiven und auch die Verwendung neuer Kanäle und Medien, um diesen Perspektiven eine Stimme zu geben. Wir haben jetzt eine Situation, in welcher die Nutzer selbst die Kommunikation im Medium übernehmen (das Internet), z.B. Freiwillige moderieren Foren und organisieren Nutzergruppen Treffen (siehe dazu den nächsten Abschnitt, wie Entwickler sich die Kommunikationskanäle zu Nutze machen können).

Vernetzungsarbeit über Systemgrenzen hinweg

In traditionellen Projekten sind die Zuständigkeiten in der Entwicklung eher klar definiert, d.h. es gibt ein gemeinsames und stabiles Verständnis davon, was die Grenzen des technischen Systems und seiner Interfaces sind. In den meisten Fällen ist die Software das Eigentum des Auftraggebers und läuft auf seinen Geräten. Der Austausch mit anderen Systemen ist klar definiert durch Dateiformate und Schnittstellen (z.B. ein Abrechnungssystem generiert Buchungen und überträgt diese „en bloc“ an eine Bank oder ein Krankenhaus sammelt Behandlungsdaten und transferiert diese am Ende des Monats an eine Versicherungsgesellschaft). Im Gegensatz dazu bieten Web-Systeme eine große Anzahl von Schnittstellen, ermöglichen direkten Zugriff und integrieren absichtlich eine große Anzahl von Dienstleistungen unter einer Benutzungsschnittstelle (z.B. Content-Syndication, B2B, Microsofts .Net). Alle diese Fähigkeiten werden üblicherweise während des Ausbaus des Web-Systems erweitert (d.h. die Integration weiterer Web-Dienste in eine e-Applikation). Dies hat nicht zu vernachlässigende Auswirkungen auf den Umfang des Projektes und wie es fortgeführt wird, z.B.:

- Web-Portale, die Dienstleistungen integrieren, und andere Web-Services müssen die Anwender über den Ursprung des Dienstes informieren. Dies ist insbesondere bei Stadtinformationssystemen notwendig, weil diese – um ihre Attraktivität zu steigern – Verbindungen mit privaten Unternehmen eingehen (Public-Private-Partnership). Fragen stellen sich wie „Wer bietet mir diese Dienstleistung an?“ oder „Wer wird welche meiner (persönlichen) Daten weiterverarbeiten?“ Häufig gibt es dafür keine einfache Antwort, und um für die Privatsphäre Sorge zu tragen, ist mehr notwendig, als nur eine ausführliche Erklärung anzubieten.

- Das Erstellen und Ausbauen von technischen und organisationalen Netzwerken zur Verbesserungen des Durchsatzes stellt immer wieder die Architektur der Systeme in Frage. Kritisches Hinterfragen und regelmäßige Auswertungen sollten eine Transparenz sowohl für die Anwender als auch für die Anwendungsentwickler herstellen.

- Begründet durch den kontinuierlichen Austausch zwischen den Beteiligten und allen damit verbundenen Risiken und Abhängigkeiten von autonomen Partnern verschiebt sich der Fokus für die zukünftige Funktion der Web-Applikation ständig (dies ist normalen Projekten nicht der Fall). Das Projektmanagement muss deshalb flexibel sein und sich regelmäßig an die ständig wechselnden Umstände der Umgebung anpassen (Gaedke 2001).

Es gibt kein Labor für Prototyping im Internet! Web-Projekte sehen sich neuen Bedingungen gegenüber, die nach und nach in der Anwendung einer Applikation sichtbar werden. Reaktionen „realer“ Anwender werden erfahrbar. Deshalb kann sich die Anforderungsermittlung in Web-Projekten nicht auf traditionelles Prototyping verlassen, weil eine ganze Reihe von Annahmen nicht mehr länger gültig sind. Eine Menge relevanter Akteure und ihre Perspektiven müssen eingebunden werden, um Aufwendungen zu sparen und Innovation sicherzustellen. Wir wenden uns nun einer neuen Art zu, wie „e-Prototyping“ einen evolutionären Ansatz unterstützen kann, der sich auf kurze Entwicklungszyklen abstützt. Darüber hinaus werden wir explizit Nutzer-Feedback organisieren und behandeln, indem wir aktive Kommunikationskanäle anbieten, die die Entwicklung mit wertvollen Informationen versorgen.

