18 Heidelberg Eye Explorer - Die technologische Neuausrichtung Erfahrungsbericht aus einem Reengineering-Projekt

Martin Moro
Lehrstuhl für Wirtschaftsinformatik III, Universität Regensburg
Martin.Moro@wiwi.uni-regensburg.de

Abstract

In diesem Papier wird sowohl die gewählte Vorgehensweise als auch die erlebten Schwierigkeiten in einem Reengineering-Projekt vorgestellt. Der Schwerpunkt liegt dabei auf der Vermittlung von Erfahrungen und der Motivation für eine Standardisierung der Entwicklungsprozesse. Abgerundet wird mit Vorschlägen für einen Bewertungsprozess zur Unterstützung bei der Entscheidung zwischen divergierenden Ansätzen.

Schlüsselworte: Software Engineering, Prozess, nichtfunktionale Anforderungen, Qualität, Bewertung, Designalternativen

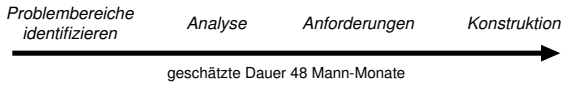
18.1 Einleitung und Problembereich

Die Firma Heidelberg Engineering entwickelt seit 10 Jahren Technologien zur Bildaufnahme im Bereich des Auges. Es entstanden Lasererkennungssysteme zur Abtastung von Auge, Netzhaut, Sehnerv und Blutgefäßen. Zur Visualisierung der Messdaten wurde die Software Eye Explorer [1] entwickelt.

Kunden von Heidelberg Engineering sind Augenarztpraxen und Kliniken. Während Augenarztpraxen nur ein Aufnahmegerät und einen PC benutzen, unterhalten Kliniken mehrere Aufnahmegeräte sowie eine netzwerkweite Software-Installation. Der historisch gewachsene Eye Explorer ist technologisch nicht für den Einsatz in Kliniken vorbereitet. Die wachsende Nachfrage gab aber den Impuls für ein Reengineering. Für die Bewältigung des Reengineerings wurde eine Kooperation aus der Firma Heidelberg Engineering, der Uniklinik Regensburg und dem Lehrstuhl für Wirtschaftsinformatik III geschlossen.

18.2 Der Weg zur neuen Softwarearchitektur

Schnell wurde klar, dass das Vorhaben weite Teile des Eye Explorers betreffen wird. Zu Beginn wurde daher folgende grundsätzliche Vorgehensweise überlegt:



18.2.1 Problembereiche identifizieren

Der erste Arbeitsschritt definierte anzustrebende Verbesserungen aus der Sicht des Softwareanwenders. Wichtig war die intensive Kommunikation mit allen Beteiligten, insbesondere mit dem Uniklinikum Regensburg. Ergebnis war eine Liste an konkreten Schwierigkeiten in der Anwendung des Eye Explorers. Unterstützung lieferte hierzu eine bereits bei der Firma Heidelberg Engineering geführte Liste an Supportanfragen. Viele Informationen stammen aus Interviews mit den Systembetreuern und Ärzten der Klinik.

Die Punkte auf der so erhaltenen Liste wurden priorisiert und nach ihrer Machbarkeit abgeschätzt. Einge-

brachte Wünsche zur Erweiterung der Funktionalität wurden aussortiert aber für eine spätere Berücksichtigung vorgemerkt. Relevante Anforderungen finden sich demnach vielmehr im nicht-funktionalen Sektor, bspw. eine Verbesserung der Flexibilität im Bezug auf unterstützte DB-Schnittstellen oder auch der Anbindung von Clients über das Internet.

18.2.2 Analyse

Ziel der Analyse war es, die Software zu verstehen und Probleme auf der Ebene des Sourcecodes zu erkennen. Als Struktur der bisherigen Softwarelösung wurde dabei extrahiert:

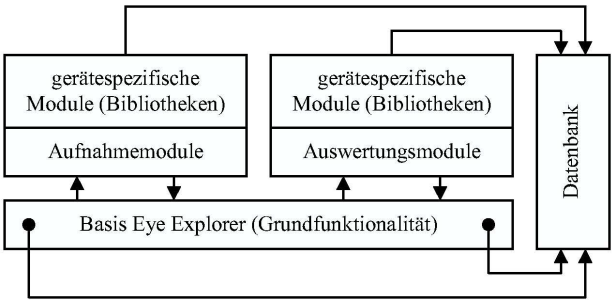


Abbildung 18.1: Architektur der bisherigen Software

Die wichtigsten Probleme ließen sich wiederkehrend in der Software nachweisen. Hauptproblempunkte waren die hochgradigen Verflechtungen innerhalb der Module (in der Abbildung 18.1 als Pfeile angedeutet), insbesondere der Zugriff auf die DB war verstreut. Viele Module benutzten eigene Datenstrukturen und speicherten Ergebnisse zwischen, so dass die DB nicht immer aktuell anzutreffen war. Die Probleme gaben einen Hinweis auf die betroffenen Softwaremodule (448.000 von 563.000 LOC, Anteil von ca. 80 %).

