User Interface Prototypen in Usability Engineering und Software Entwicklung: Probleme und Synergien

Markus Weber
Centigrade GmbH
Science Park 2
66123 Saarbrücken
markus.weber@centigrade.de

Thomas Immich
Centigrade GmbH
Science Park 2
66123 Saarbrücken
thomas.immich@centigrade.de

Abstract

Probleme beim praktischen Einsatz von User Interface Prototypen in den Bereichen Usability Engineering und Software Entwicklung können sich unter anderem in folgenden Feldern manifestieren: Detaillierungsgrad des Prototyps, Erwartungshaltung der Rezipienten von Prototypen, Wiederverwendung prototypischer Entwicklungen sowie Abbildung von Interaktionen und Workflows.

Zur Eliminierung dieser Probleme bieten sich verschiedene Ansätze an.

Neben Maßnahmen in den beiden Einzelbereichen sollte im Rahmen der expliziten Kommunikation zwischen Usability Engineer und Entwickler zur Identifizierung von Synergien die Möglichkeit der gemeinsamen Verwendung von Prototypen ebenso geprüft werden wie ein koordiniertes Erwartungsmanagement nach außen. Durch die Abkopplung der User Interface Entwicklung von anderen Backend-Entwicklungsarbeiten kann die Effizienz eines Projektes erhöht werden.

Keywords

Prototyping, Erwartungsmanagement, kooperative GUI Entwicklung, agile Entwicklung

1.0 Einleitung

Prototypen werden bei der Entwicklung von interaktiven Systemen auf vielfältige Weise eingesetzt. So können sie beispielweise dazu dienen, technische Machbarkeiten zu prüfen oder im Rahmen eines Usability Tests Erkenntnisse zur Optimierung eines Systems zu gewinnen.

Häufig werden im Entwicklungsprozess Prototypen von verschiedenen Beteiligten eingesetzt, die wiederum jeweils spezifische Ziele und Anforderungen bezüglich des Prototyps haben.

Der vorliegende Beitrag befasst sich mit User Interface Prototypen, also mit solchen Prototypen, deren Zielsetzung darin besteht, zu einer Beurteilung bestimmter Aspekte des User Interface eines Systems zu gelangen und auf dieser Grundlage Entscheidungen zu treffen. Hierbei wird der praktische Einsatz aus den zwei exemplarischen Perspektiven Usability Engineering und Software Entwicklung betrachtet, um Problemfelder aufzuzeigen und der Frage nach Berührungspunkten bei der Verwendung von Prototypen in den beiden Bereichen nachzugehen.

2.0 Anwendungskontexte

Exemplarisch können bezüglich des Einsatzes von Prototypen zwei Anwendungskontexte und Stakeholdergruppen unterschieden werden, die sich in ihren Zielsetzungen und Anforderungen hinsichtlich des Einsatzes von Prototypen unterscheiden.

2.1 Usability Engineering

Zielsetzung des Usability Engineering Prozesses ist es, Erkenntnisse zu gewinnen, die zur Verbesserung der Anwenderfreundlichkeit des interaktiven Systems beitragen und diese Erkenntnisse in entsprechende Maßnahmen und Handlungsempfehlungen umzusetzen.

Prototypen kommen hier zum Beispiel dann zum Einsatz, wenn repräsentative Anwender mit dem Prototyp typische Arbeitsaufgaben durchführen, damit auf der Grundlage der hierdurch gewonnenen Daten Wissen zur Optimierung des Systems gewonnen werden kann.

2.2 Software Entwicklung

In der Software Entwicklung geht es darum, ein funktionsfähiges System zu erstellen, das den technischen Anforderungen genügt.

Prototypen kommen hier beispielsweise zum Einsatz, wenn die Tauglichkeit von Implementierungen, zum Beispiel spezieller Widgets, geprüft wird oder die grundsätzliche Machbarkeit von Ansätzen evaluiert werden soll.

