R. Keil-Slawik, H. Selke, G. Szwillus (Hrsg.): Mensch & Computer 2004: Allgegenwärtige Interaktion. München: Oldenbourg Verlag. 2004, S.

Requirements-Analyse und Spezifikation bei Individualsoftware

Friedrich Strauß

sd&m AG

Zusammenfassung

Die Anforderungsermittlung insb. zur Benutzerschnittstelle ist bei Individualsoftware eng mit ihrem Design verbunden. Durch den direkten Kontakt zu den Benutzern kann die Anforderungsermittlung und Spezifikation effektiv miteinander verschränkt werden. Diese Verschränkung ist wichtig, um die Umsetzbarkeit von Anforderungen prüfen zu können und widersprüchliche Anforderungen frühzeitig zu entdecken. Durch geeignete Techniken kann mit den Benutzern geprüft werden, ob seine genannten Anforderungen sinnvoll sind und ob eine Spezifikation und insb. das Design der Benutzerschnittstelle ihren unterschiedlichen Anforderungen geeignet erfüllt.

1 Einleitung

Die hier beschriebenen Erfahrungen beziehen sich auf betriebliche Informationssysteme, die als Individualsoftware von sd&m konzipiert und implementiert werden. Typischerweise sind die Projekte zwischen 5 und 50 Bearbeiterjahre groß.

In unseren Projekten strukturieren wir in die Phasen, Spezifikation (Fachkonzept), Konstruktion (DV-Konzept / technisches Design), Realisierung, System-Test & Integration, Einführung. Vorgelagert gibt es Studien und Grobkonzepte, die ein Thema strukturieren oder Strategien konkretisieren und geeignete Projekte definieren.

Statt einer vollständigen Anforderungs-Analyse als eigenständige Phase vor der Spezifikation, wird die Anforderungs-Analyse mit der Spezifikationsphase verwoben. Am Anfang der Spezifikation – bei komplexen oder unklaren Themen ggf. in einer Vorab-Studie – steht die Festlegung einer kurzen Anforderungs-, Ausgrenzungs- und Prämissenliste (AAP-Liste) im Vordergrund. Diese AAP-Listen von typischerweise 3 mal 5-25 Punkten definieren vor allem den Umfang des Systems, als Basis für eine Beauftragung durch den Kunden.

Die detaillierten Anforderungen werden implizit dokumentiert, indem die Spezifikation des Systems entsprechend erweitert wird. Dies hat zwei wichtige Vorteile: Zum einen werden

inkonsistente Anforderungen viel früher und einfacher erkannt, da sie direkt in das entstehende Daten- GUI- oder Funktionsmodell integriert werden.

Abbildung 1 verdeutlicht die zeitliche Verschränkung von Anforderungsermittlung und Spezifikationsphase. Nach den AAPs werden zuerst die Anwendungsfälle identifiziert aber dann parallel zu Benutzerschnittstelle und Datenmodell spezifiziert und detailliert. Geschäftsprozessanalyse und -modellierung sind optionale Aktivitäten. Zu beachten ist, dass die Anforderungsanalyse nicht direkt mit der Erstellung des AAP-Dokuments endet. Zur Spezifikationstechnik von sd&m siehe auch [1,2].

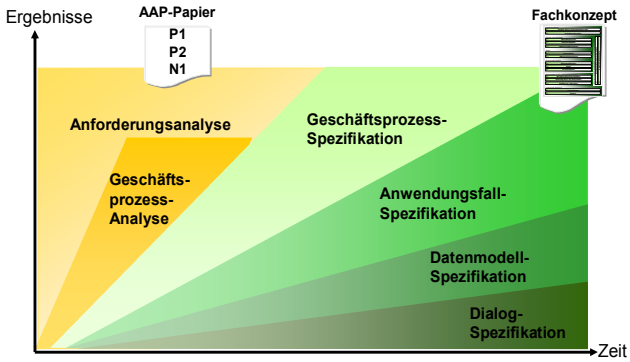

Abbildung 1: Zeitlicher Aufriss in der Spezifikation.

These 1: Steht der Kunde bei der Anforderungsermittlung und Spezifikation „ausreichend" zur Verfügung, dann sollte mehr Wert auf eine widerspruchsarme und verständliche Spezifikation gelegt werden, als zuerst eine detaillierte Anforderungsermittlung zu erarbeiten.

Mögliche Probleme: Ist der Systemumfang nicht klar abgestimmt, kann sich die Spezifikation in die Länge ziehen (ungeplante Themen enthalten oder Schleifen drehen). Gegenmaßnahme: AAP-Dokumente.

