LUDWIG MAXIMILIANS UNIVERSITÄT MÜNCHEN

FÜR

FAKULTÄT INFORMATIK

Lehrstuhl für Kommunikationssysteme und System-programmierung

Seminararbeit für das Hauptseminar: Neue Ansätze im IT-Servicemanagement (ITIL/eTOM)

Thema: Application Management

Vorgelegt von: Torsten Friedrich

Betreuer:	Thomas Buchholz
Verantwortlicher Hochschullehrer:	Prof. Dr. Heinz-Gerd Hegering
Tag der Abgabe:	16.02.2004

INHALTSVERZEICHNIS

1. Einleitung 5

2. IT-Systeme und Unternehmen 7

2.1 Das Strategie-Ausrichtungsmodell 7

2.2 *Key Business Driver* 8

2.3 *Application Portfolio* 9

3. Application Management 11

3.1 Einführende Betrachtungen 11

3.2 Die Phasen des Lebenszyklus 12

3.3 Anforderungsanalyse 14

3.4 Design 18

3.5 Realisierung 21

3.6 Einführung 25

3.7 Betrieb 30

3.8 Optimierung 34

3.9 Abschließende Betrachtungen 38

3.9.1 Vorteile, die sich aus dem *Application Management* ergeben 38

3.9.2 Probleme, die bei der Einführung auftreten können 39

3.9.3 Der *Change Management Prozess* 39

3.9.4 Testprozesse 41

3.9.5 Der *Release Management Prozess* 41

4. Überwachen von Anwendungen 43

4.1 Messung der Antwortzeit / *Application Response Measurement* 43

4.2 Überwachung des Serverbetriebes 45

4.3 Das Java-Management 46

4.3.1 Instrumentierung der *Java™ Virtual Machine* – JSP-174 46

4.3.3 Die *Java Management Extension* (JMX) 47

4.3.3 Das *Java Dynamic Management Kit* (JDMK) 49

4.4 Definition von *Logfiles* 50

5. Das BMW-Extranet-Szenario 52

5.1 Einführung 52

5.2 Der *Car-Konfigurator* 53

5.2.1 Definition 53

5.2.2 Systemdesign 53

5.2.3 Die Antwortzeit – ein QoS-Parameter 55

5.2.4 Messung der Antwortzeit 56

5.2.5 Betrieb des *Car-Konfigurators* 56

5.2.6 Optimierung / Weiterentwicklung des *Car-Konfigurators* 58

6. Zusammenfassung 60

7. Literaturverzeichnis 61

1. Einleitung

Application Management bietet eine *End-to-End-Beschreibung* für alle Managementprozesse, die im Laufe des Lebenszyklus einer Applikation anfallen. Das Spektrum reicht dabei von der Konzeption bis zur Ablösung. *Application Management* führt eine *Roadmap* ein, um neue oder existierende Anwendungen in nachhaltigen Weise in eine IT-Umgebung einzuführen. Es handelt sich dabei um eine Zusammenfassung von Aktivitäten, die die Phasen der Anwendungsentwicklung und die Aktivitäten des Servicemanagements betreffen. Durch die Integration beider Disziplinen entstehen Anwendungen, die einfach betrieben und gewartet werden können.

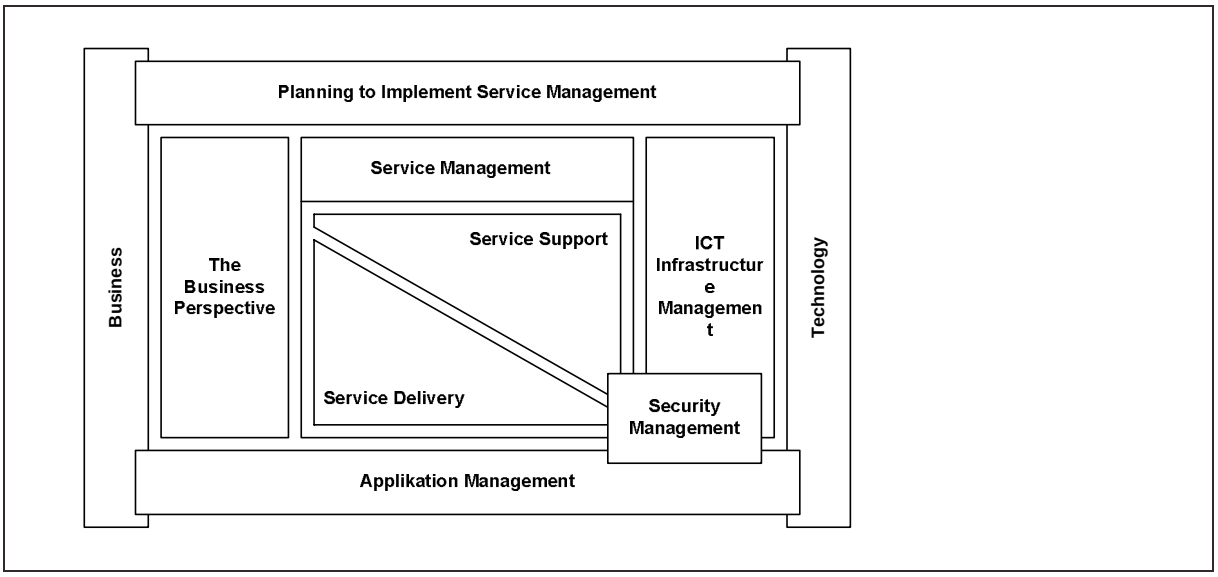

Bild 1.1: Das ITIL-Framework (vgl. [1])

Application Management ist gleichzeitig ein Titel der jetzt 7 Bände umfassenden *ITIL-Bibliothek*, die *Best-Practice-Strategien* für das IT-Service-Management anbietet. Mitarbeiter von IT-Serviceabteilungen sollen einen Einblick in die Arbeitsweise von Softwareentwicklern erhalten, wenn sie einen neuen Service definieren. Dies war einer der Hauptgründe für die Veröffentlichung dieses Bandes. Der Herausgeber dieser Schriftensammlung ist das *Office of Government Commerce* (OGC) mit seinem Sitz in England. Bild 1.1 ist diesem Buch entnommen und zeigt die Einordnung dieser ITIL-Publikationen zueinander auf.

Obwohl Anwendungsentwicklung, Servicemanagement und *Application Management* in [1] als verschiedene Prozesse dargestellt werden, sind sie in der Realität verflochten. Faktoren, die dies beeinflussen, reicht vom Markt, in dem eine Organisation operiert, bis zu Standards

und Richtlinien die innerhalb dieser Unternehmung gelten.

Ziel diese Seminararbeit ist es nicht, eine Abschrift bzw. Übersetzung von [1] anzufertigen. Vielmehr wird in einem ersten Teil versucht, wichtige theoretische Zusammenhänge aufzuzeigen. Der Schwerpunkt der Untersuchungen soll dabei auf dem *Application Management Lifecycle* liegen. Alle betriebswirtschaftlichen Begriffe werden nur erläutert, soweit sie für das Verständnis erforderlich sind.

Teil 2 wendet diese Überlegungen an praktischen Beispielen an. Der Aufgabenstellung entsprechend soll dies an dem *BMW-Extranet* erfolgen, das auch als Grundlage von [9] dient. Am *BMW-Car-Konfigrator* als eine Web-Service-Anwendung wird aufgezeigt, welche Einfluss die Spezifikation einer Anwendung auf das Design einer IT-Infrastuktur nimmt. Vorangestellt ist eine Beschreibung von Verfahren, die zur Überwachung von Anwendungen herangezogen werden können.

2. IT-Systeme und Unternehmen

2.1 Das Strategie-Ausrichtungsmodell

Die Ausrichtung von Unternehmen und IT-Systemen ist außerordentlich wichtig. Sonst besteht die Gefahr von:

- nutzlosen IT-Systemen auf Basis falsch verstandener Anforderungen,
- mangelhaft unterstützten Geschäftsanforderungen und
- nicht genehmigte „technische“ Projekten für die Verbesserung der IT-Infrastruktur.

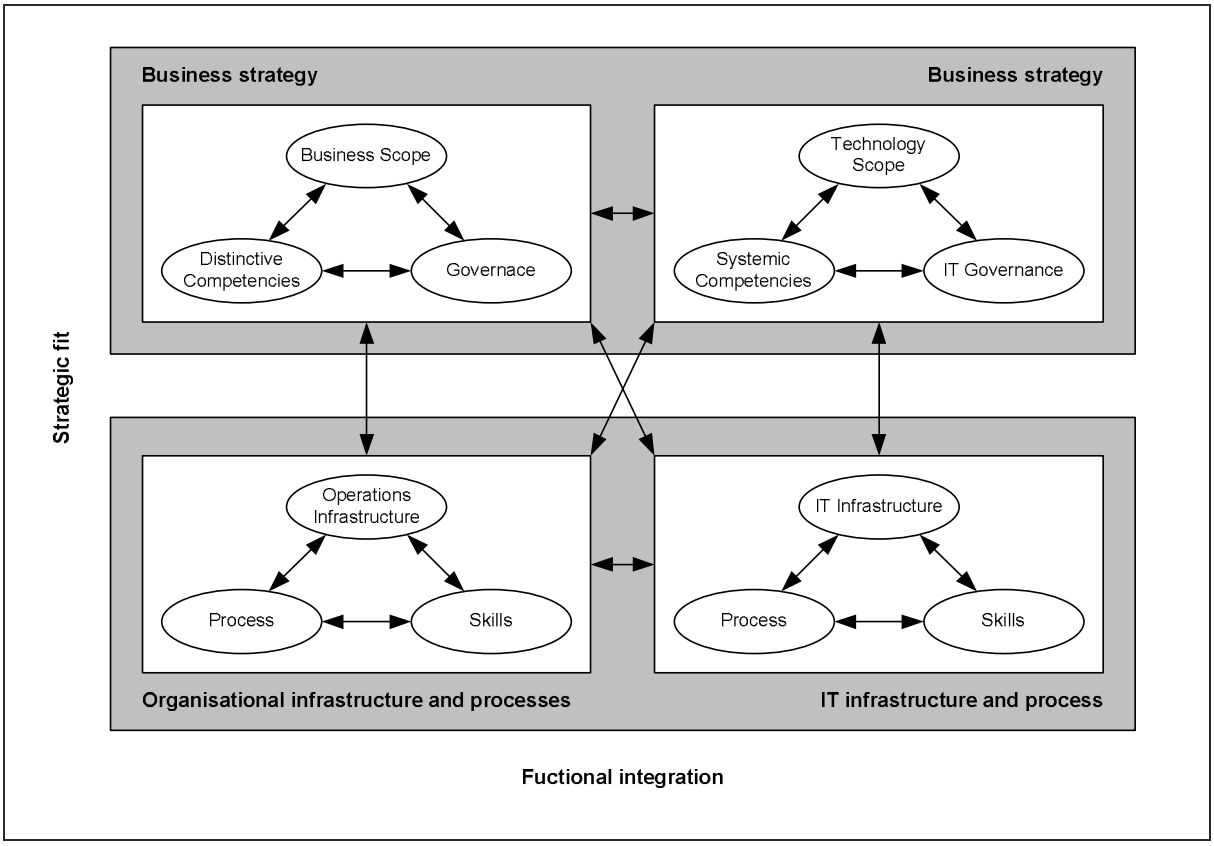

Bild 2.1: Das Strategie-Ausrichtungsmodell (vgl. [1])

Dies trifft insbesondere auch auf die Applikationen zu, da sie die Geschäftsprozesse in einem Unternehmen widerspiegeln. An der Universität Harvard wurde ein strategisches Geschäfts-IT-Ausrichtungsmodell (*strategic business-IT-alignment model* (SAM)) entwickelt, das einen strukturierten Mechanismus für die Verbindung von Geschäft und IT zur Verfügung stellt (vgl. Bild 2.1). Dieses Modell besitzt 2 Dimensionen, die funktionale Integration und die strategische Passung. Alle 12 Teilbereiche sind miteinander verknüpft und beeinflussen sich in

Abhängigkeit der Rolle von IT im Unternehmen gegenseitig. Die Entscheidung, welches dieser 4 Richtungen gewählt wird, ist nicht Aufgabe des *Application Managements*. Es handelt sich dabei um eine strategische Entscheidung der Geschäftsleitung des Unternehmens.

Das Modell definiert 4 grundsätzliche Betrachtungsweisen:

- IT als *Cost Centre*,
- IT als *Profit Centre*,
- IT als *Investment Centre* sowie
- IT als *Service Centre*.

Die beiden zuerst genannten Modelle sind häufig in der praktischen Anwendung, während *IT als Investment Centre* und *IT als Service Centre* Spezialfälle beschreiben. Tabelle 2.1 gibt eine kurze Definition dieser 4 Begriffe an.

Tabelle 2.1: Sichtweisen beim Strategie-Ausrichtungsmodell

IT als Cost Centre	***IT als Profit Centre***	***IT als Investment Centre***	***IT als Service Centre***
• Ausrichtung der IT an der Geschäftstrategie • Minimierung der Kosten steht im Mittelpunkt	• Die IT-Strategie bestimmt die IT-Technologie. • optimale Nutzung der vorhandenen IT-Infrastruktur	• Die IT-Strategie nimmt Einfluss auf die IT-Organisation.	• Im Mittelpunkt steht die optimale zur Verfügungstellung eines IT-Services.

2.2 *Key Business Driver*

Ein *Key Business Driver* ist die Eigenschaft einer Geschäftsfunktion, die das Verhalten und die Implementierung dieser Funktion maßgeblich steuern, um die strategischen Ziele des Unternehmens zu erreichen. Es ist ein Indikator, der beiden Gruppen (Geschäftsleitung und IT-Abteilung) einen gemeinsamen Satz von Zielen gibt, die innerhalb eines Projektes als Richtlinie benutzt werden können. Sie beeinflussen die Vision einer Firma und definieren den Auftrag und die Ziele von IT-Projekten innerhalb dieser Organisation.

Key Business Driver müssen an verschiedenen Eigenschaften festhalten, um praktisch an-

wendbar zu seien. In [1] werden 5 Schlüsselattribute festgelegt, die für die Bestimmung zur Anwendung kommen sollen. Jede Spezifikation sollte *SMART* formuliert seien.

- S - spezifisch (*specific*),
- M - messbar (*measurable*),
- A - abgesprochen (*agreed to*),
- R - realistisch (*realistic*),
- T - zeitspezifisch (*time specific*).

Diese Eigenschaften (*SMART*) sollten nicht nur auf die *Key Business Driver* angewendet werden, sondern auch auf funktionale Anforderungen, *Service Level Agreements* (SLA) oder Technologieanforderungen. Weitere Details der Definition und Anwendung können in [1] nachgeschlagen werden.

2.3 Das *Application Portfolio*

Nachdem die Ausrichtung von IT und Geschäft abgeschlossen ist, bleibt folgende Frage offen. Wie verwaltet eine Organisation ihre Anwendungen und Services. Es ist offensichtlich, dass eine Geschäftsfunktion, die durch einen Service unterstützt wird, ständigen Änderungen unterliegt. Gegeben durch die Komplexität von integrierten Anwendungen, ist die Fähigkeit einer effektiven Verwaltung dieser dynamischen Umgebung eine große Herausforderung für einen *IT-Service-Provider*.

Eine sehr erfolgreiche Methode der Verwaltung dieser komplexen Anwendungen ist der Einsatz eines *Application Partfolios*. Es bietet einen Mechanismus, um Applikationen in einem gegebenen Geschäftsumfeld weiterzuentwickeln. Falls richtig entworfen und verwaltet, kann mit einem *Application Portfolio* in einer sehr effektiven Art und Weise das Management von Investitionen in die IT-Infrastruktur durchgeführt werden. Es ist ein betriebliches Informationssystem (Datenbank), das alle Schlüsselattribute einer Anwendungen enthält. Alle Anwendungen, die für das Tagesgeschäft eines Unternehmens notwendig sind oder als *mission-critical* betrachtet werden, sollten in diese Datenbank aufgenommen werden. In der Anwendung eines *Application Portfolios* bestehen viele Gemeinsamkeiten zu einem Investment Portfolio. Tabelle 2.2 zählt Eigenschaften auf, die in einem solchen Informationssystem enthalten seien sollten.

Tabelle 2.2: Eigenschaften, die ein Application Portfolio enthalten sollte (nach [1] bzw. [19])

Relation	Erläuterung
allgemeines	▪ Name, ▪ ID, ▪ Beschreibung, ▪ wie kritisch für das Geschäft?
Relationen	▪ ausgeübte Geschäftsfunktionen, ▪ unterstützte Systeme, ▪ abhängige Anwendungen, ▪ ausgelagerte Funktionen, ▪ Partner der Auslagerung (Outsourcing);
Personen	▪ verantwortliche Manager, ▪ Fachbereichsleiter, ▪ verantwortliche Betreiber, ▪ Entwicklungsleiter (Projektmanager), ▪ Ansprechpartner für den Support;
technische Dokumente	▪ Systemarchitektur, ▪ Netzwerktopologie, ▪ Implementierungstechnologien;
organisatorische Dokumente	▪ Service Level Agreements (SLAs), ▪ Operational Level Agreements (OLAs);
Kosten	▪ für eine Neuentwicklung, ▪ jährliche Betriebskosten, ▪ Supportkosten, ▪ Wartungskosten;
Metriken	▪ Produktionsmetriken, ▪ Supportmetriken;

3. Application Management

3.1 Einführende Betrachtungen

Traditionelle Ansätze unterscheiden zwischen Anwendungsentwicklung (Serviceentwurf) und Servicemanagement (Servicebetrieb). Die Praxis hat gezeigt, dass dieses Vorgehensmodell nicht immer zu optimalen Resultaten geführt hat. Der hier vorgeschlagene Ansatz sieht beides als zusammenhängende Teile, die gegeneinander ausgerichtet werden müssen. Die Betrachtungsweise erfolgt dabei aus der Perspektive des Servicemanagements. Eine frühzeitige Berücksichtigung dieser Fragestellungen innerhalb des Lebenszyklus einer Anwendung hat einen großen Einfluss auf die effiziente Serviceerbringung, die sich an die Entwicklung anschließt.

