Anpassbare RE Werkzeuge – Eine Fallstudie

Holger Eichelberger, Klaus Schmid
Universität Hildesheim, Institut für Informatik
Marienburger Platz 22, D-31141 Hildesheim, Deutschland
{eichelberger, schmid}@sse.uni-hildesheim.de

Kurzfassung

Die Anpassung von Anforderungswerkzeugen an den Projektkontext ist im Regelfall durch die Anpassungsfähigkeiten des jeweiligen Werkzeuges stark eingeschränkt. In diesem Beitrag stellen wir einen generativen Ansatz zur Anpassung von Anforderungswerkzeugen vor und demonstrieren den Ansatz im Rahmen einer Fallstudie. Unser Ansatz ermöglicht die einfache, zielgerichtete und weitreichende Anpassung an die Anwendungssituation.

1. Einleitung

Verschiedene Unternehmen nutzen teilweise sehr unterschiedliche Ansätze zur Beschreibung von Anforderungen [1]. Um dies zu unterstützen bieten die meisten Anforderungswerkzeuge umfangreiche Möglichkeiten zur Abbildung der Anforderungsstruktur. Trotz sehr aufwändiger Modellierungsmöglichkeiten zeigt sich oft, dass gerade Einschränkungen in den zur Verfügung stehenden Informationstypen für die Auswahl von Werkzeugen entscheidend sind [2].

Diese Erfahrung war ein Auslöser, dafür eine Produktline von Anforderungswerkzeugen zu entwickeln. Einerseits ist auf diese Weise ein konkretes Werkzeug sehr leichtgewichtig, da es nur die für den konkreten Einsatz notwendigen Funktionen enthält. Anderseits erlaubt dieser Ansatz für Fälle, in denen die vorgesehenen Ausdrucksmittel ausreichen, eine automatisierte Anpassung. In den Fällen, bei denen zusätzliche Anpassungsmöglichkeiten benötigt werden, ist eine einfache Erweiterung der möglichen Informationstypen (bspw. Multimediaartefakte), aber auch der unterstützten Funktionalität möglich.

Zur Realisierung der Produktlinie verfolgen wir einen generativen Ansatz. Zentrales Modell ist dabei das Informationsmodell. Ein Code-Generator erzeugt aus dem Informationsmodell u.a. die Implementierung der Datenhaltungsschicht und Teile der Benutzeroberfläche. Der generierte Teil wird durch einen generischen Applikationskern ergänzt, der die wesentliche Funktionalität realisiert. Dieser ist selbst erweiterbar, um eine Unterstützung unterschiedlicher Situationen sicherzustellen. Außerdem unterstützt die flexible Infrastruktur die Nutzung als Standalone-System, Web-Client und Eclipse-PlugIn.

In diesem Beitrag stellen wir kurz den Ansatz und dessen Anwendung für die Erfassung und das Management von Anforderungen dar.

2. Der Ansatz

Mit hinreichend genau spezifizierten Modellen ist es unter Anwendung modellbasierter Techniken möglich vollständige Programme zu generieren [3]. Der Preis für diesen Grad an Allgemeinheit ist ein hoher Aufwand bei der Modellbildung und bei der Entwicklung der Code-Generierung.

Für eine Produktlinie von RE-Werkzeugen definiert vor allem das Informationsmodell die wesentlichen Eigenschaften der konkreten Ausprägung. Man kann das Informationsmodell daher als zentrales Element einer domänenspezifischen Sprache sehen. Weitere Mitglieder der Produktlinie ergeben sich aus den Varianten des Informationsmodells, d.h. durch Hinzfügen, Löschen oder Ändern von Attributen, Relationen oder Klassen. Im Gegensatz zu klassischen Produktlinien [4] handelt es sich bei unserer Vorgehensweise um eine offene Produktlinie: Durch die Möglichkeit das Informationsmodell nahezu beliebig zu verändern ist es nicht möglich, alle ableitbaren Produkte aufzuzählen, wie dies bei klassischen Variabilitätsmodellen der Fall ist. Auch kann unsere Modellierungssprache einfach erweitert werden, um neue Informationstypen zu realisieren.