Eine etablierte Requirementsdokumentation wie zum Beispiel die Volere Snowcards [4] wird bei sd&m nicht genutzt. Bei Volere werden Anforderungen einzeln als Snowcard mit Begründung, Messkriterium für die Umsetzung, Klassifikation und Anforderer beschrieben. Dies führt bei unseren Projektgrößen zu mehr als Hundert ggf. auch mehr als Tausend Anforderungen die dokumentiert und verwaltet werden müssen.

Probleme bei diesem Verfahren sind der hohe Dokumentationsaufwand und vor allem die Unübersichtlichkeit: Bei diesem Vorgehen können Anforderungskonflikte bei der Analyse nur schwer entdeckt werden, da die einzelnen Anforderungen unverbunden nebeneinander stehen. Viele fachliche Konflikte ergeben sich aber erst aus einer Kombination von mehreren

Anforderungen. Benutzer haben zudem Schwierigkeiten, die Auswirkungen von einzelnen Anforderungen abzuschätzen und können Anforderungen auch schlecht priorisieren. Insbesondere entstehen so schnell viele sogenannte goldene Henkel Anforderungen.

2 Anforderungen an die Benutzerschnittstelle

Wir haben oben beschrieben, dass das explizite Dokumentieren von detaillierten Anforderungen aufwändig ist. Konkrete Anforderungen an die Benutzerschnittstelle sind besonders kompliziert zu dokumentieren: Eine Anforderung zu beschreiben ohne eine Lösung vorwegzunehmen, führt schnell zu sehr abstrakten oder zu umständlich Texten. Use-Szenarien sind vielleicht noch die praktikabelste Technik um vor allem Anforderungen an zu unterstützende Abläufe zu formulieren.

Gerade für Dialogoberflächen gibt es sehr effiziente Methoden, um statt einer expliziten Anforderungsermittlung direkt ein GUI-Design zu erstellen bzw. abzuändern sowie das erstellte Design zu evaluieren. Bei sd&m wird also das in ISO 13407 definierte Vorgehen im Groben durchgeführt, allerdings ohne eine echte Dokumentation der Anforderungen zur Benutzerschnittstelle.

Für die Designerstellung bieten sich Paper-Mockups für den Ablauf und das Grob-Design an, sowie GUI-Prototypen für die Detailphase der Spezifikation. Beim Durchspielen eines Anwendungsfalls anhand eines Mockups oder Prototypen prüft man, ob das Design die kritischen Anwendungsfälle geeignet unterstützt. Benutzer des späteren Systems können erheblich einfacher Probleme eines GUI-Designs identifizieren als Anforderungen benennen. Im folgenden erläutern wir, wie sd&m diese Methoden beim Erstellen der (GUI-) Spezifikation einsetzt und welche Vorteile und Nachteile sie besitzen.

These 2: Konkrete Anforderungen zur Benutzerschnittstelle können nur anhand erster Designs explizit genannt werden. Hier ist eine Anforderungsermittlung meist nur als abstrakte Anforderungen zu einfachen Erlernbarkeit, Geschwindigkeit der Bedienung, Fehlerrate usw. möglich. Das Erarbeiten eines möglichen Designs liefert beim Durchspielen /Bewerten wichtige nicht-erfüllte Anforderungen

Mögliche Probleme:

1. Details der GUI werden frühzeitig diskutiert (Farbe, Detail-Layout, usw.) → Nutzung von Low-Fidelity Prototypen wie Paper-Mockups (Beschreibung zur Nutzung bei sd&m und möglicher Probleme siehe unten)

2. Neuartige Designlösungen werden nicht entwickelt, da der Kunde ihm bekannten Lösungen haben will. → Design-Iteration ggf. intern und nur Design-Evaluation mit dem Kunden durchführen.

In den folgenden Abschnitten beschreiben wir die wichtigsten Erfahrungen für die Methoden zur Erstellung und Iteration von Benutzerschnittstellen (Paper-Mockups, GUI-Prototypen) sowie für den Evaluationsschritt (User-Boards) bei der Iteration nach 13407.

2.1 Paper-Mockups

Schwerpunkt bei den Paper-Mockups ist das Grob-Design inkl. Ablauflogik, d.h. wie verteilt sich die Bearbeitung auf einzelne Fenster bzw. Karteikarten und wie verhält sich der Dialog. Durch die Nutzung von Papier, Ausdrucken, Post-Ist, Schere usw. wird ein Dialog viel schneller änderbar als mit DV-Unterstützung. Eine gute Einführung zu Mockup-Techniken bietet [5].

Echte Paper-Mockups werden dabei in manchen Projekten nicht zusammen mit dem Fachbereich, sondern nur intern im Team erstellt. Dies hat den Vorteil, dass Design-Ideen in Ruhe ausprobiert und umgesetzt werden können; die Fachbereichsmitarbeiter des Kunden prüfen, ob das neue Design geeigneter ist. Hierzu setzen wir häufig User-Boards ein, die jedoch abhängig vom Kunden unterschiedlich durchgeführt werden (siehe unten).