Beispiele für diese gegenseitigen Beeinflussungen sind:

- Anwendungskomponenten und ihre Abhängigkeiten,
- Planung von *Releases* (inkrementelle Softwareentwicklung),
- Definition der Strukturen für Filesysteme,
- Skripte, die vor bzw. nach der Einführung einer Anwendung ausgeführt werden,
- Skripte, um eine Anwendung zu starten bzw. zu stoppen,
- Skripte, die die Hardware- und Softwarekonfiguration auf dem Zielsystem vor der Installation prüfen,
- Spezifikation von Metriken und Ereignissen, die die Anwendung liefert, und zur Status- und Leistungseinschätzung herangezogen werden können,
- Nutzung von Skripten durch den Systemadministrator zur Verwaltung der Anwendung,
- Vereinbarung von *Service Level Agreements* (SLAs),
- Anforderungen, die sich aus dem Betrieb, dem Support bzw. dem Servicemanagement ergeben;

Eine Definition, wie die (neue) Einzelanwendung mit den anderen Applikationen interagieren soll, ist oft notwendig.

Anwendungsentwicklungs- und Servicemanagementabteilungen müssen eng zusammenarbeiten, um sicherzustellen, dass in jeder Phase des Lebenszylus Servicedefinition und -betrieb als

Einheit betrachtet wird. Dieser Denkansatz kann einen längeren Veränderungsprozess nach sich ziehen, der viele Fallen in sich birgt.

Eine IT-Organisation muss die Qualität des *Application-Management-Lifecycles* überwachen. Der Informationsfluss zwischen den einzelnen Phasen beeinflusst seine Qualität. Das Verständnis jeder einzelnen Phase ist für die Verbesserung der Qualität des ganzen Kreislauf entscheidet. Methoden und Werkzeuge, die in der einen Phase benutzt werden, können Einfluss auf eine andere nehmen. Die Optimierung einer Phase kann eine Suboptimierung des gesamten Kreislauf bewirken.

3.2 Die Phasen des Lebenszyklus

Bild 3.1 gibt den *Applications-Management-Lebenszyklus* wieder, der [1] zu Grunde liegt. Die Phasen Anforderungsanalyse (*Reqirements*), Design (*Design*) und Realisierung (*Build*) gehören zur Anwendungsentwicklung. Das Servicemanagement ist in die Abschnitte Einführung (*Deploy*), Betrieb (*Operate*) und Optimierung (*Optimise*) einer Applikation involviert.

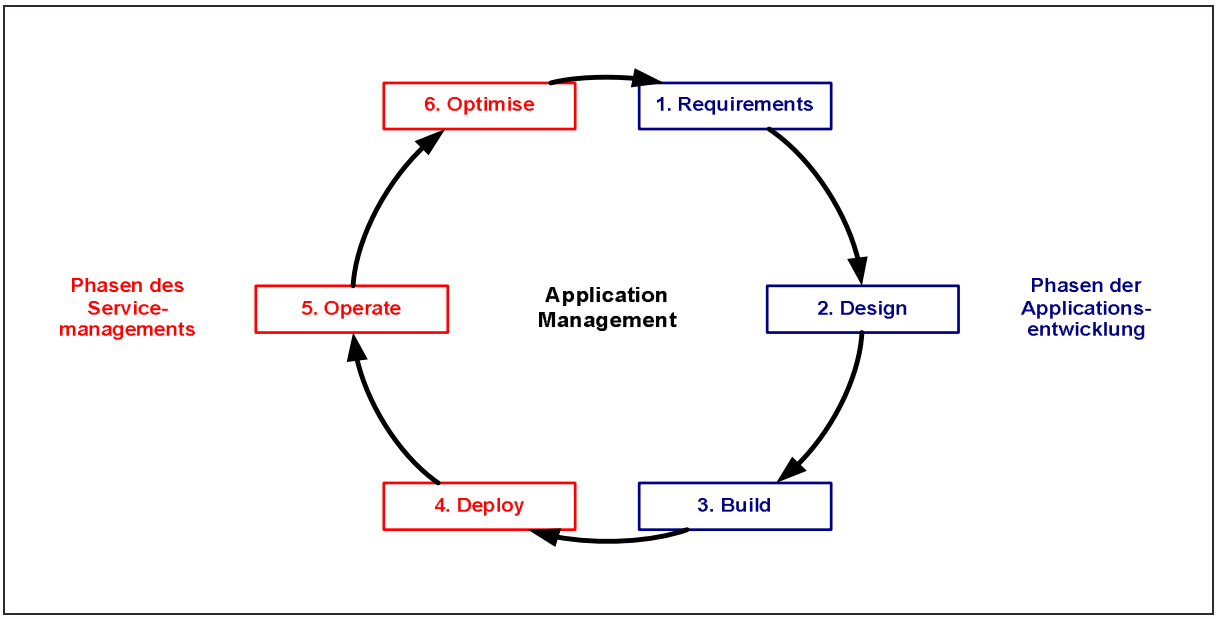

Bild 3.1: Der Application-Management-Lebenszylus (vgl. [1])

Die Phase der Stilllegung (*retirement*) einer Anwendung ist in Bild 3.1 nicht dargestellt. Trotzdem ist sie zu berücksichtigen. Unter Stilllegung wird das endgültige Entfernen einer Applikation aus der IT-Infrastruktur verstanden, nicht aber die seltene Nutzung. Ggf. sind auch für diese Phase Planungsaktivitäten durchzuführen.

Diese Lebenszyklusmodell ist linear bzw. „*wasserfall-basierend*“. Trotzdem kann sich eine Anwendung in mehreren Zuständen zur selben Zeit befinden. Dies erfordert eine strenge *Versions-*, *Konfigurations-* und *Releasekontrolle*.

Für ein besseres Risikomanagement wird dieser Ansatz um Inkremente oder Zyklen erweitert. Risiken entstehen durch ungenaue Anforderungen und Änderungswünsche. Die Nutzung von Inkrementen impliziert, dass das System Stück für Stück entwickelt wird. Jede dieser Komponenten unterstützt eine Geschäftsfunktion des Gesamtsystems. Dieser Entwicklungsansatz kann die Serviceeinführung (*time to market*) verkürzen. Ein Entwicklungsteam besitzt in diesem Fall weitere Entscheidungsmöglichkeiten. Prototypen helfen, ein besseres Verständnis für nicht verstandene Anforderungen zu gewinnen.

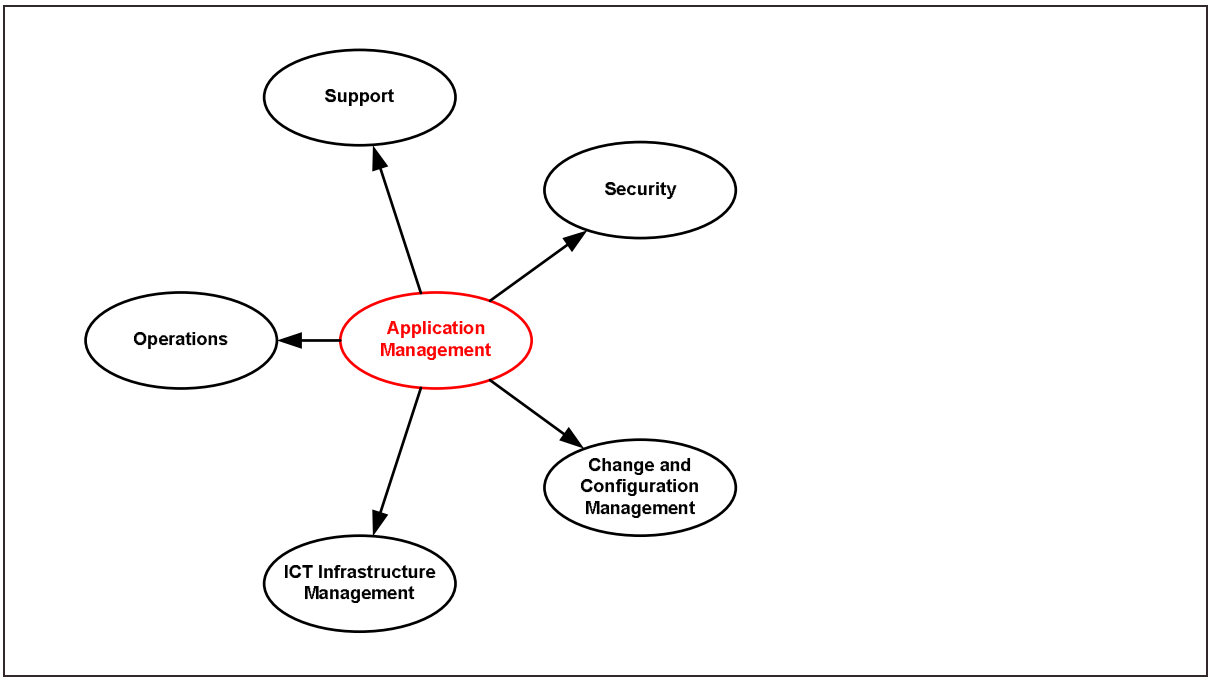

Bild 3.2: Abhängigkeiten der Rollenverteilung beim Application Management

In den einzelnen Phasen des Lebenszyklus sind neben den Softwareentwicklern folgende Rollen aus dem Servicemanagement beteiligt:

- *Operation*,
- *Support*,
- *Security*,
- *Change- and Configuration Management* sowie

- *ICT Infrastructure Management*.

Diese Abhängigkeiten verdeutlicht ebenfalls Bild 3.2. Die Besetzung jeder Rolle ist durch eine oder mehrere Personen möglich. In kleinen IT-Service-Abteilungen kann eine Person auch mehrere Rollen übernehmen. In den folgenden Unterkapiteln wird immer wieder darauf hingewiesen, welche Aufgaben die einzelnen Rollen wahrnehmen müssen.

3.3 Anforderungsanalyse

Jede Applikation tritt in dieser Phase in den Zyklus ein. Das Entwicklungsteam arbeitet eng mit dem Management einer Organisation zusammen, um deren Anforderungen an die neue Anwendung festzulegen. Diese Phase identifiziert die Funktionalitäten, Performance-Level sowie andere Charakteristiken, die die Applikation besitzen soll. Anforderungen zur Modifikation einer bestehenden Anwendung, die sich in der Optimierungsphase befindet, haben ihren Ursprung in einem Antrag. Diesen bezeichnet ITIL als *Request for Change* (RFC). Es wird zwischen folgenden Anforderungstypen unterschieden:

- funktionale Anforderungen,
- nichtfunktionale Anforderungen,
- Nutzeranforderungen,
- Change Cases.

- ***Funktionale Anforderungen***

Funktionale Anforderungen definieren den Service, den die Anwendung erbringen soll. Sie werden auch als fachliche Anforderungen bezeichnet. ITIL empfiehlt die Verwendung von *System-Context-Diagrammen* bzw. *Use-Case-Modellen* (vgl. [8]).

- *System-Context-Diagramme*:

Sie verdeutlichen alle Informationen, die zwischen der Anwendung und der Umwelt ausgetauscht werden. Dies betrifft sowohl Datenquellen als auch –senken.

- *Use Case Modelle*:

Sie legen die Interaktionen zwischen den externen Akteuren und dem in Planung befindlichen

System fest. Akteure sind Parteien außerhalb des Systems, die mit ihm interagieren. Sie können aber auch Klassen von Nutzern bzw. andere Systeme repräsentieren. Eine Veranschaulichung von Rollen, die Nutzer spielen können, ist ebenso möglich.

Mit Hilfe von *Use Cases* ist eine Darstellung der Kommunikationsbeziehungen zwischen Geschäftsleitung und Anwendungsentwicklern möglich. Sie bilden die Basis für die Dimensionierung, die Planung des Entwicklungsaufwandes und helfen bei der Definition von Nutzeranforderungen. Ebenso können sie aufgrund ihrer leichten Verständlichkeit zum Bestimmen der funktionalen Aspekte von *Service Level Agreements* verwendet werden. Da diese *Use Cases* alle Szenarien beschreiben, die eine Anwendung unterstützt, können sie zur Definition von Testfällen herangezogen werden.

Für Detailplanungen sind weitere Softwaremodellierungstechniken zulässig. Ein konzeptionelles Datenmodell beschreibt die verschiedenen Komponenten eines Systems, deren Beziehungen untereinander sowie deren interne Struktur. *State-Transition-Diagramme* können dynamische Abläufe veranschaulichen. Sie geben die verschiedenen Zustände zusammen mit Ereignissen, die zu diesen Zuständen führen, wieder. Interaktionen zwischen den Komponenten einer Anwendung können durch *Object-Interaction-Diagramme* modelliert werden.

CASE-Tools helfen diese Modelle auf einem konsistenten Stand zu halten.

- ***Nichtfunktionale Anforderungen***

Nichtfunktionale Anforderungen bestimmen die technischen Anforderungen an ein IT-System. Sie bilden die Basis für die Systemdefinition, Kostenabschätzung und können für Machbarkeitsstudien des IT-Projektes herangezogen werden. Sie haben großen Einfluss auf die Systemarchitektur. Nichtfunktionaler Anforderungen sind in Tabelle 3.1 aufgelistet. Sie beschrieben die Qualitätsmerkmale der zukünftigen Anwendung. Diese Merkmale können für das Erarbeiten von Testplänen nichtfunktionaler Anforderungen verwendet werden.

Tabelle 3.1: Kategorien nichtfunktionaler Anforderungen

Kategorie	Erläuterung
Steuerbarkeit	▪ -
Effizienz	▪ Wie viele Ressourcen werden verbraucht?
Effektivität	▪ Sind die Geschäftsziele richtig verstanden?
Interoperabilität	▪ Zusammenarbeit mit anderen Systemen
Verfügbarkeit, Zuverlässigkeit	▪ -
Kapazität, Leistung	▪ -
Sicherheit	▪ -
Behebbarkeit	▪ Wie schnell kann der Service nach einem Fehler wiederhergestellt werden? ▪ Gibt es Prozeduren oder Werkzeuge für die Wiederherstellung?
Installierbarkeit	▪ Wie groß ist der Installationsaufwand für die Anwendung? ▪ Stehen automatisierte Installationsprozeduren zur Verfügung?
Überwachbarkeit	▪ Besitzt sie standardisierte Konfigurationen?
Wartbarkeit	▪ Wie gut kann die Anwendung abgepasst werden?
Betreibbarkeit	▪ Beeinflusst die Applikation andere Anwendung in ihrer Funktionalität?
Absicherbarkeit	▪ Ist die Anwendung sicher vor unberechtigtem Zugriff?

- ***Nutzeranforderungen***

Dieser Anforderungstyp soll die Nutzerfreundlichkeit der Anwendung sicherstellen. Vorhandene Standards zum Gestalten von Benutzerschnittstellen und -oberflächen fließen in diese Arbeit ein. Nutzeranforderungen bestimmen Qualitätsmerkmale einer Anwendung, die wiederum für den Softwaretest benutzt werden können.

- ***Change Cases***

Change Cases spezifizieren Erweiterungen oder Veränderungen der Funktionalität einer Anwendungen in der Zukunft. Im Vorfeld sind sie mit dem Auftraggeber in Art und Umfang abzusprechen. Diese Evaluierungen verursachen Kosten. Deshalb ist ihre Definition auf ein sinnvolles Maß zu begrenzen.

- ***Testen der Anforderungen***

Dies ist eine häufig übersehene Möglichkeit, die Qualität einer Anwendung, die sich in der

Entwicklung befindet, zu verbessern. Techniken, die hierfür verwendet werden können, sind:

- das Anfertigen von Prototypen (Dummy-Anwendungen),
- Präsentationen,
- erneutes Überarbeiten der Dokumente, Diagramme und Modelle (*Reviews*).

Die Herausforderung des Testens in diesem Stadium ist es, die enorme Vielfalt der Anforderungen zu berücksichtigen. Erschwerend kommen die unterschiedlichen Meinungen der beteiligten Parteien hinzu (Inhaber der Anwendung, Nutzer, Systemdesigner).

- ***Besetzung des Teams in der Phase der Anforderungsanalyse***

- *Change- and Configuration Management*:

Es verwaltet die *Configuration Management Database* (CMDB), die innerhalb des gesamten Anwendungslebenszyklus verwendet wird. Das *Change and Configuration Management* stellt Informationen zur Verfügung, um für die neue Anwendung das *Change*, *Release* und *Configuration Management* festzulegen. Weitere Informationen befinden sich in [2].

- *Support*

Diese Rolle ist für die Supportunterstützung der neuen Anwendung verantwortlich. Darin eingeschlossen ist das *Incident* und *Problem Management*. *Incident* und *Problem Management* werden im Detail in [2] untersucht.

- *Operations*

Die Rolle *Operations* ist für die tägliche Wartung einer Anwendung zuständig, sobald sich diese im Produktionsbetrieb befindet. Sie hilft bei der Definition von betrieblichen Anforderungen. Dazu gehört auch die Kontrolle, dass Kompatibilität zu den anderen Anwendungen einer Organisation besteht, die sich zur Zeit im Betrieb befinden. Zusätzlich ist es Aufgabe dieser Rolle, Standards für die tägliche, wöchentliche, monatliche und *Ad hoc*-Wartung festzusetzen.

- *Security*

Das primäre Ziel der *Security*-Rolle innerhalb einer IT-Organisation ist die Sicherstellung von Datenvertraulichkeit, -integrität und -verfügbarkeit. Es ist Aufgabe dieser Rolle, eine effizien-

te Backup-Strategie einzuführen. Kritische Daten (Definition durch Gesetze, Industrie- und Firmenrichtlinien) müssen in festgelegten Intervallen sicher gespeichert werden. Alle andere anfallenden Daten sollten von Zeit zu Zeit gelöscht werden, um die Kosten der Archivierung zu minimieren.