3.0 Problemfelder in der Praxis

Im Folgenden wird auf typische Problemfelder beim praktischen Einsatz von Prototypen eingegangen. Es wird erläutert, wie sich Probleme in den beiden Bereichen Usability Engineering und Software Entwicklung manifestieren

Brau, H., Diefenbach, S., Hassenzahl, M., Koller, F., Peissner, M. & Röse, K., (Hrsg.): Usability Professionals 2008

können und welche Ansatzpunkte zur Beseitigung es gibt.

3.1 Detaillierungsgrad von Prototypen

Das Konzept „Detaillierungsgrad“ eines Prototyps bezieht sich darauf, wie sehr visuelle oder funktionale Aspekte in der prototypischen Umsetzung ausdifferenziert werden (low fidelity vs. high fidelity).

Im Usability Engineering besteht die Gefahr, dass der Fokus eines Teilnehmers in einem Usability Test durch einen ungeeigneten Detaillierungsgrad von der eigentlichen Fragestellung abgelenkt wird. Dies kann beispielsweise dann der Fall sein, wenn ein prototypisches User Interface visuell sehr differenziert gehalten ist und hierdurch sehr viele Kommentare zur visuellen Gestaltung provoziert werden, obwohl die Fragestellung des Tests dem Interaktionsdesign galt. Aber auch ein zu gering aufgelöster Prototyp kann problematisch sein, wenn er für die Anwender zu abstrakt bleibt und das entsprechende Feedback sich in der Folge auf einem vergleichbar abstrakten Niveau bewegt.

In der Software Entwicklung kann das Risiko von zu detaillierten Prototypen darin bestehen, dass sie einen hohen Implementierungsaufwand mit sich bringen, der dem eigentlichen Einsatzzweck nicht angemessen ist. Dies ist etwa dann der Fall, wenn sich Entwickler dazu animiert sehen, einen (funktionstechnisch) detaillierten Prototypen weiterzuentwickeln, „um keine Arbeit zu vergeuden“, obwohl die Ausdetaillierung des Prototyps ausschließlich deshalb erforderlich war, um fundiert zu dem Schluss zu kommen, dass ein bestimmter Implementierungsansatz nicht trägt. Durch die hohe Detaillierung des Prototyps und eine eventuell unzureichende Kommunikation im Projektteam kann sich daher die Wahrscheinlichkeit des berechtigten Verwerfens eines Prototyps verringern. Letztendlich werden jedoch Ressourcen vergeudet, da die Arbeiten, die aufbauend auf dem detaillierten Prototyp durchgeführt wurden, nicht zielführend waren.

In der Praxis sollte der Detaillierungsgrad eines Prototyps der jeweiligen Fragestellung angemessen sein. Dies setzt voraus, dass diese vor der Erstellung des Prototyps definiert und den relevanten Personen mitgeteilt wird, was im praktischen Projektalltag durchaus nicht immer gegeben ist. Ist dies nicht der Fall, so besteht die Gefahr, dass ein Prototyp entsteht, dessen Ausgestaltung sich vor allem an umsetzungstechnischen Aspekten orientiert, nach dem Motto: „Je einfacher etwas prototypisch umgesetzt werden kann, desto eher wird es im Prototyp realisiert“.

Für den Usability Engineer ist es wichtig, ein Gespür dafür zu entwickeln, dass Teilnehmer in einem Usability Test unterschiedlich mit dem Detaillierungsgrad eines Prototyps umgehen. So findet man in der Praxis Teilnehmer, die sich in Feedback zu Details verlieren und denen es von sich aus unmöglich ist, davon zu abstrahieren, während andere Teilnehmer damit kein Problem haben. Neben der möglichst optimalen Ausrichtung des Detaillierungsgrads auf die Fragestellung ist es damit für den Usability Engineer also auch wichtig, in der konkreten Situation mögliche negative Tendenzen bei den Teilnehmern zu beobachten und angemessen darauf zu reagieren. Dies kann er zum Beispiel tun, indem er den Fokus der Teilnehmer durch Hinweise und Erläuterungen wieder auf die eigentliche Fragestellung richtet. Im Vorfeld des Tests kann auch die Erfahrung des Usability Engineers mit vergleichbaren Tests und Teilnehmergruppen zu einem Urteil über den geeigneten Detaillierungsgrad des Prototyps beitragen.