Im Projekt basierte die Analyse auf dem Sourcecode und geringer technischer Dokumentation. Dies erwies sich bei der Analyse durch Externe (Lehrstuhl) als Defizit, ein Softwaremodell wäre zum besseren Verständnis hilfreich gewesen. Kompensiert wurde dies durch eine Vielzahl an Gesprächen und Reviews.

18.2.3 Anforderungen

Neben den funktionalen Anforderungen waren die Qualitätsanforderungen ausschlaggebend:

1. Sowohl für Einzelplätze als auch für Netzwerke geeignet (ca. 80 % sind aber Einzelplätze).
2. Bisheriger Eye Explorer leicht migrierbar.
3. Datenbank flexibel austauschbar.
4. Auf wachsende Anforderungen skalierbar.
5. Offen für die Ankopplung angrenzender Systeme.

Zusätzliche Anforderungen des Reengineerings:

1. Die bereits entwickelten Module sollen beibehalten werden (Schutz des geleisteten Aufwandes).
2. Als Entwicklungssprache gilt weiterhin C++.

Erst die iterative Überarbeitung in einer Reihe von ausführlichen Gesprächen führte zu einer ausreichend vollständigen Liste. Für die Entscheidungsfindung einer optimalen Architektur war diese aber notwendig.

18.2.4 Konstruktion

Begonnen wurde bei der Konstruktion mit einer informellen Sammlung von Lösungsideen. Die potentiellen Ansätze wurden als Story-Cards notiert, um eine Diskussion am runden Tisch zu ermöglichen. Die Story-Cards enthielten dabei eine Bezeichnung der Lösungsidee, eine Beschreibung, eine schematische Zeichnung und eine Auflistung aller bis dahin bekannten Vor- und Nachteile im Projekt. Für einen schnellen Überblick bewährte sich ein Umfang von zwei Seiten. Im Projektverlauf wurden die Story-Cards (vgl. auch [2]) mit neuen Erkenntnissen bestückt, so dass mitunter auch ein Umfang von bis zu 50 Seiten zu finden war (vgl. Abbildung 18.2 für ein Beispiel einer Story-Card).

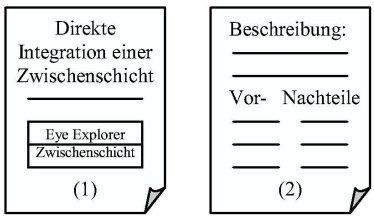


Abbildung 18.2: Beispiel einer Story-Card

Der Vergleich der verschiedenen Lösungsansätze stellte sich insbesondere bei der Prognose des Erfüllungsgrades an nicht-funktionalen Anforderungen als schwierig heraus. Wie sollte z.B. die geforderte Flexibilität sichergestellt werden ? Und: Wie sollte dies den Lösungsansätzen in dieser frühen Phase angesehen werden ? Gelöst wurde dies durch Rücksprache mit Experten, Recherchen in Erfahrungsberichten zu den Technologien lieferten wertvolle Informationen aus der Praxis. Verfahrensweisen zu denen keine Materialen gefunden werden konnten, wurden durch Prototypen verifiziert.

Als nützlich erwiesen sich Szenarios zur Quantifizierung nicht-funktionaler Anforderungen (vgl. [3, S.267], [4, S.33]). durch die jeweils interessierten Personen. Beispielsweise wurde die nicht-funktionale Anforderung der Flexibilität der DB-Schnittstellen auf Oracle, MySQL und MS SQL Server beschränkt. Die nicht-funktionalen „weichen“ Anforderungen wurden damit erst konkret greifbar. Ergebnis war eine Kriterienmatrix der Lösungsansätze mit einer Gegenüberstellung ihrer Eigenschaften. Der Vergleich führte zu der Erkenntnis, dass nur eine Mischung aus den Ideen zum Erfolg führen kann.

