Design for Future (DFF 2011) – Workshop Report

Christof Momm[1], Stefan Sauer[2], Mircea Trifu[3]

[1] SAP Research, Karlsruhe, Germany
christof.momm@sap.com
[2] Universität Paderborn, s-lab – Software Quality Lab, Paderborn, Germany
sauer@s-lab.upb.de
[3] Forschungszentrum Informatik (FZI), Karlsruhe, Germany
GI-Arbeitskreis „Langlebige Softwaresysteme“ (AK L2S2)
der Fachgruppen „Software-Architektur“ und „Software-Reengineering“
(http://akl2s2.ipd.kit.edu/)
mtrifu@fzi.de

Abstract: Software altert und erodiert, obwohl sie ein immaterielles Gut ist. Denn sowohl die Anforderungen an die Software als auch ihre Umgebung ändern sich mit der Zeit. Bei Informationssystemen ist dieses Phänomen als „Legacy“ wohlbekannt. Das Problem existiert aber auch bei eingebetteten Systemen, wo komplexe Software in langlebigen technischen Geräten eingesetzt wird. Die ökonomischen Auswirkungen der Software-Alterung sind beträchtlich. Wissenschaft und Industrie sind deshalb aufgefordert, neue Methoden der Softwaretechnik für zukunftsfähige Software zu entwickeln und die erheblichen Investitionen in große Softwaresysteme zu schützen. Beim 3. Workshop „Design for Future“ des Arbeitskreises „Langlebige Softwaresysteme“ (L2S2) diskutierten Wissenschaftler und Praktiker Herausforderungen, Erfahrungen und Lösungen aus praktischer wie aus wissenschaftlicher Sicht.

1 Einleitung

Aktuelle Ansätze der Softwaretechnik wie modellbasierte Entwicklungsmethoden, Softwarearchitektur, Evolution und Lifecycle-Management, Qualitätsmanagement und Software-Reengineering können dazu beitragen, die Herauforderungen der Software- Alterung anzugehen und die Situation zu verbessern, wenn sie geeignet weiterentwickelt und angewandt werden. Für den 3. Workshop „Design for Future“, der im Rahmen der Konferenz „Software Engineering 2011“ stattfand, wurden vier Themenblöcke als inhaltliche Schwerpunkte im *Call for Papers* vorgegeben:

- Integrierte und modellgetriebene Entwicklungsmethoden für langlebige Software
- übergreifendes Application-Lifecycle-Management
- Anpassungsfähige und zukunftssichere Software-Architekturen
- Qualitätsmanagement

2 Ablauf des Workshops

Der Workshop „Design for Future 2011“ gliederte sich in zwei wesentliche Teile. Im ersten Teil wurden aktuelle Fragestellungen und Lösungen aus Wissenschaft und Praxis in Vorträgen präsentiert und ausgiebig diskutiert. Eröffnet wurde der Workshop mit einem eingeladenen Vortrag. Im Anschluss präsentierten die Autoren der akzeptierten Workshopbeiträge ihre Arbeiten. Im zweiten Teil wurden Themen, Ziele und Fragestellungen einer Forschungsagenda für langlebige Softwaresysteme aus drei thematischen Perspektiven – modellgetriebene Entwicklungsmethoden, Application- Lifecycle-Management, Software-Architektur- und Qualitätsmanagement – in parallelen Kleingruppen diskutiert, die abschließend ihre Ergebnisse im Plenum vorstellten.

3 Vorträge und Diskussionen

In seinem eingeladenen Vortrag bediente sich Dr. Axel Uhl, *Chief Development Architect* der SAP AG, der Analogie des alternden Kartoffelsalates um in sehr anschaulicher und unterhaltsamer Weise die Probleme der Software-Alterung und Mittel dagegen herauszustellen. Basierend auf seinen Erfahrungen mit großen betrieblichen Informationssystemen identifizierte er drei wesentliche Gründe für die Alterung von Software: (1) sich ändernde Anforderungen an die Software seitens der Kunden bzw. des Marktes, (2) die Weiterentwicklung/Ablösung der verwendeten Plattformen sowie (3) die unbedachte Wiederverwendung bestehender Komponenten bei der Weiterentwicklung des Systems.