Für Entwickler ist es wichtig, den Stellenwert eines Prototyps zu kennen. Mit diesem Wissen können sie ein fundiertes Urteil darüber fällen, ob eine relativ hohe Detaillierung daher rührt, dass der Prototyp als Grundlage weiterer Implementierungsarbeiten genutzt werden soll oder daher, dass sie erforderlich war, um eine implementierungstechnische Frage umfassend bewerten zu können. Insbesondere in größeren Entwicklerteams sollten daher die entsprechenden Informationen klar kommuniziert werden.

3.2 Erwartungsmanagement

Häufig wird der Prototyp einer nicht unmittelbar am Projekt beteiligten Öffentlichkeit zugänglich gemacht. Dies geschieht beispielsweise im Rahmen von Vorstandsmeetings oder auch bei Tests mit repräsentativen Anwendern im Kontext eines Usability Tests. Immer werden durch die Konfrontation mit einem Prototyp auch die Erwartungen des Publikums an das System beziehungsweise an das Entwicklungsprojekt beeinflusst.

Für Teilnehmer an einem Usability Test kann es schwierig sein, den Stellenwert eines Prototyps richtig einzuschätzen. Beispielsweise wird unter Umständen durch visuelle Gestaltung und scheinbare Funktionalität des Prototyps der Eindruck erweckt, dass die Entwicklung schon sehr weit fortgeschritten ist und das Release kurz vor der Fertigstellung steht. Dies kann insbesondere bei inhouse Entwicklungen den Druck auf die Projektverantwortlichen erhöhen, da unter Umständen die Akzeptanz für Revisionen und „Verzögerungen“ der Entwicklung eines „weit gediehenen“ Systems reduziert wird. Eine andere Seite der selben Problematik zeigt sich, wenn es den Rezipienten nicht möglich ist, von auftretenden Bugs zu abstrahieren

und sie diese als schwerwiegende Mängel des Endsystems ansehen, obwohl es sich lediglich um die zu erwartenden Fehler einer frühen Implementierung handelt.

Aus der Entwicklerperspektive ist Erwartungsmanagement speziell dann bedeutsam, wenn ein hauptsächlich vom Usability Engineer verantworteter Prototyp interaktive Elemente oder Paradigmen vorstellt, deren technische Machbarkeit bislang noch nicht geprüft wurde, diese aber durch ihre Verwendung in Tests und Präsentationen implizit Teil des Anforderungskatalogs werden. Dies kann unter Umständen dazu führen, dass eine vom Publikum erwartete Programmeigenschaft nicht nur spät, sondern im schlimmsten Fall überhaupt nicht realisierbar ist, was sich negativ in der abschließenden Bewertung des Systems niederschlägt.

Der Usability Engineer sollte aktives Erwartungsmanagement betreiben, um sicherzustellen, dass der Stellenwert des Prototyps von jeweiligen Publikum richtig erkannt und der Prototyp richtig eingeordnet wird. Dies kann beispielsweise durch eine Erläuterung dahingehend geschehen, dass tiefgreifende Veränderungen des Prototyps durchaus möglich sind und er lediglich ein Artefakt für einen bestimmten, umschriebenen Projektzweck darstellt, wie zum Beispiel Kommunikation und Sammlung von Feedback im Rahmen eines Usability Tests. Der häufig im Rahmen von Usability Tests gegebene Hinweis, dass während der Interaktion Fehler auftreten können, die auf noch existierende Schwachstellen der Technik hinweisen, dient ebenfalls dem Erwartungsmanagement.