4. Anleitung fürs „e-Prototyping“

In diesem Abschnitt beschreiben wir unseren Ansatz zu „e-Prototyping“. Zunächst diskutieren wir, wie aktuelle Software Engineering Trends zu kürzeren Entwicklungszyklen, zu einer Verflechtung von Prototyping und zu Release Management führen. Dann erklären wir die einzelnen Schritte unseres Prototyping-Ansatzes in Relation zu „klassischen“ Prototyping Aktivitäten. Abschließend erläutern wir wie Schwierigkeiten in der Anwender–Entwickler-Beziehung durch aktive und in den Entwicklungsprozess integrierte Unterstützung von Kommunikation überwunden werden können.

Aktuelle Trends im Software Engineering

Die Verkürzung von Entwicklungszyklen ist in verschiedenen Teilbereichen des Software Engineering thematisiert worden. Ein Beispiel für eine neue Methode, die eben dies propagiert ist Extreme Programming (Beck 2000). Um die Qualität eines Softwareprodukts zu verbessern, fordert Beck kürzere Entwicklungszyklen auf allen Ebenen des Software Engineering. Weiterhin schlägt er vor, bei der Implementation mit der Basisfunktionalität zu beginnen, die dem Kunden den größten (monetären und produktiven) Nutzen bringt. Nach Beck soll das System durch kontinuierliche Integration und häufiges Veröffentlichen von neuen Versionen stetig wachsen.

Häufiges Veröffentlichen von neuen Versionen ist auch gängige Praxis in Projekten der Open Source Gemeinde. Jørgensen (2001) beschreibt diesen Ansatz für das FreeBSD Projekt. Kommunikation in Form von Feedback spielt bei dieser Klasse von Entwicklungsprojekten eine wichtige Rolle. Im all-

gemeinen wird das Feedback gut dokumentiert und verwaltet (siehe bspw. das Mozilla Projekt). Dies führt in solchen Projekten zu neuen Management-Herausforderungen.

Wichtige neue Herausforderungen für das Management wurden auch von einer aktuellen Studie für die deutsche Software Industrie ermittelt (Stahl et al. 2000, S. 10). Diese beschreibt u.a. die Strategie, mit der Unternehmen die Schwierigkeiten zu überwinden versuchen: „Generell kann ein Trend zu risikominimierenden Entwicklungsprozessen (z.B. inkrementelle Prozesse) festgestellt werden."

Hier wird bereits darauf hingewiesen, dass immer kürzere Entwicklungszyklen gewählt werden, um den sich schneller ändernden Anforderungen im Bereich der Anwendungsentwicklung für das Internet besser gegenüber zu stellen. Wir sehen hier eine Wegbewegung vom Relase-Management – „Zudem haben verschiedene Unternehmen darauf hingewiesen, dass ‚die Zeit der großen Releases' vorbei sei" (ebd., S. 129) – und meinen, dass Prototyping bereits den notwendigen Charakter besitzt und hilft, kürzere Zyklen und kleinere Schritte zu unterstützen. Allerdings ist es notwendig, dies methodisch zu untermauern, um nicht in kleinschrittige Beliebigkeiten abzugleiten.

Zusammenfassend lässt sich festhalten, dass evolutionäre Ansätze und die Verwendung von Benutzer-Feedback etabliert sind. Allerdings müssen diese Methoden, und dies gilt besonders für Web-Projekte, mit dem Einsatz der entwickelten Software verwoben werden. Ebenso wie Prototyping aus unserer Sicht ein Mittel zur Unterstützung eines auf zyklischer Entwicklung basierenden evolutionären Ansatzes ist (siehe Abschnitt 2), so ist auch „e-Prototyping" ein evolutionärer, auf kurzen Entwicklungszyklen basierender Ansatz zur Unterstützung von Web-Projekten, in dem jede Version als ein e-Prototyp für den nächsten Entwicklungszyklus angesehen wird.

