Domänenspezifische Editoren für versicherungsmathematische Berechnungen

Matthias Gutheil
gutheil@itemis.de
itemis AG

Bernhard Stadler
stadler@itemis.de
itemis AG

Zusammenfassung

In diesem Artikel beschreiben wir, wie wir in zwei Projekten mit Hilfe von domänenspezifischen Editoren evolutionär die Art und Weise verbessert haben, wie versicherungsmathematische Berechnungen umgesetzt werden. Im einen Fall wurde eine bestehende, schwer lesbare textuelle Syntax ersetzt durch eine an mathematische Notation angelehnte Darstellung. Im anderen Fall lösten wir Textverarbeitung als Werkzeug zur Spezifikation von versicherungsmathematischen Berechnungen durch domänenspezifischen Editoren ab. Die Modelle wurden schrittweise aus Programmcode extrahiert und automatische Codegeneratoren wurden eingeführt. Dies führte beim Kunden zu einer erheblichen Verkürzung der Time-to-Market und einer starken Reduktion der Entwicklungskosten.

1 Einleitung

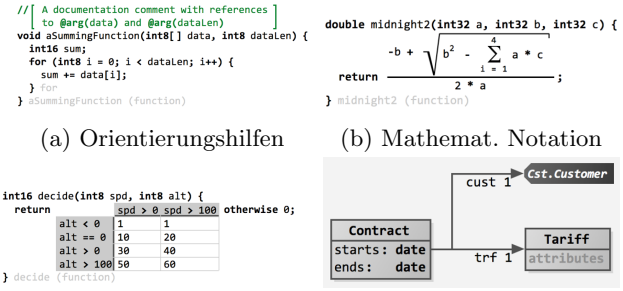
Die Entwicklung von Versicherungsprodukten ist im Kern mathematischer Natur und die entsprechenden Berechnungen für die Versicherungsportfolios sind in Softwaresystemen umgesetzt. Dementsprechend wichtig sind produktive und effiziente Werkzeuge und Prozesse für die Entwicklung dieser Softwaresysteme. Wir zeigen in zwei Beispielen, wie domänenspezifische Editoren die Effizienz und Benutzerfreundlichkeit der Entwicklung verbessern und so die Time-to-Market und die Entwicklungskosten reduzieren können.

2 Domänenspezifische Editoren

Während manche Editoren domänenunabhängig sind, wie etwa Baum-Editoren oder Tabellen, zeichnen sich domänenspezifische Editoren dadurch aus, dass die Darstellung und Bearbeitung an die Domäne angepasst ist. Als speziell auf die Entwicklung domänenspezifischer Sprachen und Editoren ausgerichtete Platformen gibt es Language Workbenches wie das Meta Programming System von JetBrains [1]. Abbildung 1 zeigt verschiedene mit MPS entwickelte Editoren. Besonders die mathematische Notation und die Entscheidungstabelle finden Verwendung in Editoren für die Versicherungsbranche.

Zu den Stärken von MPS zählen:

- Konsequente Integration von Metamodellierung, domänenspezifischen Sprachen und Editoren, Modelltransformationen und Programmierung
- Die Benutzerfreundlichkeit bei der Bearbeitung von domänenspezifischen Modellen
- Die unkomplizierte Erstellung und Wartung von Sprachen und Editoren
- Die Möglichkeit, bestehende Sprachen durch eigene Sprachelemente und zugehörige (Teil-)Editoren beliebig zu erweitern.



(a) Orientierungshilfen (b) Mathemat. Notation

(c) Entscheidungstabelle (d) Diagramme

Abbildung 1: Editor-Ausschnitte in MPS

3 Projekt 1 – GIC

In einem Projekt für einen deutschen Versicherungskonzern GIC (anonymisiert) haben wir erfolgreich MPS eingesetzt, um Entwicklungsprozess und -werkzeuge zu verbessern.

Ursprünglich wurden versicherungsmathematische Berechnungen aus fachlichen Konzepten in Form von Spezifikationen in einer Textverarbeitung konkretisiert. Diese Spezifikationen bestanden aus einer Mischung von natürlichsprachlichem Text und Pseudocode und wurden von einem externen Dienstleister ausprogrammiert. Dies hatte sowohl entwicklungs- als auch prozessbezogene Nachteile: Die Textverarbeitung bot keine adäquate Benutzerunterstützung wie Auto-Vervollständigung, Überprüfung von Referenzen oder „Go To Definition“. Darüber hinaus war die Spezifikation unpräzise, da beispielsweise die Pfade zu Attributen nicht immer eindeutig angegeben waren. Auch neigten Spezifikation und Implementierung dazu, auseinanderzudriften, da bisweilen abweichende Umsetzungen in der Spezifikation nicht nachgeführt wurden. Die manuelle Implementierung der Spezifika-

tion führte zu Verzögerungen und zu hohen Entwicklungskosten.

