Generative Programmierung

Ulrich W. Eisenecker (eisenecker@informatik.fh-kl.de), Mathias Henss (mat_tes@gmx.net), Markus Lang (prinz-william@gmx.de), Max Schlee (schlee2@web.de), Fachhochschule Kaiserslautern, Zweibrücken

Die Softwareentwicklung ist überwiegend auf die Erstellung einzelner Systeme ausgerichtet, obwohl überall auf die Notwendigkeit von Wiederverwendung hingewiesen wird. Doch wie soll die Entwicklung von Einzelstücken etwas Wiederverwendbares hervorbringen? Zwar bringen Frameworks, Entwurfsmuster und Softwarekomponenten in dieser Hinsicht Besserung, aber keinen Durchbruch. Einen wesentlichen Fortschritt verspricht hingegen die Entwicklung von Softwaresystemfamilien. Sie ermöglicht die anforderungsspezifische Fertigung einer großen Zahl von Systemvarianten, und sie kann weitgehend automatisiert werden. Dies erlaubt die drastische Senkung von Entwicklungszeiten und –kosten sowie wesentliche Verbesserungen der Qualität.

Die generative Programmierung (GP) umfaßt aufeinander abgestimmte Methoden und Techniken zur Modellierung von Softwaresystemfamilien, so daß ausgehend von einer Anforderungsspezifikation mittels Konfigurationswissen aus elementaren, wiederverwendbaren Implementierungskomponenten ein hochangepaßtes und optimiertes Zwischen- oder Endprodukt nach Bedarf automatisch erzeugt werden kann.

In jüngster Zeit wurde die Initiative *Model Driven Architecture* (*MDA*) der *Object Management Group* (*OMG*) gestartet. Die MDA zielt auf die Entwicklung abstrakter, plattformunabhängiger Fachmodelle, die durch plattformspezifische Modelle und manuelle Annotationen ergänzt werden. Dadurch können unter Verwendung abstrakter Spezifikationen entwickelte Anwendungen bei Angabe des plattformspezifischen Modells und der erforderlichen Annotation in konkrete Implementierungen transformiert werden. Damit reiht sich die MDA in die Ansätze zur Entwicklung von Softwaresystemfamilien ein.

Inhalt des Vertiefungsvortrags

Nach einer Einführung, welche die Entwicklung von Einzelsystemen der von Systemfamilien gegenüber stellt, wird ein Überblick der generativen Programmierung vermittelt, wobei auch der Bezug zur MDA hergestellt wird. Anhand eines einfachen Beispiels werden die wichtigsten Schritte der generativen Programmierung beschrieben:

- Die *Merkmalmodellierung* dient der Analyse und Dokumentation der Merkmale einer Systemfamilie, unabhängig von der Implementierungstechnik.
- *Implementierungskomponenten* sind die elementaren Bausteine, aus denen die Produkte einer Familie gefertigt werden. Daher müssen sie maximal kombinierbar und minimal redundant sein. Nach ihrer Identifikation werden sie sogenannten Äquivalenzkategorien zugeordnet. Die *Abhängigkeitsanalyse* identifiziert die Abhängigkeiten zwischen Implementierungskomponenten und ist damit die Grundlage zur Festlegung einer gemeinsamen Architektur für alle Systeme einer Familie.
- Die *Systemarchitektur* beschreibt den Aufbau aller Mitglieder einer Systemfamilie. Hier haben sich im Kontext von C++ Schichtenarchitekturen, die auch in der Implementierung eingesetzt werden, gemäß dem GenVoca-Ansatz bewährt. Bei Verwendung von anderen Sprachen oder Entwicklungswerkzeugen kann es vorkommen, daß

eine Systemarchitektur technisch nicht implementiert werden muß. In diesen Fällen ist sie jedoch konzeptionell zur Strukturierung dennoch unverzichtbar.

