Business Technology

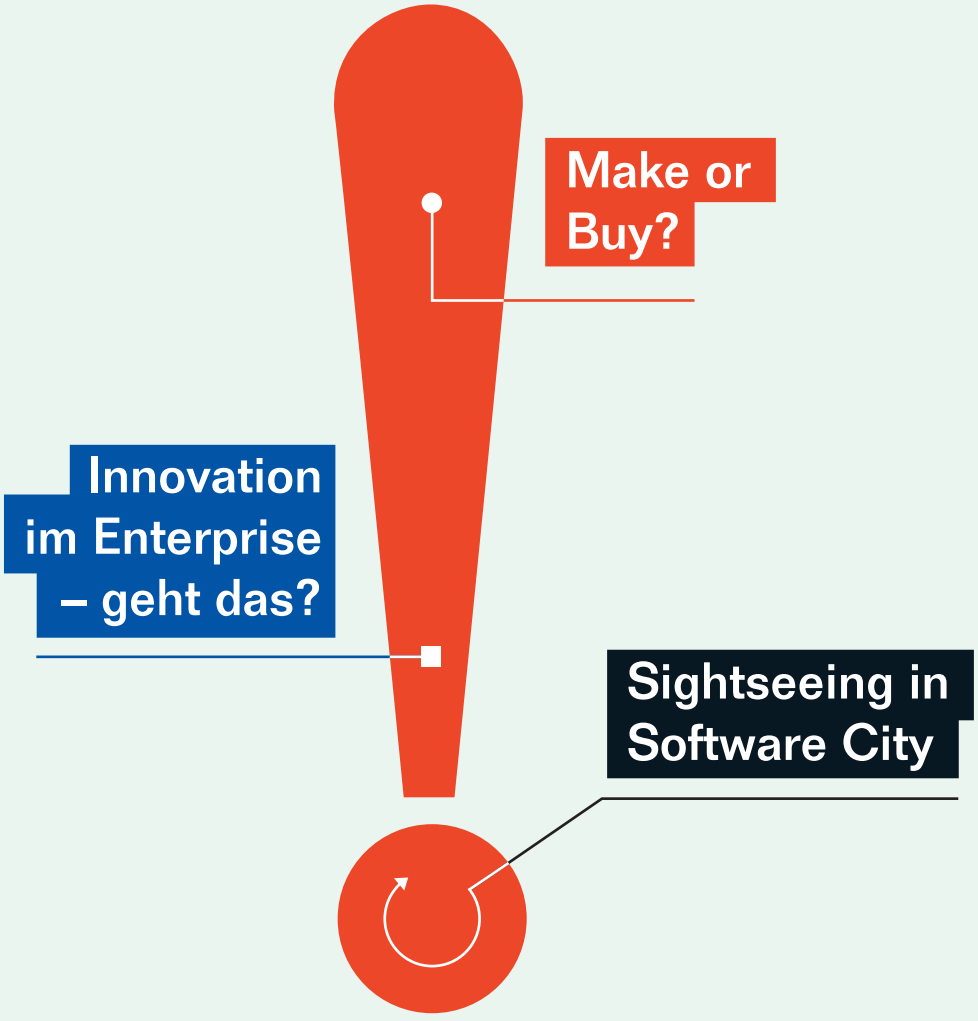
www.bt-magazin.de 3.2013 Heft 14

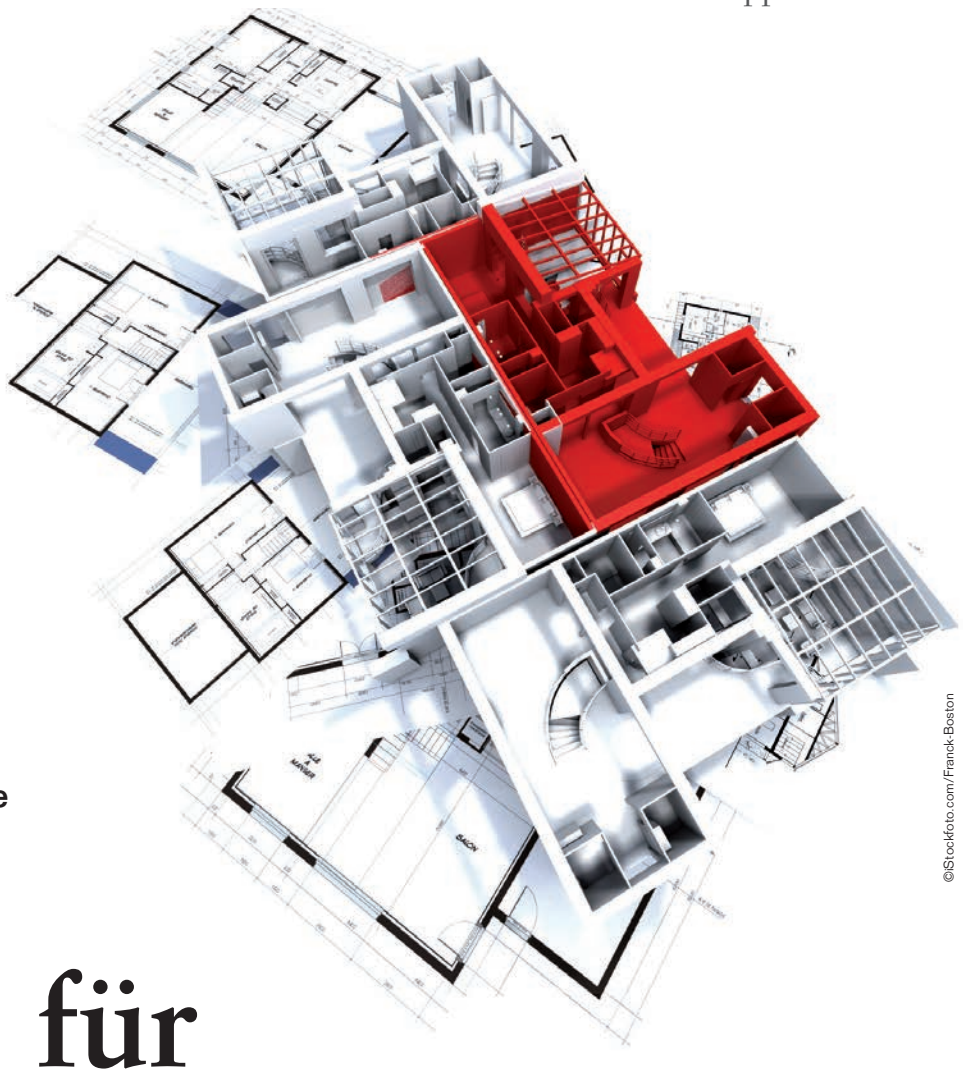
€ 4,90 S&S

Business Technology

Lean Innovation & Transformation

INNOVATIVE ARCHITEKTUR





Architekturbeispiel für eine „User-owned Application“

Gebaut für beständigen Wandel

Betrachtet man eine Fachapplikation als ein Haus, so übernimmt die IT-Abteilung jede Art von Instandsetzungs- und Umbauarbeiten, und die Fachabteilungen müssen jede noch so kleine Arbeit beauftragen, warten bis sie umgesetzt wird und beschränken sich auf das reine „Bewohnen“. Wie kann eine Architektur einer User-owned Application aussehen, in der die Fachabteilungen wieder „Herr im eigenen Hause sind“?

AUTOREN: KONSTANTIN DIENER UND DANIEL KAMINSKY

Kontinuierliche Innovation und der wachsende Umfang gesetzlicher Regelungen erfordern eine steigende Anzahl kurzfristig umzusetzender Änderungen an den Fachapplikationen der Unternehmen. In der Regel sind Prozesse zur Umsetzung einer solchen Änderung langwierig und kostspielig, da die Übersetzung von der Fach- in die IT- und schlussendlich in die Maschinensprache sehr aufwändig und verlustbehaftet ist. Ein möglicher Ansatz zur Beschleunigung der Umsetzungsprozesse ist die Überführung der Hoheit über die Anwendung von der IT- in

