Konsolidierung von Software-Varianten in Software-Produktlinien — ein Forschungsprogramm

Rainer Koschke
Universität Bremen
http://www.informatik.uni-bremen.de/st/
koschke@informatik.uni-bremen.de

Zusammenfassung

Software-Varianten entstehen oft, indem ganze Software-Systeme im großen Stil kopiert und an die neuen, spezifischen Anforderungen angepasst werden. Diese Software-Varianten erfahren dann eine eigene Wartung. Dadurch entsteht jedoch unnötige Mehrarbeit in der Wartung. Dieselben Änderungen, die die ursprüngliche gemeinsame Code-Basis betreffen, müssen in allen Varianten wiederholt werden.

Durch kontrolliertes Management von Varianten in Software-Produktlinien können diese Probleme vermieden werden. Jedoch müssen hierzu die existierenden Varianten erst zu einer Produktlinie konsolidiert werden. Dazu müssen Gemeinsamkeiten und Unterschiede (Variabilitäten) der Varianten identifiziert werden.

Dieser Beitrag stellt unser auf drei Jahre ausgelegtes Forschungsvorhaben vor, das Methoden und Techniken erarbeiten soll, um existierende Software-Varianten in Software-Produktlinien zu konsolidieren. Wir werden dieses Problem sowohl auf der Quellcodeebene als auch auf der Architekturebene untersuchen. Hierbei bauen wir auf einer Reihe von Techniken auf, die wir bereits erfolgreich für die Analyse einzelner Systeme eingesetzt haben. Dazu zählen unsere Techniken zur Merkmalsuche, Klonerkennung, Protokollerkennung und reflektionsbasierter Architekturrekonstruktion. Diese Techniken werden wir für eine inkrementelle Untersuchung von Varianten adaptieren und erweitern.

Das Ziel dieser Erweiterung ist die Unterstützung eines inkrementellen Vorgehens, bei dem zu einem Zeitpunkt, an dem bereits n Systeme untersucht wurden, nur die Unterschiede des nächsten Systems zu den bisherigen n Systemen gezielt analysiert werden müssen.

1 Motivation

In softwareentwickelnden Unternehmen wird eine immer schnellere Reaktion auf neue Anforderungen für Produkte gefordert. Der Markt der in der Softwareproduktion tätigen Unternehmen ist dabei geprägt durch das Bestreben, die folgenden, sich teilweise widersprechenden Aspekte zu optimieren: Schnelle „Time-to-Market“ bzw. „Time-to-Delivery“, niedrige Entwicklungskosten, hohe Software-Qualität, niedrige Wartungs- und Evolutionskosten und hohe Vorhersagbarkeit bei Projektablauf und Qualität. Wiederverwendung vorhandener Software-Komponenten wird – neben generativen Ansätzen – als der prädominante Weg gesehen, diese Ziele zu realisieren.

Opportunistische, weitgehend ungeplante Wiederverwendung hat sich aber als ungeeigneter Weg erwiesen. Vielmehr ist eine aktive Gestaltung der wiederverwendbaren Komponenten notwendig, die auch Variabilität zur Anpassung an die spezifischen Erfordernisse des Einzelprodukts vorsieht.

Die Planung und Entwicklung technischer Produkte als Mitglieder von Produktfamilien sind in Branchen wie dem Anlagenbau oder in der Automobilindustrie heute bereits Stand der Technik und nicht mehr wegzudenken. Im Bereich der Softwaresysteme verstärkt sich seit einigen Jahren der Trend hin zu dieser Art der Wiederverwendung. Durch die starke Zunahme des Anteils software-intensiver Systeme und Produkte in nahezu jeder Branche, beispielsweise im PKW-Bereich oder in der Telekommunikation, ist eine starke Zunahme des Interesses an der produktfamilienbasierten Software-Entwicklung zu verzeichnen.

Speziell für Firmen, deren einzelne Softwareprodukte große Gemeinsamkeiten aufweisen, liegt dieser Weg nahe. Andererseits scheuen viele Firmen diesen Weg, weil sie bereits beträchtliche Summen Geld in die Entwicklung ihrer Einzelsysteme investiert haben. Sie können es sich nicht leisten, eine neue Software-Produktlinie aufzusetzen, ohne existierende Implementierungen einzubeziehen. Dies würde zu viel Entwicklungszeit verlangen und ein zu hohes Risiko darstellen. Ein gangbarer Weg ist es, die Produktlinie statt dessen evolutionär und inkrementell einzuführen, indem existierende System konsolidiert und schrittweise in eine Produktlinie integriert werden. Hierzu werden Techniken für die Analyse existierender Software-Varianten benötigt.

Forschungsziel. Der Forschungsbeitrag des Projekts ist es, Analyse- und Restrukturierungstechniken bereitzustellen, um die für eine Produktlinie notwendigen, anpassungsfähigen Komponenten aus existie-

renden Produkten, die sich am Markt bewährt haben, zu extrahieren und in eine Produktlinienplattform zu integrieren.

