Preprint Version: The original publication is available at www.springerlink.com
http://dx.doi.org/10.1007/s00287-009-0349-2

Aktuelles Schlagwort: Software-Evolvability

Matthias Riebisch, Stephan Bode
Technische Universität Ilmenau
{matthias.riebisch|stephan.bode}@tu-ilmenau.de

Einleitung

Von vielen Softwaresystemen wird ein langer Nutzungszeitraum verlangt. Dazu muss die Software den ständigen Forderungen nach Veränderungen nachkommen können, z. B. aufgrund der Optimierung von Prozessen oder durch Einbindung in andere Systeme. Im Zuge der Software-Wartung sollen solche Änderungen mit geringem Aufwand und in kurzer Zeit ausgeführt werden. Dabei tritt jedoch ein als Architekturerosion (engl. architectural decay, architectural drift) bezeichneter Effekt auf, indem die Änderungen zu einer Verschlechterung der Struktur führen, die weitere Änderungen erschwert oder verhindert. Der Ersatz eines betroffenen Systems durch eine Neuentwicklung ist aufgrund der Risiken, der Kosten und des Zeitbedarfs keine gute Alternative. Daraus entsteht *über die Software-Wartung hinaus* die Forderung, die Software langfristig in einem Zustand zu erhalten, der schnelle und einfache Änderungen erlaubt. Die Eigenschaft Evolvability (dt. etwa Weiterentwickelbarkeit) kennzeichnet diesen Zustand. Aufwand, strategisches Vorgehen und Komplexität zu ihrer Erhaltung sind wesentlich anspruchsvoller.
Der Begriff Evolution wurde im Zusammenhang mit Software bereits seit den 1970er-Jahren benutzt, als die Änderungsproblematik bei ersten großen Softwaresystemen deutlich wurde, und erlangte in den 1990er-Jahren stärkere Beachtung. Ein wichtiger Beitrag stammt z. B. von Rajlich und Bennet [11], die mit dem Staged Model (Abb. 1) die Software-Wartung beschrieben haben.

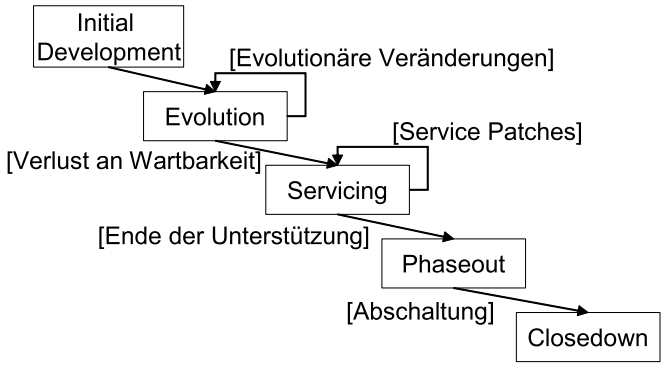

Abb. 1: Staged Model mit Zuständen während der Wartung nach [11]

Begriff und Bezug zu verwandten Themen

Evolution als Begriff steht eng im Zusammenhang mit den Termini *Wartung* (engl. maintenance), *Wartbarkeit* (maintainability) sowie *Evolvability. Reengineering* beschreibt Maßnahmen zur Verbesserung der Wartbarkeit. Die differenzierte Betrachtung der Eigenschaften Wartbarkeit und Evolvability macht den Aspekt der Langfristigkeit deutlich.

Evolvability

Für Evolvability gibt es derzeit verschiedene Definitionen, von denen Rowe et al. [12] einige zusammengetragen haben. Die Unterscheidung zur Wartbarkeit ist von besonderer Bedeutung, weil ein Softwaresystem nicht notwendigerweise gut weiterentwickelbar ist, wenn es sich leicht warten lässt. Im Gegensatz zu anderen trägt die Definition von Breivold et al. [3] dem Rechnung, indem sie für Evolution typische Aspekte wie strukturelle Änderungen und Erhaltung der Architekturintegrität berücksichtigt:

Preprint Version: The original publication is available at www.springerlink.com
http://dx.doi.org/10.1007/s00287-009-0349-2

„Software evolvability is the ability of a software system to adjust to change stimuli, i.e. changes in requirements and technologies that may have impact on the software system in terms of software structural and/or functional enhancements, while still taking the architectural integrity into consideration." [3]
Die Evolvability eines Softwaresystems sollte hoch sein, denn dadurch kann verhindert werden, dass ein System zu einem sogenannten Legacy-System wird [9], indem es die Fähigkeit zu Evolution verliert. Aktivitäten für die Erhöhung der Evolvability tragen immer auch zur Verbesserung der Wartbarkeit bei, was umgekehrt nicht gilt. Beispiele sind die Entwicklung einer Plugin-Schnittstelle, die den Austausch von Komponenten statt der Weiterentwicklung durch Änderungen ermöglicht, oder die Einführung einer Abstraktionsschicht für Verteilung oder Plattformwechsel.