Abbildung 2: Domänenspezifischer Editor

In Abbildung 2 ist eine Spezifikation des im Projekt entwickelten domänenspezifischen Editors zu sehen. Äußerlich sieht sie ähnlich aus wie in der vorher verwendeten Textverarbeitung, jedoch werden die vorher aufgezählten Nachteile vermieden: Auto-Vervollständigung ist in MPS immer inklusive. Ungültige Referenzen kann man gar nicht erst eingeben. Die Spezifikation ist eindeutig und wird verwendet um ausführbaren Programmcode zu erzeugen. Hierdurch können zum einen Spezifikation und Implementierung nicht mehr auseinanderdriften und zum anderen entfällt der separate Implementierungsschritt. Dies reduziert Entwicklungsdauer und -kosten drastisch.

4 Projekt 2 – i2S

In einem größeren Projekt für i2S[1], einem portugiesischen Hersteller von Versicherungssoftware, hatten wir die Aufgabe, die IBML (Insurance Business Modelling Language) neu zu implementieren. Die IBML (siehe Abbildung 3) ist eine von i2S selbst entwickelte Sprache zur Berechnung von Werten (z.B. Prämie, Steuer, Provision), zur Validierung und zur Definition von Workflows.

Über all die Jahre entstanden in verschiedenen Systemteilen unterschiedliche IBML-Dialekte, verschiedene IBML-Editoren und Ausführungsengines. Die IBML konnte nur textuell bearbeitet werden, was für Versicherungsmathematiker, die Formeln gewohnt

[1] http://i2s.pt

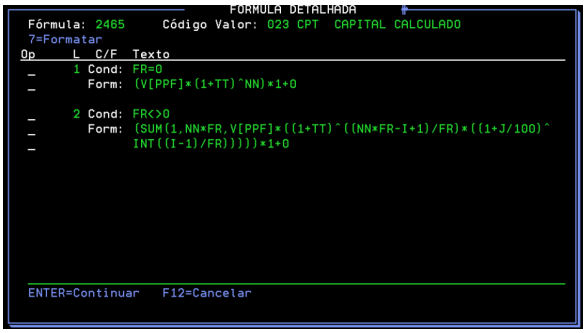


Abbildung 3: IBML

sind, sehr beschwerlich war. Neben den funktionalen Anforderungen stand die Wartbarkeit und die Benutzerfreundlichkeit der IBML V2 an oberster Stelle.

Durch einen auf MPS basierenden Editor (siehe Abbildung 4) ist es uns gelungen die Wartbarkeit und die Benutzerfreundlichkeit zu verbessern.

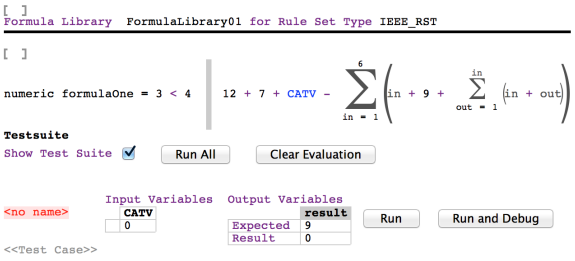


Abbildung 4: Beispielformel mit Test

Wie an den Formeln in der Abbildung zu erkennen ist, können Versicherungsmathematiker nun in gewohnter Formelschreibweise direkt editieren. Der Nutzer kann im Editor Testfälle spezifizieren und diese können direkt im Editor getestet und debugged werden. Durch die Neuentwicklung existiert nur eine Ausführungsengine in Java, was die Wartbarkeit verbessert hat.

5 Fazit

In beiden Projekten ist es uns gelungen, die bestehende Lösung durch eine wesentlich effizientere abzulösen. Im ersten beschriebenen Projekt konnte der Entwicklungsprozess erheblich verbessert werden, was zu einer besseren Time-to-Market und Kosteneinsparungen führte. Im zweiten Projekt wurde vor allem die Wartbarkeit stark verbessert. In beiden Projekten konnte die Benutzungsfreundlichkeit erhöht werden und die Anwender können ihre Berechnungen ohne jede Schwierigkeit im neuen Editor definieren.

Literatur

[1] Markus Voelter. “Language and IDE Modularization and Composition with MPS”. In: *GTTSE IV: Intl. Summer School, GTTSE 2011. Revised Papers*. Springer Berlin Heidelberg, 2013.