Schritte des e-Prototyping

In der evolutionären und partizipativen Softwareentwicklung wurden zyklische Vorgehensmodelle bereits Anfang der achtziger Jahre vorgeschlagen. Alle diese Ansätze legen einen Schwerpunkt auf die Kommunikation zwischen Entwicklern und Benutzern. Beispielsweise beinhaltet das STEPS Modell (Floyd et al. 1989) Zyklen bestehend aus (D1) Revisionsetablierung, (D2) Herstellung, (D3) Freigeben einer Version und (D4) Einsatz der Software.

Im folgendem wird unter Berücksichtigung der vier Schritte des evolutionären Prototyping – Funktionsauswahl, Konstruktion, Evaluation, weitere Verwendung (vgl. Abschnitt 2) – beschrieben, wie sich „e-Prototyping", basierend auf der oben genanten Klasse von Ansätzen, zur Realisierung eines evolutionären Softwareentwicklungsprozesses innerhalb von Web-Projekten verwenden lässt (P1-4, siehe Abbildung 1):

1. Grundlage der **Funktionsauswahl** bilden die zuvor gesammelten Anforderungen der Benutzer. Die „klassischen" Verfahren zur Anforderungsermittlung greifen allerdings nicht in der Domäne der Web-Anwendungen mit ihren e-Prototypen, da die Benutzergruppe im Vorwege nicht (oder zumindest nur undeutlich) bestimmt werden kann, was ein systematisches Vorgehen praktisch unmöglich macht (siehe Abschnitt 3). Daher müssen die ersten Anforderungen von den am Projekt beteiligten Akteuren, der sogenannten „Steuerungsgruppe" (bestehend aus Mitgliedern der Entwicklergruppe, der (Web-) Anbieter-Organisation und Geschäftspartnern, siehe auch Abbildung 1) antizipiert werden (Jeenicke 2001). Ziel ist es, möglichst schnell eine Version für die Benutzer freizugeben, um so ebenfalls schnell in eine öffentliche Diskussion mit den Benutzern einzutreten; die Benutzer also umgehend in den Entwicklungsprozess zu integrieren.

   Der Entwicklungsplan für die erste benutzbare Version des Systems sollte nur die Umsetzung derjenigen Funktionen beinhalten, die von den Entwickler mit Blick auf die oben genannten Wissens Barrieren (technologie-, projekt- und anwendungsbezogen) umgesetzt werden können. Eine angemessenen Funktionsauswahl bildet die Grundlage für eine zyklische Entwicklung, mit deren Hilfe

die Barrieren überwunden werden können. Die Erfahrungen, die während jedes Zyklus gewonnen werden, helfen dem Entwicklerteam die nächsten Schritte der Entwicklung zu meistern. Grundlage für die Funktionsauswahl der nächsten Zyklen ist jeweils die auf einer Evaluation des Benutzer-Feedbacks basierenden Entscheidungen der Steuerungsgruppe (siehe unten, Schritt 3 und 4).

2. Die **Konstruktion** in jedem Zyklus konzentriert sich auf die Funktionen, die im vorhergehenden Schritt ausgewählt worden sind. Nach Abschluss der Konstruktion wird die Software verfügbar, d.h. über das Internet zugänglich gemacht. Von da an wird es von den Benutzern wie ein Produktionssystem behandelt, auch wenn es von Seiten der Entwickler als ein Prototyp und Studienobjekt angesehen wird. Im Gegensatz zu konventionellem Prototyping wird die Software unter realen Bedingungen verwendet und nicht als Prototyp gekennzeichnet. E-Prototypen, d.h. Versionen, müssen daher höheren Qualitätsanforderungen gerecht werden als „klassische" Prototypen, was zu einer zusätzlichen Gewichtung des technischen Entwurf führt. Ziel der Konstruktion muss daher ein einsetzbares System sein, welches die Voraussetzung für die Erhebung und Evaluation von Benutzer-Feedback bildet.