Evolution

Evolution umfasst nach Lehmans *„laws of software evolution"* [8] die Entwicklung eines Softwaresystems über den gesamten Lebenszyklus hinweg von der Erstellung bis zur Stilllegung und schließt damit initiale Entwicklung, Wartung und Reengineering ein. Der Begriff betont durch den Gegensatz zu Revolution den Charakter schrittweiser, kontinuierlicher Änderungen. Rajlich und Bennett unterscheiden in ihrem *Staged Model* des Software-(wartungs-)lebenszyklus (Abb. 1) den Zustand *Evolution,* in dem evolutionäre Veränderungen möglich sind, vom Zustand *Servicing*, der nach Verlust von Wartbarkeit erreicht wird. Evolvability beschreibt also die Fähigkeit, Wartbarkeit langfristig zu erhalten [11].
Evolutionäre Veränderungen umfassen zusätzlich zur Wartung auch Modifikationen struktureller Art. Typische Beispiele sind – im Gegensatz zu unten genannten Wartungsmaßnahmen – die Integration neuer Technologien und Plattformen wie die Kapselung eines bestehenden Systems als Service einer service-orientierten Architektur (SOA), die Umsetzung neuer qualitativer Anforderungen wie die Steigerung von Skalierbarkeit und Mehrsprachigkeit für einen weltweiten Einsatz, oder eine Änderung des Architekturstils wie die Einführung einer neuen Abstraktionsschicht für die Speicherung von Geschäftsobjekten. Die Betonung liegt nicht auf der Lebenserhaltung des Systems, sondern auf der Weiterentwicklung der Architektur. Evolution ist nicht auf Legacy-Systeme begrenzt, sondern ist für alle Systeme erforderlich, die einen hohen Wert für ein Unternehmen haben.

Wartung

Software-Wartung bezeichnet im Sinne der Aufrechterhaltung die Modifikation eines Softwareproduktes nach dessen Auslieferung zur Anpassung an veränderte Anforderungen (auch Qualitätsattribute), an eine veränderte Umgebung sowie zur Fehlerbehebung [6]. Die Aktivitäten umfassen meist kleine Änderungen, weil Zeit- und Kostenforderungen eine wichtige Rolle spielen. Dieser Zeit- und Kostendruck führt häufig zu unvollständigen Änderungen und zu Folgefehlern. Wartungsaktivitäten tragen nicht notwendigerweise zu Wartbarkeit und zu Evolution bei. Beispiele für solche Aktivitäten sind Erweiterungen zur Behandlung zusätzlicher logischer Bedingungen, was zu verringerter Wartbarkeit führt, wenn komplizierte Fallunterscheidungen oder eine Mehrfachnutzung von Parametern die Verständlichkeit verringern. Führen Erweiterungen zu sehr langen Methoden oder Blöcken, zu Code-Duplikaten oder zu vielen lokalen Variablen, ist eine verringerte Wartbarkeit die Folge. Daraus resultieren Folgefehler, die letztlich zur Architekturerosion führen.

Wartbarkeit

Die Wartbarkeit bezieht sich auf den Aufwand für Änderungen. Das Qualitätsmerkmal ist in der ISO-Norm 9126 [7] als Fähigkeit eines Softwareproduktes definiert, modifiziert zu werden. Dazu gehören Korrekturen, Verbesserungen oder Anpassungen der Software an Änderungen der Umgebung sowie der Anforderungen und der funktionalen Spezifikationen. Das Qualitätsmodell der ISO 9126 verfeinert die Wartbarkeit durch die Untermerkmale Analysierbarkeit, Änderbarkeit, Stabilität, Testbarkeit und Standardkonformität.
Diese Untermerkmale sind keinesfalls als vollständig anzusehen. Die Autoren beziehen Verständlichkeit ein, denn Programmverstehen macht bis zu 50 Prozent des Wartungsaufwandes aus. Traceability spielt für die Wartbarkeit ebenfalls eine große Rolle, denn Traceability Links unterstützen die Nachvollziehbarkeit von Entwurfsentscheidungen und damit die Verständlichkeit. Gute Wartbarkeit erhöht die Lebensdauer, indem die Entwicklungsrisiken verringert werden. Trotzdem fehlt diesem Qualitätsmerkmal der Fokus auf langfristige Aufrechterhaltung, weil sie den aktuellen Aufwand für Änderungen in den Vordergrund stellt. Die Wartbarkeit kann beispielsweise über die Code-Qualität verbessert werden, jedoch ohne wesentliche Auswirkungen auf Evolvability. Letztere muss deshalb als separates Ziel betrachtet werden.