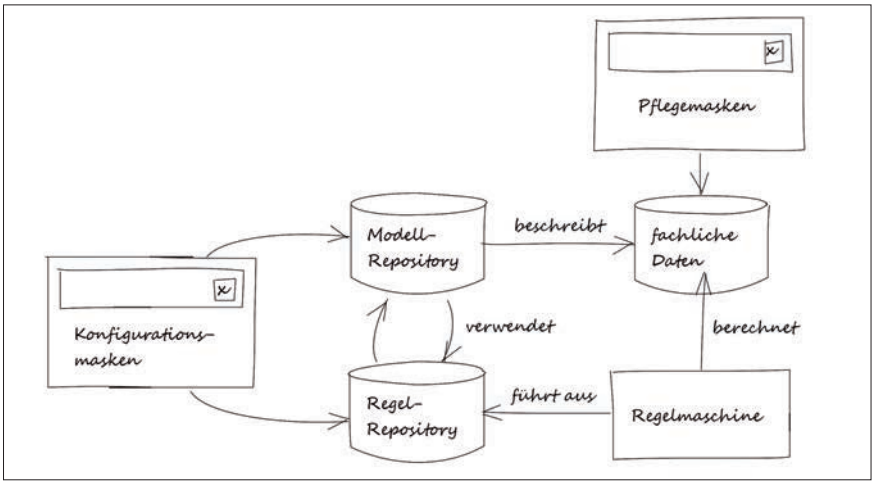


Abb. 1: Architekturblaupause des fachlichen Anwendungsframeworks

die Fachabteilungen. Wir möchten in diesem Zusammenhang von einer User-owned Application sprechen. Sind die Fachbenutzer klassischerweise für die Pflege der Daten in „ihrer" Anwendung verantwortlich, erstreckt sich der Verantwortungsbereich in einer User-owned Application zusätzlich auf das Domänen(daten)modell und die Geschäftsregeln. Die IT-Abteilung stellt für diese Arbeit ein fachliches Anwendungsframework als Plattform zur Verfügung, das Aspekte wie Nachvollziehbarkeit, Versionierbarkeit, ein Lebenszykluskonzept und Ähnliches bietet [1]. Das Anwendungsframework basiert auf der in **Abbildung 1** dargestellten Architekturblaupause.

ZUSÄTZLICHE ROLLE FÜR DEN FACHANWENDER

Grundlegend lassen sich in der Interaktion der Benutzer mit einer Fachapplikation zwei Rollen unterscheiden. Recht anschaulich können diese Rollen durch einen Seitenblick auf den Sprachgebrauch der SQL abgegrenzt werden: Die Data Definition Language (DDL) dient dem Erzeugen, Verändern und Löschen von Datenstrukturen und die Data Manipulation Language (DML) dem Erstellen, Verändern oder Löschen von Inhalten für diese Datenstrukturen. In einer klassischen Anwendung liegt die Data-Definition-Rolle in der Verantwortung der IT- und die Data-Manipulation-Rolle bei der Fachabteilung. In User-owned Applications hat die Fachabteilung beide Rollen inne. Das Konzept der „Definition"-Rolle gilt dabei sowohl für das Domänenmodell als auch für die Geschäftsregeln. Die Regeln werden nicht nur durch den Fachanwender ausgeführt, sondern auch definiert und gepflegt. Tabelle 1 enthält eine Zuordnung der Blaupausekomponenten zu den einzelnen Rollen.

AUSWIRKUNGEN AUF DIE TECHNOLOGIEAUSWAHL

Die neue Rollenverteilung bei der Anwendungsbetreuung wirkt sich auch auf die Technologieauswahl für die Umsetzung einer User-owned Application aus. Normalerweise gehen mit der klassischen Aufgabenteilung zwischen IT und Fachabteilung auch unterschiedliche Anforderungen an die „Definition"- und die „Manipulation"-Teile der Anwendung einher. Erstere dürfen ruhig etwas spartanischer und „technischer" sein. Liegen beide Rollen in der Verantwortung der Fachabteilung, sind die Anforderungen an alle Komponenten gleich hoch, und die Technologieplattform sollte im Optimalfall all diese Komponenten „aus einem Guss" bieten. Es gibt eine Reihe von Optionen für die Einzelkomponenten (Tabelle 1). Einige kommerzielle Produkte, wie bspw. Visual Rules [2], bieten einen Großteil der geforderten Komponenten sogar unter einem Dach. Für diesen Artikel haben wir uns bewusst dafür entschieden, einen auf Open-Source-Technologien aus dem Eclipse-Ökosystem basierenden Ansatz (aus einem Guss) vorzustellen.

