it 2/2008

Die Rolle von Features und Aspekten in der Softwareentwicklung

The Role of Features and Aspects in Software Development

Sven Apel, Universität Passau, Preisträger 2007 des Software-Engineering-Preises der Ernst-Denert-Stiftung*

Zusammenfassung Feature-orientierte Programmierung (FOP) und Aspekt-orientierte Programmierung (AOP) sind komplementäre Technologien. Obwohl beide auf die Modularität von so genannten querschneidenden Belangen abzielen, so tun sie dies auf unterschiedliche Art und Weise. Im Rahmen der Arbeit wurde beobachtet, dass FOP und AOP kombiniert werden können, um ihre individuellen Schwächen zu überwinden. Die Arbeit schlägt mit Aspekt-basierten Featuremodulen und Aspektverfeinerung zwei Techniken zur Symbiose von FOP und AOP vor. Beide Techniken werden in einer Fallstudie evaluiert und entsprechende Programmierrichtlinien zum Einsatz von FOP und AOP werden abgeleitet. Schlussendlich wird mittels der Analyse von acht AspectJ-Programmen unterschiedlicher Größe die Frage beantwortet, wie Implementierungsmechanismen der FOP und der AOP heutzutage Verwendung finden.

►►► **Summary** Feature-Oriented Programming (FOP) and Aspect-Oriented Programming (AOP) are complementary technologies. Though they aim at crosscutting modularity, they do so in different ways. We observed that FOP and AOP can be combined to overcome their individual limitations. Consequently, we propose with Aspectual Feature Modules (AFMs) and Aspect Refinement (AR) two techniques to unify FOP and AOP. We use AFMs and AR in a non-trivial case study to create a product line of overlay networks. We also present a set of guidelines to assist programmers in using FOP and AOP techniques for implementing product lines in a stepwise and generative manner. Finally, we answer the question of how FOP and AOP-related implementation techniques are used today by analyzing a representative set of eight AspectJ programs of different sizes.

KEYWORDS D.1 [Software: Programming Techniques], D.2 [Software: Software Engineering], D.3.3 [Software: Programming Languages: Language Constructs and Features] Feature-orientierte Programmierung; Aspekt-orientierte Programmierung, Aspekt-basierte Featuremodule, Aspektverfeinerung, Softwareproduktlinien, schrittweise Verfeinerung / feature-oriented programming, aspect-oriented programming, aspectual feature modules, aspect refinement, software product lines, stepwise refinement

1 Einleitung

Der Begriff *Softwaretechnik* und die damit verbundene Offensive erwuchs in den 60ern und 70ern aus den anhaltenden Problemen bei der Entwicklung von Software, welche unter dem Begriff *Softwarekrise* zusammengefasst werden [7]. Obwohl sich seitdem einiges bewegt hat, ist die derzeitige Situation in der Softwareentwicklung alles andere als zufriedenstellend. Laut dem aktuellen Bericht der Standish Group werden nur 34% aller Softwareprojekte erfolgreich zum Abschluss gebracht [5].

Seitdem werden zwei Prinzipien eng mit der Überwindung der Softwarekrise in Verbindung gebracht: *Trennung von Belangen (separation of concerns)* und *Modularität (modularity)* [3; 8]. Finden diese Prinzipien in der Entwicklung von Software Beachtung, lässt sich die Verständlichkeit, Wartbarkeit, Wiederverwendbarkeit und Maßschneiderbarkeit von Software signifikant verbessern. Allerdings stellte sich schnell heraus, dass

* Die Dissertation von Herrn Dr.-Ing. Sven Apel wurde mit dem Dissertationspreis 2007 der Ernst-Denert-Stiftung für Software-Engineering ausgezeichnet. Das Promotionsverfahren wurde an der Universität Magdeburg durchgeführt. Gutachter der Dissertation waren Prof. Dr. Gunter Saake, Universität Magdeburg, Prof. Don Batory, University of Texas at Austin, und Prof. Dr. Christian Lengauer, Universität Passau.

es weit komplizierter ist, adäquate Konzepte, Methoden, Formalismen und Werkzeuge zu entwickeln, als zunächst angenommen.

2 Zielstellung

Die Dissertation des Autors hat zum Ziel, zu diesem Bereich der Forschung beizutragen [1]. Dieser Artikel fasst seine Arbeit kurz zusammen. Im Speziellen beschäftigt sich die Arbeit mit zwei derzeitig diskutierten Programmierparadigmen, der *Feature-orientierten Programmierung (FOP)* [2; 9] und der *Aspekt-orientierten Programmierung (AOP)* [4; 6]. Beide Paradigmen konzentrieren sich auf eine bestimmte Klasse von Entwurfs- und Implementierungsproblemen, die so genannten *querschneidenden Belange (crosscutting concerns)*. Ein querschneidender Belang entspricht einer einzelnen Entwurfs- oder Implementierungsentscheidung bzw. einer Fragestellung oder eines Ansinnens, dessen Implementierung typischerweise über weite Teile eines Softwaresystems verstreut ist. Aus diesem Grund widersprechen querschneidende Belange den Prinzipien der Trennung von Belangen und der Modularität.