- *ICT Infrastructure Management*

Das *ICT Infrastructure Management* formuliert in dieser Phase die Anforderungen das *Capacity* und *Availability Management*. Dazu zählt die Identifikation von *Service Level Agreements* (SLAs). Weitere Informationen finden sich in [2] und [3].

3.4 Design

Diese Phase ist eine der wichtigsten innerhalb des Lebenszyklus. Sie nutzt die Resultate der Anforderungsanalyse und erstellt daraus eine Spezifikation. Sie bildet die Basis für die Implementierung. Es findet eine Verfeinerung der Dokumente aus der Anforderungsanalyse statt. Ein Architekturmodell, das das Zusammenwirken der funktionalen Komponenten der Anwendung mit dem IT-System (Desktops, Server, Datenbanken, Netzwerke) veranschaulichen, muss erstellt werden.

Eine Abschätzung der Last, die die Anwendung erzeugt, dient der Dimensionierung der IT-Infrastruktur. Für eine zügige Einführung bzw. einen sicheren Betrieb ist die Kenntnis der Charakteristik der Anwendung erforderlich. Funktionale und nichtfunktionale Anforderungen sollen mit der gleichen Intensität bearbeitet werden. Es ist zu prüfen, in welchen Punkten der Entwurf gegen firmenspezifische Designrichtlinien verstößt. Diese Konflikte sind aufzulösen.

In dieser Phase werden erste Vorimplementierungen vorgenommen. Sie helfen nicht nur den Softwareentwicklern, sondern dienen auch der Projektabstimmung, da aufkommende Fragen leicht mit dem Kunden abgeklärt werden können. In keiner anderen Projektphase verursachen Modifikationen und Zusätze weniger Aufwand, als zu diesem Zeitpunkt. Falls es das Projekt zulässt, sollten von Anfang an *Application Frameworks* (vgl. Abschnitt 2.2.5) zum Einsatz kommen.

- ***Projektmanagement***

Das Ziel besteht darin, ein flexibles Design zu erarbeiten, in dem Änderungen in akzeptabler Zeit eingebracht werden können. Ein Zeitplan über die Projektlaufzeit ist in dieser Phase anzufertigen. Eine Risikobewertung der einzelnen Entwicklungsabschnitte ist anzustellen. Kritische Aktivitäten erhalten mehr Aufmerksamkeit. Mit ihrer Bearbeitung soll begonnen werden, damit genügend Zeit für unvorhersehbare Ereignisse vorhanden ist. Die Balance zwischen den verfügbaren Ressourcen (Mitarbeiter, Geld), dem Projektzeitplan und den realisierten Funktionen der Anwendung ist herzustellen.

- ***Testen der Anforderungen***

Es ist zu prüfen, ob die gefundene Spezifikation mit den funktionalen und nichtfunktionalen Anforderungen aus Phase 1 vereinbar ist. Für diese Aufgabe schlägt ITIL den Einsatz verschiedener *Code-Inspection*-Verfahren vor (*Inspection*, *Desk check*, *Walk-through*, *Review*), die in [1] definiert sind. Diese statischen Tests sollen vor der formalen Genehmigung des Designs stattfinden. Mitarbeiter der Geschäftsleitung sowie der IT-Serviceabteilung sind in diese Tests einzubeziehen.

- ***Besetzung des Teams in der Designphase***

- *Change- and Configuration Management*:

Diese Rolle unterstützt das Entwicklerteam mit Informationen, wie die Applikation von der Entwicklungs- und Testumgebung in die Produktionsumgebung überführt werden soll. Sie wird in Fragen der Versionskontrolle, der Softwareverteilung, der Lizenzverwaltung, der Nutzerüberwachung und Stilllegung konsultiert (*Change Control*, *Status Reporting*). Zusätzlich untersucht das *Change and Configuration Management*, in welcher Form die Anwendung Konfigurationsdaten zur Verfügung stellt, um sie in die *Configuration Management Database* (CMDB) zu übernehmen. Bilden hierfür entsprechende Standards (vgl. WMI, DMI) die Grundlage, lässt sich dieser Datenaustausch automatisieren.

- *Support*

Diese Rolle unterstützt den Entwicklungsprozess in dieser Phase mit Informationen, wie die Supportfähigkeit der neuen Anwendung erleichtert werden kann. Dies beinhaltet einen einfachen Zugriff auf Informationen für den *Service Desk* oder direkte Unterstützung von *Incident*- und *Problem Management*-Funktionen. Das Mitschreiben von kritischen Betriebszuständen in

einem gemeinsamen Format fällt ebenso in diese Verantwortlichkeit. Das Implementieren von Methoden, die die Servicewiederherstellung nach einem Fehler beschleunigen, die Supportkosten minimieren oder spezielle Hilfe-Tools bieten, ist zu beaufsichtigen.

- *Operations*

Die Rolle *Operations* ist an einem störungsfreien Betrieb der neuen Anwendung interessiert. Folgende Maßnahmen können dies garantieren:

- Gestalten der Applikation, die die Richtlinien für Nutzerschnittstellen einer Organisation berücksichtigen,
- Vorsehen einer Möglichkeit zur Integration in eine zentrale Managementplattform,
- Aufstellen von Zeitplänen und wiederkehrenden Prozessen, so dass unternehmensweite Managementstandards eingehalten werden,
- Sicherstellen, dass die Betriebsdokumentation der neuen Anwendung den Organisationsrichtlinien entspricht,
- Definition von *Operational Level Agreements* (OLAs) sowie deren Sicherstellen durch das Softwaredesign.

- *Security*

Die Rolle *Security* konzentriert sich darauf, dass die Sicherheitsanforderungen der Organisation erfüllt werden. Hierzu zählen:

- Überwachung der korrekten Arbeitsweise der IT-Ressourcen,
- Funktionen zum *Intrusion Detection* und *Virus Protection*,
- keine Anfälligkeit im Hinblick auf *Denail-of-Service*-Attacken,
- Unterstützung von Gruppenstandards und Festlegen von Verfahren zur Datensicherung und sicherem Löschen von Daten,
- liefern von Informationen über Betriebszustände und Ereignisse,
- keine Angriffsmöglichkeiten für Netzwerkattacken,
- Implementieren von Alarmen zur Identifikation von Eindringlingen in das Netzwerk,
- Unterstützung von geeigneten Sicherheitstechnologien,
- Unterstützung der Organisationsrichtlinien zur Benutzerverwaltung (Passwörter),
- Treffen von externen und physikalischen Sicherheitsmaßnahmen (z.B. Zugang zu Rechner-Pools),

- Berücksichtigen der Protokollierung von sicherheitsrelevanten Ereignissen;

- *ICT Infrastructure Management*

Diese Rolle ist in eine Reihe von Festlegungen involviert, die das *Capacity*-, *Availability*- und *Service Conitinuity Management* betreffen. Dabei liegen die Schwerpunkte auf:

- Vorsehen eines *Capacity Managements* für die Applikation und der unterlagerten IT-Infrastruktur,
- Überwachen der Verfügbarkeit der Infrastruktur,
- Bereitstellen eines Kostenmanagements zum Verwalten der IT-Ausgabe,
- Ausliefern von *Server Builds*,
- Standardinstallationen, Softwareinstallation.

3.5 Realisierung

Ist die Phase des Designs abgeschlossen, nutzt das Entwicklerteam die erarbeitete Spezifikation für die Implementierung und den Test der Anwendung.

- ***Einheitliche Richtlinien zum Abfassen des Quelltextes (Coding Conventions)***

Das Ziel einheitlicher Richtlinien zum Abfassen von Quelltexten besteht in seiner einfachen Verständlichkeit. Dies erleichtert Personen, die nicht an der Entwicklung der Anwendung beteiligt waren, die Einarbeitung. Gutem Softwaredesign liegt auch lesbarer Quellcode zu Grunde, der den Richtlinien des Unternehmens entspricht. Sie dokumentieren logische Abhängigkeiten (*Preconditions*, *Postconditions* von Methoden und Funktionen) und ermöglichen automatische Tests. Aus der Erfahrung heraus rät ITIL nur eine begrenzte Anzahl dieser Richtlinien zu definieren, die für alle Softwareentwickler einer Organisation verbindlich sind. Im Gegensatz dazu würde ein komplexes Regelwerk stehen, das versucht jeden Aspekt abzudecken, aber nicht konsistent innerhalb eines Unternehmens angewendet wird.

- ***Anwendungsunabhängige Implementierungsrichtlinien***

- *Templates, Code-Generatoren und Application Frameworks*

Eine Vielzahl von Entwicklungswerkzeugen erleichtert das Erstellen von Quellcode. Prinzi-

piell stehen folgende Möglichkeiten zur Verfügung:

- *Code-Generatoren*, die Code-Skelette erzeugen,
- *Templates*, die Teile von Komponeneten definieren und
- *Application Frameworks*, die komplette Komponenten zur Verfügung stellen.

Falls es die Spezifikation zulässt, ist die Nutzung dieser Codebausteine einer Eigenentwicklung in jedem Fall vorzuziehen. *Application Frameworks* können viele der nichtfunktionalen Anforderungen berücksichtigen, die für Anwendungen mit vergleichbarer Charakteristik gelten (vgl. hierzu [1]).

Code-Generatoren, *Templates* und *Application Frameworks* zählen zum betriebswirtschaftlichen Vermögen (Aktivposten) eines Unternehmens. Sie enthalten das geistige Kapital aus früheren Projekten. Je mehr standardisierte Programmteile in einer Applikation Verwendung finden, desto schneller kann die Applikation erstellt werden. Dies trägt zu einer Kostenreduzierung bei.

- *Einbettung von Schnittstellen zur Verwaltung und Diagnose von Anwendungen*

Zur Definition von einheitlichen Managementschnittstellen für Applikationen haben folgende Ausschüsse Standards erarbeitet (vgl. hierzu auch [5]).:

- *IBM Application Management Specifikation* (AMS),
- *Distributed Management Task Force* (DMTF),
- *Desktop Management Instrumentation* (DMI).

Sie sollen im Rahmen der Implementierung einer Applikation berücksichtigt werden.

Die Integration von Diagnosepunkten (*Diagnostic Hooks*) ermöglicht es *Debugging-Werkzeugen* die Ausführung von verteilten Systemen zu beobachten. Sie liefern Informationen über die Performanz der Anwendung, Fehlerzuständen und Konfigurationssätzen. Sie werden implementiert, um die Systemverfügbarkeit und –zuverlässigkeit zu erhöhen und die Ausfallzeit von Applikationen zu minimieren. Sie helfen, die Qualität von Anwendungen zu erhöhen. *Diagnostic Hooks* lassen sich in 3 Kategorien unterteilen:

- Ereignisse, die von der Hardware und dem Betriebssystem gemeldet werden,
- Meldungen, die von beteiligten Softwaresystemen (z.B. Datenbanken, Web-Server) verfügbar sind,
- Informationen, die die Applikation selbst erzeugt.

Diese Diagnosepunkte sind ein wichtiges Instrument innerhalb der Testphase, die sich an die Implementierung anschließt.

- ***Test***

Ist die Implementierung der Applikation beendet, ist es notwendig, sie einem gründlichen Test zu unterziehen. Im Mittelpunkt steht dabei die Frage, ob alle Anforderungen und Funktionen erfüllt werden, die im Rahmen der Anforderungsanalyse erarbeitet wurden.

Der Test nichtfunktionaler Anforderungen ist mit der gleichen Intensität durchzuführen, wie dies für funktionale Anforderungen der Fall ist. Dieser Sachverhalt wird oft übersehen. Jeder Testplan soll eine Abschnitt enthalten, der jede nichtfunktionale Anforderung beschreibt. Darin wird auch das Testverfahren jeder einzelnen Anforderung definiert.

Die Testverfahren können in großem Umfang die Qualität einer Anwendung beeinflussen. Sehr viele Projekte verlegen diese Tests an das Ende der Realisierungsphase. Infolge von Zeitverzug kommt es immer wieder zum Abbruch der Tests, da der Termin für die Auslieferung/Einführung nicht verschoben werden kann. Um ein qualitativ hochwertiges Softwareprodukt auszuliefern, ist das Testen in dieser Phase eine allumfassende Aktivität. Grundlage für das Erstellen von Testplänen bilden die Dokumente aus der vorangegangenen Phasen (*Design*).

Es stehen verschiedene Testtypen zur Verfügung. Tests, die vom Entwicklungsteam durchgeführt werden, sind:

- *Unit*-Tests,
- Systemtests und
- Integrationstests.

Interne Design-Spezifikationen können zum Testen von *Units* oder des Systems herangezogen werden. ITIL schlägt in [1] ein Beispiel für einen Unit-Testplan vor. Die Nutzung externen Designs ist zur Entwicklung von Abnahmetestplänen verwendbar.

Bei der Kundenabnahme eines Software-Systems können folgende Tests stattfinden:

- *Functional Acceptance Test*,
- *User Acceptance Test* und
- *Production Acceptance Test*.

Falls eine Anwendung iterativ entwickelt wird oder für die mehrere Versionen geplant sind, sollen alle Wiederholungstests gegen die originalen Testspezifikation durchgeführt werden. Eine Möglichkeit um sicherzustellen, dass die Testumgebung eine Nachbildung der Produktionsumgebung darstellt, ist der Betrieb unter *Change* und *Configuration Management*. Dies ist aber nur zulässig, falls an der Applikation keine Anpassungen notwendig sind.

- ***Besetzung des Teams in der Phase der Realisierung***

- *Change- and Configuration Management*:

Diese Rolle überwacht das Implementieren und Testen von Funktionen, die das geforderte *Change- and Configuration Management* erbringen. Dies schließt folgendes ein:

- Komponentenidentifikation,
- Veränderungskontrolle (*Change Control*),
- Statusaufzeichnung (*Status Control*),
- Versionskontrolle,
- Softwareverteilung,
- Lizenzkontrolle,
- Nutzerüberwachung,
- Informationen zum Stilllegen der Anwendung.

- *Support*

Die Rolle *Support* arbeitet eng mit dem Testteam zusammen, damit sie im späteren Betrieb Wartungsarbeiten ausführen kann. Ideal ist es, wenn die Personen, das diese Rolle wahrnehmen müssen, in der Testumgebung erste Erfahrungen mit der neuen Anwendung sammeln

können. Sie arbeiten unter Anleitung der Softwareentwickler.

Der *Service Desk* kann am *User Acceptance Test* teilnehmen. Dies hat den Vorteil, das die Endbenutzer *Incidents* über die normalen Kanäle propagieren. In [1] wird die Meinung vertreten, dass erfahrene Servicemitarbeiter *Incidents* schneller bearbeiten, als dies Softwareentwickler tun.

- *Operations*

Die Rolle *Operations* hat in dieser Phase mit der Rolle *Support* vergleichbare Aufgaben. Sie soll aktiv am Test der Applikation mitarbeiten. Die Betreiber sollen an der Testumgebung Kenntnisse im Umgang mit der neuen Applikation erwerben.

- *Security*

Durch eine enge Zusammenarbeit mit dem Entwicklungsteam ist zu garantieren, dass die neue Anwendung nicht die Sicherheitsstandards der Organisation verletzt. Mindestens soll an folgenden Aktivitäten mitgearbeitet werden.:

- Risikobewertung der neuen Anwendung,
- Überarbeitung (*Reviews*) von Quellcode und Dokumentationen,
- Erarbeitung von Testplänen zum Überprüfen der Einhaltung von Sicherheitsstandards,
- Hinweise zu *Trade-off*-Entscheidungen, die die Sicherheit beeinflussen.

- *ICT Infrastructure Management*

Das *ICT Infrastructure Management* berät die Entwicklungs- und Testteams beim Implementieren von Methoden für das *Capacity* und *Availability Management*. In der Testumgebung wird geprüft, ob die in der Phase der Anforderungsanalyse vereinbarten *Service Level Agreements* verifiziert werden können.

3.6 Einführung

Haben alle Parteien den Abnahmetests zugestimmt, kann die Einführung in das Unternehmen beginnen. Viele Probleme, die während einer Einführung auftreten, haben ihre Ursachen in Änderungen. Deshalb müssen im Vorfeld sorgfältige Planung angestellt werden, um diesen Problemen erfolgreich entgegenzutreten.

- ***Planung der Einführung***

Ein guter Ansatz zur Planung der Einführung einer neuen Applikation ist die Anwendung eines Rahmenwerkes. Es muss sich auf die Gegebenheiten einer kleinen Organisation genauso skalieren lassen, wie an ein multinationales Unternehmen. Eine ausführliche Diskussion dieser Zusammenhänge findet sich in [2] und [3].

Dem Team, das die Einführung der Anwendung überwacht, sind Informationen in Form eines *Deployment Kit* zur Verfügung zu stellen. Bestandteile, die diese Dokumentensammlung enthalten muss, sind in Tabelle 3.2 (vgl. [1]) angegeben.

Tabelle 3.2: Dokumente eines Deployment Kits

Dokument	Erläuterung
Deployment-Checkliste	Es beschreibt die Aktivitäten, die für einen Abschluss der Einführung erbracht werden müssen. Es ist ein Projektplan, der alle Schlüsseldaten für die Systemeinführung enthält.
Deployment-Rollback-Dokument	Es beschreibt Einzelheiten, wie die Einführung in jedem Schlüsselstadium abgebrochen werden kann. Es nennt Personen, technische Ressourcen und Aktivitäten, die hierfür notwendig sind.
Abschlusszertifikat	Es hält die erfolgreiche Einführung fest. Es wird durch einen Vertreter der Anwendungsnutzer unterzeichnet.

- ***Genehmigung der Einführung***

Voraussetzung für die Genehmigung der Einführung ist die Vollständigkeit der Dokumente. Eine Hauptaufgabe des *Change Management* Prozesses ist es, dass alle Parteien, die von der Einführung betroffen sind, von der bevorstehenden Änderung Kenntnis haben. Viele Systeme beeinflussen sich gegenseitig. Das *Change Management* versucht alle Systeme zu identifizieren, die von der bevorstehenden Änderung betroffen sind. Gleichzeitig versucht es, die Auswirkungen auf in Betrieb befindliche IT-Systeme zu minimieren.