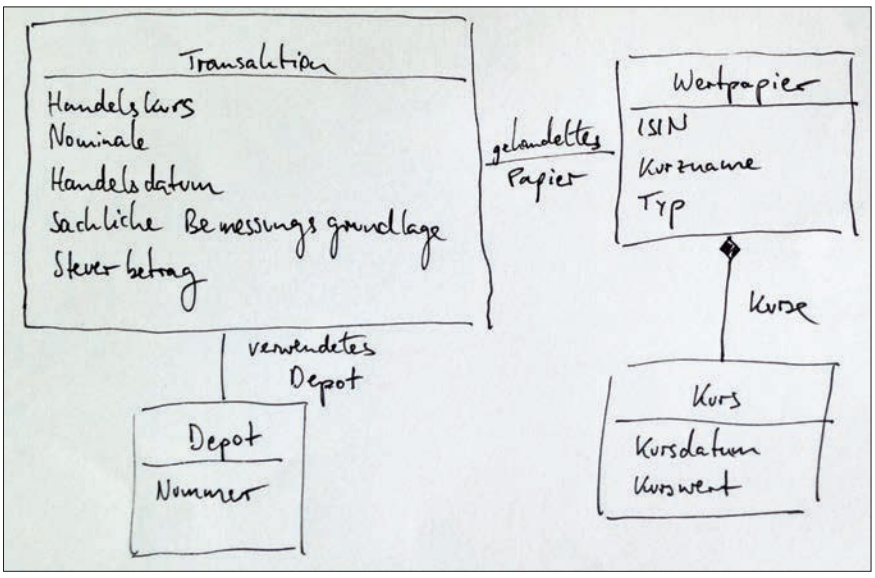


Abb. 2: Domänenmodell der Anwendung zur Steuerberechnung

Data Definition und Data Manipulation

- *Eclipse Modeling Framework*: Das Domänenmodell wird über Pflegemasken des Eclipse Modeling Frameworks (EMF) [3] in Form eines Ecore-Modells bereitgestellt.
- *EMFStore*: Zur Ablage der Modelle und zur Umsetzung des Lebenszyklus kommt EMFStore [4] zum Einsatz (siehe Kasten „EMFStore"). Im Kasten „Persistenz dynamischer Modelle" sind weitere Persistenzstrategien beschrieben.
- *EMF Client Platform*: Die dynamischen Pflegemasken werden basierend auf der EMF Client Platform [5][6] realisiert.

Rule Definition und Rule Execution

- *Xtext*: Aufgrund der guten Interoperabilität mit dem EMF kommt als Basistechnologie für die Geschäftsregeln Xtext zum Einsatz.
- *Git*: Die Regeln werden in einem Git Repository versioniert. Für das Hochladen von Regeländerungen werden für die Fachbenutzer Masken in Eigenentwicklung erstellt, die die Versionierung nahtlos in die Regeleditoren integrieren.
- *Regelmaschine*: Die Regelmaschine ist eine Eigenentwicklung.

Die gewählten Technologien werden im Folgenden anhand eines fachlichen Beispiels vorgestellt.

FACHLICHER KONTEXT

Im Jahr 2009 wurde zur vereinfachten Besteuerung von Kapitalerträgen die so genannte Abgeltungssteuer eingeführt. Seitdem werden alle Gewinne aus Kapitalgeschäften einheitlich mit einem Steuersatz von 25 Prozent besteuert. Relativ einfach nachvollziehbar wird dieser Gewinn beim Verkauf von Aktien ermittelt. Ist der Kurs der Aktie beim Verkauf höher als beim Erwerb, muss die Kursdifferenz als Gewinn versteuert werden – Verluste werden in diesem Beispiel der Einfachheit halber vernachlässigt. Der Gewinn wird dabei auch als „sachliche Bemessungsgrundlage" bezeichnet. Auf den Gewinn sind Kapitalertragsteuern in Höhe von 25 Prozent zu entrichten. **Abbildung 2** enthält einen Ausschnitt aus einem vereinfachten Domänenmodell zur Steuerberechnung.

Die einzelnen Attribute des Domänenmodells sind mit entsprechenden Geschäftsregeln zur Berechnung hinterlegt. **Abbildung 3** zeigt beispielhaft die Geschäftsregel zur Ermittlung des konkreten Steuerbetrags (pro Transaktion) anhand der sachlichen Bemessungsgrundlage.

EMFStore

EMFStore [4] ist ein Projekt im Bereich der Eclipse-Modeling-Technologie. Es handelt sich dabei um einen Repository-Server für die semantische Versionierung. Die Technologie ist mit Versionierungssystemen für textuelle Artefakte wie Subversion oder Git vergleichbar, dient aber explizit der persistenten Speicherung und Verwaltung von Modellen. Der EMFStore bietet mit entsprechenden Client-Plug-ins für SWT von klassischen Versionierungssystemen bekannte Funktionen wie eine Artefakthistorie oder das Auflösen von Versionskonflikten. Über ein API lässt sich der EMFStore in eigene Anwendungen integrieren.

