Systemanalyse in einem Scrum-Projekt

Christian Kraemer, Bernhard Tausch
TNG Technology Consulting GmbH
Betastraße 13a
85774 Unterföhring
{christian.kraemer, bernhard.tausch}@tngtech.com

Thomas Matzner
Berater für Systemanalyse
Beltweg 22
80805 München
tam@tamatzner.de

Abstract: In einem größeren Scrum-Projekt wurden gezielt Konzepte der Systemanalyse, hauptsächlich für das Zusammenspiel mit anderen Anwendungen, erstellt. Wir beschreiben das von uns praktizierte Vorgehen und die erzielten Resultate. Die gewonnenen Erfahrungen zeigen, dass die zu analysierenden Themen sorgfältig ausgewählt werden wollen, um unnötige Aufwendungen zu vermeiden, dass es jedoch eine Reihe von Themenfeldern gibt, für die sich der Analyseaufwand lohnt.

1 Vorstellung des Projekts

Ziel des Projekts ist die Entwicklung mehrerer ineinandergreifender Applikationen zur Umsetzung eines neuen Geschäftsmodells in der Telekommunikationsbranche. Die Applikationen sind eine Web-GUI, ein Backendsystem und ein Datawarehouse. Dabei werden einige externe Schnittstellen zur Abrechnung und zu weiteren Systemen bedient, die nicht mit Scrum entwickelt werden.

Insgesamt besteht das Gesamtentwicklungsteam aus etwa 20 externen und internen Mitarbeitern. Das Gesamtentwicklungsteam teilt sich in drei bis vier Scrum-Teams auf, welche von zwei bis drei Scrum-Mastern betreut werden (die Anzahl der Teams hat sich im Laufe des Projekts mehrfach geändert). Alle Teams besitzen denselben Product Owner. Die Sprintlänge beträgt 2 Wochen, wobei in der Regel nach jedem Sprint die Software auf Produktion installiert wird. Der Projektstart erfolgte im Herbst 2010.

Das Geschäftsmodell arbeitet mit indirekter Vermarktung über Partner. Die Leistungen der entwickelten Anwendung werden sowohl dem auftraggebenden Unternehmen wie auch dessen Vertriebspartner und z.T. den Endkunden zur Verfügung gestellt. Da es sich um ein neues Geschäftsfeld handelt, müssen parallel zur Software auch die Geschäftsabläufe mit Kunden und Partnern erarbeitet werden.

Zur Erstellung des „Look and Feel“ der Applikation wurde eine Design-Agentur beauftragt. Zum Entwurf des initialen Product Backlogs wurden anfangs mit Hilfe des Personas-Konzepts [MY06] imaginäre Benutzer des Systems erstellt. Basierend auf deren Anforderungen wurden User Stories geschrieben und anschließend in einer Story Map [Pat01] strukturiert. Diese erlaubte die Priorisierung und Auswahl der User Stories, so dass im ersten Release alle relevanten Geschäftsprozesse zumindest berührt wurden.

2 Systemanalyse: Arbeitsweise und Ergebnisse

Nach einigen Wochen Entwicklungszeit sah der Product Owner die Notwendigkeit, bestimmte Themen außerhalb des Sprint-Zyklus gründlich zu analysieren. Schwerpunkt hierbei waren, auch in Übereinstimmung mit den Erfahrungen aus anderen agilen Entwicklungen, Schnittstellen zu benachbarten Systemen. Bei diesen konnte nicht vorausgesetzt werden, dass die notwendigen Ansprechpartner aus anderen Unternehmensbereichen oder aus Partnerunternehmen innerhalb der kurzen Sprint-Laufzeiten alle auftretenden Fragen klären würden. Also musste vorausschauend geplant und konzipiert werden.

2.1 Einbindung des Analytikers in den Scrum-Prozess

Zu diesem Zweck wurde über 6 Monate hinweg ein Systemanalytiker part-time zu 60% eingesetzt. Er war räumlich bei dem Scrum-Entwicklungsteam angesiedelt, nahm jedoch nicht an allen Meetings des Teams teil. Seine Einbindung gestaltete sich folgendermaßen:

- Er nahm an den täglichen Stand-ups teil. Dies diente zum einen der allgemeinen Information des Teams über die Themen, zu denen Konzepte entwickelt wurden. Damit war es möglich, etwa vorhersehbare Schwierigkeiten oder Themen, auf die speziell geachtet werden sollte, frühzeitig zu kommunizieren. Außerdem bildeten sich bei dieser Gelegenheit mehrfach ad-hoc-Teams als Sparringspartner des Analytikers, etwa um bestimmte Entwürfe kurzfristig zu reviewen und auf Verständlichkeit und technische Machbarkeit zu untersuchen.