Die Entwicklung neuer Methoden und Werkzeuge für die technische Unterstützung der Konsolidierung ist notwendig, um die Gemeinsamkeiten und Variabilitäten zwischen den Software-Produkten analytisch erfassen und in benutzerverständlichen Darstellungsformen präsentieren zu können. Grundlegende Forschung ist nötig zur Klärung der Frage, wie aus den analytischen Ergebnissen auf vereinheitlichte Software-Produktlinien-Architekturen geschlossen werden kann. Unsere bereits existierenden Analysen für Einzelsysteme sollen hierzu entsprechend angepasst und erweitert werden. Die Umsetzung in realistisch einsetzbare Werkzeuge ist zentrales Ziel unserer Forschung.

2 Ziele und Arbeitsprogramm

Insgesamt ist festzuhalten, dass es eine aktive und umfangreiche Forschung zu Software-Produktlinien gibt. Die Rekonstruktion einer Produktlinienarchitektur aus existierenden ähnlichen Systemen und Konsolidierung der Software-Varianten in der Plattformarchitektur werden allerdings nur ungenügend behandelt.

Hier setzt nun die vorgeschlagene Forschung an: Durch den Einsatz moderner Analysetechniken sollen die Architekturen der Software-Varianten inkrementell extrahiert und einander gegenüber gestellt werden. Inkrementell bedeutet in diesem Zusammenhang, dass die Rekonstruktion für Variante N auf die bereits vorhandenen Informationen für die Varianten $1 \ldots N-1$ zurückgreifen kann. Gemeinsamkeiten und Unterschiede von Aspekten sowohl der Struktur als auch des Verhaltens sollen identifiziert und explizit repräsentiert werden.

Hierzu sollen spezifisch jene Techniken herangezogen werden, mit denen wir bereits bei der Analyse von Einzelsystemen gute Erfahrungen gemacht haben. Sie müssen an die Besonderheiten von Software-Produktlinien angepasst werden.

Komponentenkorrespondenz. Beim Vergleich zweier Varianten stellt sich zunächst die Frage, welche Komponenten der Varianten miteinander korrespondieren. Da wir von Varianten ausgehen, die einer gemeinsamen Code-Basis entstammen, können hier Korrespondenzen über Namen und syntaktische Übereinstimmungen hergestellt werden. Für letztere kann unsere Implementierung zur Klonerkennung herangezogen werden. Allerdings ist auch der Fall zu erwarten, dass Komponenten in einer Variante komplett reimplementiert wurden. In diesem Fall müssen die Schnittstellen manuell aufeinander abgebildet werden.

Die Klonerkennung muss an die spezifischen Bedürfnisse des neuen Kontexts der Analyse von System-Varianten angepasst werden. Außerdem ist man in unserem Anwendungsfall nicht nur an den Übereinstimmungen, sondern gerade auch an den Differenzen interessiert. Man interessiert sich also für den Quellcode, der keine Entsprechung findet.

Letztlich lässt sich nicht automatisch entscheiden, ob zwei Komponenten dieselbe Semantik haben, so dass hierfür eine Einbeziehung des Benutzers von Nöten sein wird.

Erweiterung Merkmallokalisierung. Unsere bisherige Merkmallokalisierung muss erweitert werden, so dass sie produktübergreifend durchgeführt werden kann [1]. Der Kern der Analyse ist die mathematische Begriffsanalyse, mit deren Hilfe man binäre Relationen analysieren kann. Hierzu muss der formale Kontext für die Begriffsanalyse festgelegt werden, d.h. – in der Terminologie der formalen Begriffsanalyse gesprochen – die Objekte, Attribute und die dazwischen bestehende Relation. In der Analyse von Einzelsystemen verwenden wir als Objekte die Menge der Routinen eines Systems und für die Attribute die Menge der Merkmale. Die Relation beschreibt, welche Routine für welches Merkmal ausgeführt wird. Sie lässt sich durch wiederholte Programmläufe mit den unterschiedlichen Merkmalen gewinnen, bei denen mitprotokolliert wird, welche Routinen zur Ausführung gelangen. Der aus der Begriffsanalyse resultierende Begriffsverband gibt Hinweise auf die Routinen, die spezifisch für unterschiedliche Mengen von Merkmalen sind.

Erweiterung Reflektionsmodell. Das Reflektionsmodell wird für Software-Varianten erweitert [3]. Hierzu sind sowohl konzeptuelle Erweiterungen an der Ausdrucksmächtigkeit für Soll-Architekturen (die in diesem Kontext zur Produktlinienarchitektur wird) als auch an der Abbildung der Implementierungskomponenten auf Komponenten der Produktlinienarchitektur notwendig. Die Abbildung muss inkrementell ausgelegt werden.

Erweiterung Protokollerkennung. Die Herleitung der Protokolle für Einzelsysteme wird um eine Phase erweitert, die Unterschiede und Gemeinsamkeiten der Komponenten in Bezug auf ihr Protokoll herausarbeitet [2].

Literatur

[1] T. Eisenbarth, R. Koschke, and D. Simon. Locating features in source code. *IEEE Transactions on Software Engineering*, 29(3), 2003.

[2] T. Eisenbarth, R. Koschke, and G. Vogel. Static object trace extraction for programs with pointers. *Journals of Systems and Software*, 2005.

[3] R. Koschke and D. Simon. Hierarchical reflexion models. In *Working Conference on Reverse Engineering*, pages 36–45. IEEE Computer Society Press, Nov. 2003.