Persistenz dynamischer Modelle

In vielen Unternehmen sind relationale Datenbanken der vorherrschende Persistenzmechanismus. Alternativ zu EMFStore ist es auch möglich, dynamische Modelle darin zu speichern:

- Metamodell: Das Datenbankmodell entspricht dem Metamodell der Anwendung (Tabellen: Typ, Attribut, Attributwert ...). Diese Alternative ermöglicht eine intuitive Versionierung von Daten und Strukturen, muss für große Datenmengen durch Datenbankmechanismen speziell optimiert werden.
- Automatische DDL-Statements: Das Domänenmodell ist direkt in der Datenbank reflektiert (Tabellen: Wertpapier, Transaktion ...). Ändert der Benutzer das Domänenmodell, wird das Datenbankmodell automatisch durch die Anwendung verändert. Diese Alternative ist sehr performant, setzt der Flexibilität aber Grenzen und erfordert eigene Implementierungen für die Zugriffsschicht und Funktionen wie Versionierung und Co.

Rolle	Komponenten	Beispielhafte Technologieoptionen
Data Definition	Model-Repository, Konfigurationsmasken	RDBMS, NoSQL, EMF/Ecore ...
Data Manipulation	Fachliche Daten, Pflegemasken	RDBMS, Oracle Application Express, EMF Client Platform ...
Rule Definition	Regel-Repository, Konfigurationsmasken	Groovy/Scala/Xtext+Subversion/Git, JetBrains MPS, BRMS ...
Rule Execution	Regelmaschine, Pflegemasken	BRMS ...