Um das Risiko zu minimieren, dass in einem Prototyp vorgestellte Konzepte nicht umsetzbar sind, empfiehlt es sich, dass Usability Engineer und Entwickler frühzeitig miteinander in Kontakt treten, so dass potenzielle Probleme erörtert werden können oder die Umsetzbarkeit gegebenenfalls in Form von technischen „Spikes“[1] geprüft werden kann. Der Entwickler (beziehungsweise der Projektmanager) sollte die Implementierung eines Spikes im Voraus zeitlich begrenzen, damit die Studie ausschließlich auf die wesentlichen Risikoquellen fokussiert wird.

3.3 Wiederverwendung von Prototypen

Zuweilen wird die Wiederverwendung von Prototypen als Ideal gesehen, das im Softwareentwicklungsprozess verfolgt werden sollte. So werden dann beispielsweise bei der Erstellung eines Systems Prototypen von Usability Test zu Usability Test sukzessive erweitert oder evolutionäre technische Prototypen in der Entwicklung eingesetzt.

Im Rahmen des Usability Engineering Prozesses kann die Gefahr bestehen, die jeweils in Usability Tests behandelten Fragestellungen dem Primat der Wiederverwendung von Prototypen unterzuordnen und damit den Erkenntnishorizont von vornherein zu beschränken. Dies ist etwa dann der Fall, wenn nur wenig Ressourcen zur Verfügung gestellt werden, um Umstrukturierungen eines Prototyps vorzunehmen, die eine revidierte Informationsarchitektur reflektieren. Dies kann so weit gehen, dass zur Untersuchung einer Fragestellung auch kein Papierprototyp eingesetzt wird, da dies scheinbar einen „Rückschritt“ im Vergleich zu einem bereits vorhandenen interaktiven HTML-Prototyp darstellt.

Bei der Software Entwicklung besteht das Risiko darin, dass durch das Ideal der Wiederverwendung von Prototypen und aufgrund enger Projektzeitpläne Implementierungen „mitgeschleppt“ werden, die von ihrer Qualität her unzureichend sind oder die auf ein bestimmtes, zum Entwicklungszeitpunkt aktuelles, Framework abgestimmt waren, welches sich über die Projektdauer jedoch verändert hat oder durch ein anderes abgelöst wurde.

Der Usability Engineer sollte vor dem Einsatz von Prototypen prüfen, ob der zu erwartende Erkenntnisgewinn den Aufwand für eine Überarbeitung oder Ergänzung eines Prototyps rechtfertigt. Je günstiger das Kosten-Nutzen Verhältnis ist, desto eher sollten die erforderlichen Ressourcen dafür zur Verfügung gestellt werden. Insbesondere wenn Papierprototypen eingesetzt werden, können Änderungen oder „Neuentwicklungen“ leicht umgesetzt werden. Allerdings kann diese Art von Prototyp nicht für alle Fragestellungen Verwendung finden. Beispielsweise sind sie ungeeignet zur Evaluation von Detailinteraktionen mit bestimmten Controls.

Im Rahmen der Software Entwicklung muss für ein Projekt festgelegt werden, ob die Wiederverwendung von Prototypen ein explizites Ziel darstellt. Ist dies der Fall, so sollte die Codequalität durch automatisierte Tests (zum Beispiel „Unit Tests“[2]) und explizite „Refactoring“[3]

1 Der Begriff „Spike“ wird oft in agilen Entwicklungskontexten verwendet und beschreibt eine fokussierte Studie, in der Entwickler technisch relevante Aspekte und Designideen in Form von kleinen abgeschlossenen Programmteilen umzusetzen versuchen (Proof of Concept).

