Technische
Universität
München

Bachelor-Arbeit

Softwarekartographie: Analyse und Darstellung der IT-Landschaft eines mittelständischen Unternehmens

Bearbeiter: Stephen Lauschke

Betreuer: Josef Lankes

Aufgabensteller: Prof. Dr. Matthes

Technische Universität München
Fakultät für Informatik

Abgabedatum: 15.07.2005

Erklärung

Ich versichere, dass ich diese Bachelor-Arbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

Datum, (Stephen Lauschke)

Abstract: Die IT-Landschaften moderner Unternehmen sind sehr komplex und oftmals schlecht dokumentiert. Ein unzureichender Überblick über die eigene IT-Landschaft ist für die meisten Unternehmen jedoch mit Risiken und Kosten verbunden, was es für die Unternehmen notwenig macht ihre IT-Landschaften in geeigneter Form besser zu beschreiben.

Das Gebiet der Softwarekartographie beschäftigt sich mit der Darstellung von IT-Landschaften durch Softwarekarten, wobei der Schwerpunkt auf Anwendungslandschaften liegt.

Die vorliegende Arbeit beschreibt das Vorgehen bei der Analyse und Darstellung der IT-Landschaft eines mittelständischen Unternehmens durch Softwarekarten. Dabei werden die Phasen und deren Aktivitäten beschrieben, die bei der Erstellung von Softwarekarten durchlaufen werden. Darauf aufbauend wurde versucht, diese Phasen in aus der Softwaretechnik bekannte Prozessmodelle einzuordnen.

Neben dem hier vorliegenden theoretischen Teil wurden Softwarekarten für die Krones AG erstellt. Ergebnisse aus dem praktischen Teil, die dem besseren Verständnis dienen, sind zum Teil anonymisiert dem Anhang beigefügt.

Inhalt

Abbildungen

Tabellen

1 Motivation

Unternehmen begeben sich zunehmend in eine starke Abhängigkeit von einer funktionierenden IT-Landschaft[1], was sich nicht zuletzt an der stetig steigenden Zahl von Informationssystemen, die zunehmend auch unternehmenskritische Vorgänge unterstützen, abzeichnet. Die große Anzahl von Anwendungen und deren mitunter starke Vernetzung durch unterschiedlichste Technologien führen zu sehr komplexen Anwendungslandschaften[2] die oftmals kaum dokumentiert sind.
Für produzierende Betriebe wie die Krones AG, sowie für viele Betriebe anderer Sparten ist die volle Funktionsfähigkeit von unternehmenskritischen Anwendungssystemen essentiell, was letztlich eine Dokumentation der IT-Landschaft unumgänglich macht. Häufig steht eine globale Sicht auf die IT-Landschaft nicht zur Verfügung; Wissen über diese existiert höchstens lokal und ist in der Regel stark personenbezogen, was unweigerlich zu Insellösungen führt. Darüber hinaus entstehen Wissenslücken, wenn Schlüsselpersonen das Unternehmen verlassen. Ein anderes Einsatzszenario für die Dokumentation von Anwendungslandschaften stellt der Ausfall kritischer Komponenten dar. In einem derartigen Fall muss in kürzester Zeit reagiert werden; Fragen nach betroffenen Systemen, verantwortlichen Personen und Auswirkungen im Allgemeinen lassen sich jedoch leider oft nur durch profunde Kenntnis der IT-Landschaft, die häufig nur bei Einzelpersonen vorhanden ist, vollständig und schnell beantworten.
Neben Aspekten der Verfügbarkeit hat das Management ein großes Interesse an einer Dokumentation der IT-Landschaft, da mit steigender Komplexität der Landschaft deren Evolution und damit verbunden IT-Investitionen schlechter planbar werden. Auf der anderen Seite haben Informationssysteme eine hohe Lebensdauer, was deren langfristige Einsatzplanung nötig macht. Anwendungssysteme unterstützen eine Vielzahl von Geschäftsprozessen. Durch das Erkennen und Eliminieren redundanter Systeme kann zum Beispiel die Abarbeitung von Prozessen optimiert und Kosten eingespart werden, indem eine durchgängige, geradlinige Unterstützung der Geschäftsprozesse erreicht wird, die möglichst geringe Kosten verursacht.
Offensichtlich ist nach obigen Ausführungen die Beschreibung von Anwendungslandschaften ein wichtiges Thema, welches Gegenstand der vorliegenden Arbeit ist.

