CARL VON OSSIETZKY universität OLDENBURG

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Abteilung Software Engineering

Modellbasierte erreichbarkeitsoptimierte Rekonfiguration komponentenbasierter Softwaresysteme zur Laufzeit

Dissertation

zur Erlangung des Grades eines
Doktors der Ingenieurwissenschaften

(Dr.-Ing.)

von

Dipl.-Ing. Jasminka Matevska

Gutachter:

Prof. Dr. Wilhelm Hasselbring (Universität Kiel)
Prof. Dr. Ralf Reussner (Universität Karlsruhe)

Tag der Disputation: 8. Juli 2009

Kurzfassung

Operative Softwaresysteme unterliegen ständigen Veränderungen im Laufe ihres Lebenszyklusses. Sie müssen kontinuierlich den veränderten Anforderungen angepasst werden, um deren Funktionalität bzw. deren Dienste zu erweitern oder zu optimieren. Weiterhin sind Veränderungen unumgänglich, um die Qualitätseigenschaften der Systeme zu verbessern. Schließlich ist es oft notwendig Fehler zu beseitigen. Der Prozess der Durchführung der notwendigen Veränderungen (Rekonfiguration) führt im Regelfall zum vorübergehenden Ausfall der Systeme. Insbesondere bei geschäftskritischen Web-basierten Anwendungen kann die fehlende Verfügbarkeit der Dienste zu finanziellen Verlusten führen. Das Hauptziel einer Rekonfiguration zur Laufzeit ist die Sicherung bzw. Erhöhung der Verfügbarkeit der Systemdienste. Weiterhin spielt eine transparente Durchführung einer Rekonfiguration eine entscheidende Rolle für die Zufriedenheit der Benutzer eines Systems. Eine Rekonfiguration kann als transparent betrachtet werden, wenn während deren Durchführung das System innerhalb der vertraglich festgelegten Reaktionszeiten korrekt antwortet.

Diese Arbeit, erschienen als Buch im Vieweg+Teubner Verlag mit dem Titel „Rekonfiguration komponentenbasierter Softwaresysteme zur Laufzeit“ unter der ISBN 978-3-8348-1001-4, liefert einen neuen modell- bzw. architekturbasierten Ansatz zur Planung und transaktionalen Durchführung einer Rekonfiguration komponentenbasierter Softwaresysteme zur Laufzeit unter voller Verfügbarkeit und möglichst geringer Beeinflussung der Reaktionsfähigkeit der Systemdienste. Hauptstrategie dabei ist die Verschiebung des Rekonfigurationszeitpunkts bis zu einem optimalen, analytisch bestimmten Zeitpunkt, zu dem die Beeinträchtigung (Störung) des Systems als minimal erwartet wird. Der wissenschaftliche Beitrag besteht aus drei Teilbeiträgen:

1. **Anwendungsmodell**, das ein Architektur-Sichtenmodell als Grundlage definiert und eine Beschreibung der statischen und dynamischen Sicht einer System-Architektur beinhaltet. Ein sog. Component-Connector-Container (C3) Meta-Modell definiert dabei die strukturelle Zuordnung zwischen den beiden Sichten.

2. **Optimierungs- und Analysemodell**, das die Grundlage für eine auftragsbezogene Optimierung der Erreichbarkeit bzw. Reaktionsfähigkeit der System-

dienste während der Laufzeit-Rekonfiguration darstellt. Die Optimierung wird als Knapsack-Minimierungsproblem aufgefasst und durch Analyse der Laufzeitabhängigkeiten zwischen Instanzen von Komponenten und Berücksichtigung zusätzlicher Faktoren, wie das Benutzungsmodell des Systems, die Dauer und die Dringlichkeit der Rekonfiguration, durchgeführt.

3. **Rekonfigurationsmodell**, das durch Lebenszyklus- und Redeployment-Protokolle ein detailliertes Konzept für die Durchführung der Rekonfiguration als sog. Redeployment-Transaktion zur Laufzeit definiert und somit eine Erhaltung der Konsistenz des Systems und die volle Verfügbarkeit der Systemdienste während der Rekonfiguration gewährleistet.