3. Die **Evaluation** des Prototyps hängt stark von den Kommunikationsmitteln ab, die parallel zu der Benutzung jeder e-Prototyp-Version eingerichtet werden (siehe nächsten Teilabschnitt). Feedback zu der aktuellen Softwareversion gliedert sich u.a. in

   - Fehlermeldungen, die von Benutzern und Administratoren gesammelt werden,
   - Benutzbarkeitsprobleme, die aus Diskussionsbeiträgen gefiltert werden, und
   - Zusätzliche (und neue) Anforderungen der Benutzer (technology pull).

   Fehlermeldungen, Benutzbarkeitsprobleme und zusätzliche Anforderungen werden mittels diverser Kommunikationskanäle gesammelt und dokumentiert. Forderungen durch die am Projekt beteiligten Akteure nach neuen „strategischen" Anwendungen, mit denen ein Wettbewerbsvorteil erreicht werden soll (technology push), werden gesammelt und in der Entwicklergruppe und der Steuerungsgruppe diskutiert (siehe Abbildung 1).

4. Entscheidungen betreffend der **weiteren Verwendung** der Softwareversion berücksichtigen die Evaluation. Prototyping sollte dazu verwendet werden die Interessen der in der Anwendungsdomäne anzutreffenden Benutzer, Anbieter und andere am Projekt beteiligte Interessensgruppen in den Entwicklungsprozess zu integrieren. Letztendlich werden die Entscheidungen über die weiter Verwendung, die die Basis der Funktionsauswahl des nächsten Zyklus bilden, allerdings aus der Management-Sicht getroffen (Steuerungsgruppe).

Diese vier Schritte können als ein Zyklus-Durchlauf in einem durch Prototyping gesteuerten evolutionären Entwicklungsprozess angesehen werden. Die Entscheidungen, die nach der Evaluation getroffen werden, bilden den Startpunkt für den nächsten Zyklus, der wiederum vor der eigentlichen Konstruktion mit einer Auswahl der fachlichen und technischen Funktionen beginnt. Die Anforderungen, die in der nächsten Version realisiert werden sollen (basierend auf den notwendigen Fehlerkorrekturen und ausgewählte innovative Veränderungen), sollten vom Umfang her so ausgelegt sein, dass die Konstruktion und das Release der Software (bzw. e-Prototyp) innerhalb von 3 Monaten durchführbar ist.

