Reengineering-Ziele:

Kapselung, Anpaßbarkeit und Funkionserweiterung

has-program service AG, Spohrstraße 6, 22083 Hamburg

Ingrid Wetzel, Universität Hamburg, Fachbereich Informatik, Arbeitsbereich Softwaretechnik, Vogt-Kölln-Str. 30, 22527 Hamburg, wetzel@informatik.uni-hamburg.de

1 Einleitung

Im folgenden wird der Reengineeringprozeß für ein Kontokurrentbuchhaltungssystem vorgestellt. Ziel diese Beitrags ist es, in einem ersten Schritt die vielfältigen Aufgaben, die innerhalb eines Reengineeringprozesses auftreten können, zu identifizieren. Obwohl in einem konkreten Projekt entstanden, erscheinen diese Aufgaben in gewisser Weise stellvertretend für Anforderungen/Aktivitäten in vielen aktuellen Projekten. In einem zweiten Schritt wird das konkrete Vorgehen im Projekt kurz aufgezeigt. Eine Interim-Auswertung weist auf wesentliche Probleme und Einschätzungen des Projektes hin. Das Projekt wird von einem größeren Softwarehaus durchgeführt.

Ausgangspunkt ist ein in COBOL geschriebenes Nebenbuchungsprogramm für den Versicherungsbereich, das bisher in Projekten unter großem Anpassungsaufwand eingesetzt wurde.

Zu dem Reengineeringprojekt kam es im Wesentlichen aus zwei Gründen:

- Die Buchhaltung soll im Rahmen einer OO Neuentwicklung eines Standardbestandssystems für Makler (mit Partner-, Produkt-, Vertrag-, Schaden- und Objektkomponenten) integriert werden. Die Integration steht wegen der Ersteinsatzfähigkeit bei einem konkreten Kunden vor der 2000 Wende unter enormem Zeitdruck.
- Die Buchhaltung soll als Stand-alone System in verschiedenen Softwareprojekten (also über das Standardbestandssystem hinaus), die in dem Softwarehaus durchgeführt werden, zum Einsatz kommen. Weiter ist geplant, die Nebenbuchhaltung für Versicherer und Makler als eigenständiges Produkt auf den Markt zu bringen.

2 Identifikation von Aufgaben

Um den beiden obengenannten Zielen (Integration in Bestandssystem, Stand-alone Produkt) zu genügen, wurde eine Architektur geschaffen, die weitreichende Änderungen an den COBOL-Sourcen zur Folge hat. Die einzelnen Aufgaben sind hierbei die folgenden:

- *Kapselung der Buchhaltung*

  Die Kapselung der Buchhaltung ist mit verschiedenen Standardschnittstellen zu versehen. Insbesondere ist eine Standardschnittstelle vom Bestandssystem zur Buchhaltung zu konzipieren, durch die die Buchhaltung „versorgt“ wird, ohne – wie bisher – auf ein Bestandssystem direkt zuzugreifen. (Die Buchhaltung benötigt aus dem Bestandssystem z.B. detaillierte Informationen über Verträge mit Verteilungsplänen, etc.). Dies bedeutet eine grundlegende Veränderung aller Zugriffe innerhalb der einzelnen COBOL Verbucher. Es erfordert die Auswahl und Festlegung der Standardschnittstelle, wobei Erweiterungen des Standards wegen komplexer Vertragsstrukturen, die innerhalb des Bestandssystems angeboten werden, notwendig sind.

  Weitere Schnittstellen hinsichtlich Hauptbuch, Personalabrechnung und Ex- und Inkasso sind ebenfalls zu bedienen.

  Eine eigene Problematik tritt auf, Zusatzkonzepte für das Identifizieren wesentlicher Objekte der beiden zu integrierenden Systeme zu erstellen, z.B. Kunden und Kontokurrentkonten. Wann entstehen Kontokurrentkonten, wieviele Kunden sollen Kunden zuordenbar sein. Wie werden aus Sollstellungen mit bestimmten Vorgängen die in der Buchhaltung notwendigen Belegarten und Buchungsschlüssel ermittelt?