Diese Systemanalyse kann aus einem *Request for Change* (RFC) resultieren, der in einer früheren Phase des Entwicklungszyklus erzeugt wurde. Das *Change Advisory Board* (CAB) bearbeitet diesen Antrag. Dabei handelt es sich um eine Gruppe von Servicemanagementmitar-

beitern der IT-Organisation, die die Weiterentwicklung und Pflege bestehender Systeme plant.

Ist die Einführung genehmigt, kann der Prozess der Verteilung der Applikation beginnen.

- ***Softwareverteilung***

Die Strategie der Softwareverteilung ist einer der kritischen Erfolgsfaktoren für die Einführung einer Anwendung. Dieses Problem ist Gegenstand von [2] und [3]. Eine Anwendung kann über Disketten, CDs oder dem Netzwerk (Intranet, Extranet) verteilt werden. Eine Analyse folgender Gesichtspunkte wird empfohlen.:

- *Packing*
  Alle Dateien einer Anwendung müssen so aufbereitet werden, dass eine einfache Verteilung zu den geforderten Zielen möglich ist. Eine Installationsalgorithmus ist zu entwickeln. Ziel muss es seien, diesen Prozess zu automatisieren, so dass er ohne fremde Hilfe abläuft.

- *Einführung*
  Entscheidend ist, dass die Installation neuer Anwendung keine Störungen existierender Softwaresysteme provoziert. Es sind Funktionen zu entwickeln, die dem Nutzer der Applikation ein einfaches Softwareupgrade ermöglichen.

- *flexible Softwaredistributionen*
  Eine Anwendung ist in Distributionen aufzuteilen, um sie in verschiedenen Stufen, die auf Nutzergruppen zugeschnitten sind, im Unternehmen zu installieren. Funktionen, die hierfür durch Netzmanagementplattformen zur Verfügung stehen, sind zu nutzen.

- *Push-Deployment-Strategie*
  Dieses Softwareverteilungsverfahren läuft zeitgesteuert ab. Dies minimiert den Ressourcenverbrauch (Serverbelastung, Netzwerkverkehr) während der Einführung. Dieses Verfahren stellt sicher, das die Softwareinstallation erst dann abläuft, wenn der Benutzer nicht mit dem IT-System arbeitet. Es ist möglich, die Softwareverteilung zu unproduktiven Zeiten ablaufen zu lassen (Abend- und Nachtstunden).

- *Pull-Deployment-Strategie*

  Diese Verteilungsstrategie ermöglicht einem System eine Anwendung sofort von einem Softwareverteilungsserver auf die Client-Rechner zu verteilen. Diese Verfahren hat den Vorteil, das die genaue Anzahl der Rechner, auf denen die Software aufgespielt werden soll, nicht bekannt seien muss.

- *Selbstheilung*

  Während einer Softwareverteilung können Fehler auftreten (z.B. Serverausfall). Verfahren, die an einem Unterbrechungspunkt wiederaufsetzen können, sind zu bevorzugen. In diesem Fall sparen sie Netzbandbreite und Zeit.

- *Patching*

  Die Grösse eines Softwareupdates (*Release*) kann den Erfolg einer Einführung beeinflussen. Dies trifft besonders auf Netzwerke mit geringer Übertragungsbandbreite zu. In diesen Fällen kann die Aufspaltung von Installationspaketen in Segmente mit anschließender Reassemblierung im Zielrechner zum Erfolg führen.

- *Protokollierung*

  Ein detailliertes Aufzeichnen aller Ereignisse während einer Softwareinstallation gibt dem Team, das in die Einführungsphase involviert ist, einen Überblick über den Verteilungsprozess.

- *Back-out Facility*

  Falls im Rahmen einer Einführung unerwartete Probleme auftreten, ermöglicht diese Funktion einer Managementplattform die Koordination eines synchronisierten Rücksprungs zu alten Softwareinstallationen auf einigen oder allen Client-Rechner.

- ***Piloteinführung***

Eine Piloteinführung ist eine Softwareverteilung in einem kleinen, aber repräsentativen Ausschnitt der Produktionsumgebung. Zeitlich ist sie nach dem Abschluss der Softwaretests und vor Einführung der Applikation in eine Organisation einzuordnen. Es ist nicht notwendig, die gesamte Systemfunktionalität im Rahmen dieser Tests nachzuweisen. Eine Piloteinführung

soll folgende Fragen beantworten:

- Dokumentation von Produktionsabläufen, die in der Testumgebung nicht simuliert werden konnten,
- Bestimmen von Nutzerinteraktionen mit dem System vor der unternehmensweiten Einführung,
- Entwicklung oder Modifikation der Schulungsunterlagen, die auf den Erfahrungen der Piloteinführung basiert,
- exakte Bestimmung von Parametern für die Einführung in der Produktionsumgebung,
- Bestimmung des Supports, der für die neue Anwendung benötigt wird,
- Validierung funktionaler und nichtfunktionaler Anforderungen.

- ***Besetzung des Teams in der Phase der Einführung***

- *Change- and Configuration Management*:

Diese Rolle ist hauptsächlich an den Arbeiten für die Einführung einer Applikation beteiligt. Sie überwacht die Prozesse des *Change*, *Release* und *Configuration Management*. Mit Hilfe der *Configuration Management Database* stellt sie Informationen zur Verfügung, welche Nutzer, Gruppen oder Server in den Verteilungsprozess einzubeziehen sind. Diese Informationen werden auch als Basis zur Planung des Einführungsprozesses genutzt. Gleichzeitig erteilt sie Hinweise, wie das *Configuration Management* System zum Erstellen von Reports verwendet werden kann, um die Einführung zu überwachen.

- *Support*

Die Rolle *Support* ist an der Planung der Einführung beteiligt. Ziel ist es, dass die angeforderten Ressourcen verfügbar sind. Gleichzeitig werden alle *Incidents*, die in Vorbereitung bzw. während der Einführung auftreten, bearbeitet. Sie ist verantwortlich, dass der *Incident* und *Problem Management* Prozess die Einführung dieser Anwendung handhaben können.

- *Operations*

Auch die Rolle *Operations* ist an der Planung der Einführung beteiligt. Sie muss den Betrieb der Anwendung übernehmen, sobald die Einführung abgeschlossen ist. Ab diesem Zeitpunkt ist sie ebenfalls für tägliche Wartung zuständig.

- *Security*

Die neue Anwendung darf auf keinen Fall einen Sicherheitsstandard der Organisation unterlaufen. Dies muss die Rolle *Security* garantieren. Sie ist ebenfalls an diesen Planung interesiert. Sie muss verhindern, das so genannter *Malicious Code* in die Produktionsumgebung gelangen kann.

- *ICT Infrastructure Management*

Die Rolle *ICT Infrastructure Management* plant und überwacht die Kapazitäten und Verfügbarkeiten der IT-Umgebung, in der die Anwendung eingeführt wird. Sie stellt Untersuchungen an, ob die Anwendung ihr vereinbartes *Service Level Agreement* (SLAs) erreicht. Diese Rolle unterbreitet Vorschläge für Modifikationen der Anwendung oder der SLAs, falls das Service Level nicht erreicht wird.

3.7 Betrieb

Es besteht ein direkter Zusammenhang zwischen unzureichender Wartung eines IT-Systems und der Zahl von Ereignissen und Problemen, die im Rahmen des Betriebs auftreten. Es gibt eine Vielzahl von Aktivitäten während der Betriebsphase, die die Erfahrung der Betreiber und der Nutzer im Umgang der mit der Anwendung erhöhen.

Ein *Service Level Agreement* (SLA) ist nach [1] eine Definition der Erwartungen des Kunden an den *Service Provider*. Es ist eine Methode, um servicerelevante Fragen zwischen beiden Parteien festzuschreiben. Die Ausfertigung erfolgt schriftlich. Von großer Bedeutung ist dabei das Aufzeichnung von Leistungsparametern einer Anwendung. Das schließt einen ständigen Soll-/Istwertvergleich ein. Diese Arbeitsweise, verbunden mit dem Ergreifen geeigneter Maßnahmen, garantiert die Einhaltung der geschlossenen Vereinbarungen. Über die Auswertung der Messdaten lassen sich Informationen zur Stabilität und Fehlerrate berechnen. Beides sind Qualitätskenngrößen einer Anwendung. Ein wichtiger Bestandteil dieser Messungen ist das Erfassen der Nutzerzufriedenheit. Sie ist durch geeignete Kriterien zu beschreiben, die sich messen und aufzeichnen lassen.

Der hier beschriebene Prozess ist ausführlich in [4] beschrieben, wobei in dieser Publikation der Schwerpunkt auf dem Betrieb einer IT-Infrastruktur liegt. Nachfolgende Ausführungen ergänzen diese Betrachtungen unter dem Gesichtspunkt des Betriebs von Applikationen.

- ***Tägliche Wartungsaktivitäten zur Einhaltung des Service Levels***

Tabelle 3.3: Beispiele von zyklischen Wartungsaktivitäten für Applikation (nach [1])

Zeitintervall der Durchführung	**Beispiele**
täglich	▪ Verfügbarkeit des Betriebspersonals, ▪ Einsehen (*Review*) der täglichen Problemberichte und Notfalländerungen, ▪ Ausführen von Programmen, die die Konsistenz von Datenbanken prüfen, ▪ Kontrolle der Datenbank, in die Server ihre Fehlermeldungen protokollieren, ▪ Überwachung des Netzwerks, ▪ Statusüberwachung der Clientrechner, ▪ Statusüberwachung der Server und des Service, den sie bereitstellen, ▪ Überwachung von Ereignisprotokollen auf Schlüsselservern, ▪ Überwachung der Systemleistung und Systemverzeichnissen, ▪ Durchführen von Sicherheitsbackups auf jedem Server;
wöchentlich	▪ Pflege des *Change Management Review Boards*, ▪ Durchführen von Gruppenbesprechungen, ▪ Informationsaustausch mit den Entscheidungsträgern der Geschäftsleitung, ▪ Kontrolle der Systemverzeichnisse auf Servern, ▪ Erhöhung der Datenbankleistung, ▪ Erstellen von Managementreports, ▪ Überwachung von Filesystemen;
monatlich	▪ Zusammenstellen von monatlichen Statusreports, ▪ Kontrolle des Systemzustandes bzw. der Systemleistung, ▪ Verbesserung der Systemleistung, ▪ Kontrolle, welche Nutzer auf die Server zugegriffen haben, ▪ Kontrolle, das die Backups zur Servicewiederherstellung ständig verfügbar sind;
im Bedarfsfall (ad hoc)	▪ Veranstaltung von Meetings (*Brainstorming),* ▪ Erstellen von Fehlerberichten (*Troubleshooting Reports*), ▪ Analysen zur Verbesserung von Prozessen, ▪ Einsehen von Statusberichten zur Sicherheitsverletzung, ▪ Verbesserung von Sicherheitsfunktionen;

Zur Vorbeugung von Serverausfallzeiten können eine Vielzahl von Aktivitäten beitragen. Sie lassen sich klassifizieren in Aktivitäten, die täglich, wöchentlich, monatlich oder im Bedarfsfall (*ad hoc*) auszuführen sind. In diesem Zusammenhang ist ebenfalls festzulegen, wer sich für die Ausführung verantwortlich zeichnet bzw. innerhalb welchem Zeitrahmens diese Arbei-

ten abzuschließen sind. Der Inhalt von Tabelle 3.3 ist [1] entnommen und zählt zyklische Wartungsarbeiten auf.

Zustandsüberwachung von Anwendungen

Eine entscheidende Frage ist die Wiederherstellung von Anwendungen nach Systemfehlern. Der Zustand einer Anwendung wird von mehreren Komponenten bestimmt. Im einzelnen sind dies (vgl. [1]):

- die Anwendung selbst,
- Konfigurationen der Server,
- Logfiles,
- Systemdateien,
- Konfiguration, die die Nutzer betreffen,
- Nutzerdaten,
- Netzwerkverbindungen.

ITIL empfiehlt, das Wiederherstellen der Zustandes einer Anwendung ständig zu prüfen. Ein ideales Zeitpunkt für diese Aktivitäten ist jeder Versionswechseln, mindestens jedoch einmal aller 6 Monate. Weiterführende Ausführungen zu diesem Thema befinden sich in [2].

- ***Der Nutzen einer Anwendung***

Es wurde schon ausgeführt, das in der Betriebsphase die Leistungsdaten einer Anwendung ständig gemessen werden sollen. Folgende Fragen sind dabei zu berücksichtigen (vgl. [1]).:

- Sind die Erwartungen der Nutzer in dem Umfang erfüllt, wie sie vereinbart wurden?
- Stimmt das gemessene *Service Level* mit dem überein, das in den Dokumentationen festgeschrieben ist?
- Bewegen sich die Betriebskosten im vorgesehenen Limit?
- Unterstützt die Applikation die vorgesehene Geschäftsfunktion?

Die Ergebnisse sind gemeinsam mit Nutzerbefragungen in *Management Reports* festzuhalten. Deren Inhalt könnte bestehen aus:

- Anzahl und Typ von *Incidents* nach Versionen einer Applikation geordnet,
- einer Liste von Problemen ihrer Priorität entsprechend geordnet,
- einer Liste von Nutzer, die die meisten *Incidents* verursacht haben sowie
- einer Liste von Betriebsfehlern, die einen neuen Lösungsansatz erfordern.

Die Applikationsentwickler sollen diese Berichte in regelmäßigen Abständen erhalten. Forschungen haben gezeigt, dass 80% aller Probleme durch die Applikationen verursacht werden.

- ***Besetzung des Teams in der Phase des Betriebs***

Die nachfolgend aufgeführten Rollen garantieren den Betrieb von Applikationen in einer IT-Umgebung, die sich im Produktionsbetrieb befindet.

- *Change- and Configuration Management*:

Die Rolle *Change- and Configuration Management* stellt in dieser Phase sicher, das Applikationen stets mit der korrekten Version betrieben werden. Zugleich ist sie dafür verantwortlich, dass die Anwendungen im Produktionsbetrieb richtig konfiguriert sind. Betriebsanleitungen, die das Betreiberteam nutzt, sind auf einem aktuellen Stand zu halten. Das *Change- and Configuration Management* muss eng mit der Rolle *Operations* zusammenarbeiten, um anstehende *Releases* zu planen. Dies schließt eine Diskussion aller Probleme, die im Zusammenhang mit einer neuen Softwareversion stehen, ein. Im Fall einer fehlgeschlagenen Einführung, liegt es in der Verantwortung des *Change- and Configuration Managements*, dass die Betreiber eines IT-Systems die letzte funktionsfähige Version nutzen, um einen Service wiederherzustellen.

- *Support*

Die Rolle *Support* arbeitet eng mit der Rolle *Operations* zusammen. Sie zeichnet jeden Störfall während des Betriebes auf und informiert die Systembetreiber über mögliche Beeinflussungen.

- *Operations*

Dieser Rolle fällt die Hauptaufgabe innerhalb der Betriebsphase zu. Sie organisiert den tägli-

che Betrieb. Dazu zählt die Datenbank- und Systemadministration für Applikationen und IT-Systeme. Sie arbeitet Pläne für täglich wiederkehrende Prozesse aus. Darunter fällt die Datensicherung und –speicherung, das Ausgabemanagement, die Systemüberwachung oder das Print- und Fileservermanagement. Es liegt in der Verantwortung der Rolle *Operations*, dass die für die jeweilige Anwendung gültige Betriebsanleitung eingehalten wird.

- *Security*

Diese Rolle führt in periodischen Zyklen und zum Teil ohne Vorankündigung Prüfungen der Einhaltung von Sicherheitsverfahren und –bestimmungen durch. Eine Basis bilden dabei die Daten, die in den Sicherheits-Logfiles gespeichert sind.

- *ICT Infrastructure Management*

Das *ICT Infrastructure Management* konzentriert sich in dieser Phase hauptsächlich auf das Sammeln von Informationen, die die Kapazität und Verfügbarkeit des IT-Systems betreffen. Diese Rolle ist für den *Third-Level-Support* zuständig, falls komplizierte *Incidents* dies erfordern. Sie ist an der Aufzeichnung von Betriebsdaten interessiert, die zu einer Verbesserung des IT-Service in der anschließenden Optimierungsphase herangezogen werden können.

3.8 Optimierung

Jedes kommerzielles Softwarepaket, das sich im Produktionsbetrieb befindet, ist in regelmäßigen Abständen einer Prüfung (*Review*) zu unterziehen. Im Mittelpunkt steht dabei die Frage, ob Modifikationen notwendig sind oder einer Weiterbetrieb ohne Änderungen möglich ist.

- ***Der Prozess des Application Reviews***

Folgende Voraussetzungen können diesen Prozess anstoßen (vgl. [1]):

- eine festgesetzte Zeitspanne ist seit der letzten Prüfung vergangen (z.B. 12 – 24 Monate),
- das *Problem Management* hat einen Fehler in der aktuellen Version identifiziert,
- die Geschäftsanforderungen haben sich geändert oder
- die IT-Infrastuktur, auf der die Anwendung läuft, wurde modifiziert.

Die *Review-Prozesse* sind in Klassen (Nutzer, Geschäftsprozesse, Technologie) zu zusam-

menzufassen. Tabelle 3.4, die auf [1] basiert, nennt typische Frage, die innerhalb der jeweiligen Gruppe zu beantworten sind.