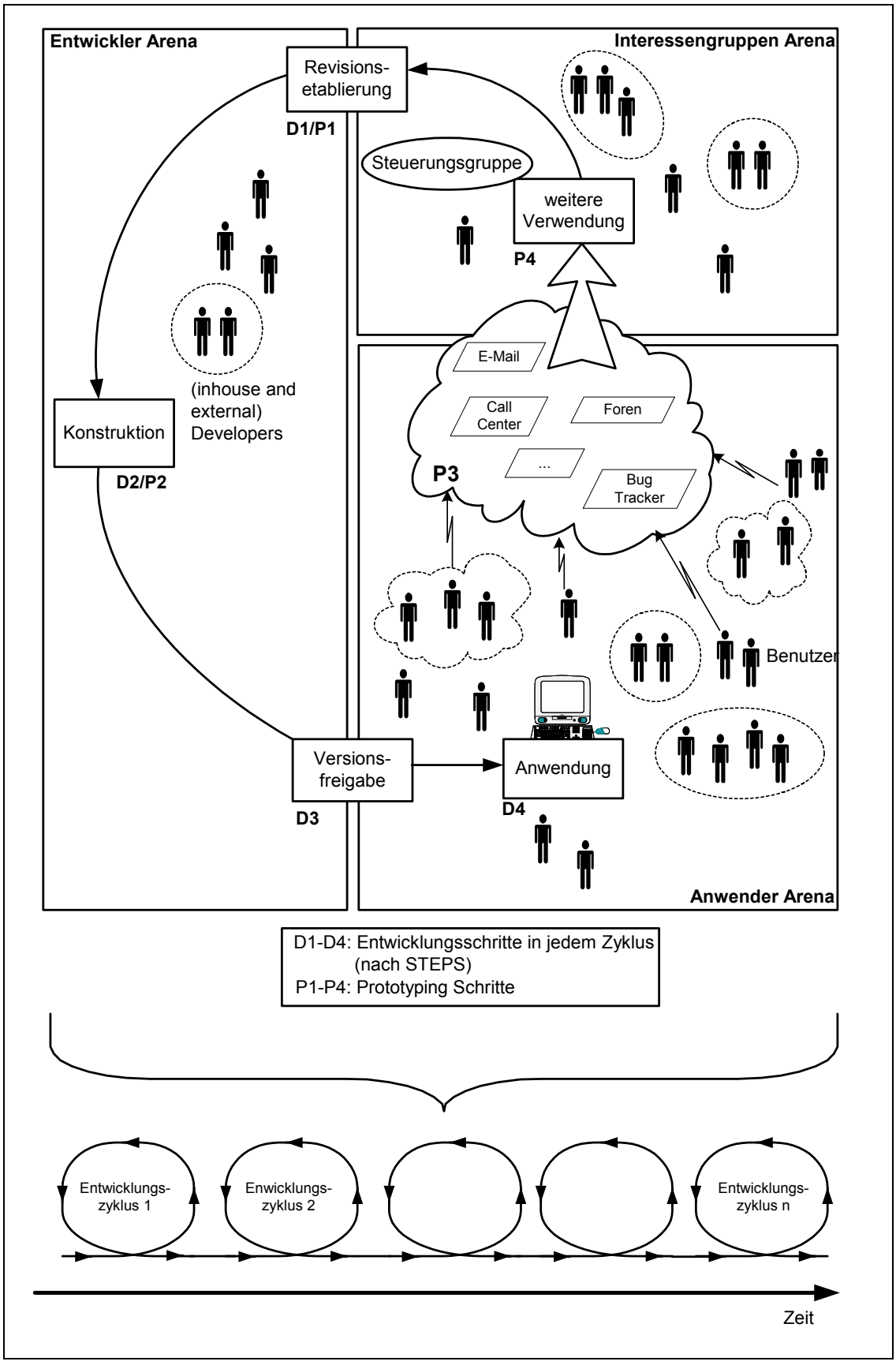

Abbildung 1 Kommunikationsfluss in einem Internet-bezogenem Entwicklungsprojekt

Kommunikation und Management

Die **Kommunikation** (oder zumindest der Austausch von Informationen) zwischen Benutzern und Entwicklern ist unerlässlich für den Fortschritt des Prototyping-Prozesses. Daher bedarf es zur Etablierung und Unterstützung der Interaktion mit den (zum größten Teil) „unbekannten" Web-Nutzern spezieller Kommunikationsmittel. Insbesondere die folgenden Kommunikationskanäle haben sich als besonders sinnvoll erwiesen: E-Mail, die an eine speziell für diesen Zweck reservierte Adresse gesendet werden kann (z.B. feedback@web-organization.de), ein Call-Center, das Benutzerprobleme und -vorschläge aufnimmt, sowie eine Web-Seite, über die Formulare zur Fehlermeldung sowie Diskussionsforen zugänglich sind. So weit möglich sollten alle Beiträge und Anrufe beantwortet werden.

Innerhalb der Internet-Gemeinde kommt es häufig vor, dass Benutzer, weil sie z.B. Interesse an einer speziellen Software haben, ohne nennenswerte Gegenleistungen eine aktive Rolle innerhalb eines Projekts einnehmen (siehe bspw. Newsgruppen). Zur erfolgreichen Interaktion zwischen Entwicklern und Nutzern ist es wichtig, dass sich diese speziellen Benutzer ernst genommen fühlen und die Software verlässlich arbeitet (dies beinhaltet u.a. eine implizite Garantie, dass im Falle eines Softwarefehlers, der etwa zu kritischen Datenverlusten führt, diesen freiwilligen (Test-) Benutzern Hilfe angeboten wird, die über das normale Supportangebot hinausgeht).