- Die Konzepte wurden frühzeitig, d.h. schon vor der konkreten Beauftragung an die Entwicklung, in Planning Meetings vorgestellt. Daran nahm jeweils ein signifikanter Teil des Entwicklungsteams teil. Ziel war, dem Product Owner frühzeitig ein Gefühl für den groben Aufwand und etwa vorhersehbare Schwierigkeiten bei der Umsetzung zu geben.

- Finalisierte Konzepte wurden in einem Sprint Planning vorgestellt und geschätzt.

Zentrale Dokumentationsplattform für das Projekt war ein dediziert hierfür aufgesetztes Wiki. Da die externen Ansprechpartner meist keinen Zugang zu diesem Wiki hatten, wurden die meisten Schnittstellendokumente als Textdateien erstellt. Auch sie waren,

wie die gesamte Projektdokumentation, zu jedem Zeitpunkt mit ihrem momentanen Bearbeitungsstand über das Wiki einsehbar und kommentierbar.

Für den Umfang der geplanten Konzeptionsarbeiten wurde keine voll ausgebaute Modellierungsplattform für nötig gehalten. Graphische Modelle wurden mit dem freien Editor yEd erstellt, der neben informellen Graphiken auch einfache Teilnotationen für Daten- und Klassenmodelle sowie Prozessmodelle in BPMN anbietet. Da diese Modelle ausschließlich der menschlichen Kommunikation dienten und keine Generierungsmechanismen gefordert wurden, war ein unkompliziert für alle Teammitglieder verfügbares Zeichentool ausreichend.

2.2 Ergebnisse der Analyse

Folgende Inhalte wurden über einen Zeitraum von 6 Monaten erarbeitet:

- Schnittstellenspezifikation zur Versorgung des FiBu-Systems mit Abrechnungsdaten. Hier ging es vor allem darum, die je Geschäftsmodell nötigen Datenumfänge und die genaue Befüllung der einzelnen Datenfelder mit dem empfangenden System abzustimmen. Weiterhin war der technische Übertragungsmechanismus zu klären und zu beschreiben.

- Schnittstellenspezifikation zu Versorgung der Marketing-Anwendung mit Daten über die erreichten Verkäufe in den einzelnen Teilmärkten; gleiche Aufgabenstellung wie beim FiBu-System.

- Im Zusammenhang mit diesen beiden Schnittstellen entstand die Forderung nach einer generellen Historisierung geschäftsrelevanter Daten. Grund war, dass etwa zur Überprüfung von Abrechnungen nicht nur der beauftragte Leistungsumfang eines Kunden zum aktuellen Zeitpunkt, sondern zu einem beliebigen historischen Zeitpunkt nachvollzogen werden musste. Hier wurde eine Reihe von Anforderungen für vorhersehbare Anwendungsfälle erarbeitet, aufgrund derer das Entwicklungsteam ein Framework (Hibernate Envers) für die Historisierung auswählen und ein Nutzungskonzept dafür erarbeiten konnte.

- Spezifikation des Zusammenspiels mit der zentralen Benutzerverwaltung. Dies erforderte intensive Analysearbeit, da es, je nach Partnerunternehmen, zwei Sorten von Benutzern gibt: Solche, die in der zentralen Benutzerverwaltung und nur dort gepflegt werden und solche, die nur in unserem System von Bedeutung waren und nur dort gepflegt werden. Den zentral gepflegten Benutzern, d.h. solchen, die auch andere Anwendungen des Unternehmens nutzen können, soll über das zentrale System ein Single Sign On geboten werden. Da jedoch das zentrale System nicht die geforderte Verfügbarkeit bieten kann, muss in Notfällen auch ein Sign On ohne die zentrale Benutzerverwaltung geboten werden. Dies führte zu einer Reihe von nichttrivialen Prozessabläufen für die einzelnen Einsatzszenarien.