- Eine *domänenspezifische Sprache* (engl. *Domain Specific Language*, Abk. *DSL*) dient der Spezifikation von Systemen oder höherwertigen Bausteinen; dadurch können Systeme "bestellt" werden. Dabei wird dem Anwender die vertraute Umgebung des Problembereichs bereitgestellt und sie verbirgt den Blick auf den Lösungsbereich mit möglicherweise komplexen und unbekannten Implementierungskomponenten.
- Ein *Konfigurationsgenerator* automatisiert die Montage von Produkten: Er übernimmt eine DSL-Spezifikation, analysiert diese, führt falls erforderlich eine Baubarkeitsprüfung durch und fügt die Implementierungskomponenten zu einem fertigen Produkt zusammen

Anschließend werden aktuelle Entwicklungen der Merkmalmodellierung, die Generierung grafischer Benutzungsoberflächen und eine Fallstudie zur Entwicklung einer Softwaresystemfamilie für E-Learning/Web-Learning vorgestellt.

Das Merkmalmodell ist ein zentraler Bestandteil des Domain Engineerings in der generativen Programmierung. Es wird primär zur Modellierung von gemeinsamen und unterschiedlichen Merkmalen der Domäne und ihren Wechselwirkungen genutzt, bildet aber auch die Basis zur Beschreibung der Konfigurationsmöglichkeiten der Implementierungskomponenten. Von zunehmenden Interesse ist die Möglichkeit, Merkmalmodelle als domänenspezifische Sprache zur Beschreibung einer Systemkonfiguration zu verwenden. Zu diesem Zweck müssen Merkmalmodelle sowohl leicht weiterverarbeitet als auch spezialisiert und konfiguriert werden. Aus diesem Grund wurde an der Fachhochschule Kaiserslautern der Merkmalmodelleditor AmiEddi entwickelt. Die aktuelle Version dieses Programms ermöglicht die grafische Editierung von Merkmaldiagrammen, die Erweiterung des Metamodells durch vorgegebene Zusatzinformationen, sowie die Weiterverarbeitung der Merkmalmodelle beispielsweise durch Generatoren, da die in einem Repository gehaltenen Daten mittels XML gespeichert werden.

Zurzeit ist die Neuentwicklung eines Nachfolgers dieser Software im Gange. Hierbei stehen folgende Aspekte im Vordergrund.

- Anpassung der Merkmaldiagramme an Forderungen aus der Praxis: Als Beispiel sei hierfür die Einführung von Kardinalitäten für Merkmale genannt. Bisher war nur die Unterscheidung zwischen optionalen und notwendigen Merkmalen möglich.
- Flexibilisierung und leichte Erweiterbarkeit des Metamodells: Es ist möglich das Metamodell durch Merkmaldiagramme zu beschreiben. Diese Diagramme des Systemkerns können durch den Benutzer zu einem domänenspezifischen Metamodell erweitert werden. Durch diese Vorgehensweise kann der Benutzer mit dem Merkmalmodelleditor Merkmaldiagramme sowie das Metamodell in gleicher Weise bearbeiten. Dies macht die Merkmalmodellierung zu einem in sich geschlossenen und orthogonalen System, da sie sich selbst definiert.
- Spezialisierung und Konfiguration von Merkmaldiagrammen: Grundsätzlich entsteht bei der Spezialisierung eines Merkmaldiagramms ein weiteres Merkmaldiagramm geringerer Komplexität. Wird ein Merkmaldiagramm solange spezialisiert bis es keinerlei Variabilität mehr enthält, resultiert daraus eine Konfiguration. Konfigurationen dienen im Sinne der generativen Programmierung als Eingabe für Generatoren, können prinzipiell aber auch zur Steuerung von beliebigen Programmen eingesetzt werden. Insbesondere das Zusammenspiel zwischen Spezialisierung und einem erweiterten Metamodell stellt hohe Anforderungen an die neue Software und ihre Implementierung.

Die Neuentwicklung des Merkmalmodelleditors wird in Java realisiert. Die Persistenz und der Export von Merkmaldiagrammen sowie Konfigurationen erfolgen unter Verwendung von XML.