Qualitätsattribute und Untermerkmale von Evolvability

Um Evolvability genauer zu beschreiben, werden wie für Wartbarkeit auch hier Untermerkmale als Qualitätsattribute zur Verfeinerung genutzt. Breivold et al. [3] diskutieren diesbezüglich Analysierbarkeit, Integrität der

Preprint Version: The original publication is available at www.springerlink.com
http://dx.doi.org/10.1007/s00287-009-0349-2

Architektur, Änderbarkeit, Portabilität, Erweiterbarkeit, Testbarkeit sowie domänenspezifische Attribute. Im Vergleich zur Wartbarkeit spielt bei der Evolvability also beispielsweise die Erweiterung der Software eine wichtige Rolle über Änderungen hinaus. Die Anpassungsfähigkeit an geänderte Umgebungen, die Portabilität, welche hier ebenfalls als Untermerkmal auftaucht, ist auch als Qualitätsattribut in der ISO 9126 [7] definiert. Wie bei Wartbarkeit auch erachten die Autoren das Qualitätsattribut Traceability als wichtig. Für die Weiterentwicklung sind dokumentierte Entscheidungen, die Verknüpfung verschiedener Artefakte und somit die Nachvollziehbarkeit der bisherigen Entwicklung unabdingbar. So kann z. B. die Entscheidung darüber, wie ein neues Feature einzubauen ist, leichter getroffen werden.
Die bisher genannten Qualitätsattribute beziehen sich vorrangig auf Artefakte wie Quellcode, Modelle und Beschreibungen. Auch das Vorgehen und die Entwicklungstätigkeiten müssen Qualitätsattribute aufweisen, die Evolvability in Hinsicht auf den Prozess unterstützen. Dazu gehören die Prozess-Reife (Planung, Analysierbarkeit, Selbstoptimierung), die Einflüsse auf Produktivität, die Merkmale bezüglich Regelungsgrad und Informationsweg-Länge, die Güte des Erfahrungsmanagements (z.B. Einarbeitungsaufwand und Grad der Fehlervermeidung) und weitere.

Entwurfsprinzipien mit Einfluss auf Evolvability

Die genannten Qualitätsattribute werden von den seit Langem untersuchten Entwurfsprinzipien des Software-Engineerings (positiv oder negativ) beeinflusst. Generell kann festgestellt werden, dass die Evolvability von allen Prinzipien unterstützt wird, die eine Beherrschung der Komplexität fördern, wie Abstraktion, Hierarchisierung, Modularität, Kapselung, konzeptionelle Integrität und Zerlegung nach Zuständigkeiten (Separation of Concerns). Eigenschaften wie Komplexität und Kopplung wirken sich negativ auf Evolvability aus, können jedoch nicht beeinflusst werden, wenn Sie immanent zur Aufgabenstellung gehören. Eine detaillierte Darstellung der Zusammenhänge und der Zuordnung von solchen Prinzipien zu Qualitätsattributen ist in [2] angegeben.
Die Qualitätsattribute mit Bezug zu Prozessaspekten werden von den Prinzipien für Entwicklungsprozesse und Entwurfstätigkeiten gefördert, die ein iteratives und risikovermeidendes Vorgehen unterstützen. Einige dieser Prinzipien werden beim Reengineering erfolgreich eingesetzt, wie beispielsweise die als Reengineering Patterns bezeichneten Prinzipien *„Grow Test Base Incrementally“*, *„Distinguish Public from Published Interface“*, *„Conserve Familiarity“* und *„Write Tests to Enable Evolution“* [4].