2 Unit Tests (auch „Modultests“ genannt) dienen zur automatisierten Überprüfung der Korrektheit von Modulen in einer Software. Nach jeder Codeänderung werden alle Unit Tests automatisch durchlaufen und geben so Aufschluss darüber, ob die Software immer noch in einem stabilen Zustand ist.

3 Mit „Refactoring“ bezeichnet man Änderungen am Quellcode einer Software, die nur dessen Lesbarkeit und Struktur verbessern, ohne jedoch sein Verhalten zu ändern. Fest eingeplantes und durch Tests gesichertes Refactoring, senkt die „Hemmschwelle“ bei Entwicklern, Imple-

Phasen gesichert werden. Zumindest auf der User Interface Schicht sollte hierbei nicht der klassische „Test First“[4] Ansatz verfolgt werden – das Erstellen von Tests kurz vor dem Refactoring bietet eine hinreichende Absicherung und erlaubt ein viel agileres Reagieren auf veränderte Anforderungen an den Prototyp.

Ist die Wiederverwendung von Prototypen ein Ziel, so sollte zunächst festgelegt werden, ob Wiederverwendung über verschiedene Einsatzphasen von Prototypen im Entwicklungszyklus hinweg oder Wiederverwendung des Prototyps hinsichtlich des letztendlich implementierten Systems oder beides gemeint ist.

Bezüglich des ersten Aspekts sollte der Einflussfaktor der Kopplung des User Interface Prototyps mit konkreten Daten des Backend betrachtet werden. Durch eine feste „Verdrahtung“ kann die Ursache für bestimmte Effekte im User Interface unter Umständen nur unzureichend isoliert werden, oder aber bestimmte Effekte treten aufgrund einer konkreten Daten-Konstellation gar nicht erst auf, wodurch Probleme des User Interface lange unentdeckt bleiben. Wenn möglich, so empfiehlt es sich, die Entwicklung des User Interface möglichst abgekoppelt von der Backend Entwicklung zu betreiben, um die genannten Effekte zu verhindern.

Für den zweiten Aspekt der Wiederverwertbarkeit ist es bedeutsam, dass die technische Plattform für den Prototyp weitestgehend mit derjenigen des Endproduktes identisch ist, da ansonsten ein Transfer der prototypischen Entwicklungen für das Endprodukt nicht nahtlos durchgeführt werden kann. Ein derartiger Ansatz, bei dem das Produkt quasi nahtlos aus Prototypen entsteht, beziehungsweise bei dem der Prototyp zu jeder Phase des Prozesses das Produkt *ist*, wird im Allgemeinen in agilen Ansätzen verfolgt.

3.4 Abbildung von Interaktionen und Workflows

Das User Interface eines Systems dient in der Regel dazu, dem Anwender die Erledigung von Arbeitsaufgaben im Rahmen von Interaktionen und Workflows zu ermöglichen. Workflows stellen dabei umfassendere Sammlungen von Arbeitsschritten im Dienste einer bestimmten Zielsetzung dar. Interaktionen sind kleinere Arbeitseinheiten, bei denen der Anwender mit dem System interagiert und die wiederum Bestandteil mehrerer unterschiedlicher Workflows sein können. So kann etwa die Interaktion „Eingabe einer Adresse in ein Formular“ in einem System Teil der Workflows „Anlegen eines neuen Kunden“ sowie „Ergänzen eines Kundendatensatzes“ sein. In Prototypen sind die möglichen Interaktionen und Workflows mehr oder weniger umfänglich abgebildet.

