Adaptierbarkeit und Adaptivität durch Laufzeitmodelle

Jürgen Ebert
Universität Koblenz-Landau
Institut für Softwaretechnik
Universitätsstraße 1
56070 Koblenz
ebert@uni-koblenz.de

Abstract

Änderungen von Softwaresystemen können grundsätzlich durch regelmäßige externe Anpassungsmaßnahmen (Makro-Evolution), aber auch durch permanente Selbstanpassung (Mikro-Evolution) erreicht werden. Interpretierbare Laufzeitmodelle bieten Unterstützung für beide Ansätze.

Der Vortrag beschreibt eine generische Architektur und berichtet von ersten Fallstudien zur Nutzung von Laufzeitmodellen zur Verbesserung der Änderbarkeit.

1 Einleitung

Software altert kontinuierlich relativ zu ihrem Kontext. Die Komplexität des Codes steigt und die Struktur des Systems geht durch zahlreiche Änderungen nach und nach verloren [LPRW97].

Um Software langlebiger zu machen muss auf vorhersehbare, aber auch auf unvorhersehbare Änderungsanforderungen reagiert werden. *Adaptierbarkeit* (adaptibility) ist die Fähigkeit eines Systems, bei veränderten Kontext-Bedingungen leicht geändert werden zu können. *Adaptivität* (adaptivity) hingegen bezeichnet die Fähigkeit eines Systems, sich geänderten Bedingungen gegenüber selbst anpassen zu können. Während Adaptierbarkeit leichte manuelle Änderbarkeit erfordert, ohne dass eine konkrete Form von Änderung erwartet wird, geschieht Adaptivität eher „automatisch“ und setzt vorhersehbare Änderungen voraus.

Ziel einer auf Langlebigkeit ausgerichteten Softwaretechnologie muss es sein, sowohl Adaptierbarkeit als auch Adaptivität in ausgewogener Form zu unterstützen. Adaptierung betrifft größere Änderungen im Zuge von Wartungsaktivitäten. Adaptivität bezieht sich auf kleinere Änderungen, die automatisch und permanent geschehen. Die Evolution eines Softwaresystems kann somit als Folge von wenigen, eher größeren *Makro*-Evolutionsschritten und vielen, eher kleinen *Mikro*-Evolutionsschritten aufgefasst werden.

2 Modelle

Modelle werden verwendet, um relevantes Wissen über Softwaresysteme in konziser Weise zu repräsentieren. Modelle werden häufig in visuellen und (semi-)formalen Sprachen beschrieben. Sie können als Graphen repräsentiert werden, deren Gestalt durch Metamodelle festgelegt ist.

Model-basierte Software baut auf dem in Modellen kodierten Wissen auf. Durch Generierung wird aus Modellen Code erzeugt, in dem die Modelle selbst *implizit* enthalten sind. Wenn implizite Modelle geändert werden sollen, ist eine Re-Generierung erforderlich.

Modelle können aber zur Laufzeit *explizit* gehalten und dynamisch interpretiert werden. Bei der Adaption solcher *Modell-zentrierten* Software werden die expliziten Modelle zur Laufzeit geändert. Dadurch ändert sich direkt auch das Verhalten des Systems. Diese Änderbarkeit unterstützt also die Adaptivität der Software.

3 Laufzeit-Modelle

Durch die Verwendung von Software-Komponenten, die den für die vorhersehbaren Änderungen relevanten Anteil in expliziten Modellen enthalten, die interpretiert und aufgrund vordefinierter Regeln zur Laufzeit verändert werden, sind adaptive Komponenten möglich, die ihr Verhalten zur Laufzeit verändern können.

Durch klare Schnittstellen zwischen den Laufzeitmodellen und dem konventionell kodierten Code, bleibt weiterhin für unvorhergesehene Änderungen der Spielraum einer tiefergehenden Anpassung der Komponenten.