Paper-Mockups werden bei sd&m für GUIs erst in letzter Zeit häufiger eingesetzt. Das Haupthindernis ist die scheinbar einfache Nutzung von GUI-Buildern. Ohne weitere Ergonomie-Kenntnisse greifen viele Mitarbeiter gerne schon am Anfang zu einem GUI-Builder, um schnell optisch ansprechende Oberflächen zu entwerfen. Mittlerweile wird aber die Paper-Mockup-Technik in den sd&m-internen Schools GUI-Gestaltung und Spezifikation geschult (seit 3 bzw. 2 Jahren). Ein einmaliges Ausprobieren in einer Schulung reicht bei vielen Kollegen schon aus, um es dann später auch in echten Projekten einzusetzen.

Wichtigster Vorteil ist neben der einfachen Änderbarkeit der klare Low-Fidelity Ansatz: Ablenkende Diskussionen um Farben, Ausrichtung von Widgets Beschriftungen, fehlende Elemente wie Statuszeilen, Menüs usw. werden so gut wie nie geführt: Den Kundenmitarbeitern ist klar, dass dies noch nicht das endgültige Design ist. Dieser Vorteil bleibt erhalten, wenn man beim Feindesign nicht zu einem echten GUI-Builder, sondern zu Mal-Werkzeugen wie Powerpoint wechselt.

2.2 GUI-Prototypen

Keine Spezifikation kommt ohne GUI-Prototypen aus. Ein GUI-Prototyp umfasst zumindest die statische Präsentation einzelner Fenster, er kann ggf. auch funktionale Ausschnitte umsetzen und vereinfachte Datenbanktabellen enthalten (zum Beispiel mit Microsoft Access programmiert). In diesem Sinne sind die oben beschriebenen Paper-Mockups genauso GUI-Prototypen, wie Visual-Basic Entwürfe oder die Programmierung der Oberfläche in der Zielplattform.

Fast alle Informationssysteme sind sehr datenlastig: Das Dialog-Design ist hier immer auch ein Kampf um das Layout, wobei viele Daten gleichzeitig darzustellen sind und man trotzdem eine strukturierte, übersichtliche Oberfläche mit geeigneten Widgets definieren möchte.

Mit einem Paper-Mockup (bzw. einem Malwerkzeug) kann man nicht sinnvoll entscheiden, ob noch drei weitere Felder auf eine Karteikarte passen oder nicht. Ist das (erste) Grobdesign fertig, dann lohnt bei solchen Fragestellungen der Wechsel zu einem GUI-Builder. Sind doch noch Änderungen am Grobdesign nötig, sollte man ruhig zur Papier-Mockup Technik zurückwechseln – unter Nutzung von zurechtgeschnittenen Ausdrucken!

Problematisch ist der Bau von Oberflächenprototypen mit GUI-Buildern und insb. Access, weil es dazu verleitet Funktionalität einzubauen. Bei Nutzung von Access werden gerne Tabellen angelegt und mit Testdaten gefüllt, sowie elementare Funktionen in Basic programmiert.

Der Aufwand dieser Prototypprogrammierung – die über die Erstellung statischer Präsentationen hinausgeht – zahlt sich häufig nicht aus. Da das System sehr datenlastig ist und sich die Anforderungen in der Spezifikation doch noch häufig ändern, erreicht der Prototyp selten eine ausreichende Qualität, um ernsthafte Fragestellungen zu beantworten. Hier sind ein oder zwei zusätzliche User-Board WKS effektiver, außerdem wird die Konzentration auf die Spezifikation so auch besser sichergestellt – beim Prototyp-Programmieren aber ohne ein paar schöne Schnörkel auszukommen, fällt wahrscheinlich nicht nur sd&m-Mitarbeitern schwer.

Auch das von Projektleitern manchmal anvisierte Sparziel, den Prototyp-Code aus der Spezifikation in der Programmierung weiterverwenden zu können, wird selten erreicht, da doch viele Korrekturen und ggf. Redesigns nötig sind.

Sinnvoll ist ein funktionaler Prototyp, um neue Konzepte *exemplarisch* vorzuführen. Die Nutzung von dynamischen Fensterbereichen und Split-Panes versteht ein Sachbearbeiter am einfachsten am Beispiel.