Im Usability Engineering muss vorab geklärt werden, inwieweit Interaktionen und Workflows im Prototyp fokussiert werden sollen, da dies unmittelbar Technik und Gestaltung des Prototyps beeinflusst. Steht beispielsweise die Evaluation von Workflows im Vordergrund, so können hierfür Papierprototypen in Frage kommen, mit denen dem Anwender bestimmte Screensequenzen präsentiert werden können. Für die Evaluation von Detailinteraktionen sind Papierprototypen dagegen nicht geeignet, da sie keine ausreichend hohe Interaktivität zur Verfügung stellen können. Werden Papierprototypen unreflektiert eingesetzt, so kann die Betrachtungsebene der Detailinteraktionen zu kurz kommen, da diese für Testteilnehmer nur unzureichend greifbar wird. Für derartige Evaluationen können beispielsweise HTML- oder technisch noch aufwändigere Prototypen zum Einsatz kommen.

Für die Software Entwicklung kann die Unterscheidung zwischen Workflows und Interaktionen beispielsweise deshalb bedeutsam sein, weil für die Evaluation von Workflows reale Datenumgebungen zu simulieren sind und diese erst nach Bereitstellung entsprechender Informationen (realistische Daten) durch Domänenexperten umgesetzt werden können. Für Interaktionen anhand bestimmter Controls müssen realistische Datenstrukturen dagegen nicht unbedingt eine Rolle spielen. Je nach Fokus der Evaluation muss eventuell also berücksichtigt werden, dass den Entwicklern tiefergehende Information zur Verfügung gestellt werden muss und entsprechende Ressourcen dafür eingeplant werden sollten.

4.0 Zusammenspiel von Usability Engineering und Software Entwicklung

Die vorangegangenen Erläuterungen zeigen, dass sich die Probleme, die beim praktischen Einsatz von Prototypen in Usability Engineering und Software Entwicklung auftreten, in vergleichbaren Problembereichen abspielen, wobei die konkrete Erscheinungsform eines Problems abhängig ist davon, ob das Ziel des Prototyps eher in einer Usability Evaluation oder einer technischen Evaluation besteht.

Es stellt sich die Frage, wie Synergien von Usability Engineering und Software Entwicklung geschaffen werden können, so dass der Einsatz von Prototypen in beiden Bereichen effizienter erfolgt.

mentierungen in mehr oder weniger großen Teilen zu verwerfen und verhindert somit idealerweise das „Mitschleppen“ von minderwertigen oder nicht nachhaltig getesteten Entwicklungsarbeiten.

4 „Test First“ bezeichnet einen Ansatz, bei dem zunächst Unit Tests geschrieben werden und dann erst mit der Entwicklung des eigentlichen Systems begonnen wird.

Die Grundlage für derartige Synergieeffekte ist die Kommunikation zwischen Usability Engineer und Software Entwickler. Diese ist in der Praxis durchaus nicht selbstverständlich, da aufgrund von Projektstrukturen nicht immer ein direkter Austausch gegeben ist und die Kommunikation daher oft indirekt und/oder implizit erfolgt, beispielsweise durch die Begutachtung von Prototypen des jeweils anderen Teams. In diesem Kontext sollte auf eine direkte und explizite Kommunikation hingearbeitet werden, beispielsweise im Rahmen regelmäßiger Teammeetings, bei denen relevante Themen besprochen und Entscheidungen getroffen werden. Auf diese Weise können negative Effekte impliziter Kommunikation vermieden werden, die zum Beispiel auftreten können, wenn der Software Entwickler „seinen" Prototyp erheblich ausdifferenziert, um „vorzuarbeiten", weil er einen visuell sehr detaillierten Prototyp fälschlicherweise als User Interface Spezifikation erachtet.

Umgekehrt kann auch der Einblick des Usability Engineers in die Arbeit des Software Entwicklers nützlich sein, etwa wenn der Usability Engineer hierdurch erkennt, dass er einen Prototyp des Software Entwicklers ganz oder in Teilen für seine Zwecke einsetzen kann. Unter Umständen kann sogar ein Prototyp, der vom Software Entwickler bereits verworfen wurde (da dieser beispielsweise überholte Architekturen nutzt) für den Usability Engineer hilfreich sein, da er Funktionalitäten enthält, die im Fokus eines Usability Tests stehen – unabhängig davon, wie diese Funktionalitäten technologisch umgesetzt sind.