Als wesentliches Mittel gegen solche schädlichen Einflüsse sieht er die Schaffung zeitloser Designs auf Grundlage guter Abstraktionen für die entsprechende Domäne. Geeignete Modelle haben seiner Ansicht nach wesentlich dazu beitragen, solche Designs über Generationen sich ändernder Plattformen und Anforderungen hinweg zu transportieren. Darüber hinaus bedarf es natürlich einer kontinuierlichen Pflege des Systems, welche durch wohldefinierte *Governance*-Prozesse gesteuert wird. Dies hängt jedoch stark von der Wertschätzung des durch die Software gelieferten „Inhalts“ zusammen. Bricht die *Community* weg, ist die Alterung und letztendlich Ausmusterung der Software vorprogrammiert.

Im Anschluss an den eingeladenen Vortrag ging Harry Sneed in seinem Vortrag der Frage nach, welche Eigenschaften von Software-Services den Evolutionsprozess beeinflussen oder beeinträchtigen. Als wesentliche Hindernisse identifizierte er (1) Mehrsprachigkeit der Software, (2) Schnittstellenkomplexität und (3) Lieferantenabhängigkeit – und stellte ihnen geeignete Maßnahmen gegenüber: mehrsprachige Werkzeuge und Mitarbeiter, strenge Konventionen, saubere Architekturen, harte, aber faire Service Level Agreements sowie redundante Services.

In der zweiten Vortragsrunde stellte Heiko Koziolek ein Regelwerk für die nachhaltige Entwicklung langlebiger softwareintensiver Systeme in der industriellen Automatisierung auf Basis wiederkehrender Evolutionsszenarien vor. Das Regelwerk ist das Ergebnis einer umfassenden Literaturrecherche und soll nun im praktischen Einsatz evaluiert werden. Ziel des von Fabian Christ vorgestellten Ansatzes zur Kompatibilitätsanalyse frame-

workbasierter Anwendungen ist es, die Kompatibilität einer Anwendung zu einer neuen Frameworkversion ohne eine „Migration auf Probe“ zu bewerten. Stattdessen wird ein automatisierter, dreistufiger Prozess auf Grundlage eines Metamodells für Framework-Beschreibungssprachen vorgeschlagen. Der Prozess aus Benutzungsanalyse, Differenzanalyse und Kompatibilitätsanalyse liefert Hinweise auf Kompatibilitätsverletzungen. Die Kompatibilität von OSGi Service-Schnittstellen ist Gegenstand des Ansatzes, den Marco Müller vorstellte. Mit Hilfe sequentieller Kontrakte, die als Zustrandsautomaten formalisiert sind, soll die korrekte Interaktion von Komponenten beschrieben, analysiert und durchgesetzt werden.

Die Nutzung eines graphbasierten Adaptivitäts-Frameworks im Kontext der Software-Evolution stellte Mahdi Derakhshanmanesh in der dritten Vortragsrunde vor. Neben der selbst-adaptiven Anpassung an kleinere vorhergesehene änderungen (Micro Adaptation) wird so auch die Anpassung der Software an größere, unvorhergesehene änderungen im Rahmen von Wartungsaktivitäten (Macro Adaptation) unterstützt. Das Framework nutzt ein Laufzeitmodell der anpassungsfähigen Software und eine Anpassungs-Middleware, um die Software zur Laufzeit mittels Zustandsvariablen und Anpassungsregeln zu überwachen und anzupassen. Macro Adaptation besteht dann in der Veränderung der Regelmenge. Die Kombination von Clustering- und Pattern-basierten Reverse-Engineering-Ansätzen empfahl Markus von Detten in seinem Vortrag. Mit Hilfe des Clustering soll die Systemarchitektur zurück gewonnen werden, sogenannte „Bad Smells“ können aber die Clustering-Ergebnisse verfälschen. Deshalb sollen „Bad Smells“ zuvor mittels Mustererkennung entdeckt und entfernt werden, um so die Qualität der Architektur zu verbessern.

Im Anschluss an die Präsentation der akzeptierten Workshop-Papiere stellte Prof. Dr. Ursula Goltz das beantragte Schwerpunktprogramms „Design for Future – Managed Software Evolution“ vor.

4 Ergebnisse der Arbeitsgruppen

Ziel der abschließenden Diskussionsgruppen war es, wesentliche Forschungsfragen im Kontext der Entwicklung, Evolution und des Betriebs langlebiger Softwaresysteme zu identifizieren und herauszustellen. Damit sollte der Grundstein für eine Forschungsagenda gelegt werden, die im Rahmen weiterer Arbeitstreffen sukzessive weiter ausgearbeitet werden soll.

Die Einteilung in Diskussionsgruppen erfolgte auf Basis der inhaltlichen Schwerpunktbildung, die sich an den im *Call for Papers* aufgeführten Themenblöcken orientierte: modellgetriebene Entwicklungsmethoden (MDD), Application-Lifecycle-Management sowie Software-Architektur- und Qualitätsmanagement.