Tabelle 1: Rollen, Komponenten und Beispiele für Technologieoptionen

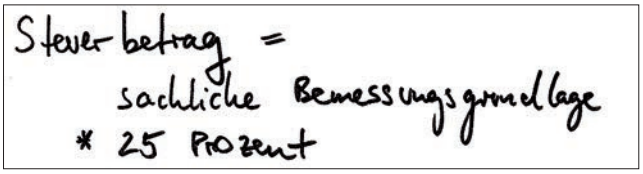


Abb. 3: Beispielhafte Geschäftsregel der Anwendung zur Steuerberechnung

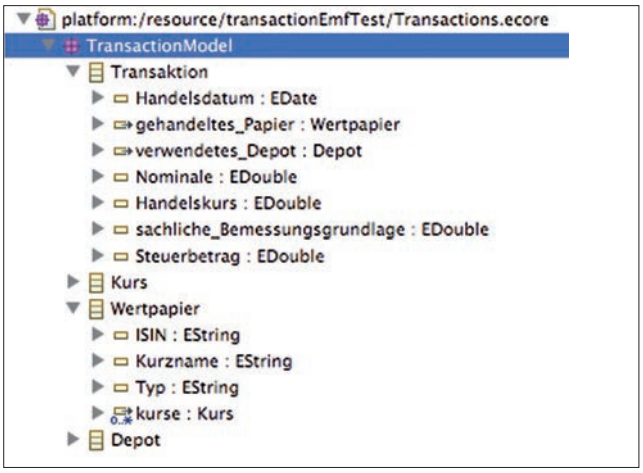


Abb. 4: Ecore-Modell der Anwendung zur Steuerberechnung

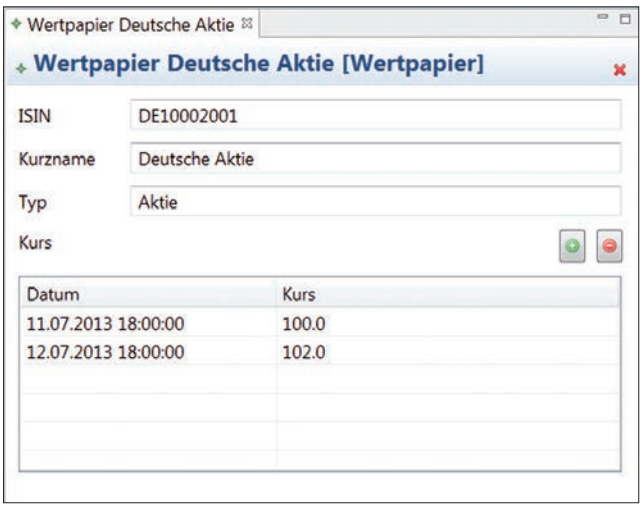


Abb. 5: Wertpapierpflegemaske

Abb. 6: Transaktionspflegemaske

UMSETZUNG DES DOMÄNENMODELLS MIT DEM ECLIPSE MODELING FRAMEWORK

In seiner Data-Definition-Rolle konfiguriert der Fachanwender das vorgestellte Domänenmodell als Ecore-Pendant. Zu diesem Zweck existiert im Eclipse Modeling Framework ein Plug-in für die Eclipse Rich Client Platform, das ausschnittweise in **Abbildung 4** zu sehen ist. Die Abbildung zeigt, dass der Standardeditor unser Domänenmodell als Baum darstellt. Sollte für die Fachbenutzer eine Darstellung in einer grafischen Form (wie auf der Zeichnung) intuitiver sein, lässt sich ein entsprechender Editor über das Graphical Editing Framework (GEF) [7] realisieren. Ein Zugeständnis musste der Anwender bei der Konfiguration des Modells machen: Statt Leerzeichen wurden in den Bezeichnern Unterstriche verwendet.

Auf Basis des Ecore-Modells erzeugt die EMF Client Platform automatisch passende Pflegemasken für die Data-Manipulation-Rolle. Diese Masken sind nicht durch Quellcode implementiert, sondern werden reflektiv aus dem Modell abgeleitet. Änderungen des Modells schlagen sich dadurch unmittelbar in den Datenpflegemasken nieder.

In der in **Abbildung 5** dargestellten Pflegemaske erfasst der Fachbenutzer beispielsweise im oberen Teil die Stammdaten eines Wertpapiers und im unteren Bereich dessen Kurse. Da die Kurse fest mit dem Wertpapier verbunden sind und potenziell in einer recht großen Menge vorliegen können, werden sie als Tabelle dargestellt. Hinzufügen oder Löschen von Kursen wird durch die Steuerelemente („+“ und „-“) oberhalb der Tabelle ermöglicht. Da die Kurse mit fachlich korrekten Datentypen für das Datum und den eigentlichen Kurswert modelliert wurden, ist eine Sortierung der Tabelle direkt möglich. Zusätzlich werden dadurch fachlich falsche Eingaben unterbunden. In ein Datumsfeld lassen sich ausschließlich Datumswerte eintragen.

Abbildung 6 zeigt beispielhaft eine zweite Pflegemaske für Transaktionen. Die Transaktion verweist auf das gehandelte Papier und das verwendete Depot. Der Editor unterstützt den Benutzer bei der korrekten Bearbeitung der Transaktion: Beim Setzen der Verweise können nur Elemente des richtigen Datentyps ausgewählt werden. So ist es bspw. nicht möglich, für das gehandelte Papier ein Depot auszuwählen; strukturelle Fehler werden verhindert. Zudem ermöglicht der Editor die integrierte Anlage der referenzierten Elemente: Möchte der Benutzer auf ein Depot verweisen, das noch nicht existiert, so kann es in einem integrierten Bearbeitungsschritt angelegt, referenziert und direkt bearbeitet werden. Eine Unterbrechung des Arbeitsflusses durch eine separate Anlage ist nicht notwendig.

UMSETZUNG DER GESCHÄFTSREGELN MIT XTEXT

Das Domänenmodell beschreibt zunächst die fachliche Struktur der Anwendungsdomäne. In unserem Beispiel ist hier unter anderem festgelegt, aus welchen Attributen ein Wertpapier oder eine Transaktion bestehen. Die Werte dieser Attribute werden entweder durch den Fachbenutzer direkt im Rahmen seiner Data-Manipulation-Rolle eingetragen oder in Form von Geschäftsregeln abgeleitet.

Die Geschäftsregeln sind durch eine Xtext-basierte domänenspezifischen Sprache (DSL) umgesetzt. Sie definiert

- logische Bedingungen („Wenn […] dann […] sonst […]“)
- die Referenzierung von Elementen im Domänenmodell und Navigation zwischen diesen Elementen („Objekt -> Unterelement“)
- Berechnungsausdrücke („Kaufpreis * 119 Prozent“)
- Domänenfunktionen (höherwertige Funktionen)

In **Abbildung 3** wurde eine beispielhafte Geschäftsregel vorgestellt, die in der DSL formuliert aussieht wie in Listing 1. Ebenso lassen sich aufwändigere Regeln mit der DSL umsetzen (Listing 2). Diese zweite Variante nutzt auch eine der fachlichen Domänenfunktionen (*Anschaffungskurs*) zur Bestimmung des korrekten Anschaffungskurses.