Statt eines GUI-Builders wird manchmal auch Visio oder auch Powerpoint für das Design der Oberfläche genutzt. Hier ist die Bedienung noch einfacher, die Konzentration auf das Wesentliche fällt wegen der Low-Fidelity Technik einfach. Zudem sind die Grafiken leider bequem skalierbar, was zu einer gefährlichen Situation führen kann: Erst die spätere Programmierung mit Standardcontrols zeigt, ob man alle Widgets in der Standardgröße unterbringen kann. Nicht immer findet sich eine so einfach Lösung wie in einem ernsten Problemfall im Bankbereich: statt eines aufwändigen Redesigns, wurden die 25 Benutzer mit extra großen Monitoren ausgestattet.

2.3 User-Boards

User-Boards sind bei uns Workshops mit 1-6 Benutzern des späteren Systems. In diesen Workshops werden Dialogentwürfe auf gute Bedienbarkeit und Vollständigkeit bzgl. der nötigen (GUI-) Funktionalität evaluiert. Außerdem nehmen 2-4 Personen des Projektteams (inkl. Kunden-IT) teil, ein Ergonomie-Experte wird nur in kritischen Projekten hinzugezogen. Die Ausgestaltung der Workshops ist in Abhängigkeit von Kunde und Projekt sehr unterschiedlich. Zwei wichtige Kernelemente gibt es aber immer: Die Dialogentwürfe werden auf großen Papierausdrucken (teilweise größer als DIN A0) oder auf Overhead-Folie

mitgebracht. Zu einem Teilbereich werden jeweils von den Benutzern praktische Anwendungsfälle genannt (oder vorbereitet mitgebracht). Die Bearbeitung wird dann mit den Dialogentwürfen anhand konkreter Daten durchgespielt. Dabei werden bei Papierausdrucken gerne Post-Its zum Beschreiben genutzt. Dagegen war das Durchspielen eines Prototyps am Beamer häufig weniger effektiv. Anscheinend hilft der Einsatz von Low-Fidelity Techniken, um sich auf die wichtigen Aspekte zu konzentrieren.

Das *konkrete* Durchspielen zusammen mit dem Fachbereich ist eine der effektivsten Maßnahmen. Dadurch können wir überflüssige Funktionen finden, inkonsistente oder unvollständige Abläufe entdecken und Hinweise zu anderen Bereichen der Spezifikation wie dem Datenmodell oder auch zu Geschäftsprozessen erhalten.

Die durchgespielten Beispielanwendungsfälle (Use Szenarien) haben wir manchmal explizit dokumentiert, sind damit aber wenig glücklich geworden. Im Nachhinein stellte sich häufig raus, dass die Dokumentation nach der Erstellung nicht weiter genutzt wurde. Die richtige Granularität der Beschreibung zu finden, ist eine der Herausforderungen. In den Workshops einigte man sich in der Regel schnell auf bestimmte interessante fachliche Konstellationen und für das Spezifikationsteam reichte in der Regel die Dokumentation der Datenkonstellation oder ggf. die nötige Bearbeitungsreihenfolge.

Diese User-Boards sind in der wissenschaftlichen Literatur als Review- bzw. Evaluationstechniken mit unterschiedlichen Schwerpunkten unter vielen Namen bekannt. Cognitive-Walkthrough ist die gängige Bezeichnung bei der Evaluation durch einen einzelnen Benutzer. Unser Vorgehen entspricht meistens dem *simplified Thinking aloud* wie es zum Beispiel im *Discount Usability Engineering Approach* von J. Nielsen [3] beschrieben ist.

Zusammenfassung

Wir haben erläutert, warum eine Mischung aus Anforderungsdokumentation durch Anforderungen, Ausgrenzungen und Prämissen AAPs für den Systemumfang zusammen mit einer Spezifikation sinnvoller ist als eine eigenständige Anforderungs-Analyse und - dokumentation. Voraussetzung ist jedoch eine stabile Definition des groben System-Umfangs und ein direkter Zugang zu den späteren Benutzern. Ersteres wird zumindest ein externer Auftragnehmer schon aus Eigeninteresse möglichst gut definieren, dies ist bei neuen Kunden bzw. neuen Themen einer der größten Fallstricke für die erfolgreiche Projektabwicklung. Letzteres ist wegen der Individualentwicklungssituation meist auch mehr oder weniger einfach möglich.

References

Kretschmer, D., Krug, W., Stutz, C. Siedersleben, J.: Analysis beyond UML. In Proceedings of the RE'02: IEEE Joint International Requirements Engineering Conference 2002, S. 215.

Krug, W., Siedersleben, J: Bausteine zur Spezifikation. In Siedersleben J. (Hrsg.), Softwaretechnik, Hanser (2002).

Nielsen, J., Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier. In Bias, R. G., Mayhew, D. J., Cost-Justifying Usability. Academic Press (1994).

Robertson, J. und S.: Mastering the Requirements Process. London: Addison-Wesley (1999).

Snyder, C.: Paper Prototyping. Morgan Kaufmann (2003)