Für die System-Architekturbeschreibung werden UML 2 Komponenten-, Zustands- und Sequenzdiagramme, zur Komponenten-Verhaltensspezifikation Protokollautomaten eingesetzt. Das Benutzerverhalten wird durch ein Benutzungsmodell dargestellt, wobei Ausführungssequenzen als Markov-Ketten und mit Mengen der beteiligten Komponenten modelliert werden. Diese Informationen werden in ein Knapsack-Minimierungsproblem kombiniert, um eine Analyse der Laufzeitabhängigkeiten durchführen zu können. Zur Berechnung der Laufzeitabhängigkeiten werden graphentheoretische Konzepte eingesetzt. Schließlich werden Lebenszyklusprotokolle auf System- und Komponentenebene für die transaktionale Durchführung der Laufzeit-Rekonfiguration definiert.

Es fand eine zweigeteilte empirische Evaluation statt. Zum einen wurde das transaktionale Redeployment für die Java EE Plattform realisiert und evaluiert. Zum anderen wurde die Optimierung der Erreichbarkeit der Systeme während der Rekonfiguration von Web-basierten Java-Anwendungen evaluiert. Dabei wurde ein probabilistisches Benutzungsverhalten simuliert und Monitoringdaten aufgezeichnet. Diese Monitoringdaten wurden anschließend zur Analyse der Laufzeitabhängigkeiten eingesetzt, um geeignete Nutzungsszenarien und Zeitpunkte für die Durchführung einer auftragsbezogenen Laufzeit-Rekonfiguration zu bestimmen und zur Laufzeit wieder zu erkennen.

System-Architekturbeschreibung – Das Anwendungsmodell

Das Anwendungsmodell ist nach ausführlicher Betrachtung, Analyse und Vergleich verschiedener Techniken, Formalismen und Notationen zur Software-Architekturbeschreibung entstanden. Dabei fiel die Entscheidung auf einen semi-formalen Ansatz, der UML 2 mit formalen Spezifikationstechniken, wie Automaten-

und Graphentheorie, erweitert. Diese Entscheidung wurde durch die praktische Ausrichtung der Arbeit begründet. Das Ziel dabei war, ein System nur soviel wie nötig bzw. so wenig wie möglich zu beschreiben und dabei eine Möglichkeit zu einer detaillierten Spezifikation zuzulassen. Dieses, bedingt durch die Tatsache, dass ein operativ eingesetztes Anwendungssystem zu dem Zeitpunkt der angeforderten Rekonfiguration in der Regel kaum eine formale Beschreibung dessen Architektur besitzt. Eine vollständige und formale Spezifikation eines Systems während der Wartung bzw. vor einer Rekonfiguration ist praktisch nicht durchführbar. Sie würde, wenn überhaupt möglich, verhältnismäßig viel Zeit in Anspruch nehmen und somit eine enorme Verzögerung der Reaktionszeit bei einer Rekonfiguration erzeugen. Eine rudimentäre UML-Dokumentation ist dagegen eher in der realen Welt der Software-Anwendungen zu erwarten. Diese, erweitert durch Monitoringdaten, die entweder bereits vorhanden sind, oder mit entsprechenden Tools relativ schnell gesammelt werden können, stellen schon eine gute Grundlage für weiterführende Analysen dar, insbesondere mit Einsatz der Graphentheorie. Sollte eine Erzeugung von System- bzw. Komponenten-Zustandsautomaten möglich sein, sind sogar weitere Analysen zur Wiedererkennung und Vorhersage von bestimmten Laufzeitzuständen möglich.

