Wiedergewinnung von Geschäftsregeln aus einem Legacy Anwendungssystem

Harry M. Sneed
SoRing Kft. H-1221 Budapest
Harry.Sneed@T-Online.de

Abstrakt: Dieser Beitrag ist ein Praxisbericht über ein Reverse-Engineering Projekt für eine internationale Bank mit einem Legacy Software System. Das System, um das es hier geht ist mit der Sprache COBOL implementiert und stützt sich auf das IBM-IMS Datenbanksystem, ein hierarchisches Datenbanksystem aus den 70er Jahren. Mit der gleichen Datenbanksoftware wird auch der Dialog mit den Endbenutzern unterstützt. Da diese Middleware, bzw., DB/DC System mittlerweile in die Jahre gekommen ist, wird geplant die Anwendungssoftware abzulösen. Als Voraussetzung dafür soll das bestehende System nachdokumentiert werden. Diese Nachdokumentation, bzw. Reverse Engineering, ist ein eigenständiges Projekt mit eigenen Zielen. Ein Ziel ist die Wiedergewinnung der fachlichen Logik aus dem alten Code, bzw. die Extraktion der Geschäftsregel. Dieser Kurzbericht schildert das Problem im Allgemeinen und beschreibt, wie hier vorgegangen wird, die Regeln wiederzugewinnen.

Schlüsselwörter: Migration, Reverse Engineering, Business Rules, Rule Extraktion, COBOL, Gherkin.

1 Was sind Geschäftsregel?

Der Begriff „Business Rule“ wurde in der 80er Jahren von IBM im Rahmen ihres Business Process Planning Projekt. Aus der Sicht der damaligen Business Systemplaner war eine Geschäftsregel eine betriebliche Vorschrift, nach der alle IT-Anwendungen zu richten hatten, z.B.:

- Kunden sind wöchentlich über den Stand ihres Kontos zu informieren, oder
- Rechnungen sind am Ende eines jeden Monats auszusenden.

Diese Regel sind in allen IT-Systemen einzubauen, die von dem Anwender entwickelt werden.
Geschäftsregel können eine betriebsinterne oder eine externe Quelle haben. Extern vorgegebene Regeln folgen aus allgemeinen Gesetzen und branchenspezifischen Regulationen wie der Mindeststand des Eigenkapitals einer Bank. Interne Regeln werden von dem Anwenderbetrieb selbst vorgegeben, z.B. die Grenze eines Überziehungskredits. Geschäftsregeln sind unabhängig von der jeweiligen Anwendung, da sie übergeordnete Gültigkeit haben. Daher wird angestrebt sie in eine übergeordnete Rule-Repository abzulegen und auf der Codeebene von einem sogenannten Rule-Engine ausführen zu lassen.

Normalerweise sind Geschäftsregeln *präskriptiv* und werden Top-Down vorgegeben. Für die einzelnen Applikationen sind sie verbindlich und als solche Teil der Anwendungsspezifikation. D.h. zuerst besteht die Regel und es obliegt den Entwicklern sie in dem Code einzubauen oder von dem Code aus auf sie zuzugreifen. Geschäftsregel können aber auch deskriptiv sein. In diesem Falle beschreiben sie, wie ein Ereignis zustande kommt, z.B. ein Kontobesitzer erhielt eine Mahnung wenn sein Koton die Überziehungsgrenze überschreitet. Ein mittelgroßes IT-System produziert mehrere Hundert solcher Einzelergebnisse. Eine Angestellte bekommt eine Gehaltserhöhung, eine Kunde bekommt eine Rechnung, ein Kontobesitzer erhielt eine Mahnung. *Deskriptive* Geschäftsregel beschreiben, wie diese Ergebnisse zustande kommen, bzw. wann und wie sie erstellt werden.
Wer ein bestehendes IT-System ablösen möchte, sie es durch eine Neuentwicklung, eine Re-Implementierung, eine Konversion oder durch eine Standardsoftware, muss wissen, was das abzulösendes System leistet, d.h. welche Ergebnisse es wann und wie produziert, d.h. er braucht die deskriptiven Geschäftsregel.