- Modelle für die grundsätzlichen Geschäftsabläufe. Schwerpunkt hierbei war die Verteilung der Funktionalität zwischen den beteiligten Partnerunternehmen und die daraus resultierenden Datenflüsse. Hierfür gab es unterschiedliche Varianten, je nach dem Grad der IT-Unterstützung, die unterschiedlichen Partnern geboten werden soll. Partner, die einen hohen Grad eigener zusätzlicher Leistungen vermarkten, haben ein Interesse daran, möglichst viel Funktionalität in ihren eigenen Anwendungen abzubilden. Sie nutzen die angebotene SaaS-Plattform mit dem minimalen technisch notwendigen Leistungsumfang und übernehmen Daten frühzeitig in ihre eigenen Systeme. Partner mit reiner Reseller-Funktion hingegen nutzen einen Maximalumfang der SaaS-Funktionalität und übernehmen nur die unbedingt notwendigen Daten, etwa für ihre eigene Abrechnung und Buchhaltung.

- Aus den geschilderten Prozessvarianten ergaben sich generische Schnittstellen für die Übernahme und Übergabe von Daten zu den Partnersystemen. Diese wurden, um eine State-of-the-art-Lösung anbieten zu können, im Json-Format definiert. Zusätzlich wurde eine vereinfachte Datenstruktur im CSV-Format angeboten, da erfahrungsgemäß nicht alle IT-Partner die aktuellen Technologien unterstützen.

3 Umsetzung der Konzepte

Aus der Aufzählung im vorigen Abschnitt ist ersichtlich, dass kein vollständiges Fachkonzept im traditionellen Sinn entstand. Vielmehr beauftragte der Product Owner nur solche Konzepte, die nicht innerhalb des regulären Sprint-Zyklus erarbeitet werden konnten und deren Notwendigkeit absehbar war. Aus der Abkopplung vom Sprint-Zyklus folgt umgekehrt, dass die Konzepte nicht unbedingt direkt nach ihrer Fertigstellung umgesetzt wurden.

Generell ist zu beobachten, dass die Stakeholder den Schwerpunkt ihrer funktionalen Wünsche auf die Kernleistung der Anwendung selbst, also die Telekommunikations-Funktionalität, legen und nicht auf die geschilderten prozessübergreifenden Funktionen. Auch machen sie, durchaus erwünscht, von der Möglichkeit, Funktionalität erst kurzfristig zu priorisieren, regen Gebrauch. Auf diese Weise wurden einige Konzepte, für die ursprünglich aktueller Bedarf gesehen wurde, nach ihrer Fertigstellung bis auf weiteres vertagt, da sich inzwischen andere Schwerpunkte bei den Anforderungen ergeben hatten.

Für den Product Owner ergab sich daraus ein schwieriger Balanceakt. Er musste einige Themen vorsorglich analysieren lassen, um im Fall plötzlichen Bedarfs, etwa durch hohe Prozesslast, nicht blank dazustehen. Andererseits konnte er nicht genau vorhersehen, auf welche Themen er sich mittelfristig vorzubereiten hatte. Dies führte ca. 6 Monate nach Abschluss der Konzeptionstätigkeit zu folgenden Ständen der Umsetzung:

- Die FiBu-Schnittstelle wurde kurz nach ihrer Spezifikation realisiert, anhand der Spezifikation getestet und in Produktion genommen. Hier war die

Dringlichkeit offensichtlich, da das Produkt von Anfang an kaufmännisch solide abgerechnet werden muss.

- Die Versorgung des Marketing-Systems wurde mit etwas größerer Verzögerung von einigen Wochen zur Realisierung beauftragt. Hier ergaben sich in Test und Produktion noch Änderungsanforderungen gegenüber dem Konzept, da, anders als bei den operativen Daten für die Abrechnung, die genaue Semantik der Informationen je Teilmarkt erst anhand von realen Daten wirklich verstanden wurde.

- Das Historisierungs-Framework wurde zeitnah eingeführt. Hintergrund hierbei war die Einsicht, dass historische Daten, selbst wenn sie derzeit nicht flächendeckend gefordert werden, hinterher nicht mehr rekonstruiert werden können. Hier wurde also, abweichend von der reinen agilen Lehre, ein Feature ohne direkte Kundenanforderung eingebaut. (Die Anforderungen der beiden genannten zu versorgenden Systeme waren jedoch Spezialfälle der allgemeinen Forderung nach historischen Daten.)

- Das Zusammenspiel mit der zentralen Benutzerverwaltung ist ein Beispiel für die Verschiebung von Prioritäten. Zu Projektbeginn galt sie als absolutes Muss: „Alle unsere Anwendungen machen von dem zentralen Benutzerpool nebst Single Sign On Gebrauch.“ Diese Aussage gilt im Prinzip immer noch – doch lagen die aktuellen Prioritäten bei jedem Sprint auf anderen Funktionen, so dass die Anbindung an die zentrale Benutzerplattform noch nicht absehbar ist. Je nach Betrachtungsweise kann man dies als Verschwendung von Analyseressourcen oder als Einsparung bei der Entwicklung sehen, da bei klassischer Entwicklung vermutlich die Anbindung an das zentrale System frühzeitig gebaut worden wäre, obwohl unsere heutige Erfahrung zeigt, dass das Unternehmen zumindest in der Anfangszeit gut ohne sie auskommt.