3 FOP und AOP

Feature-orientierte Programmierung (FOP) beschäftigt sich mit der Modularität von Features im Softwareentwicklungsprozess. Ein Feature entspricht einer Anforderung an ein Softwaresystem und stellt ein Inkrement in dessen Funktionalität dar. Features werden benutzt, um Gemeinsamkeiten und Unterschiede von Softwaresystemen zu kommunizieren. Üblicherweise lassen sich Features nicht durch einzelne Prozeduren oder Objekte implementieren. Oftmals betrifft die Implementierung eines Features viele Stellen im Code eines Softwaresystems. Features sind somit typische Beispiele für querschneidende Belange. FOP stellt Techniken zur modularen Implementierung von Features bereit.

Aspekt-orientierte Programmierung (AOP) beschäftigt sich mit ähnlichen Themen. Aspekte sind Mechanismen, die, per Definition, querschneidende Belange so implementieren, dass der entsprechende Code wohl modularisiert vorliegt. Aspekte sind nicht an Anforderungen bzw. Gemeinsamkeiten oder Unterschiede von Softwaresystemen gebunden, können wohl aber zu deren Repräsentation verwendet werden. Ein Unterschied zu FOP ist die Verwendung der Programmiermechanismen. Während FOP hauptsächlich auf die Erweiterung von Klassen und Methoden mittels Objekt-orientierter Techniken abzielt, verwendet AOP auch Techniken aus dem Bereich der Metaprogrammierung, z. B. zur Identifikation der Punkte in einem Programm, die erweitert bzw. modifiziert werden sollen.

4 Symbiose von FOP und AOP

FOP und AOP stellen beide methodische und programmiersprachliche Mittel und Werkzeuge bereit, gehen das Problem der querschneidenden Belange aber auf sehr unterschiedliche Weise an. In der Dissertation wird festgestellt, dass FOP und AOP keine konkurrierenden Ansätze sind, sondern dass ihre Kombination die individuellen Schwächen überwinden kann. Diese Einsicht wird untermauert durch eine Klassifikation von querschneidenden Belangen und eine Evaluierung von FOP und AOP hinsichtlich der verschiedenen Klassen querschneidender Belange. Ergebnis ist ein Satz von Programmierrichtlinien in Form eines Katalogs, der die Stärken und Schwächen von FOP und AOP gegenüberstellt.

5 Aspekt-basierte Featuremodule

Um von den individuellen Stärken beider Paradigmen zu profitieren, wird in der Dissertation die Symbiose von FOP und AOP vorgeschlagen. Insbesondere wird der Ansatz der *Aspekt-basierten Featuremodule (AFM: aspectual feature modules)* vorgeschlagen. AFMs setzen die Symbiose von FOP und AOP um, indem sie ihre Entwurfsphilosophien, Sprachmechanismen und Werkzeuge kombinieren. Eine Evaluierung und eine Gegenüberstellung mit traditioneller FOP und AOP demonstrieren die Überlegenheit von AFMs.

6 Verfeinerung von Aspekten

Des Weiteren wird in der Dissertation herausgestellt, dass derzeitige AOP-Sprachen nicht uneingeschränkt geeignet sind, in die schrittweise Entwurfsphilosophie von FOP integriert zu werden. Konsequenterweise wird der Ansatz der *Aspektverfeinerung (AR: aspect refinement)* vorgeschlagen, welcher AOP und schrittweise Softwareentwicklung [11] à la FOP vereinheitlicht. Weiterhin werden entsprechende Sprachkonstrukte und Werkzeuge zur Verfügung gestellt.

7 Fallstudien und Analysen

Mittels einer nicht-trivialen Fallstudie wird die praktische Anwendbarkeit von AFMs und AR auf ein mittelgroßes Softwareprojekt demonstriert. Die Studie wirft weiterhin eine fundamentale Frage auf: Wie werden Mechanismen von FOP und AOP heutzutage verwendet? Hintergrund ist, dass eine spezielle Klasse von querschneidenden Belangen eng mit FOP verknüpft ist, die so genannten *Kollaborationen* [10]. Durch die fehlende Unterstützung von Kollaborationen in aktuellen Programmiersprachen wird dafür heute oft AOP benutzt.