Zu der DSL gehören Eclipse-Texteditoren, die den Benutzer bei der Erstellung der Regeln durch kontinuierliche Validierung und eine Vorschlagsfunktion unterstützen. Durch diese Hilfestellung werden strukturelle Fehler bereits bei der Eingabe vermieden.

Abbildung 7 zeigt, wie die Vorschlagsfunktion nur die an dieser Stelle gültigen Werte der DSL auflistet – in diesem Fall ausschließlich die Attribute für den aktuellen Datentyp. Dabei ist auch eine Verschachtelung über mehrere Datentypen hinweg möglich. Wir werden später noch eine weitere mächtige Hilfestellung des DSL-Editors sehen.

Die Regelmaschine ist die einzige vollständige Eigenentwicklung in unserem Entwurf. Basierend auf dem hinter den Xtext-Regeln liegenden Modell ermittelt sie den Abhängigkeitsgraphen der verknüpften Attribute (in die Berechnung eines Attributs gehen die Werte anderer berechneter Attribute ein usw.), leitet daraus die Reihenfolge der Regelausführung ab, verknüpft die fachlichen Daten mit der DSL-Regel und führt sie aus. Die Ergebnisse werden ins Modell zurückgeschrieben und sind in den Pflegemasken als grau markierte Felder dargestellt (**Abb. 6**). Ändert der Benutzer ein in die Regeln eingehendes Attribut, wird die DSL-Regel automa-

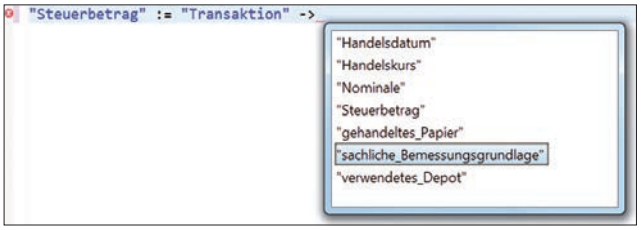


Abb. 7: Liste mit Vorschlägen für gültige Attributnamen

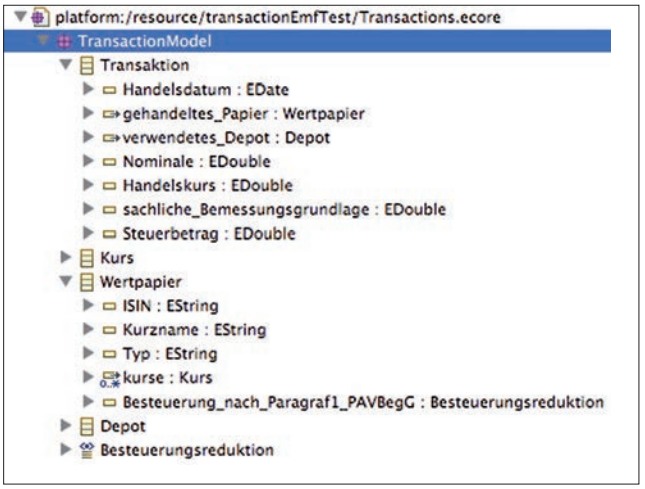


Abb. 8: Erweitertes Domänenmodell der Anwendung

Listing 1

```
"Steuerbetrag" := "Transaktion" -> "sachliche_Bemessungsgrundlage"
                                                    * 25 Prozent
```

Listing 2

```
"sachliche_Bemessungsgrundlage" :=
  "Handelskurs"    // Kurswert beim Verkauf
 - Anschaffungskurs("Transaktion" -> "gehandeltes_Papier",
                    "Transaktion" -> "verwendetes_Depot")
```

Listing 3

```
"Steuerbetrag" :=
 Wenn "Transaktion" -> "gehandeltes Papier" ->
   "Besteuerung_nach_Paragraf1_PAVBegG"
 ist "gemäß Abs. 1":
   "sachliche Bemessungsgrundlage" * 12 Prozent
 ist "gemäß Abs. 2":
   "sachliche Bemessungsgrundlage" * 7 Prozent
 Sonst:
 "sachliche Bemessungsgrundlage" * 25 Prozent
```

```
"Steuerbetrag" :=
    Wenn "Transaktion" -> "gehandeltes_Papier" -> "Besteuerung_nach_Paragraf1_PAVBegG"
        ist "gemäß Abs. 1":
            "sachliche Bemessungsgrundlage" * 11 Prozent
        ist
```

"gemäß Abs. 1"
"gemäß Abs. 2"
"nicht gesetzt"

Abb. 9: Eingabeunterstützung im Kontext der Enumeration

tisch neu ausgewertet und das Ergebnis in der Anzeige aktualisiert.