- *Anpaßbarkeit und Funktionserweiterung*

  Die neue Kontokurrentbuchhaltung soll nicht nur, wie bisher, Versicherer in ihrer Nebenbuchhaltung unterstützen, sondern auch maklerfähig sein. D.h. die Verbucherlogik ist an dezidierten Stellen zu verändern bzw. so zu gestalten, daß sie je nach Einsatz in unterschiedlicher Weise verbucht. Unterschiede treten z.B. bei der Prämien- und Provisionsverbuchung auf, da einerseits – bei Versicherern – Prämien als Einnahme des Versicherer gebucht werden, andererseits – bei Maklern – die Weiterleitung von Prämien an Versicherer notwendig ist.

  Daneben wirken sich Erweiterungen hinsichtlich eines komplexeren und flexibleren Vertragskonzeptes des Standardsystems auf die Verbucherlogik aus, die zu Erweiterungen führen (z.B. mehrere Versicherungsnehmer pro Vertrag, mehrere Versicherer pro Vertragsbaustein). Hier muß geprüft werden, inwieweit diese Erweiterungen die Verbuchungslogik der „üblichen“ Vertragskonzepte subsumieren.

- *Neuimplementierung der Dialogschnittstellen*

  Die Buchhaltung benötigt Dialogschnittstellen für manuelles Buchen und Kontenpflege, die mit einer graphischen Oberfläche gestaltet werden. Sie werden im Rahmen des Projektes auf der vorhandenen OO-Plattform realisiert und sind daher abhängig von einer bestimmten Frameworkarchitektur, was einen Bruch bei der Einbindung in andere Projekte bedeutet. Außerdem erweist sich als Problem, inwiefern Teile des manuellen Buchens eher zum Bestand- oder zum Buchhaltungssystem gehören, da sie auf das Bestandssystem zugreifen müssen und dadurch eine strenge Kapselung durchbrechen.

- *Anpaßbarkeit an kundenindividuelle Buchungseinstellungen*

  Die Anpaßbarkeit hinsichtlich kundenspezifischer Buchungseinstellungen (Kontokurrentkonten, Buchungsarten, - schlüssel, Belegarten), die durch eine umfangreiche Steuerungstabelle ermöglicht wird, ist bei dem Reengineering zu restrukturieren, da die Tabelle durch ständige Erweiterungen nahezu nicht mehr wartbar ist [2].

- *Allgemeine fachliche Erweiterungen*

  Bei dem Reengineeering wird die Buchhaltung zusätzlich um Mehrwährungsfähigkeit und Mehr sprachlichkeit erweitert.

- *Zukünftige Zertifizierung*

  Alle Aktivitäten sollen so durchgeführt werden, daß sie eine zukünftige Zertifizierung unterstützen.

3 Vorgehen im Projekt

Langfristiges Projektziel ist es, die Nebenbuchhaltung durch sukzessive Veränderungen schrittweise in C++ neu zu implementieren. Zu Beginn des Projektes stand die Überlegung an, direkt eine Neuimplementierung vorzunehmen. Dies wurde jedoch wegen der Unüberschaubarkeit des Aufwandes und der Personalressourcen nicht in Angriff genommen.

Die vorhandenen Aufgaben/Zielsetzungen führen zu folgenden, z.T. parallelen Aktivitäten:

- *Wahl einer Standardschnittstelle und ihre Erweiterung*

  Die Auswahl und Festlegung von Erweiterungen des Standards (GDV) um spezifische Sätze (800-Sätze) für das Vertragskonzept des Bestandssystems ist im Zusammenhang mit den Erfordernisses des Verbuchers festzulegen. Da das Vertragssystem parallel entwickelt wird, lassen sich mehrfache Überarbeitungen nicht vermeiden.

- *„Liften“ der Zugriffe auf Bestand und Steuerungstabelle*

  Alle Zugriffe der COBOL-Programme sind auf fachlicher und auf technischer Ebene zu ändern. Hierzu ist die Festlegung von Datenmodellen für die Standardschnittstelle und die Steuerungstabelle erforderlich. Auch hier ist die Festlegung durch häufige Änderungen geprägt.