Tabelle 3.4: Fragen des Application Review Prozesses

Klasse	Fragestellung
Nutzer	▪ Werden die Bedürfnisse der Anwender in vollem Umfang berücksichtigt, und falls nicht, an welchen Stellen sind Änderungen notwendig? ▪ Ist die Bedienung der Applikation intuitiv oder Bedarf es ein hohes Maß, um die Nutzer zu schulen? ▪ Machen die Nutzer verhältnismäßig viele Fehler beim Anwenden dieser Software? ▪ Ist die Bedienung identisch mit anderen Applikationen, die in einer IT-Organisation zum Einsatz kommen?
Geschäftsprozesse	▪ Hat sich das Geschäftsziel geändert, und falls dies der Fall ist, wie unterstützt die Applikation diese neuen Geschäftsziele. ▪ Welche Auswirkung hat eine Veränderung in der Organisationstruktur auf die Funktionalität der Applikation? ▪ Berühren externe Veränderungen den Betrieb der Anwendung? ▪ Wurden die *Service Level Agreements* (SLAs) angepasst?
Technologie	▪ Ist die eingesetzte Technologie die Ursache, dass eine Anwendung ihr definiertes *Service Level* nicht erreicht? ▪ Gibt es technische Verbesserungen oder Änderungen die eine Modifikationen der Applikation notwendig machen? ▪ Wie gut lässt sich die Applikation betreiben? ▪ Werden die Vereinbarungen, die in den *Service Level Agreements* definiert sind, eingehalten? ▪ Wie werden betriebliche Ziele erreicht?

Ein *Application Portfolio* kann helfen, viele dieser Fragen zu beantworten. Im Ergebnis dieser Analyse muss eine der folgenden Alternativen gewählt werden:

1. keine Änderungen notwendig,
2. eine Modifikation ist erforderlich und in einem *Request for Change* (RFC) zu formulieren,
3. Stilllegen der Anwendung, da sie nicht mehr benötigt wird.

zu Punkt 1: keine Änderung notwendig

Falls die Prüfung mit dem Ergebnis endet, dass keine Änderungen an einer Applikation notwendig sind, besteht die einzige Aktion darin, dieses Ergebnis im *Application Portfolio*

schriftlich festzuhalten. Gleichzeitig wird der Termin der nächsten Prüfung vereinbart.

zu Punkt 2: Modifikationen erforderlich

Ist eine Änderung an einer Softwarekomponente erforderlich, muss ein Vorschlag erarbeiten werden. Er liefert eine ausführliche Beschreibung der Modifikationen und soll folgendes einhalten (vgl. [1]):

- die Definition der Änderungen,
- die Begründung durch die Geschäftsanforderungen,
- wer ist verantwortlich,
- eine Aufwandsabschätzung, um die Änderung durchzuführen,
- die Voraussetzungen für die Änderung,
- einen Termin, wann die Änderung abgeschlossen seien soll,
- wer sind die Eigentümer,
- die Erfolgskriterien für die Änderung.

Abhängig von ihrer Art sind die Nutzer in diese Optimierungsphase einzubeziehen. Sie können Details für neue Geschäftsanforderungen einbringen. Das Team, welches die Optimierung einer Anwendung durchführen muss, soll Zugang zu einer Testumgebung bekommen, damit die Verbesserung nicht selbst zu einer Fehlerquelle wird.

Der erarbeitete Vorschlag kann der Grund einer neuen Iteration im Lebenszyklus seien. Sie startet mit einem neuen Entwicklungsprozess. Es besteht zusätzlich die Möglichkeit der Erzeugung eines *Request for Change* (RFC) innerhalb des *Change Management Prozesses*.

zu Punkt 3: Stilllegung von Anwendungen

Falls im Ergebnis der Analyse feststeht, dass eine Anwendung nicht weiter benötigt wird, kommt es zur Ausfertigung eines entsprechenden Antrags. Er enthält einen Zeitplan, der das Entfernen der Anwendung aus der IT-Umgebung bestimmt. Falls es sich um eine komplexe Anwendung handelt, sind u.U. Werkzeuge zu entwickeln, die die Ablösung leichtern bzw. beschleunigen. Diese Neuentwicklung soll wieder dem Lebenszyklus-Modell entsprechen. Alle Komponenten der Anwendung sind aus der IT-Umgebung zu entfernen. Dazu zählen ausführbare Programmdateien, Konfigurationsdateien, Quellcode und Verweise auf die Anwendung selbst.

- ***Besetzung des Teams in der Phase der Optimierung***

- *Change- and Configuration Management*:

Das *Change- and Configuration Management* stellt Daten zur Bewertung der Anwendung bereit. Dies betrifft insbesondere:

- Informationen zum *Change Management Prozess*,
- *Release-Informationen*,
- die *Configuration Management Database*.

Sie überwacht die Ausführung der Arbeiten.

- *Support*

Die Rolle *Support* bringt Probleme in den Optimierungsprozess ein, die im Produktionsbetrieb der Anwendung entstanden sind. Sie berät das Optimierungsteam zu Fragen, die bei Ausführung der Optimierung auftreten können.

- *Operations*

Diese Rolle beteiligt sich am Bewertungsprozess durch Erkennen von Schwächen, die beim Betrieb der Anwendung aufkommen. Viele Probleme, die die Zuverlässigkeit einer Anwendung beeinflussen, stehen mit einer einfachen Bedienung einer Applikation in Zusammenhang. Zusätzlich nimmt der Wartungsaufwand (täglich, wöchentlich, monatlich, usw.) Einfluss auf die Effizienz und Verfügbarkeit. Im Rahmen der Optimierung ist deshalb ein Vergleich des Zeitaufwandes für die Wartung mit anderen Anwendungen einer IT-Organisation anzustellen.

- *Security*

Die Rolle *Security* bringt Informationen zur Verbesserung der Sicherheit in die Diskussion ein. Dies betrifft die Anwendung, die Integrität der Geschäfts- und Nutzerdaten und die Erhöhung des Schutzes anderer betroffener Aktivposten. Diese Rolle ist besonders an den Vorschlägen interessiert, die im Rahmen der Optimierung vorgeschlagen werden. Eine Veränderung oder Verbesserung darf nicht die unternehmensweiten Sicherheitsrichtlinien verletzen.

- *ICT Infrastructure Management*

Diese Rolle nimmt eine herausragende Stellung innerhalb der Optimierungsphase ein. Sie erstellt eine Vielzahl von Expertisen zu Fragen, die das Servicemanagement betreffen. Dies schließt Vorschläge:

- zum *Capacity Management*,
- zur Erhöhung der Verfügbarkeit durch Designmaßnahmen,
- mit welchen Technologien und Werkzeugen eine Optimierung der Anwendung durchgeführt werden soll,

ein.

3.9 Abschließende Betrachtungen

Vorteile, die sich aus einer Ausrichtung von Anwendungsentwicklung und Servicemanagement ergeben, sollen nachfolgend genannt werden (vgl. [1]).

3.9.1 Vorteile, die sich aus dem *Application Management* ergeben

Ein Verständnis der Abhängigkeiten zwischen Applikationsentwicklung und Servicemanagement führt zu ausgewogenen Designentscheidungen in frühen Phasen des Lebenszyklus. Die integrierte Planung einer Anwendung senken die Kosten (*Total Cost of Ownership* (TCO)). Eine umfassende Analyse dieser Kosten ermöglicht es einer IT-Abteilung, zu einer exakten Kenntnis der IT-Kosten zu gelangen. Die ist für die Bewertung im Hinblick auf die Geschäftsunterstützung eines Services sehr hilfreich. Betrachtungen zu Kapazitätsanforderungen sind in einer frühen Phase gegeben. Sie können gleichzeitig mit der Applikationsentwicklung vollzogen werden. Die Verwaltung der Einführungs-, Test und Produktionsumgebung wird vereinfacht. Damit verbunden ist eine Kostensenkung für ihre Definition, Realisierung, Veränderung und Ablösung. Projektmanager für die Anwendungsentwicklung und des Servicemanagements können phasenübergreifende Indikatoren entwickeln und dies fortlaufend überwachen. Referenzarchitekturen und/oder Frameworks können entwickelt werden, die Fragen der Servicedefinition und Serviceerbringung in Betracht ziehen. Applikationsentwickler können in ihrem Design die Aspekte des Servicemanagements berücksichtigen.

Der zeitliche Rahmen dieser Seminararbeit lässt es nicht zu, diese Thesen zu bewerten.

3.9.2 Probleme, die bei der Einführung auftreten können

Für die erfolgreiche Einführung des *Application Managements* in eine IT-Organisation müssen einige Voraussetzungen erfüllt seien. In der Einleitung wurde darauf hingewiesen, dass dieses Managementverfahren Schnittstellen zu anderen Disziplinen des IT-Service-Managements besitzt und ebenso die Entwicklung einer Applikation tangiert. In [1] wird ausdrücklich darauf hingewiesen, dass es nicht möglich ist, ein prozessübergreifendes IT-Management einzuführen, solange die individuellen Prozesse keine eigenen Mess- und Steuermechanismen aufweisen. Für das *Application Management* von außerordentlich großer Wichtigkeit ist ein funktionierendes:

- *Configuration Management*,
- *Software Configuration Management* sowie,
- *Release Management*.

Die Implementierung von *Application Management* darf nicht mit der Einführung anderer Servicemanagementprozesse (*Service Level Management*, Testen) verwechselt werden. Eine IT-Organisation, die sich bei der Einführung nicht exakt an [1] orientiert, muss mit sehr hohen Kosten, verbunden mit einem erheblichen Zeitaufwand, rechnen.

Nach einer erfolgreichen Einführung des *Application Managements* innerhalb einer IT-Organisation, bestehen die Abhängigkeiten zu anderen Serviceprozessen weiter. Verbesserungen eines Prozesses müssen stets in Übereinstimmung mit dem *Application Management* erfolgen. Einige Schlüsselaktivitäten, die [1] nennt, werden nachfolgend kurz vorgestellt. Sehr detailliert sind diese Prozesse in [2] und [4] beschrieben.

3.9.3 Der *Change Management Prozesses*

Zu diesem Punkt befindet sich in [1] folgende Zusammenfassung.

Es ist ein *Change Manager* einzusetzen, der einen umfassenden Überblick über die Geschäftslage hat. Er soll die Kraft und Fähigkeit besitzen, eingebrachte Änderungsanträge zu

prüfen, zurückzuweisen oder sein Veto einzulegen. Dies ist insbesondere wichtig, falls das Aufkommen der Änderungsanträge die Fähigkeiten der Organisation übersteigt, diese zu realisieren. Die Veröffentlichungen einer *Change Management Strategie* stellt folgende Punkte klar:

- die explizite Verantwortlichkeit für die Bewilligung der Änderung,
- die Bewertung der Änderung, Festsetzung von Prioritäten und Kategorien,
- Strategien für die Implementierung von kleinen bzw. trivialen Änderungen, langfristige Aktivitäten, Änderungen mit großen Auswirkungen, Notfallmaßnahmen,
- eine Strategie für die zeitliche Abfolge der *Releases*,
- das Testen der Änderung sowie Implementierungsrichtlinien,
- das nachträgliche Prüfen einer Implementierung, Berichterstattung sowie die Erfolgskontrolle der Änderungen.

In einem Änderungsplan sind die Risiken und Wirkungen der Änderungen einzuschätzen. Alle *Request for Changes* (RFCs) sollen zentral registriert werden. Beide Parteien, das IT-Management sowie die Geschäftsleitung, müssen jeden RFC formal genehmigen. Das *Change Advisory Board* (CAB) spielt eine aktive Rolle innerhalb des *Change Managements*. Es soll Änderungen hinsichtlich ihres Einflusses auf die IT-Infrastruktur, den angebotenen Service und den benötigten Ressourcen bewerten. In Untersuchungen, die nach der Einführung einer Änderung stattfinden, ist zu klären, ob:

- die ausgeführte Änderung den gewünschten Effekt gebracht haben und keine anderen Beeinflussungen aufgetreten sind,
- die Nutzer mit den Ergebnissen zufrieden sind,
- der Ressourcenverbrauch den vorangegangenen Planungen entspricht.

Nach dem Abschluss der Arbeiten sind die Aufzeichnungen zu aktualisieren, ob das Einbringen der Änderung zufrieden stellend abgeschlossen ist oder abgebrochen wurde. Jeder offene Punkt ist zu dokumentieren. Eine Aufzeichnung der Abfolge aller Änderungen und eine rückblickende Bewertung einer vorgeschlagenen Änderungen in Bezug auf die *Service Level Agreements* (SLAs) ist durchzuführen. Eine Zusammenstellung aller Änderungen ist in regelmäßigen Abständen an die Kunden und das Servicemanagement weiterzuleiten.

3.9.4 Testprozesse

Für die Weiterentwicklung von Applikationen von besonderer Wichtigkeit ist die Durchführung von Regressionstests. Dies stellt sicher, dass Verbesserungen oder Wartungsarbeiten an einer Anwendung deren Eigenschaften in anderer Hinsicht nicht verschlechtern. In [1] werden diesbezüglich folgende Empfehlung gegeben.

Es sollen nur (getestete) Standardkomponenten Verwendung finden. Zusätzlich können Prototypen zur Elimination von Fehlern beitragen, die während der Realisierung von Änderungen auftreten. Durch eine gezielte Planung von Unit-, Integrations- und Systemtests ist die Funktionalität, Supportfähigkeit, Verfügbarkeit, Zuverlässigkeit, Sicherheit und Wartbarkeit einer Anwendung zu erhalten bzw. zu garantieren. Die Testumgebung ist komplett von der Produktionsumgebung oder anderen Testfeldern zu isolieren. Ist nur ein partieller Test in der Testumgebung möglich, ist eine Beeinflussung anderer Anwendungen in der Produktionsumgebung während der Tests zu minimieren

3.9.5 Der *Release Management Prozesses*

Das *Application Management* hängt aus folgenden 2 Gründen von dem *Release Management* ab. Die Zahl der Änderungswünsche kann die verfügbaren Ressourcen übersteigen, um diese zu bewältigen. Eine Konsequenz hieraus ist, dass ein gesteuerter *Release*-Kreislauf eine effiziente Möglichkeit ist, um Änderungen an einer Anwendung zu verwalten. Neben diesem ersten Grund existiert ein weiterer. Damit ausreichend Ressourcen verfügbar sind, ist eine größere Zahl von Änderungen zu einer *Release* zusammenzufassen.

Es muss eine Balance zwischen einer schnellen Antwort auf einen *Request for Change* (RFC) eines Anwenders und einem effektiven Einsatz der Ressourcen, die für Wartungsarbeiten verfügbar sind, gefunden werden. Das Zusammenfassen von Änderungen zu einer *Release* ermöglicht diesen effektiven Ressourceneinsatz. Andererseits ist es nicht akzeptabel, wenn ein Anwender eine unverhältnismäßig lange Zeit auf eine modifizierte Programmfunktion warten muss, die sich aus einer geänderten Geschäftsanforderung ergibt.

Die folgenden Empfehlungen gibt [1] für eine Verbesserung des *Release Management Prozesses* in einer Produktionsumgebung.

Das *Release Management* soll für große oder kritische Hardware- und Softwareeinführungen benutzt werden. Ein zusammenhängender Satz von Änderungen soll zusammengefasst werden und, falls dies möglich ist, als Teil oder komplett in einer *Release* implementiert werden. Ein Zeitplan, der die Abfolge von *Releases* regelt, ist zu erarbeiten und mit der Geschäftsleitung abzustimmen. Für jede *Release* regelt ein Dokument die auszuführenden Tests und die zugehörigen Akzeptanzkriterien. Die gesamte Software, alle Parameter, Testdaten und Laufzeitkomponenten einer *Release* sind durch das *Configuration Management* zu kontrollieren. Deinstallationsprozeduren müssen für jede Release getestet werden. Die Akzeptanz einer Release ist in einer kontrollierten Umgebung zu analysieren. Metriken sind einzuführen, um die Wirksamkeit des *Release Management Prozesses* zu überwachen. Ein Beispiel ist das Auftreten von Störfällen (*Incidence*) bei der Einführung einer *Release*.

4. Überwachen von Anwendungen

4.1 Messung der Antwortzeit / *Application Response Measurement*

IBM und *Hewlett Packard* haben 1996 ein Verfahren vorgestellt, mit dem sich die Antwortzeit von Transaktionen messen lässt. Dies führte zur Spezifikation der ARM-API 1.0 (*Application Response Measurement*). Die Weiterentwicklung dieser Softwareschnittstelle hat die *Open Group* übernommen, ein Zusammenschluss von Firmen die Anwendungssoftware, Datenbanken und Systemmanagementplattformen entwickeln bzw. diese Produkte anwenden.

Auf dem System, auf dem eine Transaktionen überwacht werden soll, wird ein Softwareagent installiert. Er implementiert die *ARM-API* und ist für die Zeitmessung der Transaktion verantwortlich. Im Hauptprogramm sind an geeigneter Stelle die Dienstaufrufe der *ARM-API* zu platzieren, um die Zeitmessung zu starten bzw. zu stoppen. Tabelle 4.1 gibt die Funktionsaufrufe an der ARM-API 2.0 an (vgl. [15]).

Tabelle 4.1: Funktionen der ARM-API 2.0

Funktion	Erläuterung
arm_init()	▪ einmaliger Aufruf bei der Initialisierung (Programmstart) des Hauptprogramms, um damit der Agent die entsprechende Umgebung einrichten kann;
arm_getid()	▪ Aufruf, falls zwischen verschiedenen Transaktionsklassen zu unterscheiden ist; Diese Funktion liefert einen eindeutigen Identifikator, der der Funktion *arm_start()* zu übergeben ist. Falls diese Differenzierung nicht notwendig ist, reicht ein einmaliger Aufruf zusammen mit *arm_init()* aus.
arm_start()	▪ zeigt dem Agenten den Start einer Transaktion an; Diese Funktion liefert einen Identifikator zurück, der bei Aufrufen von *arm_update()* bzw. *arm_stop()* benutzt werden muss.
arm_update()	▪ optionaler Funktionsaufruf, der bei längeren Transaktionen aufgerufen werden kann (vgl. *Keep-Alive-Signal*)
arm_stop()	▪ signalisiert dem Agenten das Ende einer Transaktion; Abbruch der Zeitmessung
arm_end()	▪ einmaliger Aufruf bei Beendigung des Hauptprogramms; Durch diesen Aufruf gibt der Agent reservierte Speicherbereiche frei.