- Eine nahtlose Unterstützung der Geschäftsabläufe, etwa durch automatische Übertragung von Geschäftsdaten abhängig vom jeweils gewählten Geschäftsmodell, lässt ebenfalls noch auf sich warten. Dies bedeutet keine grundsätzliche Ablehnung eines solchen Ansatzes. Jedoch waren den beteiligten Geschäftspartnern, die in die Priorisierung der Anforderungen miteinbezogen sind, stets TK-Kernfunktionen wichtiger, und sie bescheiden sich derzeit mit einer einfachen CSV-Schnittstelle, die ihnen auf manuelle Anforderung hin bestimmte Daten über den Kundenauftrag zur Verfügung stellt.

4 Erfahrungen und Empfehlungen

4.1 Besonderheiten des dargestellten Projekts

Unser Projekt hat neben der Tatsache, nach den Scrum-Regeln durchgeführt zu werden – falls dies heute überhaupt noch eine Besonderheit ist – eine Eigenschaft, die man nicht alle Tage vorfindet: Es unterstützt ein für den Auftraggeber völlig neues Geschäftsmodell. Wir müssen also unsere Beobachtungen und die Schlüsse, die wir aus ihnen ziehen, sorgfältig sortieren und uns fragen, welche davon grundsätzlich für ein Scrum-Projekt gelten können und welche dem zu Beginn unbekannten Geschäftsmodell zuzuordnen sind.

Das neue Geschäftsmodell wirkt sich auf die konzeptionelle Tätigkeit sowie auf das Priorisieren von Anforderungen teils hinderlich, teils fördernd aus:

- Hinderlich ist, dass der übliche Rückgrriff, wenn eine gute Lösung für eine Aufgabe gesucht wird, ausfällt. Es nützt nichts, zu fragen, wie der Anwender denn heute arbeitet. Dies wirkte sich bei uns insbesondere bei der Gestaltung der funktionalen Aufteilung und der Datenflüsse zwischen den teilnehmenden Partnern aus. Es gab und gibt bis heute keine Blaupause, die man als Grundlage verwenden und ggf. noch optimieren kann.

- Förderlich ist das Fehlen eines Altsystems, wenn es um die zeitnahe Priorisierung von Funktionen geht. Die Ablösung eines Altsystems bedeutet meist, dass dessen gesamter Funktionsumfang oder zumindest ein erheblicher Teil davon realisiert sein muss, bevor man das neue System erstmals in den realen Einsatz geben kann. Diese Einschränkung hatten wir nicht, d.h. wir konnten tatsächlich mit einem „minimally marketable product“ einsteigen und Feedback aus der Praxis generieren.

4.2 Einbindung eines Analytikers in in Scrum-Projekt

Zu den grundlegenden Spielregeln von Scrum gehört die Lieferung klarer Stories durch den Product Owner an das Entwicklungsteam als Vorgabe für die Implementierung. Die Kernkompetenz eines Analytikers wiederum ist das Gewinnen klarer Strukturen aus den ursprünglich unsortierten und teils widersprüchlichen Aussagen aller beteiligter Stakeholder. Diese sehen sich keineswegs in der Pflicht, ihre Anforderungen von Anfang an klar und widerspruchsfrei zu formulieren.

Bei überschaubaren Entwicklungsvorhaben mag eine einzelne Person die Rolle der Kommunikation mit den Stakeholdern, die Aufbereitung derer Wünsche in klare Stories sowie die Kommunikation mit dem Entwicklungsteam ausfüllen können. Bei größeren Mengen von Stakeholdern und Anforderungen übersteigt die Menge aufzuarbeitender Information rein quantitativ die Möglichkeiten eines Einzelnen. Eine Delegation in das Entwicklerteam hinein ist nach den Scrum-Regeln nicht möglich; eine Rückdelegation

an die Stakeholder ist meist unpraktikabel. Also bietet sich die Unterstützung des Product Owner durch einen Analytiker an.