Im Rahmen der Kommunikation kann auch geklärt werden, inwieweit gemeinsame Prototypen für beide Bereiche sinnvoll sind. In agilen Umfeldern wird oft eine durch Tests (siehe oben) abgesicherte Entwicklung praktiziert, so dass durch Refactoring Maßnahmen aus dem Prototyp ohne Weiteres das eigentliche Produkt entstehen kann. Hier muss daher keine zusätzliche Zeit für die Erstellung von „Wegwerf"-Prototypen eingeplant werden. Dies kann aber auch bedeuten, dass der Usability Engineer aufgrund der Zeitpläne keine Ressourcen hat, dedizierte Prototypen für Usability Testing zu erstellen. Hier muss er jeweils im Einzelfall prüfen, ob die akuten Fragestellungen mit den verfügbaren Prototypen verfolgt werden können, oder ob zumindest ein basales Prototyping von seiner Seite (beispielsweise in Form von Papierprototypen) angebracht und möglich ist. Außerdem sollte er im Rahmen der oben erwähnten Teammeetings ein Bewusstsein für Usability Engineering im Gesamtteam schaffen, um auf diese Weise indirekt Einfluss auf die Erstellung von Prototypen zu nehmen und so darauf hinzuarbeiten, dass Prototypen erstellt werden, die auch zur Bewertung von Usability Fragestellungen dienen können.

Werden unterschiedliche Prototypen von Usability Engineer und Software Entwickler eingesetzt und gibt es Überschneidungen in den Rezipientengruppen (zum Beispiel bei Vorstandspräsentationen) kann ein gemeinsames Erwartungsmanagement der beiden Teams nach außen sinnvoll sein. Auf diese Weise können die Rezipienten über den Stellenwert von Divergenzen zwischen den Prototypen informiert werden, damit beispielsweise der Software Entwickler nicht unter Druck gerät, wenn der Prototyp des Usability Engineers aufgrund der visuellen Ausgestaltung den Eindruck eines fast fertigen Systems erweckt, obwohl die technische Umsetzung bei weitem noch nicht abgeschlossen ist.

Auch bei separaten Prototypen kann der Informationsaustausch zwischen Usability Engineer und Software Entwickler die Effizienz des Gesamtprojekts verbessern. Dies ist beispielweise dann der Fall, wenn vom Usability Engineer Informationen zu Workflows und Datenstrukturen aus User Task Analysen oder Interviews mit Domänenexperten geliefert werden können, die der Software Entwickler für die Simulationen realer Daten-Umgebungen und Daten-Manipulationen benötigt. In einem solchen Fall liegt die Information bereits „im Team" vor und der Entwickler muss sie nicht extra von „außerhalb" anfordern.

Schließlich können Synergien auch dadurch gefördert werden, dass die Entwicklung des User Interface explizit von anderen Entwicklungsarbeiten abgekoppelt wird und spezialisierte User Interface Entwickler zum Einsatz kommen. Auf diese Weise rückt die Tätigkeit dieser Entwickler näher an diejenige des Usability Engineers und der Entwickler ist weniger durch Programmierarbeiten belastet, die nicht unmittelbar mit dem User Interface in Verbindung stehen. Durch die hierdurch entstehende größere inhaltliche Schnittmenge zwischen Usability Engineer und Software Entwickler sowie durch die Expertise und die Werkzeuge (beziehungsweise Technologien) des spezialisierten User Interface Entwicklers kann die Erstellung von Prototypen gefördert werden, die für beide Parteien sinnvoll einsetzbar sind. Zumindest jedoch wird der Aufwand für Prototyping Aktivitäten, die nur für einen der beiden Bereiche nützlich sind, im Vergleich zur „herkömmlichen" Zusammenarbeit von Usability Engineer und Software Entwickler reduziert.