Das **Management** eines Entwicklungsprozesses, welcher „e-Prototyping" benutzt, muss nach häufigen Veröffentlichen von neuen Versionen, Kommunikation und Innovation streben. Updates einer laufenden e-Anwendung sollten in kurzen Intervallen erfolgen (zwischen ein paar Wochen und drei Monaten). Fehlerkorrekturen (Patches) sind häufiger notwendig, da so die oben genannten Kommunikationskanäle frei von Meldungen (bekannter) Fehler bleiben, wodurch Raum in den unerlässlichen Feedback-Kanälen freigehalten wird. Nur ein fehlerfreies System ermöglich also die Kommunikation über Funktionalität und Benutzbarkeit. Je fehlerhafter ein System ist, desto größer ist der Anteil der Fehlermeldungen an der Kommunikation. Wettbewerbsdruck ist ein weitere Faktor, der zu sehr kurzen Entwicklungszyklen und häufigen Veröffentlichen von neuen innovativen Versionen führt. Daher ist jede freigegebenen Version einer Software im Web-Engineering i.a. „der erste ihrer Art" (vergleiche Definition von Prototyp, Abschnitt 2).

Der beschriebe Prozess ist viel schwieriger zu kontrollieren als das in „klassischen" Softwareentwicklung der Fall ist. Eine erfolgreiche Anwendung zieht beispielsweise mehr Benutzer an, was in einer größeren Systemlast resultiert. Dies verursacht wiederum Probleme und führt zu einem fehlerhaften Verhalten mit der Konsequenz, dass ein Redesign der Systemarchitektur unvermeidbar wird. Daher kann sich der Schwerpunkt der Entwicklungsaktivitäten von einem rein funktional orientierten Ansatz zu einem strukturellem Redesign verschieben, um die Anforderungen an Skalierbarkeit und Systemauslastung zu erfüllen. Zusätzliche Sicherheitsbedürfnisse auf Seiten der Benutzer können zu Sicherheitsmerkmalen des Systems führen, die so Anfangs nicht vorhergesehen und geplant waren.

Um den skizzierten Prozess zu managen, muss sämtliches aus den verschiedene Feedback-Kanälen gesammeltes Benutzer-Feedback mit einer Systemversion assoziiert und von der Steuerungsgruppe evaluiert werden. Selbige entscheidet dann was auf die Liste der zu entwickelnden Funktionen gelangt. Diese bildet das Fundament für die nächste veröffentlichte Version, mit der Fehler sofort entfernt und die Funktionalität verbessert wird. Die Personen, die eine Fehler gemeldet haben, sollten umgehend über Verbesserungen informiert werden. Es sollte weiterhin klar sein, zu welchen Zeitpunkt die Änderungen in das laufende System integriert werden. Um doppelte Meldungen zu vermeiden, müssen alle Information über bekannte Probleme für die Benutzer zugänglich gemacht werden (siehe dazu z.B. Mozilla und Bugzilla).

Viele Aspekte von „e-Prototyping" finden sich bereits heute im Projektalltag. Daher kann man auch e-Prototyping (ohne Anführungszeichen) als in weiten Teilen erprobt ansehen, wobei sich folgende Richtlinien als spürbare Hilfen für den Projekterfolg erwiesen haben:

- **Kleinschrittiges Vorgehen mit Produktiv-Versionen:**
  e-Prototyping ist ein unverzichtbarer Bestandteil eines Softwareentwicklungsprozesses für unbekannte und diffuse Nutzergruppen. Ziel ist es, mit möglichst kleinen Schritten und explizierter Kommunikation sich schrittweise einem Produkt zu nähern, welches Betreibern, Betroffenen und Nutzern gerecht wird. Die Entwicklung ist in Schritte mit einer maximalen Länge von drei Monaten gegliedert und wird von expliziter Kommunikation begleitet. Es ist großer Vorteil zu jedem Zeitpunkt ein lauffähiges System zu haben. Dies wird durch die Adaption etablierter Prototyping Methoden zu einem Produktivsystem entwickelt.

- **Einrichtung von Kommunikationskanälen und Bündelung von Feedback:**
  (ohne direktes Feedback) Beim e-Prototyping werden die Perspektiven der Beteiligten expliziert. Um mit den komplexen Perspektiven der Nutzer umgehen zu können, wird Feedback in eigens dafür eingerichtete Kommunikationskanäle gebündelt.