Optimierung der Erreichbarkeit – Das Optimierungsmodell

Das Problem der Optimierung der Erreichbarkeit der Dienste während einer Rekonfiguration zur Laufzeit wurde als ein $\mathbb{NP}$-vollständiges Minimierungsproblem behandelt. Es wurde als ein ***Knapsack-Problem*** beschrieben. Bei dem Optimierungsproblem stellt das Grenzgewicht das maximale Gesamtgewicht aller Dienste im System dar und es wird ein minimales Gewicht, der von der Rekonfiguration betroffenen Dienste, gesucht. Idealerweise ist das minimale Gewicht 0 bzw. es sind keine Dienste betroffen. Eine Rekonfiguration während des Laufzeitzustands, der dieses minimale Gewicht aufweist, ist als störungsfrei zu betrachten und gewährleistet somit eine maximale Erreichbarkeit der Dienste. Da sowohl die Anzahl als auch das Gesamtgewicht, der von der Rekonfiguration betroffenen Dienste, von mehreren Parametern abhängig sind und diese zum Teil widersprüchliche Anforderungen beinhalten, ist es nicht möglich für jeden Rekonfigurationsauftrag ein minimales Gewicht von 0 zu garantieren. Deshalb war es notwendig, eine ***Multikriterien-Optimierung*** durchzuführen, um das möglichst geringe Gewicht zu bestimmen. Dabei wurden folgende Parameter berücksichtigt: (1) der Rekonfigurationsauftrag, (2) das Benutzungsprofil des Systems, (3) das interne Laufzeitverhal-

ten des Systems, (3) die Benutzungsintensität des Systems, (4) die Dauer und (5) die Dringlichkeit der Rekonfiguration. Dieser Parameter wurden einzeln in dem Analysemodell betrachtet und eine Art approximativer Algorithmus für die Lösung des Problems vorgestellt. Als größte Abweichung vom Optimum und somit maximale Störung im System kann dabei eine Rekonfiguration zum Laufzeitzustand, bei dem das Grenzgewicht gilt, verstanden werden.

Bei dieser Optimierungsmethodik entfällt die Einschränkung bezüglich der Natur der Komponenten im System. Sie ist prinzipiell für sämtliche Systeme, die aus über Schnittstellen kommunizierenden Komponenten bestehen, geeignet.

Analyse zur Optimierung der Erreichbarkeit – Das Analysemodell

Das Analysemodell beinhaltet die Beschreibungen und Analysemöglichkeiten der berücksichtigten Parameter.

- **Rekonfigurationsauftrag** Bei der Analyse des Rekonfigurationsauftrags wird Graphentheorie eingesetzt. Der Auftrag wird als Graph abgebildet. Um transitive Abhängigkeiten zu berücksichtigen, wird für jeden Auftrag die transitive Hülle gebildet. Somit wird das betroffene Teilsystem bestimmt.

- **Benutzungsmodell** Falls ein Benutzungsmodell bekannt bzw. aus Monitoringdaten abgeleitet werden kann, kann eine Analyse dessen zur Bestimmung relevanter Anwendungsfälle bzw. Ausführungssequenzen beitragen. Dadurch kann die Menge der zu analysierenden Systemzustände reduziert werden.

- **Internes Laufzeitverhalten** Eine Analyse des internen Laufzeitverhaltens des Systems wurde zur Bildung und Gewichtung von Laufzeit-Abhängigkeitsgraphen eingesetzt. Eine Bestimmung von minimalen Laufzeit-Abhängigkeitsgraphen und deren Zuordnung zu den System-Laufzeitzuständen ist dabei das entscheidende Ergebnis zur Optimierung der Erreichbarkeit.

- **Dauer und Dringlichkeit** Diese Faktoren, die möglicherweise vertraglich geregelt sind, können die Einhaltung der analytisch bestimmten Werte für eine Erreichbarkeit unter Umständen negativ beeinflussen.

Schließlich bietet das Analysemodell auch eine formale Vorgehensweise bei der Wiedererkennung bzw. Vorhersage des optimalen Zustandsraums als günstigen Startpunkt für die Rekonfiguration zur Laufzeit mit dem Einsatz der Dienst-

Effekt-Automaten. Auch an dieser Stelle soll der praktische Ansatz betont werden: Wie in der Evaluation gezeigt, ist eine Wiedererkennung dieser Zustände auch unter Monitoring mit unvollständigen Dienst-Effekt-Automaten möglich. Sollten sie jedoch zur Verfügung stehen, ist auch eine Vorhersage und somit eine Reduzierung der Monitoringzeiträume möglich.