2 Erkennung deskriptiver Regel

Der erste Schritt auf dem Weg zur Entdeckung der deskriptiven Geschäftsregel ist die Bestandsaufnahme aller Ergebnisse. Ergebnisse sind auf zwei Ebenen zu entdecken – auf der Ebene der Datengruppen und auf der Ebene der einzelnen Daten. Eine Datengruppe wäre die Kontodaten, ein einzelnes Datum wäre der Kontostand, z.B. als Datenbanksatz gespeichert, als Bericht gedruckt oder als Bildschirmmaske angezeigt werden. Einzelne Datenwerte werden gesetzt, benutzt oder abgefragt. Hier geht es darum die gesetzten Werte zu ermitteln. Sie können errechnet oder einfach überschreiben werden.
Die Ausgabe einer Datengruppe sowie das setzen eines Datenelements kann bedingt oder unbedingt erfolgen. Ist sie bedingt, gehören die Bedingungen zu der Regel wie das Ergebnis zustanden kommt. Es können auch weitere Datenelemente dazu kommen, Datenelemente, deren Werte in das Endergebnis einfließen. Solche Zwischenergebnisse gehören auch zu der Regel, wie ein bestimmtes Endergebnis zustande kommt, z.B:
der Rechnungsbetrag enthält den Warenpreis, den Mehrwertsteuer und den Diskont. Der Diskont kommt aber nur dazu, wenn der Kunde ein Stammkunde ist und die Ware zu einem bestimmten Warensortiment gehört. Bei der Erzeugung eines Ergebnisses sind in der Regel mehrere andere Daten und mehrere Bedingungen

beteiligt. Die endgültige Geschäftsregel umfasst eine Kette Zwischendaten und eine Reihe verschachtelten Bedingungen, z.B.:
Rechnungsbetrag = Warenpreis + (Mehrwertsteuer – Diskont) if (Ware ist Mehrwertsteuerpflichtig & Ware ist Diskontware & Kundenstatus ist Stammkunde)

3 Veränderung der Datennamen

In diesem Beispiel sind die Datennamen erkenntlich. In alten Programmen sind die Datennamen oft verstümmelte Abkürzungen, die keiner mehr versteht, nicht einmal der ursprüngliche Entwickler. Eine wichtige Voraussetzung für die Wiedergewinnung der Geschäftsregel sei also die Umbenennung der Variabelnamen. In der hier geschilderten Reverse Engineering Projekt werden die Namen nicht im ursprünglichen Source ersetzt, wie im vorhergehenden Projekt in Österreich sondern erst in den Regeln selbst. Die neuen sprechenden Namen wurden aus dem Beschreibungen im Data Dictionary entnommen. Darüber wurde in einem früheren WSRE-Beitrag berichtet [Sneed WSRE-2015].