- **Permanente Balance zwischen Stabilität und Innovation:**
  Der Prozess des Prototyping spielt sich im ständigen Entwickeln einer stabilen Version mit minimalem aber vollständigem Funktionsumfang, dem Sammeln des Feedbacks aus den unterschiedlichen Kanälen und der Planung des nächsten Entwicklungsschrittes**.**

5. Zusammenfassung

Softwareentwicklung steht in Web-Projekten vor der besonderen Schwierigkeit, verlässliche Anforderungen aus der Anwendungsdomäne heraus abzuleiten – gerade auch deshalb, weil die potentiellen Anwender weit verstreut und letztlich unerreichbar sind z.B. für die Evaluation eines Prototypen in einer Laborsituation. Die neuen Bedingungen, Schwierigkeiten und Herausforderungen bei der Einbeziehung von und Kommunikation mit den relevanten Akteuren reflektierend, haben wir einen Ansatz für das e-Prototyping skizziert: Da jedes Release bzw. jede veröffentlichte Version als Produktivsystem läuft, sollten beim Management von Web-Engineering Prozessen kurze Entwicklungszyklen mit häufigen, jeweils auch innovativen Versions-Releases angestrebt und Rückmeldungen von dessen Benutzern (bzw. anderen relevanten Akteuren) über verschieden Kommunikationskanäle eingeholt werden. Aus unserer Sicht verbessert e-Prototyping das „traditionelle“ Prototyping, indem es systematisch mit einem zyklischen bzw. evolutionären Softwareentwicklungsansatz und einem auf die Einbeziehung der relevanten Akteuren gerichteten Kommunikationsmanagement verbunden wird.

Web-Projekte, die sich des e-Prototyping bedienen, profitieren in dreierlei Hinsicht:

- Das systematische Sammeln, Kanalisieren und (Aus-)Sortieren von Rückmeldungen setzt die Projektsteuerung in die Lage, für jede neue Version bzw. jeden neuen Entwicklungszyklus kurzfristig verlässliche Anforderungen zu benennen.

- e-Prototypen als Produktivsysteme zu integrieren bedeutet, Web-Anwendungen in kurzen Zyklen mit häufigen, jeweils auch innovativen Versions-Releases zu entwickeln – eine Strategie, die dazu beiträgt, insgesamt die Zeit für Entwicklungszyklen und die damit verbundene „Time-to-Market“ zu verkürzen und das Risikomanagement in Projekten zu verbessern.

- e-Prototyping trägt außerdem zu einem integrierten Wissensmanagement bei: In jedem Entwicklungszyklus treten typischerweise erneut die verschiedenartigen Barrieren bei der Aneignung von neuen Webtechnologien auf (vgl. Abschnitt 3) – aber das Vorgehen in kleinen Schritten verbunden mit einer kontinuierlichen Evaluation bringt Wissenslücken zum Vorschein (bevor sie wo-

möglich hohen Schaden verursachen), unterstützt das Lernen und die Weitergabe von Wissen zwischen den beteiligten Akteuren. Von daher lässt sich e-Prototyping auch als einen Ansatz verstehen, der die Aneignung von neuen Webtechnologien in Projekten zur Anwendungsentwicklung befördert.

Natürlich findet man auch jetzt schon in vielen Web-Projekten Elemente eines e-Prototyping, und weitere Forschungen in diesem Bereich sollten auf einer Evaluation dieser Praxis aufbauen. Unsere Kernaussage hier ist, dass e-Prototyping, basierend auf einem in der anwendungsorientierten Softwareentwicklung Konzept, eine Bereicherung des Methodenrepertoires im sogenannten Web-Engineering ist, von dem sowohl die wissenschaftliche Diskussion als auch Projekte im gesamten World Wide Web profitieren können

6. Danksagung

Wir danken Christiane Floyd für die Inspiration und Diskurs der vergangenen Jahre, der uns zu der hier präsentierten Herangehensweise geleitet hat.

7. Literatur

Beck, K.: Extreme programming explained: embrace change. Addison-Wesley, Reading, Mass., 2000

Bleek, W.-G., Kielas, W., Malon, K., Otto, T., Wolff, B.: Vorgehen zur Einführung von Community Systemen in Lerngemeinschaften. In GeNeMe 2000: Gemeinschaften in Neuen Medien, herausgegeben von Engelien, M. und Neumann, D. Lohmar, Köln: Josef Eul Verlag, 2000.