Transaktionale Rekonfiguration zur Laufzeit – Das Rekonfigurationsmodell

Das Rekonfigurationsmodell definiert ein Konzept zur transaktionalen Durchführung der Laufzeit-Rekonfiguration. Dabei wurden Lebenszuklysprotokolle sowohl auf Komponenten- als auch auf Systemebene festgelegt, deren Einhaltung eine konsistente Durchführung der Rekonfiguration unter Erhaltung der vollen Grenzverfügbarkeit des Systems sichert. Die Zustände in den Protokollen betreffen Aktivität und Kommunikation der Komponenten. Ein Austausch ist nur nach Erreichen eines gesicherten Zustands (*blocked / ready to change*) möglich. Eine Blockierung kann allerdings nur aus fest definierten Zuständen statt finden (*free* bzw. *passive & not used* und *passive & used*). Das Protokoll berücksichtigt dabei ankommende Anfragen und deren Aufbewahrung in Warteschlangen während der Rekonfiguration bzw. deren Abarbeitung nach der Rekonfiguration und sichert somit die volle Grenzverfügbarkeit des Systems während der Rekonfiguration. Da die Rekonfiguration mehrere Komponenten betreffen kann, ist eine übergeordnete Zustandskontrolle auf Systemebene, wie im System-Laufzeitprotokoll definiert, notwendig.

Ein Komponentenaustausch wurde speziell durch ein Redeployment-Protokoll als sog. Änderungstransaktion definiert. Diese ist nicht mit einer Datenbanktransaktion gleich zu setzen, weil es sich dabei um eine Änderung der Anwendung und nicht der Daten, die sie bearbeitet, handelt. Das Redeployment-Protokoll wurde auch mittels CTL formalisiert und somit eine Grundlage zur Überprüfung, durch z.B. Model Checking, bereit gestellt. Schließlich bietet das Rekonfigurationsmodell eine logische Architektur eines *plattformunabhängigen Rekonfigurationsmanagers – PIRMA*, das die transaktionale Durchführung einer Rekonfiguration zur Laufzeit ermöglicht und die Analyse zur Optimierung der Erreichbarkeit integriert.

Evaluation

Die Evaluation, des in dieser Arbeit verfolgten Ansatzes, wurde im Rahmen mehrerer Diplomarbeiten und Individueller Projekte durchgeführt. Die Java EE-

basierte Evaluation enthält zum einen die Realisierung von PIRMA als Machbarkeitsprüfung des theoretischen Konzepts. Dabei wurden für den JBoss Anwendungsserver die theoretischen Teilkonzepte in konkrete Teilsysteme umgesetzt und in ein Gesamtsystem integriert. Das implementierte System konzentriert sich auf Rekonfiguration von Enterprise JavaBeans (EJB) zur Laufzeit und ist aus Client- Sicht als Eclipse Plug-In realisiert. Zum anderen wurde die Laufzeit-Rekonfiguration Java EE-basiert evaluiert. Dabei wurden, für die Java EE Technologie typische, Rekonfigurationsszenarien auf unveränderten JBoss und Bea WebLogic Anwendungsservern getestet und ausgewertet. Zusätzlich wurde das eigenimplementierte System PIRMA in zwei Laborexperimenten auf Funktionstüchtigkeit und in Hinblick auf Dauer der Redeployment-Vorgänge bzw. der Verzögerungen in den Antwortzeiten getestet. Die Evaluation der Optimierung der Erreichbarkeit konzentriert sich auf die Bestimmung und Wiedererkennung minimaler Laufzeit- Abhängigkeitsgraphen. Dazu wurde eine Web-basierte Java Anwendung als komponentenbasiertes System unter Last gesetzt und probabilistisches Benutzungsverhalten simuliert. Das System wurde beobachtet und die Monitoringdaten aufgezeichnet und analysiert.