Das in einer Kooperation zwischen der University of Waterloo und der Universität Koblenz entstandene ‚Graph-based Runtime Adaptation Framework (GRAF)' [DAET11] (Abb. 1) unterstützt diese Trennung in Laufzeitmodelle und Anwendungscode.

GRAF basiert auf Anfragen, Transformationen und Interpretieren der (abstrakten Syntax) von beliebigen Modellen zur Laufzeit. Das Framework trennt die Laufzeitmodelle und deren Infrastruktur von der Anwendung (adaptable software).

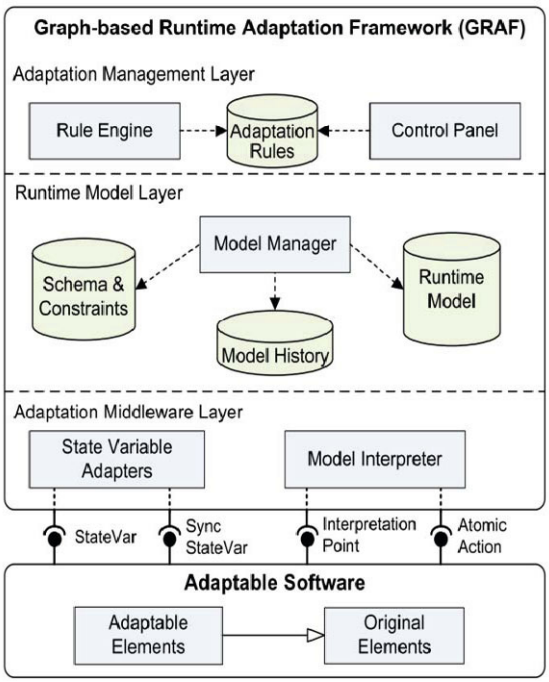

Abbildung 1 GRAF (Aufbau) [DAET11]

Mikro-Evolution besteht in GRAF aus der Reifikation ausgewählter Zustände der adaptierbaren Software im Laufzeitmodell (monitoring), der Beobachtung des Laufzeitmodells (analyzing) durch Regelbedingungen, der Auswahl einzelner möglicher Änderungsregeln (planning) und der Durchführung der von Modelltransformationen (executing) in einer MAPE-Schleife.

Makro-Evolution kann sowohl dieses Regelwerk ändern als auch Anpassungen der adaptierbaren Software vornehmen, sowie weitere Verbindungen zwischen der adaptiven Software und dem Framework herstellen.

Die Separierung von Modellen (in einem Framework) vom Code innerhalb von Komponenten kann sowohl initial zur *Konstruktion* adaptierbarer Komponenten verwendet werden, als auch im Nachhinein im Rahmen von *Reengineering*-Maßnahmen einer existierenden Komponente hinzugefügt werden.

4 Ausblick

Der vorgestellte Ansatz ist nicht auf die Verwendung konkreter Modellierungssprachen festgelegt, und die Grenze zwischen Code und Modell kann ebenfalls verschieden gewählt werden.

Zur Zeit ist GRAF konkret für die Verwendung von Aktivitätsdiagrammen als Laufzeitmodellen ausgelegt. In weiteren Arbeiten wird der Ansatz in generischer Form realisiert, um von konkreten Modellierungssprachen unabhängig zu werden.

Danksagung

Die in diesem Vortrag präsentierten Überlegungen sind in Zusammenarbeit mit Mahdi Derakhshanmanesh (Koblenz) und Ladan Tahvildari und Mehdi Amoui (Waterloo) entstanden

Referenzen

[DAET11] Mahdi Derakhshanmanesh, Mehdi Amoui, Jürgen Ebert, Ladan Tahvildari. *GRAF: Graph-Based Runtime Adaptation Framework.* Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2011), Waikiki, Honolulu, Hawaii, USA, 2011.

[LPRW97] M.M. Lehman, D.E. Perry, J.F. Ramil, P.D. Wernick. *Metrics and Laws of Software Evolution - The Nineties View.* Proceedings of the 4th International Software Metrics Symposium (METRICS '97), IEEE, 1997