Aus pragmatischen Gründen stellen wir das Informationsmodell als UML Klassenmodell dar. Dieses legt die zu verarbeitenden Daten und deren Relationen fest. Durch Annotationen, d.h. durch Stereotype und Eigenschaftswerte, werden Zusatzinformationen hinterlegt, die beschreiben ob ein Attribut in der Benutzeroberfläche dargestellt werden soll, ob der Wert durch den Benutzer verändert werden kann, etc.

Einige Bestandteile des Informationsmodells sind für alle Mitglieder der Werkzeugproduktlinie verpflichtend, bspw. in Form gemeinsamer Oberklassen. Danach wird das Modell von einem Code-Generator verarbeitet, der die Modellinformationen in Quelltext übersetzt. Der generierte Quelltext beinhaltet u.a. die Implementierung des Informationsmodells, die Datenhaltungsschicht und die Teile der Benutzeroberfläche, die zur Manipulation der Daten vorgesehen sind. Während der Übersetzung des Programms wird die Realisierung der Informationsarchitektur mit dem Programmkern verbunden, so dass ein lauffähiges Programm entsteht. Der Programmkern beinhaltet beispielsweise gemeinsame Funktionalität der Werkzeugproduktlinie. Weitere bisher nicht vorgesehene Variabilitäten können als Plug-in hinzugefügt werden.

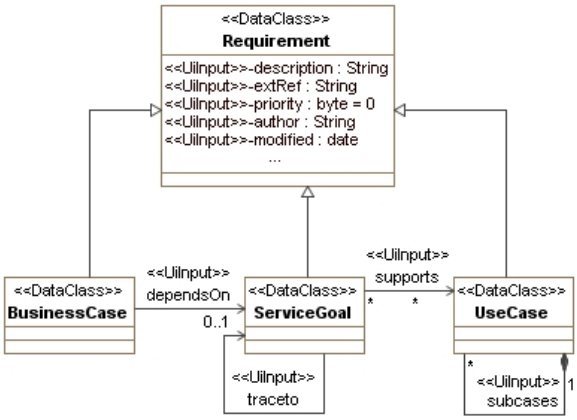

Abbildung 1 Basis-Informationsmodell der Fallstudie

4. Fallstudie

Zur Demonstration unseres Ansatzes haben wir ein einfaches Informationsmodell für Anforderungswerkzeuge definiert und davon verschiedene Werkzeugvarianten abgeleitet. Die Modelle unterscheiden sich z.B. im Detailgrad der gespeicherten Informationen, der Anzahl der Klassen und den Relationen

Abbildung 1 zeigt die Basisversion des Informationsmodells. Es definiert eine Hierarchie verschiedener Anforderungstypen und deren Beziehungen. So ist ein Geschäftsfall eine Anforderung und kann in Unterfälle gegliedert sein (`subcases`). Die Attribute in `Requirement` wie auch die Beziehungen sind so annotiert, dass sie in der Benuzeroberfläche angezeigt werden (Details der Auszeichnungen sowie auch die gemeinsamen Oberklassen der Produktlinie sind in Abbildung 1 nicht dargestellt).