Um die Zusammenführung der jeweils erarbeiteten Einzelergebnisse zu erleichtern, wurde das folgende Schema – angelehnt an den „Goal-Question-Metric“-Ansatz – für die Erarbeitung und Erfassung der Forschungsfragestellungen vorgeschlagen.

1. **Ziele** aufstellen: Herausstellen der wesentlichen Ziele für den jeweiligen inhaltli-

chen Schwerpunkt.

2. **Fragestellungen** ableiten: Für jedes aufgeführte Ziel werden wesentliche Forschungsfragen identifiziert, die zur Erreichung des Ziels beantwortet werden müssen.
3. **Bestehende Technologien/Forschungsansätze** erfassen: Für jede Fragestellung werden (soweit bekannt) bereits bestehende Technologien oder Forschungsansätze aufgeführt, die potenziell zur Beantwortung der jeweiligen Forschungsfragestellung beitragen.

Die folgenden Abschnitte fassen die Ergebnisse der drei Diskussionsgruppen zusammen. Aus Platzgründen beschränken wir uns in diesem Bericht auf eine Auswahl der wichtigsten Ziele und Fragestellungen.

4.1 Modellgetriebene Entwicklungsmethoden (MDD)

Die Diskussionsgruppe zum Thema „Modellgetriebene Entwicklungsmethoden“ vereinte praktische und wissenschaftliche Perspektiven. Das erste zentrale Ziel war die Identifikation der richtigen Modelle für langlebige Softwaresysteme. Als typische Fragestellungen in diesem Zusammenhang identifizierte die Gruppe:

- Auf welcher Granularität modellieren wir?
- Für welchen Zweck werden die Modelle erstellt?
- Welche Aspekte werden modelliert?
- Wie kann die Angemessenheit der Modellierung evaluiert werden?
- Was ist die geeignete Repräsentation eines Modells?

Lösungsansätze können Domänenspezifische Sprachen (DSL) und die systematische Entwicklung von Modellierungssprachen (Language Engineering) liefern. Ein weiteres Ziel, das betrachtet wurde, ist der übergreifende Einsatz modellbasierter Methoden für Entwicklung und Betrieb. Hierbei sind unter anderem die Fragen zu beantworten:

- Welche Modelle braucht man für den Betrieb?
- Welche Entwicklungsmodelle können im Betrieb genutzt werden?

Techniken, die für eine Annäherung an die Lösung genutzt werden können, sind beispielsweise Design-by-Contract-Modelle, Performance-Modelle, Adaptionsmodelle. Die Evolutionsfähigkeit von Metamodellen und Modeltransformationen stellt ein weiteres wichtiges Ziel für die Langlebigkeit modellgetriebener Entwicklungsmethoden dar. Typische Fragestellungen sind:

- Wie kann die Gültigkeit von Modellen nach der Evolution geprüft werden?

- Wie erhält man die Traceability von Transformationen zu Metamodell-Versionen?
- Wie langlebig sind Modellierungssprachen?
- Wie kann ich gewährleisten, dass der Technologie-Stack noch funktioniert?

Benötigt werden hierzu geeignete Richtlinien im Umgang mit MDD-Techniken. Weitere Ziele, die diskutiert und mit konkreten Fragestellungen unterlegt wurden, waren Variabilitätsmanagement, insbesondere auf Architekturebene, Roundtrip-Engineering und die Co-Evolution von Modellen (Anforderungen, Architektur) und Implementierung sowie das Management, die Evolution und die Versionierung von Modellen. Auch zu diesen Fragen konnte die Gruppe erste potenzielle Lösungsansätze identifizieren.

4.2 Application-Lifecycle-Management

Die Diskussionsgruppe zum Thema „Application-Lifecycle-Management“ (ALM) hatte eine starke Industriebeteiligung aus unterschiedlichen Domänen: Prozessleitsysteme, Automotive (Embedded) Systeme sowie betriebliche Informationssysteme. Der erste Schritt bestand daher in der Klärung des Aufgabenbereichs und der Herausstellung der wesentlichen Herausforderungen. Unabhängig von der Domäne wurden Installation, Upgrade, Update und ein übergreifendes Versionsmanagement als Kernaufgaben des ALM identifiziert. Bei weiterführenden ALM-Aufgaben, wie z.B. der überwachung und Analyse von Laufzeitinformationen oder der Steuerung bzw. den Selbstheilungsfähigkeiten des Systems, wichen dagegen die Anforderungen stark voneinander ab. Daher wurde in Hinblick auf die zu erarbeiteten Ziele und Fragstellungen der Fokus auf die Kernaktivitäten gelegt.