4 Wiedergewinnung elementarer Regeln

In diesem Rule Recovery Projekt wurden die Regel für die Erstellung der Einzelergebnisse in fünf Stufen ermittelt. In der ersten Stufe werden die Sourcen analysiert und alle Variabel die in jedem Source Modul verarbeitet werden, zusammen mit der Art ihrer Verarbeitung – ob Eingabe, Ausgabe oder beides – in einer Datentabelle abgelegt. In einem zweiten Durchgang wird die Entscheidungslogik – die Bedingungen – aus dem Code entnommen und als Entscheidungsbaum mit Zuweisungsanweisungen zwischengespeichert.
In der zweiten Stufe werden die lokalen Regeln erkannt und dokumentiert. Für jede Ausgabevariabel wird der Pfad zu ihrer Zuweisung samt Bedingungen herausgeschält. Festgehalten wird, in welchem Codeabschnitt, unter welchen Bedingungen die Ausgabevariabel gesetzt wird. Es kann vorkommen, dass die gleiche Variabel an mehreren Stellen zugewiesen wird. Sie werden alle registriert und dokumentiert für jeden Source Modul.
In der dritten Stufe geht es darum, den Pfad zu dem jeweiligen Source-Modul zu skizzieren. Ein COBOL Modul kann von beliebig vielen übergeordneten Moduln aufgerufen werden. Die Modulaufrufe – Call Anweisungen – werden aus dem Source Code herausgeholt und in eine Call-Tabelle gespeichert. Modulaufrufe können bedingt oder unbedingt sein. Falls sie bedingt sind, werden ihre Bedingungen mitgespeichert. Das Ergebnis dieser Stufe sind die Modulaufrufe samt ihren Vorbedingungen.
In der vierten Stufe werden die Modul-Aufrufspfade zu dem Modul in dem die ausgewählten Ausgabe-Variabel gesetzt ist, mit den lokalen Pfaden vereinigt. Dadurch entsteht ein durchgängiger Pfad mit Bedingungen von dem Eintritt in das Programm über mehrere Modulstufen hinweg bis hin zu dem Modul an der Stelle, wo die gewählte Variabel zugewiesen wird. Dies ist die Geschäftsregel für jene Variabel.
Der gleiche vierstufige Prozess wird für jede elementare Ausgabevariabel wiederholt.

5 Ermittlung der groben Geschäftsregel

Grobe Geschäftsregel sind die Pfade zu den Objektausgaben. Hier geht es nicht um einzelne Datenvariable, sondern um Datengruppen – Segmente, Sätze, Tupels, Masken, Berichte, usw. Als erstes Kommt es darauf an, die groben Ausgaben im Source Code zu erkennen. Sie werden in Write, Insert, Store, Display und Print Anweisungen ausgegeben. Diese Anweisungen werden wie bei den elementaren Ausgaben durch die Source Analyse erkannt und in Tabellen zwischengespeichert. Danach folgt der gleiche Prozess wie bei den elementaren Ausgaben. Die Ausgabe-Operationen werden zunächst mit ihrem Codeabschnitt und ihrer Vorbedingungen verknüpft und anschließend mit ihren Aufrufpfaden vereinigt. Das Ergebnis ist ein Pfad bzw. Codescheibe, von dem Eintritt in das Programm bis zu Ausgabestelle des Datenobjekts einschließlich aller Bedingungen.

6 Generierung der Regeldokumentation

Die Erstellung der Programmdokumente und der Programmbeziehungstabellen wird von dem Tool Softredoc gemacht. Redoc verarbeitet in diesem Falle sowohl die COBOL/IMS als auch die Natural/ADABAS Sourcen. Aus jedem Source-Modul wird ein Struktogramm zur Darstellung der Entscheidungslogik, ein HIPO Diagramm zur Darstellung der Ein- und Ausgabendaten, einen Perform-Baum aller Subroutinenaufrufe eine Call-Tabelle aller Unterprogrammaufrufe und ein Verzeichnis aller verwendeten Daten abgeleitet. Das neue Regelverzeichnis enthält die Regel zu jeder Ausgabe-Variabel. Die Regel sind nach den Namen der Ausgaben geordnet und können über diesen zugegriffen werden. Zurzeit sind die Regeln nur in der Zielsprache formuliert. Es ist aber geplant sie auch in Gherkin Ausdrücke zu übersetzen damit Anwender ohne Kenntnis der Programmiersprache sie auch verstehen können. If und Case Bedingungen werden in „Wann Dann“ und Schleifen in „Solange als“ Ausdrücke umgesetzt. Egal wie sie formuliert werden, bleiben die deskriptiven Regel am Ende wegen der originalen verschachtelten und verkorksten Programmierlogik schwer verständlich.

7 Nutzung der wiedergewonnenen Regel

Sämtliche wiedergewonnene Geschäftsregel werden in eine Business Rule Repository abgelegt. Sie sind dort über den Datennamen abzufragen. Wer wissen will, wie ein bestimmtes Ergebnis z.B. die Errechnung eines Preises zustande kommt, kann dort nachfragen. Dies ist auf jeden Fall besser als durch Tausende Codezeilen durchzukämmen um zur gleichen Aussage zu kommen.