Dieses Messverfahren ist auf Servern und Clients einsetzbar, eine gegenseitige Überwachung ist damit möglich (vgl. Bild 4.1 und Bild 4.2). Die gewonnenen Informationen können mit einem übergeordnetem Leitsystem (*Enterprise Management System*) ausgewertet werden. Momentan aktuell ist die Version 4.0, die im Oktober 2003 veröffentlicht wurde. Sie besitzt

zusätzliche Methoden, um eine Transaktion genauer zu klassifizieren. Dies betrifft beispielsweise das Erfassen von Elementen, die von einem Server übertragen wurden. Zudem lässt sich der Grund ermitteln, der zum Abbruch einer Transaktion geführt hat. Es können Zähler, Metriken, numerische Identifikatoren und Strings vereinbart werden. Zusätzlich wurden die Namen der Methoden der API modifiziert. Das Prinzip ist aber unverändert geblieben.

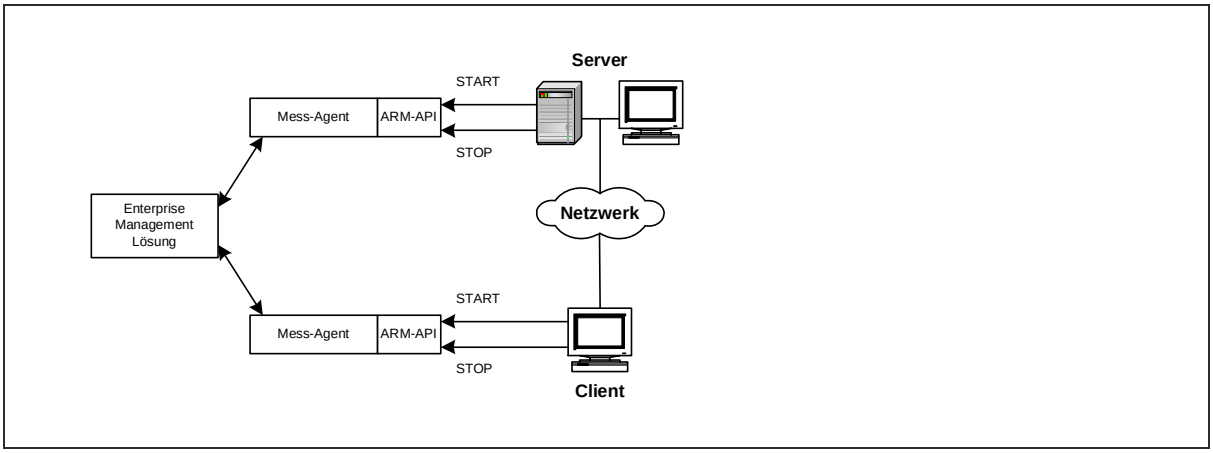

Bild 4.1: Prinzip des Application Response Measurement (vgl. [15])

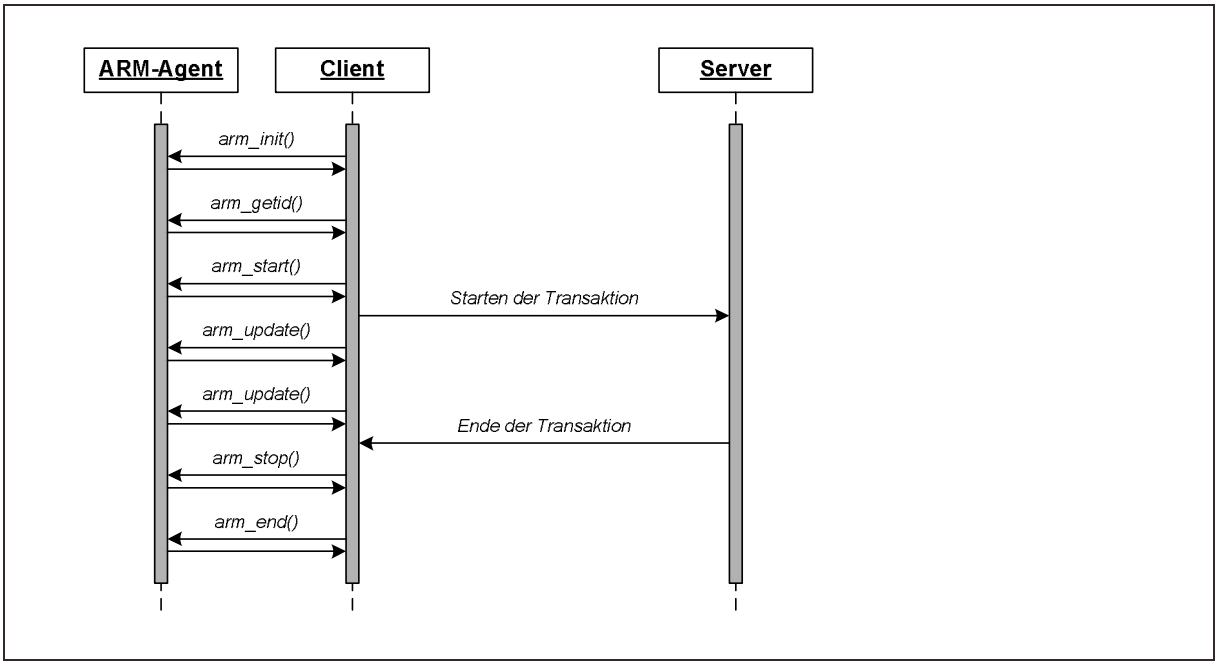

Bild 4.2: Sequenzdiagramm für den Ablauf von Transaktionen, die Funktionsaufrufe der ARM-API 2.0 integrieren

Die Version 4.0 dieser API ist nicht abwärtskompatibel. Referenzimplementierungen in C++ bzw. Java stehen unter folgender URL zur Verfügung:

http://www.opengroup.org/tech/management/arm .

4.2 Überwachung des Serverbetriebs

Das Management verteilter IT-Systeme wird zur Zeit durch das Internetmanagement dominiert. Es handelt sich dabei um eine herstellerübergreifende Managementstandard. Kennzeichnet für diese Managementarchitektur ist die Definition von Rollen, Agenten und Manager. Ein Agent ist eine Softwarekomponente, die auf der Netzkomponente installiert ist. Ein Manager fragt in regelmäßigen Abständen die Managementinformationen, die der Agent der Komponente verwaltet, ab.

Charakteristisch für diese Architektur ist ein Informationsmodell, das auf *Internet-MIBs* (*Management Information Base*) basiert (vgl. [5]). Hersteller von Netzkomponenten (z.B. Hubs, Switches, Bridges, Router) integrieren diese MIBs in ihre Geräte. Die darin enthaltene Information bestimmen die Produzenten größtenteils selbst. Eine kleine Teilmenge ist in der *Standard-MIB-II* definiert, die jedes Gerät implementieren muss. Standards zur Struktur von Managementinformationen sind in RFCs (*Request for Comment*) festgeschrieben. An dieser Stelle wird darauf hingewiesen, dass ein Unterschied zur Bedeutung der Abkürzung RFC besteht, den ITIL benutzt. Kommerzielle Managementplattformen (*HPOpenView, Tivoli*) bieten Konfigurationsmechanismen an, diese hinterlegten Informationen aus den Geräten zu importieren und zum Aufbau einer eigenen Netzdatenbank zu nutzen.

Tabelle 4.2: Informationsquellen für die Überwachung von WWW-Servern (vgl. [18])

Management Information Base	Basis für RFC2039			derzeitiger Stand		
	Fassung	Ausgabe	Status	Fassung	Ausgabe	Status
Standard-MIB-II	RFC1213	März 1991	full	RFC1213	März 1991	full
Host Resource MIB	RFC1514	Sept. 1993	proposed	RFC2790	März 2000	draft
Netw. Serv. Monit. MIB	RFC1565	Jan. 1994	proposed	RFC2788	März 2000	proposed
Application MIB	in Bearbeitung	-	-	RFC2287	Febr. 1998	proposed
				RFC2564	Mai 1999	proposed

Für das Überwachen von WWW-Servern hat die *Internet Society* den RFC2039 veröffentlicht. Dieser informelle Standard entstand im Ergebnis des 35. Treffens der *Internet Engineering Task Force* (IETF) im November 1996 (vgl. [18]). Er schlägt die Integration der in Tabelle 4.2 aufgeführten MIBs vor. Einige der Standards, auf die sich RFC2039 bezieht, wurden

zwischenzeitlich weiterentwickelt und durch neue RFCs ersetzt.

Für die Zustandsüberwachung der Hardware eines WWW-Servers sind die *Standard-MIB-II* sowie die *Host-Resource-MIB* heranzuziehen. Die *Network-Service-MIB* und die *Applikation-MIBs* dienen der Verwaltung der Softwarekomponenten des Servers. Beim Zusammenfügen aller dieser Informationen entsteht eine umfangreiche Liste von Attributen für das Servermanagement.

Insbesondere äußert RFC2039 den Wunsch, Informationen, die ein Web-Server in *Logfiles* schreibt, auf MIB-Variablen abbilden zu können. Dies eröffnet die Möglichkeit, um Betriebszustände des Servers mit Managementplattformen in gleicher Art und Weise zu erfassen, wie dies für andere Netzkomponenten schon der Fall ist. Damit wird ein Beitrag zum automatischen Betrieb eines WWW-Servers geliefert. Andererseits sind die Sicherheitsrisiken, die sich mit dem Einsatz des *Simple Network Management Protocol* (SNMP) in der Version 1 bzw. Version 2 ergeben (unverschlüsseltes Übertragen von Passwörtern für den Zugriff auf die MIB), kritisch zu bewerten.

4.3 Das Java-Management

4.3.1 Instrumentierung der *Java™ Virtual Machine* – JSP-174

Eine *Java Virtual Machine* (JVM) muss auf jedem Rechner installiert sein, auf dem kompilierte Java-Programme (Bytecode) ablaufen. *Sun* veröffentlicht in [11] eine Spezifikation zur Zustandsüberwachung der JVM. Weitere grundsätzliche Charakteristiken dieser Managementschnittstelle nennt Tabelle 4.3.

In [11] werden weitere Daten und Metriken spezifiziert, um einen Zustandsüberwachung der *Java Virtual Machine* zu ermöglichen. Es handelt sich dabei um Informationen über:

- das Betriebssystem, auf dem die JVM installiert ist,
- den Verbrauch von Betriebssystemressourcen durch die JVM,
- die *Java Virtual Machine* selbst (Laufzeitsystem, Compilereigenschaften),
- die Ausführung und Synchronisation von Teilsystemen, d.h. Kontrolle und CPU-Auslastung von *Threads*,

- das *Class-Loader-System* und
- die Ausnutzung des Speichers (*Heap*, *Non-Heap*-Speicher, *Garbage Collection*).

Tabelle 4.3: Charakteristiken der Java Management Extention (JVM) (vgl. [11])

Merkmal	Erläuterung
Überwachungsmodell	▪ sofortiges Alarmieren von einmalig auftretenden Ereignissen (*low frequency events*), ▪ Einrichten von Zählern für ständig wiederkehrende Systemzustände (*high frequency events*);
Leistungsverbrauch	▪ Der zusätzliche Leistungsverbrauch, der durch JMX hervorgerufen wird, soll nicht über 1 Prozent ansteigen (gemessen mit Standard-Benchmark-Programmen, z.B. *specJBB*, *specJVM*, *ECPerf*).
Ereignisgenerierung	▪ Die Ereignisgenerierung bzw. die Zähler können je nach Bedarf freigegeben, gesperrt bzw. zurückgesetzt werden.
Wiederanlauf	▪ Ein Wiederanlauf der JVM darf für die Freigabe, das Sperren, einem Rücksetzen sowie dem Aufsetzen eines Monitors nicht erforderlich werden.
Erkennen von Speicherengpässen	▪ Dies erlaubt einer Anwendung Speicherengpässe einer JVM zu erkennen.
Erkennung von *Deadlocks*	▪ Einbinden von Mechanismen, die es einer Anwendung oder einem Managementagenten erlauben, *Deadlocks* von Anwendungen zu erkennen.
SNMP-Unterstützung	▪ Eine SNMP-MIB-Definition soll ein SNMP-Applikationmanagement ermöglichen.
Unterstützung von Managementschnittstellen	▪ Unterstützung der *Java Management Extension MBeans* ▪ Ein Mechanismus soll es JVM-Anwendern erlauben, Erweiterungen für plattformspezifische Managementdaten zu integrieren.

4.3.2 Die *Java Management Extension (JMX)*

Mit der *Java Management Extension* (JMX) definiert *Sun Microsystems* eine Management-Architektur, die in Bild 4.3 dargestellt ist. Sie besteht aus 3 Ebenen, dem:

- *Instrumentation Level*,
- *Agent Level* sowie
- *Manager Level*.

Objekte, die überwacht werden sollen, sind auf der Instrumentierungsebene durch *MBeans* (oder *Managed Beans*) zu repräsentieren. Sie können Anwendungen, Geräte oder Software-

implementierungen eines Service oder einer Vorgabe (*Poilcy*) modellieren. *MBeans* sind Java-Objekte, die der *JavaBeans*-Spezifikation entsprechen.

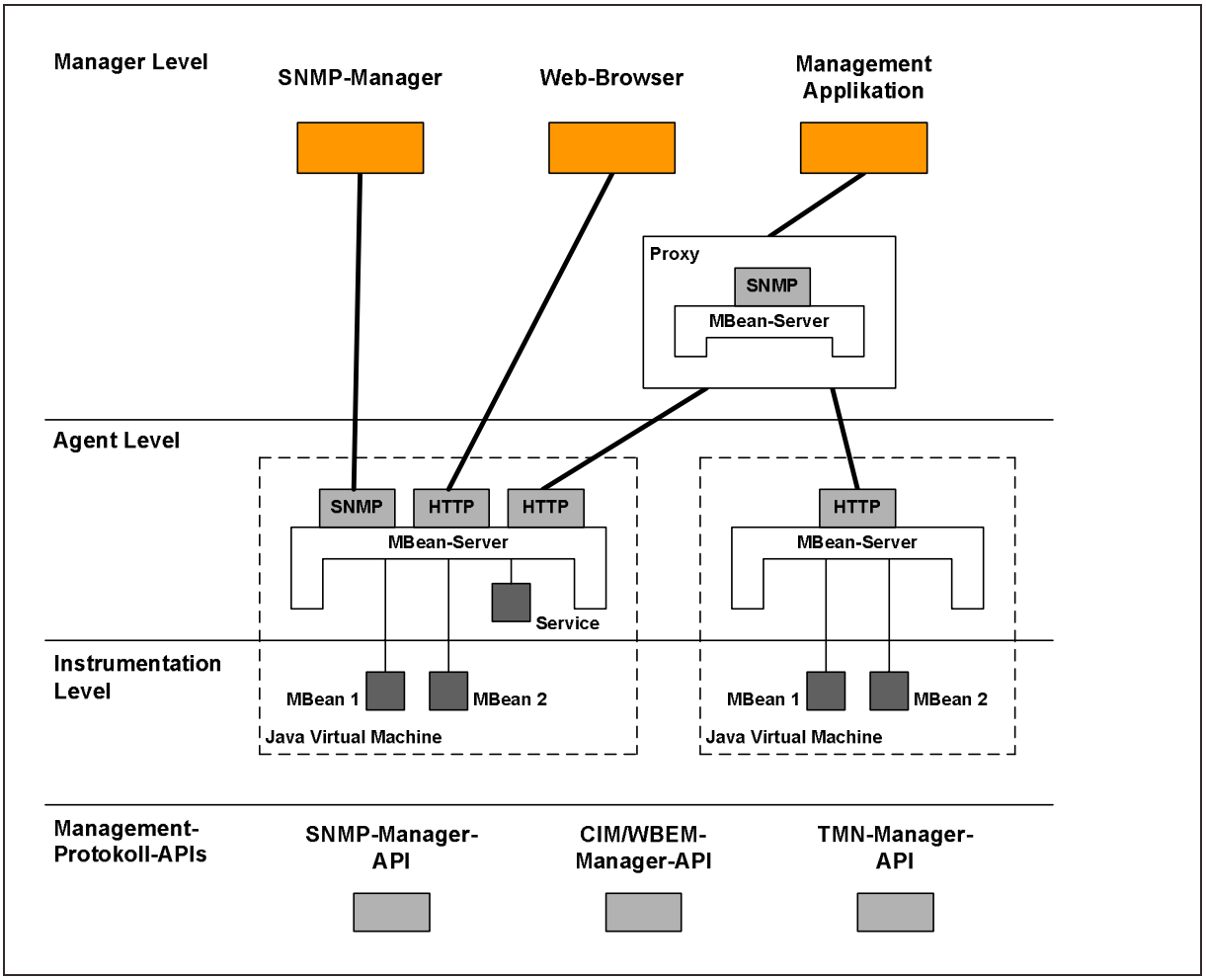

Bild 4.3: JMX-Management-Architektur (vgl. [12])

Ein *JMX-Agent* läuft innerhalb einer *Java Virtual Machine* (JVM). Er vermittelt als Bindeglied zwischen den *MBeans* (*Instrumentation Level*) und den Managementanwendungen (*Manager Level*). *MBean* und *JMX-Agent* stehen in einer Client-Server-Beziehung zueinander. Der Agent ist aus einem *MBean-Server* und mindestens einem Protokolladapter zusammengesetzt. Er kann ebenfalls Services enthalten, die durch *MBeans* abgebildet werden. Alle *MBeans* müssen sich beim *MBean-Server* registrieren und können dynamisch hinzugefügt bzw. entfernt werden. Über die Protokolladapter können die Agenten durch Managementanwendungen angesprochen werden. Sie konvertieren die Funktionalität der *MBeans* in die Repräsentation des gegebenen Managementprotokolls bzw. in ein anderes Informationsmodell (z.B. SNMP).

JMX-Manager ermöglichen es Managementanwendungen mit den *JMX-Agenten* zu interagie-

ren und Informationen auszutauschen. Ein Manager kann eine größere Anzahl an Agenten kontrollieren, so dass verteilte und komplexe Managementstrukturen realisierbar sind.