Das wichtigste Ziel im Rahmen der Kernaktivitäten ist nach Einschätzung aller Beteiligten dabei die Entkopplung de Lebenszyklen verschiedener, in unterschiedlichen Geschwindigkeiten co-evolvierender Systemteile, Schichten oder Bauteile. Daneben werden die weitergehende Automatisierung, Vorhersagbarkeit der Aufgaben bzgl. Dauer/Kosten, die Unterbrechungsfreiheit des Betriebs, die änderungstoleranz zur Laufzeit sowie die kontinuierliche Verbesserung als wichtige Ziele gesehen. Für jedes Ziel konnte bereits eine Reihe von Fragestellungen herausgearbeitet werden. So sind für eine Entkopplung der Lebenszyklen unter anderem die folgenden Forschungsfragen zu beantworten:

- Wie kann man den Impact von änderungen automatisiert ermitteln?
- Wie sollte ein Erweiterungskonzept für ALM-Funktionen aussehen?

Helfen können bei der ersten Fragestellung z.B. bestehende Techniken der Code- Analyse aus dem Bereich des Reverse-Engineering, aber auch Forschungsansätze zur Nachverfolgbarkeit in Modellen (Model Traceability).

4.3 Software-Architektur- und Qualitätsmanagement

Die Diskussionsgruppe „Software-Architektur- und Qualitätsmanagement“ hatte eine maßgebliche Beteiligung von Akademikern. Die Beteiligten setzten die Evolution von Software-Architekturen als zentrales Ziel in den Mittelpunkt. Dabei wurden zwei Unteraspekte besonders herausgestellt: das Management und die Qualitätssicherung bei der Evolution von Software-Architekturen.

Die aus diesen Zielen herausgearbeiteten wichtigsten Forschungsfragen können in drei Gruppen gebündelt werden:

- Inwieweit ist Langlebigkeit planbar/vorhersehbar und welche typischen Evolutionsszenerien gibt es?
- Wie erreicht man Langlebigkeit bei alten Systemen?
- Wie misst man Langlebigkeit? Was sind typischen Qualitätskriterien/ Qualitätsmetriken? Wie sieht der Trade-off zwischen Langlebigkeit und anderen Software-Eigenschaften (Kosten, Performance) aus?

Helfen können bei der ersten Fragengruppe bestehende Analyseansätze wie Change-Impact- und Traceability-Analysen, aber auch eine stärkere Wiederverwendung des Erfahrungswissens in Form von Produktlinien, Referenzarchitekturen und Architekturmustern. Bei der zweiten Gruppe wurden Reengineering- und Migrationsansätze im Vordergrund gesehen, und bei der letzten Gruppe ist eine Mischung von manuellen Architektur-Reviews und automatisierten Mess- und Verifikationsverfahren geeignet, die verschiedene Qualitätsstandards, Randbedingungen und Szenarien mit Hilfe von Metriken und Heuristiken überprüfen sollen.

5 Zusammenfassung und Ausblick

Der diesjährige Workshop DFF 2011 zählte mit qualitativ hochwertigen Beiträgen, gehaltvollen Diskussionen, durchgängig über 30 Teilnehmer/innen und der allgemein positiven Resonanz zu den erfolgreichsten Workshops der Konferenz „Software Engineering (SE) 2011“. Wir bedanken uns daher nochmals bei allen Autoren/innen der Workshop-Beiträge, den Teilnehmer/innen sowie den Vortragenden sehr herzlich für die spannenden Beiträge und Diskussionen und bei den Mitgliedern des Programmkomitees für die Begutachtung der eingereichten Beiträge, ohne die ein solcher Erfolg nicht möglich gewesen wäre. Besonderer Dank gilt aber auch dem Team der SE 2011 für die hervorragende Organisation der Konferenz sowie den studentischen Hilfskräften, die uns bei der Durchführung des Workshops tatkräftig zur Seite standen.

Die begonnene Arbeit an der Forschungsagenda soll nun bei folgenden Arbeitskreistreffen zu ausgewählten Themen fortgesetzt und weiter detailliert werden. Darüber hinaus ist geplant, die Workshop-Reihe „Design for Future“ im kommenden Jahr fortzusetzen, dann wieder gemeinsam mit dem „Workshop Software- Reengineering“ der gleichnamigen GI-Fachgruppe.