Als für dieses Projekt optimale Lösung etablierte sich eine Form einer nachrichtenbasierten Middleware. Diese ist auf die Arbeitsumgebung anpassbar. Für Einzelplätze ist sie direkt in die Software linkbar, Netzwerke lässt sie über Protokolle kommunizieren. Durch die Nutzung von C++ kann bereits entwickelter Sourcecode wiederverwendet werden. Die Verwendung des Protokolls XML ermöglicht eine Öffnung des Eye Explorers für angrenzende Systeme. Die Integration über den Linker hingegen sichert den Einzelplätzen die geforderte Leichtgewichtigkeit.

18.3 Die neue Softwarearchitektur

Für die neue Softwarearchitektur wurden die „Rosinen aus den zuvor gefundenen Lösungsideen gepickt“. Eingezogen wurde eine neue Plattform, die sowohl die fachlichen Klassen als auch die Kommunikation mit der DB kapselt (siehe Abbildung 18.3).

Fertiggestellt ist die neue Plattform sowie der Applikationsserver in seiner Grundfunktionalität, sowie der Test auf Funktionalität. Ausstehend ist der Test auf die Erfüllung nicht-funktionaler Anforderungen. Der Aufwand bis heute liegt bei ca. 14 Mann-Monaten. Noch durchzuführen ist die Integration der Plattform in den Eye Explorer. Bei einem geschätzten noch zu erbringenden Aufwand von 12 Mann-Monaten bewegt sich das Projekt über der Hälfte der Fertigstellung.

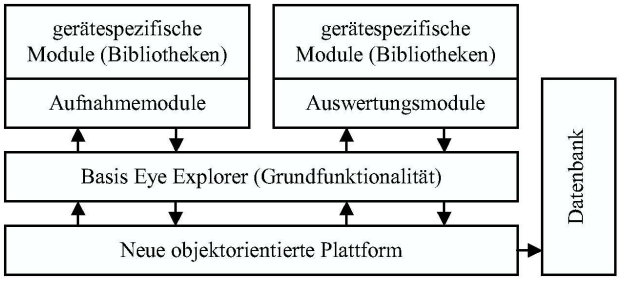


Abbildung 18.3: Struktur der neuen Softwarearchitektur

18.4 Standardisierungsmöglichkeiten im Prozess

Vorgehensmodelle wie der Rational Unified Process [5] helfen bei den ersten drei Tätigkeiten, aber wenig bei der Konstruktion. Insbesondere im vorliegenden Projekt waren Schwierigkeiten bei der Suche nach der optimalen Lösung zu verzeichnen.

Ein standardisierter Bewertungsprozess kann bei der Bewertung von Lösungsansätzen auf die Erfüllung nichtfunktionaler Anforderungen bereits während dem Softwareentwurf helfen. Er wendet visuelle Bewertungsverfahren und Metriken an, um Problembereiche aufzuspüren und Lösungsalternativen zu vergleichen. Eine zur Zeit entstehende Dissertation beschäftigt sich ausführlich mit diesem Thema.

18.5 Zusammenfassung und Ausblick

Das vorgestellte Reengineering-Projekt ist typisch für derartige Entwicklungsvorhaben. Die Balance aus der Ausrichtung auf neue Technologien und dem Schutz von entwickelten Komponenten zu finden war eines der Kernpunkte im Entwicklungsprozess. Schwierigkeiten bereitete weiterhin der Vergleich alternativer Lösungsideen. Die Prognose des tatsächlichen Verhaltens von Softwareentwürfen in der späteren Software ist ein schwer mit Werten zu stützendes Unterfangen.

Die Kernidee für eine optimale Lösung beruht auf dem Gedanken, während des Entwurfs in der Softwarearchitektur „Stellschrauben“ vorzusehen, an denen während der Implementierung bestehende Designdefizite ausgeglichen werden können. Dieser Gedanke scheint durchaus auf verwandte Projekte übertragbar. Die gewonnenen Erfahrungen fließen aktuell in eine Dissertation ein, die sich gezielt mit der Problematik der Bewertung von Softwareentwürfen beschäftigt.

Literatur

[1] Eye Explorer. `http://www.hdeng.de/h2e`, 2004

[2] Bennicke, Marcel; Rust, Heinrich: Messen im Software-Engineering und metrikbasierte Qualitätsanalyse. `http://www.visek.de`, 2004

[3] Rupp, Chris: Requirements-Engineering und -Management. Professionelle, iterative Anforderungsanalyse für die Praxis. München, Wien: Carl Hanser Verlag, 2002

[4] Clements, Paul; Bachmann, Felix; Bass, Len; Garlan, David; Ivers, James; Little, Reed; Nord, Robert; Stafford, Judith: Documenting Software Architectures - Views and Beyond. Addison-Wesley, 2002

[5] IBM Rational: Rational Unified Process Whitepapers. `http://www.ibm.com/software/rational`, 2004