Der Aufwand für die Entwicklung einer ansprechenden grafischen Benutzungsoberfläche ist nach wie vor mindestens so hoch wie der zur Implementierung des fachlichen Modells oder übertrifft diesen mitunter sogar. Ein beachtlicher Teil der Anwendungen benötigt jedoch eine grafische Benutzungsoberfläche. Sobald die Implementierung des fachlichen Kerns einer Anwendung in einer Systemfamilie weitgehend automatisch generiert werden kann, stellt sich die Frage, wie das Prinzip von *Economies of Scope* auch auf entsprechend angepasste Benutzungsoberflächen der einzelnen Mitglieder einer Systemfamilie übertragen werden kann. Im Gegensatz zu den bekannten GUI-Generatoren wie zum Beispiel JANUS ist *Angie Based guigenerAtor* (ABA) domänenspezifisch, das heißt, auch der GUI-Anteil des zu generierenden Systems wird einer Merkmalmodellierung unterzogen. Damit ist es möglich, Systeme zu erzeugen, die sowohl im generierten Quellcode als auch in der ausführbaren binären Datei nur das enthalten, was bezüglich GUI-Anteil und Funktionsumfang durch den Benutzer tatsächlich spezifiziert wurde. Es besteht weiterhin die Möglichkeit, selbsterstellte Komponenten oder die Komponenten, die von einem Designer entworfen wurden, zu benutzen. Obwohl ABA ein (Merkmal-)modellbasierter GUI-Generator ist, sind keine vertieften Kenntnisse des zugrunde liegenden Modells erforderlich, da dieses Modell vollständig auf die GUI-Elemente und den Spezifikator, welcher dem Generator vorgeschaltet ist, abgebildet wird. Dadurch kann der Generator auf eine intuitive und leicht verständliche Weise genutzt werden, wobei der gesamte Generierungsprozess für den Benutzer im Hintergrund verborgen abläuft.

Das Projekt *Generative Programmierung für web-orientierte Softwaresysteme* (*GP-WEB*) dient als Fallstudie für den Einsatz der generativen Programmierung. Das Ziel des Projekts ist es, die methodischen und technischen Voraussetzungen zu schaffen, welche die Entwicklung von Softwaresystemfamilien mittels generativer Programmierung auf dem Gebiet web-orientierter Anwendungen ermöglichen.

Im Projekt GP-WEB, welches durch das Bundesministerium für Bildung und Forschung (BMBF) gefördert wird, wurde ein Generatorprototyp entwickelt, der E-Learning/Web-Learning-Systeme sowohl hinsichtlich ihrer technischen Funktionalität als auch der gewünschten Lehrinhalte erzeugt. Neben den technischen Aspekten der automatisierten Erzeugung von Anwendungen auf der Grundlage einer Systemfamilie, ist natürlich auch interessant, wie und auf welchen Wegen ein menschlicher Anwender ein solches System nutzen kann. Zu diesem Zweck wurde unter anderem eine web-orientierte Umgebung entwickelt, mit der die einfache Benutzung möglich wird.

Als Basis zur Realisierung wurde das .NET Framework von Microsoft, speziell ASP.NET und Visual Basic, eingesetzt. Die entwickelte Web-Applikation erlaubt es dem Benutzer, ein personalisiertes Lernsystem via Internet komfortabel zu konfigurieren. Die implizite Baubarkeitsprüfung der Web-Applikation gewährleistet hierbei eine fehlerfreie Konfiguration. Das Frontend, welches als Interface zum Benutzer dient, reicht die Konfiguration an einen entsprechenden Web-Service weiter. Dort wird das gewünschte Endsystem durch einen Generator automatisch montiert und dem Anwender, der die Bestellung aufgegeben hat, per E-Mail zugestellt.

Die Aspekte der webseitigen Konfiguration, Prüfung und Generierung erscheinen durchaus auf weitere Domänen übertragbar.

Aktuelle Literaturhinweise

[CE00] gibt einen umfassenden Überblick über die generative Programmierung, wobei Methoden und Verfahren des Domain Engineerings mit geeigneten Implementierungstechniken und Vorgehensweisen für die Entwicklung von Softwaresystemfamilien durchgängig verbunden werden (siehe auch *www.generative-programming.org*).