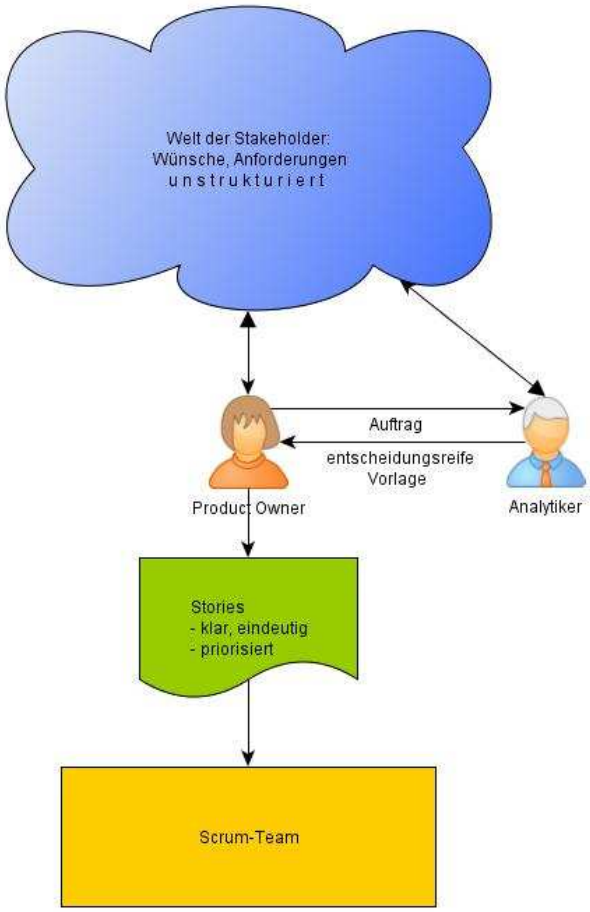

Abbildung 1: Einbindung eines Analytikers in ein Scrum-Projekt

Der Product Owner himmt dabei weiterhin seine Kernverantwortung wahr:

- Er trifft die fachliche Entscheidung für eine Lösung, wenn es zwischen mehreren Möglichkeiten auszuwählen gilt.
- Er priorisiert die Stories und legt damit den funktionalen Umfang für die einzelnen Sprints fest.

Nicht zu seiner Kernverantwortung gehört jedoch die Detailkommunikation mit den Stakeholdern und die inhaltliche Ausarbeitung der Lösungen bzw. ihrer Varianten. Aufgabe des ihn unterstützenden Analytikers kann also sein:

- Kommunikation mit den Stakeholdern, um deren Wünsche zu verstehen.
- Aufdecken und, wenn möglich, Auflösen von Widersprüchen in den Anforderungen.
- Detaillierte Formulierung von Stories.
- Wenn nötig, d.h. wenn keine Lösung erkennbar ist, die allen Wünschen gerecht wird, Erarbeitung mehrerer Lösungsvorschläge und entscheidungsreife Aufbereitung für den Product Owner.

Diese Betrachtung sehen wir für jedes Scrum-Projekt als gültig an, egal ob ein neues oder ein bestehendes Geschäftsmodell unterstützt wird.

4.3 Vorausdenken oder Refaktorisieren?

Unsere Betrachtung im vorigen Kapitel hat gezeigt: Konzeption in einem agilen Projekt ist eine Gratwanderung. Denkt man zuviel voraus, landet man im Extremfall wieder beim Wasserfall und wird schwerfällig im Aufnehmen und Umsetzen aktueller Anforderungen. Verzichtet man ganz auf übergreifende Konzepte, setzt man sich folgenden Gefahren aus, die in unserem Projekt zumindest während der ersten Sprints tatsächlich aufgetreten sind:

- Die Bedienung des Systems ist uneinheitlich. Jeder Anwendungsfall funktioniert für sich einwandfrei. Aber wie findet sich der Benutzer in der Funktionalität zurecht? Wie wird er unterstützt, welche Schritte er auszuführen hat, um auf kürzeste und einfachste Weise zu einem gewünschten fachlichen Ergebnis zu kommen?
- Die Abbildung der Prozesskette ist uneinheitlich und unvollständig. Das kann sich in Mehrfacheingaben derselben Information durch unterschiedliche Nutzer äußern oder in mangelhafter Darstellung des Prozesses und seines aktuellen Bearbeitungsstandes gegenüber den diversen Benutzerrollen.

Bei kompromisslosem Vorgehen in Sprint-Zyklen lautet die Antwort auf diese Probleme: Refactoring. Hat man in einer frühen Story zu kurz gedacht, wird sich dies in den Anforderungen an eine spätere Story niederschlagen, und dann ist die Gelegenheit gekommen, die Software zu verändern.