Praktische Einsetzbarkeit

Die Evaluation der Konzepte zeigte, dass ein pragmatischer und erweiterbarer Ansatz, wie in dieser Arbeit verfolgt, durchaus praktisch einsetzbar ist. Die dreigeteilte Evaluation bestätigte die prinzipielle Machbarkeit des Ansatzes unter Einsatz von Java EE, eine praktisch und industriell genutzte Komponententechnologie. Die vollständige Umsetzung der Konzepte ist allerdings an Techologiegrenzen gestoßen. Probleme, z.B. bei dem Austausch von stateful Session Beans, Isolierung der Classloader oder Behandlung von verteilten Transaktionen, können zum jetzigen Stand der Java EE Technologie nur teilweise und mit sehr großem Zusatzaufwand gelöst werden.

Die Evaluation zum Einsatz des Redeployment-Systems zeigte, dass das im Rahmen der Evaluation implementierte Redeployment-System dazu in der Lage ist, ein fehlerfreies Redeployment von zustandslosen Session Beans während des Betriebs, transparent aus Sicht des Clients durchzuführen. Dabei werden weder Transaktionen abgebrochen, noch ankommende Anfragen verloren. Bei einem Redeployment war nur ein Teil der Anwendung betroffen, alle übrigen Komponenten konnten die Anfragen der Benutzer weiterhin ungestört beantworten. Somit erfüllt das System die Anforderungen einer transaktionalen Durchführung einer Rekonfiguration unter voller Verfügbarkeit. Die Messungen zeigten zusätzlich,

dass die Dauer der eigentlichen Redeployments im Millisekundenbereich lag und deren Auswirkung auf die Antwortzeiten im Regelfall im Rauschen des Anwendungsbetriebs untertauchten. Gut messbare Verzögerungen wurden im Fall eines Redeployment-Versuchs von zum gegebenen Zeitpunkt benutzten Komponenten festgestellt. Sie bewegten sich jedoch in einem aus Benutzersicht nicht spürbaren Bereich von ca. $800ms$.

Letzteres könnte die Frage nach der Notwendigkeit einer zusätzlichen Analyse zur Optimierung der Erreichbarkeit aufkommen lassen. Diese Frage ist leicht zu beantworten, wenn komplexe Berechnungen bzw. längere Transaktionen in Betracht gezogen werden. Bei einem Redeployment-Versuch zu solch einem Zeitpunkt, ist die Dauer der Verzögerung sogar nicht absehbar und hängt von der Dauer der laufenden Berechnungen bzw. Transaktionen ab. Eine zusätzliche Analyse zur Bestimmung günstigerer Zeitpunkte zum Starten eines Redeployments und Verschiebung des Startzeitpunkts eines Redeployments würde zwar die Durchführung der Rekonfiguration verzögern, allerdings die Störung im System reduzieren und somit die Erreichbarkeit dessen Dienste steigern.

Schließlich zeigte der letzte Teil der Evaluation, dass es innerhalb wenigen Sekunden möglich ist, Laufzeitzustände zu bestimmen und zur Laufzeit wieder zu erkennen, um eine störungsfreie Rekonfiguration durchzuführen. Da dieses allerdings bei Komponenten, die häufig benutzt werden, mit zunehmender Anzahl von parallel aktiven Benutzern immer schwieriger wird, ist eine zusätzliche Analyse und Evaluation von Laufzeitabhängigkeiten mit einer sehr hohen Anzahl von Benutzern (≥ 1000) notwendig, um eine repräsentative Aussage bezüglich der Bestimmung minimaler Abhängigkeitsgraphen machen zu können. Letzteres war unter Laborbedingungen nicht durchführbar.

Zusätzlich zeigte die Evaluation rudimentär, dass die Zeitintervalle mit minimalen Abhängigkeiten von den Anwendungsfällen, die die Benutzer durchführen, abhängig sind. Eine Durchführung der angeforderten Rekonfiguration während den als günstig erkannten Szenarien als Menge von aktiven Anwendungsfällen, würde die Erreichbarkeit der Systemdienste maximieren. Diese Ergebnisse zeigten einen möglichen Ansatz zur Erweiterung der Abhängigkeitsanalyse auf der Ebene der Anwendungslogik.

Zusammenfassend kann der Ansatz als praktisch einsetzbar bezeichnet werden.