Der Artikel „Generatorbau in C++“ [EC99] gibt eine Einführung in die generative Programmierung mit C++. Weitere Hinweise zur generativen Programmiertechniken in C++ beinhaltet der Beitrag „Template-Programmierung in C++“ Die Entwicklung von Anwendungsgeneratoren mit Java und XML wird in [Cle01] ausführlich behandelt. [ES02] beschreibt, wie ein Frameprozessor in Verbindung mit beliebigen textbasierten Programmiersprachen zur generativen Programmierung eingesetzt werden kann.

[WL99] berichtet über die Entwicklung von Softwaresystemfamilien bei AT&T und Lucent Technologies. Daraus hervorgegangene Verfahren und Werkzeuge werden detailliert besprochen.

Ein „lebendes“ Dokument über ein Rahmenwerk zur Entwicklung von Softwaresystemfamilien stellt das Software Engineering Institute (SEI) der Carnegie Mellon University (CMU) unter *www.sei.cmu.edu/plp/framework.html* zur Verfügung (siehe auch *www.sei.cmu.edu/programs/pls/*). Das SEI hat im August 2000 auch die “First International Software Product Line Conference” (SPLC1) veranstaltet. Das SPLC1-Konferenzband [Don00] enthält zahlreiche Erfahrungsberichte aus der Industrie und vermittelt einen Überblick über den neuesten Stand der Forschung.

Das Institut für experimentelles Software-Engineering (IESE) der Fraunhofer-Gesellschaft (*www.iese.fhg.de*) und der Arbeitskreis "Generative und komponenten-basierte Software-Entwicklung" der GI-Fachgruppe 2.1.9 (*www.prakinf.tu-ilmenau.de/~czarn/generate*) bieten ebenfalls Informationen über Softwareproduktlinien und –systemfamilien sowie verwandte Themen an. Das IESE hat im November 2000 den „Ersten Deutschen Software-Produktlinien-Workshop abgehalten. Die Workshopbeiträge wurden als IESE-Bericht [KP00] veröffentlicht.

[CN01] beschreibt, wie ein Produktlinien-basierter Ansatz im Rahmen eines Softwareentwicklungsprojektes aufgesetzt werden kann. Dabei werden Architekturaspekte sowie Fragen der Organisation und des Prozesses berücksichtigt. Das Buch umfaßt eine Reihe von Praktiken sowie eine Sammlung von Mustern, die beschreiben, wie diese Praktiken bei bestimmten typischen Problemstellungen anzuwenden sind.

Literatur

[Cle01] J. C. Cleaveland. *Program Generators Using XML and Java.* Prentice-Hall 2001

[CE00] K. Czarnecki and U. W. Eisenecker. *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley 2000

[CN01] P. Clements, L. Northrop. *Software Product Lines: Practice and Patterns.* Addison-Wesley 2001

[Don00] P. Donohoe (Ed.). *Software Product Lines: Experience and Research Directions. Proceedings of the First Software Product Lines Conference (SPLC1), August 28-31, 2000, Denver, Colorado, USA*, Kluwer Academic Publishers, Dordrecht, Holland, 2000

[EC99] U. W. Eisenecker und K. Czarnecki. Generatorbau in C++. In *OBJEKTspektrum*, Nr. 6, 1999, Seiten 76-83

[ES02] U. W. Eisenecker und R. Schilling: Generative Programmierung mit einem Frameprozessor. In *iX* 10/2002, Seiten 114-121

[KP00] P. Knauber und K. Pohl (Hg.). 1. Deutscher Software-Produktlinien Workshop (DSPL1), Kaiserslautern, November 2000. Proceedings. IESE-Report No. 076.00/E, November 10, 2000

[WL99] D. M. Weiss and C. T. R. Lai. *Software Product-Line Engineering: A Familiy-Based Software Development Process*. Addison-Wesley 1999