Die *Java Management Extension* bietet Schnittstellen zu folgenden Managementarchitekturen:

- Internetmanagement,
- *web-based* Enterprise Management,
- *Telecommunication Management Network*,
- Corba,
- LDAP.

Referenzimplementierungen stehen auf der Homepage, die *SUN* für die Programmiersprache *Java* betreibt, zur Verfügung. Die URL lautet:

www.java.sun.com/products/JavaManagement .

Es sei noch darauf hingewiesen, dass das Verfahren zur Instrumentierung der *Java Virtual Machine* (JSP174) (vgl. Abschnitt 4.3.1), in die Spezifikation der JMX berücksichtigt wurde (vgl. [12]).

4.3.3 Das *Java Dynamic Management Kit* (JDMK)

Sun hat ein kommerzielles Framework (*Java Dynamic Management Kit* (JDMK)) entworfen, um die Entwicklung von Anwendungen, die auf der JMX-Spezifikation basieren, zu beschleunigen (vgl. [13]). Zusätzlich enthält es 2 separate Anwendungen (*proxygen*, *mibgen*), mit denen sich Protokollkonverter (*Proxies*) bzw. SNMP-Agenten entwickeln lassen.

Arbeiten am Lehrstuhl für Kommunikationssystem und Systemprogrammierung der Ludwig-Maximilians-Universität setzen sich mit der Managementarchitektur, die in diesem Abschnitt vorgestellt wurde, kritisch auseinander. Vor- und Nachteile werden diskutiert (vgl. [10]).

Hervorzuheben ist die Möglichkeit der Entwicklung eines SNMP-Agenten mit Hilfe eines *MIB-Compilers*. Er kann auch eine gegebene *Management Information Base* (MIB) in ein

konformes *MBean* umwandeln.

4.4 Definition von *Logfiles*

Es soll die Annahme gelten, dass auf dem Server der Web-Server *Apache* in Verbindung mit *Tomcat* zum Einsatz kommt (vgl. [6] und [7]). Obwohl auch der *Tomcat*-Server in der Lage ist, dem Client statische Seiten zu liefern, wird diese Aufgabe in der Praxis häufig von einem *Apache-Server* übernommen. *Tomcat* wird als *Apache*-Module konfiguriert und bekommt nur noch die Anfragen an *Servlets* und *Java-Server-Pages* (JSP-Seiten) zur Verarbeitung geliefert. *Servlets* sind Java-Programme, die auf Servern ablaufen und dem Client HTML-Code zurückgeben. In Gegensatz hierzu sind *Java-Server-Page* HTML-Dateien, die Java-Codestücke integrieren.

Der Web-Server *Apache* besitzt umfangreiche Möglichkeiten der Konfiguration von *Logfiles*. Tabelle 4.4 gibt einen Überblick (vgl. [6]).

Tabelle 4.4: Konfigurationsmöglichkeiten für Logfiles eines Apache-Web-Servers

Logfile	Erläuterung
ErrorLog	▪ Logfile, in dem der Server alle entdeckten Fehler protokolliert
TransferLog	▪ Logfile, in welchem alle Zugriffe auf die Web-Seiten mitgeschrieben werden
AgentLog	▪ Logfile, in das der Server alle *User-Agent-Header* eingehender Anfragen schreibt
LogLevel	▪ kontrolliert die Informationsmenge, die in der Datei *ErrorLog* festgehalten wird
LogFormat	▪ Festlegung der Informationen, die in die Log-Dateien aufgenommen werden

Das Standardformat dieser Log-Dateien entspricht dem *Common-Log-Format* (CLF). *Logfile*-Analysatoren (z.B. *wusage*, *www.boutell.com/wusage*) können es auswerten.

Das CLF-Format hat folgende Grundstruktur, die Tabelle 4.5 näher erläutert.

host ident authuser datum request status bytes .

Dieses Format kann durch Hinzufügen weiterer Informationen erweitert werden.

An dieser Stelle soll auf einen Widerspruch hingewiesen werden. Im Rahmen des Betriebs eines Web-Servers ist eine detaillierte Kenntnis der ablaufenden Vorgänge wünschenswert.

Andererseits wird in [6] hervorgehoben, dass die Leistungsfähigkeit des *Apache-Web-Servers* abnimmt, je präziser die einzelnen Aktivitäten protokolliert werden. Es besteht also ein direkter Zusammenhang zwischen dem Protokollieren von Ereignissen durch diesen Web-Server und seiner Reaktionszeit bei einer gegebenen Hardwarekonfiguration. Im Rahmen des *Application Managements* ist in der Phase *Design* dieser Widerspruch durch einen geeigneten Kompromiss zu lösen, der eine akzeptable Reaktionszeit des Servers garantiert und zum anderen die Informationen festhält, die für einen sicheren Betrieb notwendig sind.

Tabelle 4.5: Aufbau des Common-Logfile-Formats

Element	Erläuterung
host	▪ Hostname oder IP-Adresse des Clients
ident	▪ vom Client gelieferten Identitätsinformationen
authuser	▪ nach Anforderung eines passwortgeschützten Dokuments Angabe der Benutzer-ID
datum	▪ Datum der Anfrage im Format [*tag/monat/jahr:stunde:minute:sekunde zeitzonen-offset*]
request	▪ Request-Zeile des Clients in doppelten Ausführungszeichen
status	▪ Statuscode, der an der Client zurückgegeben wurde (vgl. HTTP-Protokollspezifikation)
bytes	▪ Anzahl der übertragenen Bytes

5. Das BMW-Referenz-Szenario

5.1 Einführung

Der zweite Teil dieser Seminararbeit untersucht das *BMW-Extranet* (vgl. [9]). Beim Zusammenfassen aller Informationen ergibt sich folgendes Bild.

Die BMW AG betreibt firmeneigene, abgeschlossene Netze, die auf den Internetprotokollen basieren (Intranets). Um die Zusammenarbeit zwischen Händlern, Zulieferern und IT-Partnern mit dem Mutterkonzern zu vereinfachen bzw. zu beschleunigen, erhalten sie Zugriff auf ausgewählte Dienste und Netzinfrastrukturen des BMW-Intranets. Die Partnerunternehmen bilden ein Extranet. Dabei handelt es sich um externe Netzwerke für eine abgeschlossene Benutzergruppe außerhalb von BMW. Der Zusammenschluss aller Intra- und Extranets bildet das weltweite BMW-Unternehmensnetzwerk, das in [9] als *Corporate Network* bezeichnet wird.

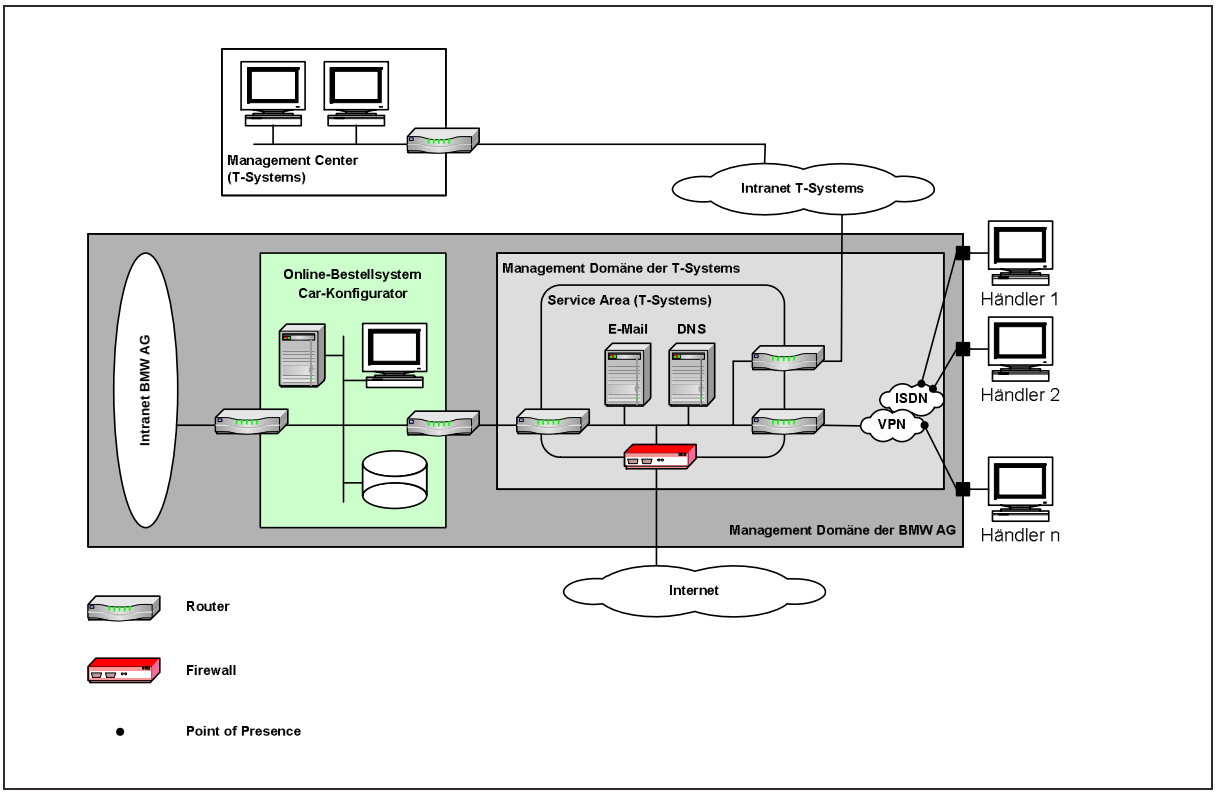

Bild 5.1: BMW-Extranet-Szenario (vgl. [9])

Das Telekommunikationsunternehmen *T-Systems* (eine Tochtergesellschaft der Telekom) stellt Netzinfrastruktur zur Verfügung, um die Händler an das für sie eingerichtete Händler-

Extranet anzubinden. Mit Hilfe dieser Netzkomponenten wird ein virtuelles privates Netzwerk (VPN) aufgebaut und den BMW-Händlern zur Nutzung zur Verfügung gestellt. Sie werden entweder über Standleitungen oder ISDN-Wählverbindungen mit einem *Point-of-Presence (POP)* verbunden (vgl. Bild 5.1).

Zusätzlich erbringt *T-Systems* im Auftrag der Händler Dienste. Diese sind u.a. DNS, E-Mail, Authentisierung oder ein Konfigurationsdienst. Für die Verwaltung des Extranets der BMW-Händler hat *T-Systems* ein eigene Servicedomäne eingerichtet. Innerhalb dieser wird den Händlern auch ein gesicherter Zugang zum Internet angeboten.

Die Dienste, die den Händlern von BMW zur Verfügung stehen, werden durch Server in einer eigenen *Service Area* bei BMW erbracht. Für die Administration dieser Dienste ist BMW zuständig.

5.2 Der *Car-Konfigurator*

5.2.1 Definition

Ein Dienst, der über das Extranet zur Verfügung steht, ist der *BMW-Car-Konfigurator*. Er unterstützt den BMW-Verkäufer beim Zusammenstellen eines neuen Autos aus der aktuellen Modellpalette entsprechend den Kundenwünschen. Eine Kopplung mit dem Produktionssteuerungssystem gestattet Aussagen über Liefertermine in Abhängigkeit von Ausstattungsvarianten. Der BMW-Händler greift mit einem Web-Browser auf den *Car-Konfigurator* zu.

Die Definition der Beziehungen, die zwischen den Vertragsparteien bestehen, ist [9] geregelt. Die Händler abonnieren den Service bei der Muttergesellschaft. Das Geschäftsmodell sieht vor, das in regelmäßigen Intervallen eine Rechnungslegung durch BMW erfolgt. Die Dienstgüte (*Quality of Service* (QoS)) ist zwischen ihnen vertraglich in *Service Leve Agreements* festgelegt. Bei der Diensterbringung nutzt BMW technische Möglichkeiten, die von *T-Systems* zugekauft werden, d.h. der Service *Car-Konfigurator* wird durch Zusammenwirken mehrerer, wirtschaftlich selbstständiger, Organisationseinheiten für die Händler erbracht.

5.2.2 Systemdesign

Es existieren zu der in Bild 3.1 abgebildeten Spezifikation Alternativen. BMW ist nicht ge-

zwungen, die Server, auf der die Anwendung *Car-Konfigurator* installiert ist, selbst zu betreiben. Aus technischer Sicht gibt es 2 zusätzliche Varianten:

- Betrieb der Server durch *T-Systems* in der schon bestehenden Service Domäne,
- Verteilung der Applikation an die BMW-Händlern (z.B. eigener Server, CDs).

Eine abschließende Bewertung, welche der 3 Varianten eine optimale Lösung darstellt, ist im Rahmen dieser Seminararbeit nicht möglich. Hierfür ist die Kenntnis der BMW-Geschäftsstrategie für den *Car-Konfigurator* notwendig.

Der Betrieb dieser Applikation an einem zentralen Ort gewährleistet eine stets aktuelle Datenbank der angebotenen Produktpalette. Dies ist bei ein Verteilung z.B. durch CDs nicht gegeben. Bei dieser Art der Anwendungsverteilung könnte das entsprechende Autohaus mit einer älteren Version weiterarbeiten. Dies widerspricht der BMW-Firmenphilosophie im Umgang mit ihren Kunden. Dieses Vorgehen ermöglicht somit einen einfachen Nutzungsausschluss der BMW-Händler, die sich im Zahlungsrückstand für diesen Service befinden. Die derzeit aktuelle Version des *Car-Konfigurators* (Stand: Januar 2004) besitzt eine Web-Schnittstelle. Über sie ist die Anwendung mit eingeschränktem Funktionsumfang (d.h. ohne Zugriff auf das Produktionssteuerungssystem) öffentlich zugänglich. Die URL lautet:

http://www.bmw.de/carconfigurator/main.htm .

Somit scheidet eine Auslagerung dieser Applikation in Richtung BMW-Händler aus. Schwierig ist es, eine Entscheidung zwischen dem Betrieb der Anwendung in der *BMW-Service Domäne* oder der *Service Area*, die *T-Systems* für das *BMW-Extranet* eingerichtet hat, zu fällen. Eine Kopplung dieser Anwendung mit dem Produktionssteuerungssystem ist unter dem Sicherheitsaspekt kritisch. Unter allen Umständen ist zu vermeiden, dass unbefugte Nutzer (Hacker) Zugang zu sensiblen Produktions- und Geschäftsdaten erhalten. Dieser Fakt spricht für den Betrieb des *Car-Konfigurators* durch BMW.

Ein Auslagern dieses Services (*Outsourcing*) bringt ein organisatorisches Problem mit sich. T-Systems müsste eine Nutzer- und Zugangsverwaltung betreiben. Die Abrechnung der Nutzung des Services erfolgt aber durch BMW. In diesem Fall muss der Autokonzern in regelmäßigen Zeitabschnitten Informationen an T-Systems übermitteln, welche BMW-Autohäuser

noch berechtigt sind, diesen Service zu nutzen. Die Vor- und Nachteile des Auslagerns von IT-Diensten werden ebenfalls in [1] diskutiert.

Das grundsätzliche Design einer Applikation wird in der Phase 1 des Lebenszyklusmodells, der Anforderungsanalyse, erarbeitet.

5.2.3 Die Antwortzeit – ein QoS-Parameter

Ein QoS-Parameter (*Quality of Service*) ist die Antwortzeit. Ein Zugriff auf den Server wird durch 3 Zeitbestandteile ($t_{Antwort}$) bestimmt. Eine Anfrage wird zunächst im Netz des BWM-Händlers ($t_{Händler}$) übertragen. Für die Weiterleitung zum Ort, an dem der Service erbracht wird, ist ein weiterer Zeitanteil ($t_{T\text{-}Systems}$) hinzuzurechnen. Schließlich benötigt auch der Server eine Reaktionszeit (t_{Server}), um die Antwort zu berechnen. Das Ergebnis wird zum Händler übertragen (vgl. Bild 5.2).

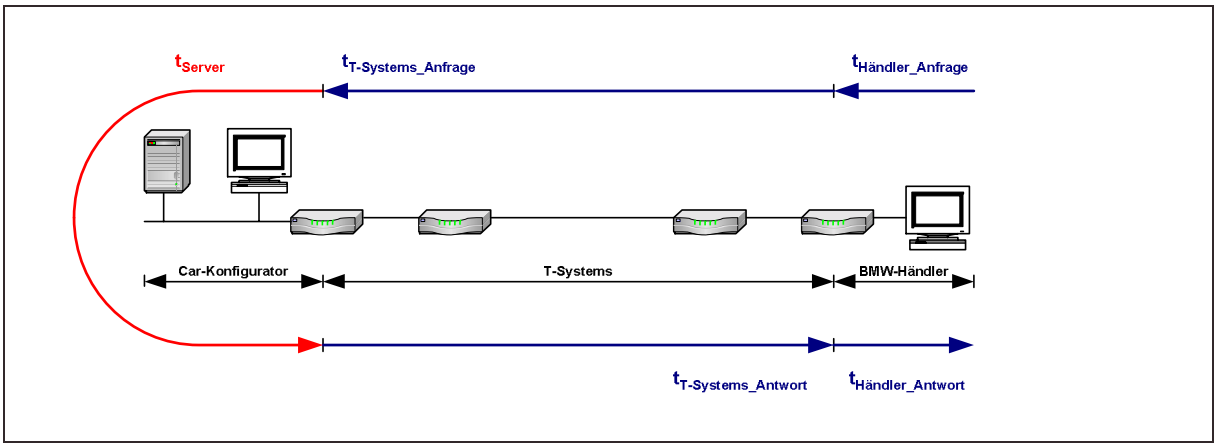

Bild 5.2: Serverzugriff

Somit lässt sich für einen Serverzugriff folgender mathematische Zusammenhang (Gl. 5.1) angeben.:

$$t_{Antwort} = 2 \times (t_{Händler} + t_{T\text{-}Systems}) + t_{Server} \qquad (5.1)$$

Die Nachrichtenlänge der Anfrage ($l_{Anfrage}$) bzw. der Antwort (l_{Anwort}) ist unterschiedlich. Es gilt fast immer:

$$l_{Anfrage} < l_{Antwort} \qquad (5.2)$$

Beide Nachrichten (Anfrage und Antwort) werden durch das Netz von T-Systems geleitet und passieren verschiedene Knotenrechner innerhalb des Transitsystems. Jeder dieser Knotenrechner entscheidet autark, auf welchem Weg ein Paket weitergeleitet wird. Es gilt:

$$t_{T\text{-}Systems\_Anfrage} \neq t_{T\text{-}Systems\_Antwort} \tag{5.3}$$

Diese Überlegungen sind auch auf das Netz des Händlers übertragbar. Somit lassen sich die Zeiten $t_{T\text{-}Systems}$ und $t_{Händler}$ in 2 Bestandteile aufteilen und für Gleichung (5.1) ein erweiterter Ausdruck angeben (Gl. 5.4).:

$$t_{Antwort} = t_{Händler\_Anfrage} + t_{T\text{-}Systems\_Anfrage} + t_{Server} + t_{T\text{-}Systems\_Antwort} + t_{Händler\_Antwort} \tag{5.4}$$

Die Zeiten zur Übertragung von Nachrichten im Netz von T-Systems wird in einer Vereinbarung (*Service Level Agreement*) zwischen BMW und T-Systems geregelt. Die Reaktionszeit der Server t_{Server} wird durch konstruktive Maßnahmen (Hardware, Software) bestimmt.