- *Transformationsprogramme zum Befüllen der Schnittstelle*

Innerhalb dieser Transformationsprogramme sind Zugriffe auf das Bestandssystem notwendig, die früher aus der Buchhaltung selbst vorgenommen wurden.

- *Erweiterung der Verbucherlogik in dreifacher Hinsicht*

  Die Erweiterungen sind zum einen hinsichtlich einer Maklersicht vorzunehmen, die in den Altsourcen völlig fehlte. Zum anderen ist eine erweiterte Funktionalität wegen des komplexeren Bestandssystems erforderlich. Daneben sind Mehrwährungsfähigkeit und Mehrsprachlichkeit zu realisieren.

- *Zusatzkonzepte*

  Zusatzkonzepte sind für das Identifizieren wesentlicher Objekte der beiden zu integrierenden Systeme festzulegen sowie die Verwaltung minimaler redundante Informationen in der Nebenbuchhaltung zu realisieren.

- *Realisierung eines Objektmodells für Tabellen der Buchhaltung*

  Zur Implementierung der Dialogschnittstellen sind für alle internen Tabellen des Nebenbuchungssystems Objektmodelle anzufertigen, auf denen die Dialoge implementiert werden.

- *Festlegung der Dialoge*

  Die manuelle Buchungsschnittstelle ist sowohl in ihren Abläufen als auch in den durch die Makler- und Bestandssystemerweiterungen notwendigen Funktionen neu zu konzipieren.

4 Interim-Auswertung

Die Fülle der auftretenden Änderungsanforderungen ist Grund genug, diese Reengineeringsmaßnahme als komplex einzustufen. Hinzu kommt die Abhängigkeit der Aufgaben untereinander (z.B. die Festlegung der Schnittstelle hat Auswirkungen auf das Datenmodell innerhalb des Buchhaltungsprogramms und die Zugriffe der Verbucherprogramme). Das Verstehen der Abhängigkeiten und die Koordination und Feinabstimmung der einzelnen Aktivitäten, die parallel von verschiedenen Teams zu konzipieren und realisieren sind, stellt somit eins der Hauptprobleme während des Prozesses dar.

Eine weitere Erschwernis besteht in der – wie so häufig – ungenügenden Dokumentation der Alt-Sourcen [1]. Wesentliche Grundlage der Koordination ist es daher, die Dokumentation des bestehenden Buchhaltungsprogramms inklusive Steuerungstabelle zu verbessern. Hier wird eine Dokumentation auf verschiedenen Granularitätsstufen ausgearbeitet, die fachliche Anforderungen mit Verbucherlogik und Anpassungsmöglichkeiten verbinden. Es wurden Schulungstage für die Anpassung der Steuerungstabelle abgehalten, aus denen heraus ein Anpassungsleitfaden sowohl für interne Mitarbeiter, die Projekte begleiten, als auch für Kundenorganisationen entsteht.

Zu dem jetzigen Projektzustand (das Projekt befindet sich in der Endphase der Integration in das Bestandssystem, wobei einige Anpassungserfordernisse nicht berücksichtigt wurden) ist noch nicht abschließend ersichtlich, inwieweit der beschrittene Weg eines schrittweisen Übergangs zu einer Neuimplementierung einer direkten Reimplementierung vorzuziehen ist. Bisher sind die Verbucher weiterhin in COBOL verändert worden, während alle weiteren Teile der Architektur in C++ realisiert sind. Aufgrund der vielfältigen Ziele beim Reengineering sind in einer eingehenderen Projektauswertung über die in [3] darstellten Maße Metriken aufzustellen, die Wiederverwendbarkeits- oder Reimplementierungsentscheidungen in Zukunft überschaubarer machen.

Referenzen:

[1] L. Richter: Wiederbenutzbarkeit und Restrukturierung o d e r Reuse, Reengineering und Reverse Engineering, in: Wirtschaftinformatik 34, 1992, S. 127-136.

[2] H. M. Sneed: Metriken für die Wiederverwendbarkeit bestehender Software-Systeme, In: Softwaretechnik-Trends, Vol. 18, No. 4, 1998, S. 19-22.

[3] H. M. Sneed: Economics of Software Re-engineering, in: Journal of Software Maintenance, Vol. 3, No. 3, 1991, S. 163-182.