Bleek, W.-G.: Situations in Life to Support the Use and Modelling of Municipal Information Systems, In: Remenyi D., Bannister, F. (Hg.): Proceedings of the European Conference on Electronic Government. MCIL, Reading, UK, 2001, S. 49-60

Brockhaus Enzyklopädie, 19.Auflage, F.A. Brockhaus, Mannheim, 1992.

Damsgaard, J., Scheepers, R.: A Stage Model of Intranet Technology Implementation and Management. In: Proceedings of the 7th European Conference on Information Systems, 1999, S. 100–116

Floyd, C.: A Systematic Look at Prototyping. In: R. Budde, K. Kuhlenkamp, L. Mathiassen, H. Züllighoven (Hg.): Approaches to Prototyping. Springer, Berlin, 1984, S. 1-18

Floyd, C., Reisin, F.-M., Schmidt, G.: STEPS to Software Development with Users. In: Ghezzi, C., McDermid, J.A. (Hg.): Proceedings ESEC ’89, Lecture Notes in Computer Science 387, Springer, Berlin, 1989, S. 48-64

Gaedke, M., Gräf, G.: Development and evolution of Web-applications using the WebComposition process model. In: Murugesan, S. Deshpande, Y. (Hg.): Web Engineering. Managing diversity and complexity of web application development. Lecture Notes in Computer Science 2016. Springer, Berlin, 2001, S. 58-76.

Hars, A., Ou, S.: Working for free? - Motivations for participating in open source projects. In: Sprague, R.H. (Hg.): Proceedings HICSS-34, IEEE Computer Society, 2001, S. 9ff.

Jeenicke, M.: Antizipative Anforderungsermittlung bei der Softwareentwicklung. Diplomarbeit, Fachbereich Informatik, Universität Hamburg, 2001

Jørgensen, N.: Putting it all in the trunk: incremental software development in the FreeBSD open source project. Erscheint in Informations Systems Journal, 2002 (Download Vorversion von http://www.dat.ruc.dk/~nielsj/research/freebsd/freebsd.pdf am 27.11.01)

Kieback, F.-M., Lichter, H., Schneider-Hufschmidt, M., Züllighoven, H.: Prototypen in industriellen Software-Projekten – Erfahrungen und Analysen. In: Informatik-Spektrum, 15 (2), 1992

Manber, U., Patel A. Robison, J.: Experience with personalization of Yahoo! In: Communications of the ACM 43 (8), Aug. 2000, S. 35-39.

Murugesan, S., Deshpande, Y. (Hg.): Web engineering: managing diversity and complexity of web application development. Springer, Berlin, 2001

Nambisan, S., Wang, Y.-M.: Roadblocks to Web Technology Adoption? In: Communications of the ACM, 42 (1), January 1999, S. 98-101.

Pape, B., Bleek, W.-G., Jackewitz, I., Janneck, M. Software requirements for project-based learning – CommSy as an exemplary solution. Proceedings HICSS-35, IEEE Computer Society, 2002

Scheepers, R.: Key Role Players in the Initiation and Implementation of Intranet Technology. In: New Information Technologies in Organizational Processes – Field Studies and Theoretical Reflections on the Future of Work. Proceedings of IFIP WG 8.2. Chapman and Hall, 1999, S. 175–195

Sherrell, L. B., Chen, L.-D.: The W Life Cycle Model and Associated Methodology for Corporate Web Site Development: Communications of the Association for Information Systems, Volume 5, Article 7, April 2001

Sommerville, I.: Software Engineering (5th edition). Addison-Wesley, Harlow, UK, 1996

Stahl, P., Wucher, R., Rombach, H., Broy, M., Friedewald, M., Kimpeler, S., Zoche, P.: Analyse und Evaluation der Softwareentwicklung in Deutschland. GfK Marktforschungs GmbH, Dezember 2000. http://www.dlr.de/IT/IV (Download 27.11.01)