5.2.4 Messung der Antwortzeit

Bezogen auf das BMW-Referenz-Szenario erhält der BMW-Händler mit diesem Messverfahren, das in Abschnitt 4.1 vorgestellt wurde, ein Werkzeug, um die Antwortzeit zu überwachen. Leider ist dieses Messverfahren nicht in Standard-*Web-Browsern* integriert (*Internet-Explorer*, *Mozilla*, *Netscape*). Vielmehr sind sie entsprechend anzupassen (vgl. [16] und [17]). Damit unterliegt diese modifizierte Standardsoftware ebenfalls dem Lebenszyklusmodell, das in dieser Seminararbeit diskutiert wurde. In diesem Zusammenhang spielen jetzt auch *Deployment-Strategien* (vgl. Abschnitt 3.6 (*Einführung*) bzw. Abschnitt 3.8 (*Optimierung*)) eine viel größere Rolle, da bei einem Versionswechsel (z.B. Änderung der Definition des Meßverfahrens für die Antwortzeit, Versionssprung der ARM-Schnittstelle) viele Arbeitsplatzrechner bei den Händlern betroffen sind, die zudem örtlich weit verteilt sind.

5.2.5 Betrieb des *Car-Konfigurators*

In [7] wird darauf hingewiesen, dass für komplexere verteilte Anwendungen eine Trennung von Präsentation und Anwendungslogik anzustreben ist. Bild 5.3 verdeutlicht die Komponen-

ten, aus denen der *Car-Konfigurator* (Web-Service) aufgebaut seien könnte. Die Aufgaben, die ein Applikation-Server wahrnimmt, sind in Tabelle 5.1 zusammengefasst.

Tabelle 5.1: Aufgaben eines Applikation-Servers (vgl. [7])

Funktion	Erläuterung
Lastbalancierung	▪ Installation als Cluster; Trotzdem bleibt die Software transparent.
Hochverfügbarkeit	▪ Sicherstellen, dass bei Ausfall ein Ersatzsystem verfügbar ist
Sicherheit	▪ Anlegen von Benutzern und Gruppen, die Zugriff auf die Programmlogik erhalten; Diese Vergabe von Nutzerrechten kann sehr differenziert erfolgen.
verteilte Transaktionen	▪ Eine einzelne Datenbank stellt bei lokalen Transaktionen sicher, dass die Integrität der Daten gewährleistet bleibt. Ein Applikation-Server steuert die Koordination von mehreren Transaktionen auf verschieden lokalen Datenbanken.
Connection Pooling	▪ Optimierung des Datenbankzugriffes durch Wiederverwendung von Datenbankverbindungen
Naming	▪ Adressierung aller wichtigen Systembestandteile durch einen Verzeichnisdienst
Entwicklungs- und Deploymentwerkzeuge	▪ Bereitstellen von Werkzeugen, die die Installation der Software auf dem Applikation-Server unterstützt

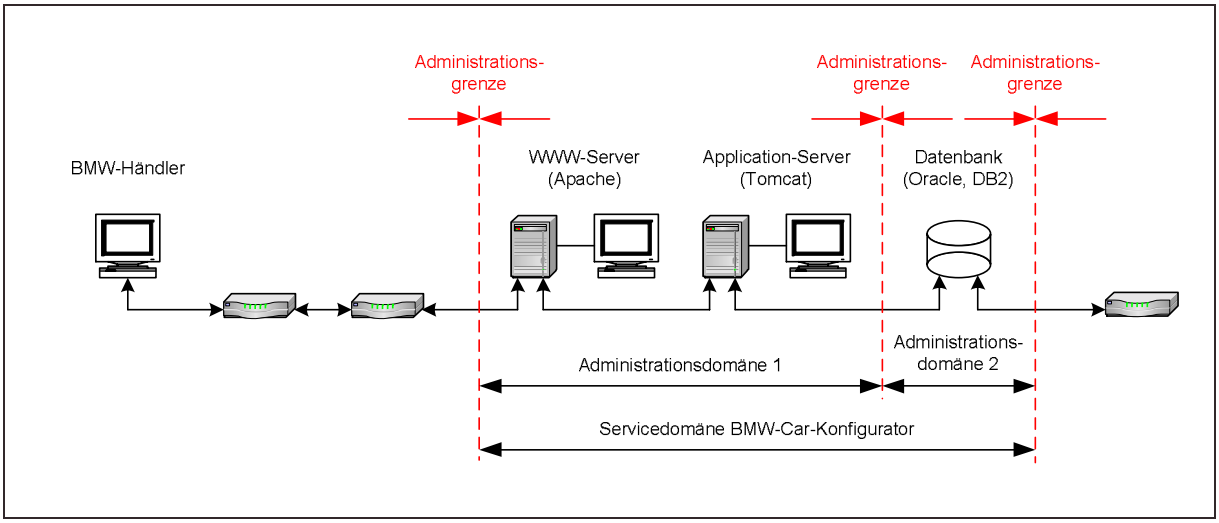

Bild 5.3: Mögliche Konfiguration der Web-Service-Anwendung BMW-Car-Konfigurator

Für die Administration ergeben sich folgende Konsequenzen. Die Software, die auf den Servern installiert ist, lässt sich in 2 Gruppen unterteilen. Zum einen kommen Standardkomponenten zum Einsatz (WWW-Server, Application-Server, Datenbanksysteme). Die gesamte Produktinformation ist in einer Datenbank hinterlegt. Die auf den Servern laufende Anwendungen werden im Rahmen der Installation einmal konfiguriert. Es ist damit zu rechnen, dass

halbjährig ein Update einzuspielen ist. Die Produktdatenbank unterliegt den Veränderungen der Produktpalette bzw. gezielten Marketingaktivitäten. Somit existieren 2 vollkommen unabhängige Updatezyklen. Durch die Trennung in 2 Administrationsdomänen kann der *Car-Konfigurator* von 2 unterschiedlichen Betreibern (Abteilungen) betreut werden. Dies eröffnet die Möglichkeit, *T-Systems* zu beauftragen, den Betrieb der Server zu organisieren. BMW konzentriert sich nur noch auf die Administration der Produktdatenbank. Es gelten die Ausführungen des Kapitels 3.8 (*Betrieb*).

Es wird noch einmal darauf hingewiesen, dass die Verfahren zur Überwachung des Betriebes einer Anwendung im Rahmen des Systemdesigns und der Implementierung (vgl. Abschnitt 3.4 bzw. 3.5) berücksichtigt werden müssen. In welcher Art und Weise eines der in Kapitel 4 eingesetzten Verfahren zum Einsatz kommt, ist in jedem Projekt neu zu entscheiden. Die in Kapitel 3.5 (*Realisierung*) genannten Managementstandards (AMS, DMTF, DMI) sind ebenso zu beachten. Das Implementieren von Managementstandards stellt einen zusätzlichen Aufwand dar, der sich letztendlich in den Entwicklungskosten niederschlägt.

So ist das Überwachen von Transaktionen ist nur möglich, wenn die Anwendung instrumentiert wurde (vgl. Kapitel 4.1 und Kapitel 4.3).

5.2.6 Optimierung / Weiterentwicklung des Car-Configurators

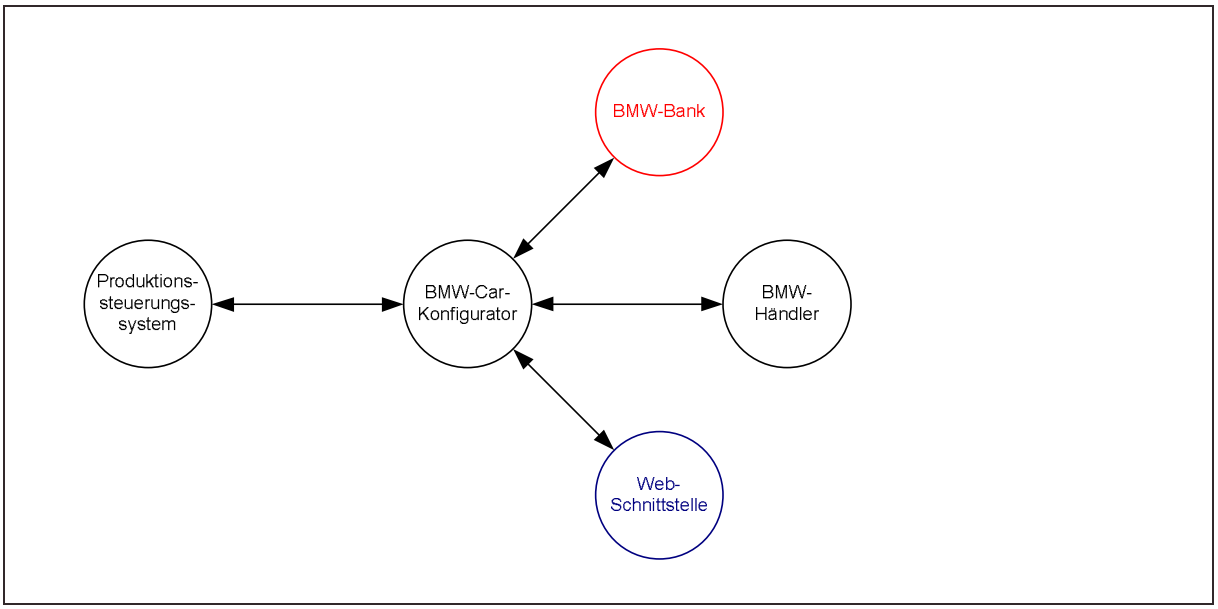

Bild 5.4: Schnittstellen des Car-Konfigurators

Die Entwicklung von Schnittstellen (extern, intern) zu anderen Systemen könnte im Mittelpunkt der Optimierung bzw. Weiterentwicklung dieser Anwendung stehen (vgl. Bild 5.4). Ein Beispiel für eine externe Schnittstelle ist die Web-Schnittstelle, die in Abschnitt 5.2.2 (*Systemdesign*) erwähnt wurde. Ein Datenaustausch mit dem Produktionssteuerungssystem ist ein Beispiel für ein internes Interface.

Eine Kopplung mit anderen Systemen ist denkbar. So sollte es dem BMW-Händler im Rahmen der Verkaufsgespräche möglich sein, einem Kunden ebenfalls Finanzierungsmöglichkeiten anzubieten. Dazu ist eine Anfrage bei der Bank notwendig, die die Finanzierung übernimmt, nachdem der Gesamtpreis feststeht. Die Kreditkonditionen können von der Wahl der Ausstattung abhängen. Gleichzeitig ist die Kreditwürdigkeit des Kunden zu überprüfen.

Für die Realisierung dieser Funktionalitäten ist ein Datenaustausch zwischen den verschiedenen Computersystemen der einzelnen Firmen notwendig. In der Literatur wird hierfür der Begriff *Business-to-Business-Anwendung* (B2B) geprägt (vgl. [7]). Für die Überwachung der Kommunikation zwischen den Unternehmen eignen sich ebenfalls die JMX-Managementarchitektur. In [14] wird hierzu eine vergleichbare E-Business-Anwendung vorgestellt. Fragen der IT-Sicherheit sind besonders kritisch zu beurteilen. In dieser Konfiguration ist über das Auslagern dieses IT-Dienstes neu zu entscheiden.

6. Zusammenfassung

Diese Seminararbeit geht auf das Lebenszyklusmodell für Anwendungen, das in [1] definiert ist, ein. Ein Schwerpunkt bildet das Beschreiben der Phasen:

- Anforderungsanalyse,
- Design,
- Realisierung,
- Einführung,
- Betrieb,
- Optimierung.

Gleichzeitig stellt die Arbeit Verfahren und Lösungsansätze vor, mit denen auf Servern installierte Anwendungen in ihrem Betrieb überwacht werden können. Dabei wird kein Anspruch auf Vollständigkeit erhoben. In einem 3. Teil wurde versucht, das *Application Management* auf das BMW-Extranet-Szenario anzuwenden, das in [9] vorgestellt wird.

Im Rahmen dieses Seminars wurden weitere Themen bearbeitet, zu denen Querbezüge bestehen. Insbesondere sind dies die Arbeiten über das *ICT Infrastructure Management* und den *ITIL Service Support*. Dabei handelt es sich um:

- *ICT Infrastructure Management – Design and Planing*,
- *ICT Infrastructure Management – Deployment and Technical Support*,
- *ITIL Service Support – Configuration Management*,
- *ITIL Service Support – Change and Release Management*.

7. Literaturverzeichnis

[1] o.V.: *Application Management*. 1. Auflage London: Her Majesty's Stationery Office, 2002 – ISBN 0-11-330866-3

[2] o.V.: *Service Support*. 1. Auflage London: Her Majesty's Stationery Office, 2000 – ISBN 0-11-330015-8

[3] o.V.: *Service Delivery*. 1. Auflage London: Her Majesty's Stationery Office, 2001 – ISBN 0-11-330017-4

[4] o.V.: *ICT Infrastructure Management*. 1. Auflage London: Her Majesty's Stationery Office, 2002 – ISBN 0-11-330865-5

[5] Hegering, Heinz-Gerd; Abeck, Sebastian; Neumair, Bernhard: *Integriertes Management vernetzter Systeme*. 1. Auflage Heidelberg: dpunkt-Verlag für digitale Technologie, Hüttig GmbH, 1999 – ISBN 3-932588-16-9

[6] Laurie, Ben; Laurie, Peter; Klicman, Peter; Wiedmann, Jochen; Lang, Jorgen W.: *Apache – Das umfassende Handbuch*. 2. Auflage Köln: O'Reilly Verlag, 2003 – ISBN 3-89721-356-7

[7] Eberhart, Andreas; Fischer, Stefan: *Web Services – Grundlagen und praktische Umsetzung mit J2EE und .NET*. 1. Auflage München: Carl Hanser Verlag München Wien, 2003 – ISBN 3-446-22530-7

[8] Oesterreich, Bernd: Objektorientierte Softwareentwicklung : *Analyse und Design mit der Unified modeling language*. 5. völlig überarbeitete Auflage Oldenbourg: Oldenbourg Wissenschaftsverlag GmbH, 2001 – ISBN 3-486-25573-8

[9] Reiser, Helmut: *Sicherheitsarchitektur für ein Managementsystem auf der Basis Mobiler Agenten*. München, Ludwig-Maximilians-Universität, Lehr- und Forschungseinheit für Kommunikationssysteme und Systemprogrammierung, Dissertation, 2001

[10] Reiser, Helmut: *Einsatz des Java Dynamic Management Kit für flexible, dynamische Managementsysteme*. München, Ludwig-Maximilians-Universität, Lehr- und Forschungseinheit für Kommunikationssysteme und Systemprogrammierung, Dezember 1999. www.nm.informatik.uni-muenchen.de/~reiser/#Publications

[11] o.V.: *JSR - 174 – Requirements for the Monitoring and Management Specification for the Java™ Virtual Machine, Public Draft 0.99.7 (25-Sep-03)*. Sun Microsystems, Inc., Palo Alto, CA, September 2003. www.jcp.org/aboutJava/communityprocess/review/jsr174

[12] o.V.: *Java™ Management Extentions Instrumentation and Agent Specification, v1.2*.

Sun Microsystems, Inc., Palo Alto, CA, Oktober 2002, www.java.sun.com/products/JavaManagement

[13] o.V.: *Getting Started with the Java Dynamic Management Kit 5.0*. Sun Microsystems, Inc., Palo Alto, CA, Juni 2002, www.java.sun.com/products/JavaManagement

[14] Kreger, Heather: Java Management Extensions for application management. In: *IBM Systems Journal Volume 40 (2001)*, No. 1, S. 104 – 129. www.researchweb.watson.ibm.com/journal/sj/401/kreger.html

[15] o.V.: Technical Standard : *Systems Management : Application Response Measurement (ARM) API*, Berkshire, 1998 – ISBN 1-85912-211-6. www.opengroup.org/tech/ management/arm

[16] Johnson, Mark; Furnas, Bill; Ruppert, Stefan: *Technical Standard: Application Response Measurement Issue 4.0 – C Binding*. Berkshire, 2003 – ISBN 1-931624-35-6. www.opengroup.org/tech/management/arm

[17] Johnson, Mark; Furnas, Bill; Ruppert, Stefan: *Technical Standard: Application Response Measurement Issue 4.0 – Java Binding*. Berkshire, 2003 – ISBN 1-931624-36-4. www.opengroup.org/tech/management/arm

[18] Kalbfleisch, C.: *Applicability of Standards Track MIBs to Management of World Wide Web Servers*. RFC 2039, IETF, November 1996

[19] Gögetap, Boran: *ITIL Application Management Express, Schulungsunterlagen*. Continental Software GmbH, Seefeld bei München, November 2003. www.continental-software.com