GESETZLICHE ÄNDERUNG

In unserem Beispiel ändert sich der Mechanismus zur Steuerberechnung durch eine fiktive gesetzliche Änderung. Entgegen dem ursprünglichen Ziel, alle Kapitalerträge einheitlich zu besteuern, gehen wir im Beispiel davon aus, dass Gewinne aus Produkten zur privaten Altersvorsorge steuerlich begünstigt werden sollen. Zu diesem Zweck erlässt der Gesetzgeber ein entsprechendes Gesetz (PAVBegG), das eine Staffelregelung enthält: Entsteht der Gewinn durch den Verkauf ...

1. ... eines zertifizierten Altersvorsorgeprodukts, ist er mit 12 Prozent zu versteuern (Abs. 1 des Gesetzes)
2. ... eines zertifizierten Altersvorsorgeprodukts, das in nachhaltige Technologien investiert, ist er mit 7 Prozent zu versteuern (Abs. 2 des Gesetzes)
3. ... eines anderen Wertpapiers, das keine der obigen Kategorien fällt, ist er weiterhin mit 25 Prozent zu versteuern (nicht im Gesetz erwähnt, da wie bislang)

Die Informationen zu den Wertpapieren erhalten ein zusätzliches Kennzeichen, das die Zuordnung zu einem der drei Steuersätze festlegt.

UMSETZUNG DER ÄNDERUNG

In seiner Data-Definition-Rolle muss der Fachanwender das neue Kennzeichen im Domänenmodell einführen. Da es für das Kennzeichen nur drei gültige Ausprägungen gibt, wird das entsprechende Attribut *Besteuerung_nach_Paragraf1_PAVBegG* im Ecore-Modell mit einer passenden Enumeration *Besteuerungsreduktion* verknüpft (**Abb. 8**).

Im nächsten Schritt „wechselt“ der Fachbenutzer in die Rule-Definition-Rolle und passt die Geschäftsregel für den Steuerbetrag an (Listing 3). Bei dieser umfangreicheren Regel ist die Unterstützung durch den DSL-Editor besonders hilfreich, die in **Abbildung 9** am Beispiel der Enumeration zu sehen ist. Auch hier werden dem Benutzer nur die möglichen gültigen Alternativen angezeigt. Gibt er trotzdem etwas Ungültiges an, wird die Stelle im Editor mit einer Fehlermeldung markiert.

FAZIT UND AUSBLICK

In User-owned Applications haben Fachanwender sowohl die „Definition“- als auch die „Manipulation“-Rolle inne. Anhand eines fachlichen Beispiels aus der Steuerberechnung konnte gezeigt werden, dass dadurch die Umsetzung von Änderungen kurzfristig durch die Anwender selbst durchgeführt werden kann. Für die Umsetzung einer solchen Applikation wurde eine Reihe von Technologien aus dem Eclipse-Ökosystem vorgestellt.

Links & Literatur

[1] Diener, Kaminsky: „Baukasten für Fachleute – Blaupause für ein fachliches Anwendungsframework“, Business Technology Magazin 2.2013
[2] http://www.bosch-si.com/de/technologie/business-rules-management-brm/visual-rules-suite.html
[3] http://www.eclipse.org/modeling/emf/
[4] http://eclipse.org/emfstore/
[5] http://eclipse.org/emfclient/
[6] http://eclipsesource.com/blogs/2013/06/13/emf-client-platform-make-it-happen-day-0-and-1/
[7] http://www.eclipse.org/gef/

Konstantin Diener

ist Leading Consultant bei der Cofinpro AG. Er beschäftigt sich seit über zehn Jahren mit der Java-Plattform und sein Interesse gilt allem, was IT und Fachabteilungen flexibel und produktiv macht (Groovy, Drools, Scrum, Kanban, DevOps, fachliche Akzeptanztestverfahren).

Daniel Kaminsky

ist als Consultant bei der Cofinpro AG beschäftigt. Er ist seit Jahren in der Java-Welt zu Hause und setzt seine Projekte mit Vorliebe nach der Scrum-Methodik um.