Modellbasierte Evolution als Schlüssel nachhaltig anpassbarer Software Systeme

Bernhard Rumpe und Steven Völkel
Software Engineering Group
Department of Computer Science, RWTH Aachen
http://www.se-rwth.de

Software veraltet. Um Software Systeme, Software-Komponenten oder Wissen über Software langlebig zu erhalten, sind ähnliche Maßnahmen wie bei Maschinen oder Gebäuden notwendig. Software Systeme wollen gepflegt, den Bedürfnissen angepasst, ggf. saniert und auch entmüllt werden.

Dabei ist das Problem nicht die Stabilität, sondern

- die Flexibilität mit einem gewandeltem Systemkontext umzugehen,
- die Migrationsfähigkeit auf neue Rechner-Hardware,
- die Anpassungsfähigkeit für neue Sensorik/Umgebungsinformation und
- die Anpassungsfähigkeit für geänderte/neue Geschäftsprozesse.

Dabei gibt es sehr wohl domänenspezifische Unterschiede, generell aber gilt, dass diese Probleme in ähnlicher Form sowohl bei eingebetteten Systemen verschiedener Coleur als auch bei langlebigen Business-Systemen auftreten.

Genereller Lösungsweg aus dieser Problematik besteht im Anheben des Abstraktionsgrades mit folgenden Hintergründen/Randbedingungen:

1. Die Abstraktion vom technischen Systemkontext führt zu stabileren, leichter migrierbaren Systemen. Abstraktion vom Systemkontext bedeutet unter anderem standardisierte Schnittstellen und Systemdienste (Betriebssystem, Network-Protokolle, Datenbanken).
2. Software und speziell ihre Strukturen sollte sich stark an Anforderungen orientieren. Idealerweise sollten Geschäftsprozesse und Anforderungen direkt abgebildet und nicht durch architektonische, optimierende oder sonstige Überlegungen zu sehr über die Codestruktur zerfasert werden. Dies führt zu deutlich wartbarerem und weiterentwickelbarem Code.
3. Abstraktion sollte auch kompaktere Programmierung/Modellierung erlauben. Damit ist trotz der Größe des Produkts eine schnellere Entwicklung und damit ein kleineres Projekt möglich. Die Entwicklungsartefakte sind kleiner und die Weiterentwicklung damit flexibler.

[RV09] B. Rumpe, S. Völkel
Modellbasierte Evolution als Schlüssel nachhaltig anpassbarer Software Systeme
In: 1. Workshop für langlebige und zukunftsfähige Softwaresysteme 2009
Informatik-Bericht 2009-05, TU Braunschweig, März 2009
www.se-rwth.de/publications

Evolution steigert die Wiederverwendung und damit die Effektivität der Entwickler sowie typischerweise auch die Qualität des Ergebnisses, weil die modifizierte Form der Software auf die Erprobungsphasen vorangegangener Fassungen zurückgreifen kann.

Evolution von Code heißt heute die Planung von Umbauten im Kopf vorzunehmen und idealerweise durch eine Serie kleiner Refactorings umzusetzen. Leider integriert diese Form der Evolution des Systems in keinster Weise Artefakte der früheren Entwicklungsphasen.

Ideal wäre deshalb die Planung und Durchführung von Evolutionsschritten auf Basis von Modellen, die durch geeignete Generierungstechniken per Konstruktion synchron zu Code sind und somit die Konsistenz zwischen dokumentierenden Modellen und Code sicherstellen. Aufgrund der den Modellen typischerweise inhärenten höheren Abstraktion und besseren Darstellungsform ist Refactoring auf Modellbasis besser planbar und effizienter durchführbar.

Refactoring auf Modellbasis bedeutet Evolution in kleinen, systematischen und überschaubaren Schritten. Systeme, die auf dieser Basis und mit der Idee der Refaktorisierbarkeit gebaut sind, sollten langlebig sein und gleichzeitig über ihre lange Lebensdauer hinweg Qualität und Zuverlässigkeit besitzen.

Voraussetzung für qualitätsgesicherte Modellevolution sind jedoch

- gute, intelligente Codegeneratoren,
- automatisierte Regressions-Tests und
- geeignete Sammlungen von Modelltransformationen.

Für einzelne UML-Notationen existieren bereits einige „Refactoring“-Techniken, teilweise als Übernahme von Refactorings aus dem Code (z.B. bei Klassendiagrammen), teilweise auch als mathematische Kalküle zur Modelltransformation (z.B. bei Statecharts, OCL).

Extreme Programming liefert die methodische Unterfütterung für die Programmierung, Eclipse, JUnit und Verwandte die Werkzeuge zur Durchführung und MDA die Methodik für die modellbasierte Softwareentwicklung. Sie nutzt Codegeneratoren und Transformationssprachen. Allerdings sind weder die Techniken ausgereift und ausgereizt, noch in breiter Masse in industrieller Stärke verfügbar.

So fehlen zum Beispiel gute Modularisierungstechniken für Modellierungssprachen ebenso wie für Transformationssprachen. Wenn wir Abstraktion, Konfiguration und Software-Planung auf Modellbasis einsetzen wollen, dann sind bessere Sprachen notwendig.

Modellbasiertes Software Engineering hat Potential für bessere Evolution, mehr Flexibilität des Produkts, stärkere Konfigurierbarkeit und Anpassbarkeit damit über den Lebenszyklus hinweg bessere Wartbarkeit von Systemen und eignet sich daher grundsätzlich für langlebige, zukunftsfähige Software.

Frage ist nur: Welche Modelle wären geeignet? Wann werden sie in welcher Form eingesetzt?