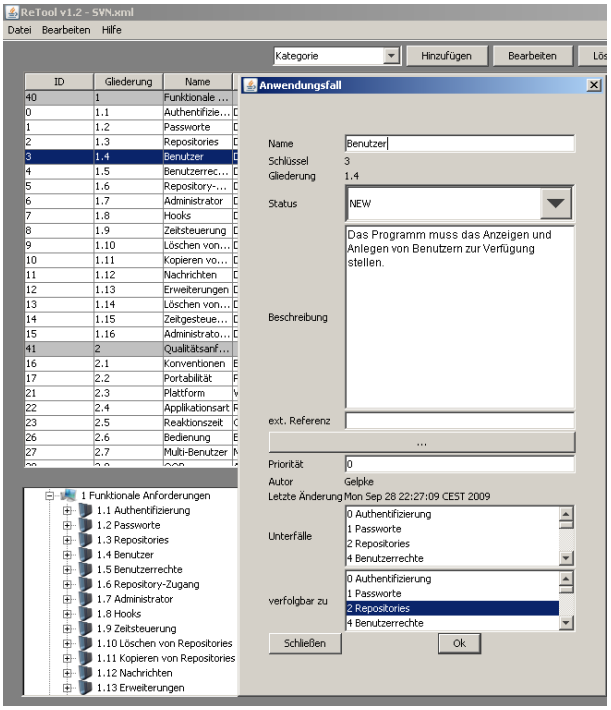

Abbildung 2 Beispiel eines Anforderungswerkzeuges mit Änderungsinformationen

Der gemeinsame Kern der Produktlinie wurde manuell erstellt und bietet mehrere Darstellungsformen, Druckfunktionalität, Editierfunktionen für Anforderungen, etc. Der Kern kann als Benutzeroberfläche für Swing, SWT, Eclipse und als Web-Client konfiguriert werden. Plug-ins ermöglichen den Datenaustausch mit externen Datenquellen, z.B. CollabNet.

Einige Varianten wurden im Praxiseinsatz getestet, z.B. in studentischen Praktika. Dabei wurde die Funktionalität des generierten Werkzeuges, dessen Eignung für den Anforderungsprozess wie auch die Anpassbarkeit beobachtet. So wurde beispielsweise eine Variante mit zusätzlichen Informationen für ein einfaches Änderungsmanagement in kürzester Zeit ohne Implementierungsaufwand umgesetzt. Abbildung 1 zeigt einen Ausschnitt der Bildschirmdarstellung der dabei generierten Werkzeugvariante.

5. Zusammenfassung

Die Anpassbarkeit des Informationsmodells ist eine wesentliche Basis um Anforderungswerkzeuge an spezielle Projektsituationen und Entwicklungsmodelle anpassen zu können. In diesem Beitrag haben wir dargestellt, dass das Informationsmodell im Sinne einer domänenspezifischen Sprache genutzt werden kann, um speziell angepasste Werkzeuge abzuleiten. Große Teile der Implementierung werden dabei aus dem spezifizierten Informationsmodell generiert. Durch Variationen des Informationsmodells ergibt sich eine offene Produktlinie von Werkzeugen, die auch leicht erweitert werden kann.

Die Fallstudie in diesem Beitrag demonstriert die konkrete Anwendbarkeit unseres Ansatzes im Bereich Anforderungswerkzeuge. Angepasste Werkzeuge wurden bisher in einem Praktikum eingesetzt um die schnelle und einfache Anpassbarkeit exemplarisch nachzuweisen (im Sinne einer Evolution kamen mehrere Varianten zum Einsatz).

Für zukünftige Arbeiten werden wir eine Erweiterung und Anpassung der Beschreibungssprache durchführen. Momentan werden die Anforderungen für die weiteren Eigenschaften identifiziert. Die dabei gewonnenen Ergebnisse werden in die nächste Generation der Anforderungswerkzeuge einfließen.

Danksagung

Die Autoren bedanken sich bei Felix Gelpke und Stephan Dederichs für die Implementierungsarbeiten.

Referenzen

[1] C. Hood, S. Mühlbauer, C. Rupp and G. Versteegen, *iX Studie Anforderungsmanagement*, Heise Verlag, 2007

[2] M. Lorenz, M. Bode and K. Schmid, *Requirements Engineering im Customer Relationship Management: Erfahrungen in der Werkzeugauswahl*, Softwaretechnik-Trends, 28 (1), 2008

[3] D. C. Schmidt, *Model-Driven Engineering*, IEEE Computer, Vol. 39, No. 2, February, 2006.

[4] P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley, 2002.