Jan Jürjens, Kurt Schneider (Hrsg.): Software Engineering 2017,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 33

Software-Reengineering in der Versicherungsbranche: Ein Praxisbeispiel

Arnaud Fietzke[1] und Benedikt Mas y Parareda[2]

1 Einleitung

Digital natives erwarten von ihren Versicherern eine radikal andere *User Experience*: Kundenportale, Automatisierung und mobile Frontends stellen neue Herausforderungen dar. Bestehende Legacy-Systeme sind oft nicht in der Lage, diesen Erfordernissen gerecht zu werden und der Druck zur Modernisierung der gewachsenen Anwendungslandschaften steigt. Modernisierungsansätze von Legacy-Systemen reichen vom kompletten Reengineering über automatisierte Ansätze bis zur Ablösung durch Produkte.

Während die Einführung von Produkten insbesondere bei der Abbildung regulatorischer Anforderungen Vorteile verspricht, erscheint das Erzielen von Wettbewerbsvorteilen durch Produkte fraglich. Zudem liegen häufig alleine die Kosten für die Datenmigration in das Zielprodukt unrealistisch hoch — schnell im zweistelligen Millionenbereich.

In automatischen Transformationsansätzen hingegen wird die Ergebnisqualität im Wesentlichen durch die Qualität des Quellsystems bestimmt, signifikante architektonische Verbesserungen lassen sich kaum erreichen. Somit bleibt unklar, ob durch automatische Ansätze lediglich ein Technologiewechsel möglich ist oder ob auch den Herausforderungen der Digitalisierung begegnet werden kann.

Angesichts der Größe und Komplexität bestehender Legacy-Systeme mag manuelles Reengineering auf den ersten Blick wenig realistisch erscheinen. Wir stellen jedoch ein konkretes Projekt vor, in dem ein umfangreiches Legacy-System effizient und risikoarm migriert wurde, wobei das Neusystem nicht nur eine bessere Qualität als das Altsystem aufweist und geringere laufende Kosten verursacht, sondern zudem eine große Anzahl neuer Anforderungen implementiert und so den Grundstein für eine moderne IT-Landschaft legt.

2 Reengineering einer Versicherungs-Software

Unser Kunde ist ein Spezialversicherer im Agrarsektor. Kernstücke der Anwendungslandschaft sind eine ursprünglich als Standardprodukt entwickelte monolithische Applikation zur Produkt-, Partner-, Vertrags- und Schadensverwaltung, sowie eine auf einer

[1] itestra GmbH, fietzke@itestra.com
[2] itestra GmbH, mas@itestra.com

proprietären GIS[3]-Komponente basierende Anwendung zur mobilen Schadenserfassung durch Sachverständige. Beide Systeme sind seit ca. 15 Jahren im Einsatz und in dieser Zeit kontinuierlich angepasst worden. Inzwischen sind sie technisch und fachlich veraltet und stellen Hindernisse für die Geschäftstätigkeit des Kunden dar. Die zu modernisierenden Anteile sind im Wesentlichen in NATURAL implementiert, modernere Systemteile sind in C# und Visual Basic umgesetzt. Vervollständigt wird der Umfang durch einige closed source-Komponenten.

2.1 Vorgehen

Schlüssel zum Erfolg ist die Use Case-basierte Extraktion von Geschäftslogik durch Reverse-Engineering der bestehenden Anwendung, unterstützt durch Tools zur statischen Analyse und Annotation des Legacy-Codes. Ergänzend findet eine Analyse vorhandener Dokumentation statt, welche jedoch nicht zwingend erforderlich ist — zu häufig entspricht die Dokumentation nicht mehr dem tatsächlichen Anwendungszustand.

Im Rahmen des Projekts wurden Hauptsoftware und mobile Schadenserfassung mit zeitgemäßer Java-Technologie neu entwickelt, inklusive der Umsetzung von ca. 300 neuen fachlichen und technischen Anforderungen. Die Vorgehensweise ist iterativ, je Iteration wird ein fachlicher Funktionsblock analysiert, realisiert und dokumentiert. So werden hohe Steuerbarkeit, minimales Risiko und schneller Mehrwert durch kurzfristige Produktivsetzung bereits migrierter Komponenten erreicht.

Um bereits migrierte Komponenten früh produktiv einsetzen zu können, müssen bis zur vollständigen Ablösung Alt- und Neukomponenten parallel betrieben werden, mit einer gemeinsamen Datenbankinstanz. Um dies zu ermöglichen und dennoch die Neukomponenten auf einem modernen, bereinigten Datenmodell aufsetzen zu können, werden in der Datenzugriffsschicht der neuen Software modulare Adapter verwendet, die Modelländerungen kapseln und nach Ablösung entfernt werden.

2.2 Ergebnis

Im vorliegenden Fall konnte die komplette Migration und funktionale Erweiterung von Clients und Backend mit einem Gesamtaufwand von ca. 2.000 PT durchgeführt werden. In Summe entstanden über 200k SLoC[4] redundanzfreier Java Code und eine Server-Komponente, die auf Basis eines einfachen Tomcat-Servers auf Linux kosteneffizient betrieben werden kann.

Es konnte zudem gezeigt werden, dass der gewählte Ansatz sehr gut skaliert, da individuelle Use Cases parallel migriert werden können. So ist auch die Migration signifikant größerer Systeme in verhältnismäßig kurzen Zeiträumen möglich.

[3] Geoinformationssystem
[4] Source Lines of Code