Analyse von Software bezüglich Evolvability

Für die Bewertung von Software stehen fragebasierte und messungsbasierte Analyseverfahren zur Verfügung. Die Verfahren beider Gruppen gehen von der Verfeinerung durch die Qualitätsattribute aus. Ein Softwaresystem ist weiterentwickelbar, wenn alle Qualitätsattribute ausreichend ausgeprägt sind – sowohl die Produkt- als auch die Prozessaspekte. Fragebasierte Verfahren eignen sich vorrangig für die Analyse komplexer Artefakte, wie von Architektur und Implementierung ganzer Systeme, oder für Qualitätsattribute mit unscharfer Beschreibung, wie innere und Prozesseigenschaften. Zu diesen Verfahren gehören die auf Review und Audit basierenden, wie die szenariobasierte Architekturbewertung bezüglich Modifizierbarkeit [1]. Geeignete Modifikationsszenarien und eine Beobachtung der Auswirkung auf Evolvability sind dabei für die zielgerichtete Analyse entscheidend. Die für agile Ansätze erarbeiteten Symptome von Qualitätsmängeln wie die von Fowler beschriebenen „Bad Smells“ [5] zielen sowohl auf Wartbarkeit als auch auf Evolvability; sie können bei Inspektionen von Architektur und Implementierung eingesetzt werden.
Messungsbasierte Verfahren lassen sich einsetzen, wenn formalisierte Artefakte oder messbare Größen aus Entwicklungsprozessen vorliegen. Metriken und mathematische Modelle werden als Grundlage solcher Verfahren verwendet. Sie müssen ebenfalls für die Qualitätsattribute bezüglich Artefakte und Prozess eingesetzt werden. In [2] werden beispielsweise Traceability-Beziehungen ausgewertet, um mittels Metriken zu Feature Tangling und Feature Scattering die Qualitätsattribute Verständlichkeit und Wiederverwendbarkeit zu bewerten. Dazu werden die Entwurfsprinzipien Zerlegung nach Zuständigkeiten, Modularität und Komplexität zugrunde gelegt.

Erhalten und Verbessern von Evolvability

Mit dem Ziel der Verbesserung von Evolvability werden Methoden, Techniken und Ansätzen des Software-Engineerings erarbeitet, indem von den Qualitätsattributen und den unterstützenden Prinzipien ausgegangen wird. Eine Reihe von Ergebnissen hat im praktischen Einsatz – beispielsweise für Produktlinien der Medizintechnik [10] – ihre Wirksamkeit bewiesen und kann zum Stand der Technik gerechnet werden, andere sind noch Gegenstand der Forschung, wie Methoden der zielorientierten Architekturentwicklung [2]. Um den Herausforderungen und der Komplexität der Aufgaben gerecht zu werden, müssen Artefakt- und Prozessaspekte gleichermaßen berücksichtigt werden. Deshalb sind ganzheitliche oder integrierende Ansätze erforderlich. Sie müssen gewährleis-

Preprint Version: The original publication is available at www.springerlink.com
http://dx.doi.org/10.1007/s00287-009-0349-2

ten, dass die Software in einem Zustand gehalten werden kann, indem noch strukturelle Änderungen möglich sind und nicht nur (korrigierende) Wartungstätigkeiten.

Beeinflussung von Artefakten: Softwarearchitektur, Modelle, Code

Zu den Mitteln und Ansätzen des Software-Engineerings mit positivem Einfluss auf Evolvability gehören Architekturmittel wie Entwurfsmuster für Variabilität, Komponentenmodelle, Plugin-Schnittstellen. Die Vorbereitung von Änderungen, z. B. mittels generativer Techniken, wirkt sich positiv aus. Ausdrucksmittel mit hohem Abstraktionsgrad wie Domain Specific Languages und beim Model-Driven Development ermöglichen Änderungen mit geringem Aufwand. Implementierungstechniken für Variabilität, die im Zusammenhang mit Domain Engineering und Software-Produktlinien [10] entwickelt wurden, verringern den Änderungsbedarf, indem Varianten vorbereitet werden. Hilfreich sind ebenso Modelle, die Informationen bereitstellen und so Architekturerosion verringern, wie durch Traceability und explizit dargestellte Abhängigkeiten. Entwurfsmethoden, die auf qualitative Eigenschaften in Architekturentwicklung und Entwurf abzielen, unterstützen ebenfalls die oben genannten Qualitätsattribute und beeinflussen so die Evolvability.