Doch das Thema Refactoring verdient auch kritische Würdigung. Es verspricht Entlastung, da der Product Owner nicht, wie bei klassischem Vorgehen, gezwungen wird, jede Entscheidung von vornherein unverrückbar richtig zu treffen. (Bzw. auch noch für richtig zu halten, wenn sie sich längst als falsch herausgestellt hat.) Doch jedes

Refactoring verursacht Aufwand, der in die Sprint-Zyklen eingeplant werden muss. Nun gewöhnen sich die Anforderer schnell an die Freiheit, in jeden Sprint ihre aktuell dringendsten Funktionen einbringen zu können. Wenn zuviel Refactoring nötig wird, hat man bald ein oder mehrere Sprints, die sich hauptsächlich dem Korrigieren zuvor schlecht getroffener Entwurfsentscheidungen widmen. Kurz gesagt: Agiles Vorgehen bedeutet, falsche Annahmen korrigieren zu können, keineswegs aber kostenneutral. Es lohnt sich nach wie vor, wichtige Entwurfsentscheidungen sorgfältig vorzubereiten.

In dem geschilderten Projekt hat Refactoring in vielen Fällen gut funktioniert. Es gibt aber auch Grenzen, vor allem dort, wo fachliche Designentscheidungen direkt den Anwender betreffen und nicht mehr zurückgenommen werden können, etwa bei Rufnummernplänen und der Struktur von Telefontarifen.

4.4 Empfehlungen für die Systemanalyse

Deshalb empfehlen wir, solche Funktionalitäten außerhalb des engen Sprint-Rhythmus vorzudenken,

- die Auswirkungen auf erheblich mehr als eine oder wenige Stories haben, insbesondere Querschnittsthemen,

- und deren Umgestaltung durch Refactoring erheblichen Aufwand erfordern würde.

Konkret sollten sich die konzeptionellen Tätigkeiten auf folgende Felder konzentrieren:

- Die grundlegenden Prozessketten, nicht bis ins kleinste Detail, vielmehr die Arbeitsabläufe, die einzelne Stories miteinander verbinden: Wer tut was, wodurch ausgelöst? Hierzu gehört auch die Steuerung der Prozesse, etwa in Form von Task-Listen für die einzelnen Benutzerrollen, und deren Visualisierung. Diese Empfehlung ist für vollständige Neuentwicklungen, wie in unserem Beispielprojekt, nur begrenzt umsetzbar, sofern die Prozessketten von den Entscheidungsträgern im Business noch nicht überblickt und entschieden werden können.

- Schnittstellen zu anderen Anwendungen, die sich idealerweise folgerichtig aus der Prozessbetrachtung ergeben: Welche Funktionen werden wo durchgeführt und welche Daten müssen zu diesem Zweck fließen?

- Das grundlegende Benutzungskonzept. Hier ist sogar Detaillierung angebracht, damit ähnliche Arbeitsabläufe nicht bei jeder Story korrekt, aber jeweils andersartig abgewickelt werden. Etwa: Wie verläuft das Zusammenspiel zwischen Prüfung eingegebener Daten, Information des Benutzers über fehlende oder unplausible Daten und die Möglichkeit, eine Bearbeitung abzuspeichern und zu beenden? Welche Arbeitsschritte kann man grundsätzlich rückgängig machen? Diese Empfehlung kann auch für neue Geschäftsfelder gelten, findet jedoch ihre Grenzen, wenn etwa noch nicht klar ist, ob eine

Funktion durch erfahrene Power-User oder durch gelegentliche Benutzer ausgeführt wird.

- Historisierung – Daten, die nicht vom ersten Produktivtag an gesammelt werden, sind unwiderruflich verloren.

- Berechtigungskonzepte, sofern die Anwendung hier vorhersehbar hohe Anforderungen an die Differenzierung stellt. Grund ist, dass es nahezu aussichtslos ist, eine differenzierte Berechtigungsprüfung, etwa anhand von Kriterien wie Benutzerrolle, Prozessschritt und diversen Dateninhalten, nachträglich durch Refactoring einzubringen.

Literaturverzeichnis

[Pat01] http://www.agileproductdesign.com/presentations/user_story_mapping/index.html (Zugriff am 12.04.2012)

[MY06] Mulder, S.; Yaar, Z: The User is Always Right: A Practical Guide to Creating and Using Personas for the Web. 2006