Durch das Aufkommen von Programmiersprachen, die Kollaborationen explizit unterstützen, sowie durch die in der Dissertation präsentierten Klassifikation und Evaluierung, stellen sich jedoch folgende Fragen: Welcher Anteil von Aspektcode implementiert Kollaborationen? Welcher Anteil implementiert querschneidende Belange, die darüber hinaus AOP benötigen? Eine quantitative Analyse von acht AspectJ-Programmen unterschied-

licher Größe ergibt, dass durchschnittlich 98% der Codebasis der analysierten Programme mit Kollaborationen verknüpft sind und nur 2% die erweiterten Mittel von AOP jenseits von Kollaborationen ausnutzen. Weiterhin wird beobachtet, dass mit steigender Programmgröße der Einfluss von AOP sinkt.

In der Dissertation wird die Frage beantwortet, warum dieses (Miss)Verhältnis zwischen AOP und FOP-Code besteht, und warum dies nicht überrascht. Weiterhin wird diskutiert, ob und wie der positive Einfluss von AOP gesteigert werden kann.

8 Schlusswort

Die Gegenüberstellung von FOP und AOP führte nicht nur zur Symbiose beider Programmierparadigmen, sondern lieferte auch tiefe Einsichten in ihre individuelle Natur, in die Einordnung in übergeordnete Paradigmen wie schrittweise Softwareentwicklung und in die aktuelle Programmierpraxis. Wenn auch beide Paradigmen ständigem Wandel unterzogen sind, werden diese Erkenntnisse doch sicher ihre Entwicklung beeinflussen.

Literatur

[1] S. Apel. *The Role of Features and Aspects in Software Development*. PhD thesis, School of Computer Science, University of Magdeburg, 2007.

[2] D. Batory, J. N. Sarvela, and A. Rauschmayer. Scaling Step-Wise Refinement. *IEEE Trans. on Software Engineering*, TSE-30(6):355–371, 2004.

[3] E. W. Dijkstra. *A Discipline of Programming*. Prentice Hall, 1976.

[4] T. Elrad, R. E. Filman, and A. Bader. Aspect-Oriented Programming: Introduction. *Communications of the ACM*, CACM-44(10):29–32, 2001.

[5] The Standish Group. Chaos Report. Technical report, Standish Group International, 2003.

[6] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. V. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-Oriented Programming. In: *Proc. of the European Conf. on Object-Oriented Programming (ECOOP)*, vol. 1241 of *Lecture Notes in Computer Science*, pages 220–242. Springer, 1997.

[7] P. Naur and B. Randell, editors. *Software Engineering: Report of the Working Conference on Software Engineering, Garmisch, Germany, Oct 1968*. NATO Science Committee, 1969.

[8] D. L. Parnas. On the Criteria to be Used in Decomposing Systems into Modules. *Communications of the ACM*, CACM-15(12):1053–1058, 1972.

[9] C. Prehofer. Feature-Oriented Programming: A Fresh Look at Objects. In: *Proc. of the European Conf. on Object-Oriented Programming (ECOOP)*, vol. 1241 of *Lecture Notes in Computer Science*, pages 419–443. Springer, 1997.

[10] T. Reenskaug, E. Andersen, A. Berre, A. Hurlen, A. Landmark, O. Lehne, E. Nordhagen, E. Ness-Ulseth, G. Oftedal, A. Skaar, and P. Stenslet. OORASS: Seamless Support for the Creation and Maintenance of Object-Oriented Systems. *Journal of Object-Oriented Programming*, 5(6):27–41, 1992.

[11] N. Wirth. Program Development by Stepwise Refinement. *Communications of the ACM*, CACM-14(4):221–227, 1971.

Dr.-Ing. Sven Apel ist akademischer Oberrat am Lehrstuhl für Programmierung der Fakultät für Informatik und Mathematik an der Universität Passau. Er promovierte im März 2007 an der Fakultät für Informatik der Otto-von-Guericke-Universität Magdeburg. Anfang 2006 war er für sechs Monate als Gastwissenschaftler an der University of Texas at Austin tätig. Sein Interesse in der Forschung gilt modernen Programmierparadigmen, Softwareproduktlinien sowie formalen Methoden für die Softwareentwicklung. Sven Apel erhielt im Januar 2007 den Forschungspreis der Fakultät für Informatik der Otto-von-Guericke-Universität Magdeburg für seine wissenschaftliche Gesamtleistung. Im September 2007 wurde ihm der Software-Engineering-Preis der Ernst Denert-Stiftung verliehen. Im November 2007 wurde ihm der Fakultätspreis der Fakultät für Informatik der Otto-von-Guericke-Universität Magdeburg für den besten Doktoranden sowie der Dissertationspreis der Otto-von-Guericke-Universität Magdeburg verliehen. Sven Apel ist Autor von über 40 referierten wissenschaftlichen Publikationen.

Adresse: Lehrstuhl für Programmierung, Fakultät für Informatik und Mathematik, Universität Passau, Innstraße 33, 94032 Passau, Deutschland,
E-Mail: apel@uni-passau.de