Beeinflussung der Entwicklungsprozesse

Alle Maßnahmen zu Software-Prozessen, die eine evolutionäre Entwicklung fördern, wirken sich auf Evolvability positiv aus. Beispielsweise werden in Produktlinien-Prozessen Koordinationsmittel zwischen Vorfertigung und aktueller Entwicklung eingesetzt, um eine langfristige Aufrechterhaltung der Wartbarkeit zu erreichen, wie das Domain Engineering und die Ansätze zu Design for Reuse. Reengineering und Refactoring zielen auf eine Verbesserung der Wartbarkeit, mit Auswirkungen auf Evolvability. Die Einführung von Traceability für Entwurfsentscheidungen vereinfacht langfristige Weiterentwicklung. Agile Prozesse fordern die Einfachheit von Lösungen. Damit vermeiden sie überflüssige Komplexität, was sich positiv auf die Evolvability der betreffenden Software auswirkt. Modellbasierte Ansätze, die Anforderungen, Architektur, Daten und Implementierung verbinden und teilweise Laufzeitevolution unterstützen, fördern Evolvability durch verringerte Ausbreitung von Änderungen. Die konsequente Beachtung des Ziels Evolvability führt zur Weiterentwicklung von Prozessen durch Verknüpfung von Analysen mit entsprechenden Maßnahmen, wie beispielsweise der Evolvability-orientierte Entwicklungsprozess in [2].

Vision: Langfristig orientierte Software- und Systementwicklung

Für alle langfristig wichtigen Softwaresysteme ist Evolvability ein zentrales Ziel, weil es den Wert der Systeme erhalten hilft und Architekturerosion verhindert. Der gegenwärtige Stand der Technik erlaubt eine direkte praktische Nutzung vor allem (Software-)technologischer Lösungen, deren Reife bereits weit fortgeschritten ist. Die größten Herausforderungen bestehen in den organisatorischen, mit menschlichen Arbeitsweisen zusammenhängenden Aufgaben. Dazu gehört die Motivation und Führung sowie die Förderung der Kommunikation. Damit lassen sich die Risiken und Konfliktbereiche beeinflussen, beispielsweise der Wissenserhalt und Transfer im Entwicklerteam oder der Erhalt der konzeptionellen Integrität bei Änderungen und bei Bearbeiterwechsel. Insbesondere die Abwägung und Koordination kurzfristiger Änderungswünsche mit langfristigen Zielen ist eine Aufgabe, die vor allem im Zuge des Managements gelöst werden muss.

Literatur

1. Bengtsson, P., Lassing, N., Bosch, J., van Vliet, H.: Architecture-level modifiability analysis (ALMA). J. Syst. Softw. 69(1-2): 129-147 (2004)
2. Brcina, R., Bode, S., Riebisch, M.: Optimization Process for Maintaining Evolvability during Software Evolution. In: Proc. 16th Int. Conf. on the Engineering of Computer Based Systems (ECBS), San Francisco, CA, USA, April 13-16, S. 196-205, IEEE 2009
3. Breivold, H. P., Crnkovic, I., Eriksson, P.: Evaluating Software Evolvability. In: Proc. 7th Conf. on Software Engineering Research and Practice in Sweden (SERPS'07), S. 96-103, ACM 2007
4. Demeyer, S., Ducasse, S., Nierstrasz, O.: Object-Oriented Reengineering Patterns. Elsevier 2003
5. Fowler, M.: Refactoring: Improving the Design of Existing Code. Addison Wesley 1999
6. IEEE Std. 1219-1998, IEEE Standard for Software Maintenance, IEEE 1998
7. ISO 9126-1:2001 Software engineering – Product quality – Part 1: Quality model. ISO 2001
8. Lehman, M. M.: Programs, life cycles, and laws of software evolution. In: Proc. IEEE 68(9), 1060-1076. IEEE 1980
9. Mens, T., Mens, K.: Assessing the Evolvability of Software Architectures. ECOOP Workshops 1998: 54-55

10. Pohl, K., Böckle, G., van der Linden, F. J.: Software Product Line Engineering - Foundations, Principles and Techniques. Springer, 2005.
11. Rajlich, V., Bennett, K. H.: A Staged Model for the Software Life Cycle. IEEE Computer 33(7): 66-71, IEEE 2000
12. Rowe, D., Leaney, J., Lowe, D.: Defining systems architecture evolvability - a taxonomy of change. In: Proc. ECBS'98, S. 45-52, IEEE 1998