1.1 Dokumentation der Anwendungslandschaft bei der Krones AG

Die Krones AG produziert und vertreibt Anlagen für die Getränkeindustrie, die zum Beispiel zum Abfüllen, Etikettieren und Reinigen von Glas- und PET-Flaschen eingesetzt werden.
Wie in der Motivation dargestellt, hat die Krones AG als produzierendes Unternehmen großes Interesse an der Verfügbarkeit ihrer Informationssysteme, da Ausfälle schnell mit hohen Kosten verbunden sind. Die Verfügbarkeit von Anwendungssystemen ergibt sich mittelbar auch aus der Funktionsfähigkeit der darunter liegenden Infrastrukturdienste. Deswegen spielt bei der Krones AG neben der Dokumentation von Anwendungssystemen auch die der Serverlandschaft eine Rolle. Ebenso wie Aspekte der Verfügbarkeit sollen Aspekte des Prozessmanagements und des Informationsmanagements dokumentiert werden, deren

[1] IT-Landschaft: Gesamtheit aller IT-Systeme in einer Organisation
[2] Anwendungslandschaft: Gesamtheit der Anwendungssysteme. Unter einem Anwendungssystem versteht man Software, die fachlich relevante Aufgaben in einer Organisation unterstützt, d.h. spezifische, im Rahmen eines Prozesses verwendete Funktionen anbietet.

Interessen an den Karten im Motivationsteil beschrieben wurden. Damit lassen sich vier zu dokumentierende Bereiche identifizieren, die Interesse an der Beschreibung der Anwendungslandschaft der Krones AG haben:

- Prozessmanagement
- Betrieb
- Applikationen
- IM Leitung

Die Stakeholder dieser Bereiche und deren Interessen an den Softwarekarten werden im Kapitel 3.1.1 genauer beleuchtet.
Dokumentationen aus dem Bereich Betrieb sollen die Abhängigkeiten zwischen Hardwaresystemen beleuchten und Aufschluss darüber liefern, welche Applikationen auf welchen Hardwaresystemen laufen, während Dokumentationen aus dem Bereich Applikationen Informationen über die Abhängigkeiten von Applikationen untereinander bereitstellen sollen, wobei ein besonderes Interesse auf detaillierten Schnittstellenbeschreibungen liegt. Darstellungen aus den Bereichen Betrieb und Applikationen adressieren also die Abhängigkeiten der einzelnen Systeme untereinander, liefern Informationen über Verantwortliche Personengruppen und können somit einen Beitrag bei der Minimierung von Ausfallzeiten leisten.
Von besonderem Interesse ist die Darstellung von Applikationen entlang der Wertschöpfungsprozesse der Krones AG. Dabei werden die vier Prozesse

- Neumaschinen
- Aufrüstung, Umrüstung
- Ersatzteil, Service, Wartung
- Forschung und Entwicklung

unterschieden. Darstellungen aus dem Bereich Prozessmanagement ordnen Applikationen Prozessen und Bereichen zu und dokumentieren auf diese Weise die Verwendung der Applikationen aus Prozesssicht.
Der Bereich IM Leitung ist mit Planungsaufgaben betraut und hegt deshalb sowohl starkes Interesse an der Darstellung des Ist-Zustandes als auch eines Soll-Zustandes der IT-Landschaft.
Die Krones AG möchte die Dokumentation ihrer IT-Landschaft konzernweit ausweiten. Hierzu sollen die zu entwickelnden Darstellungen der IT-Landschaft des Hauptsitzes der Krones AG in Neutraubling bei Regensburg eine Grundlage sein.

2 Softwarekartographie

Die Informatik hat in den letzten Jahrzehnten verschiedenste Mittel entwickelt, um Softwaresysteme und deren Entwicklung zu beschreiben. Mittel zur Beschreibung ganzer Anwendungslandschaften waren bis dato kaum bekannt. Das Gebiet der Softwarekartographie adressiert dieses Vakuum, indem es Mittel zur Verfügung stellt, mit deren Hilfe in Anlehnung an die geographische Kartographie Softwarekarten, die IT-Landschaften darstellen, gezeichnet werden können. Diese Karten sollen Fragen bestimmter Interessengruppen zur IT-Landschaft beantworten.

Die Softwarekartographie beschäftigt sich also mit der Darstellung von IT-Landschaften durch Softwarekarten. Eine Softwarekarte ist eine Repräsentation einer IT-Landschaft, bzw. eines Ausschnittes einer IT-Landschaft. Dabei sollen die Karten die IT Landschaft beschreiben und damit bei deren Bewertung und Gestaltung unterstützen, indem sie Fragestellungen bestimmter Interessengruppen beantworten und als Kommunikationsgrundlage dienen. Der Schwerpunkt der Softwarekartographie liegt dabei auf der Darstellung von Anwendungslandschaften. Eine sinnvolle, vollständige Darstellung macht es jedoch im Kontext vieler Fragestellungen notwendig, auch Infrastrukturdienste mit einzubeziehen.
Softwarekartographie orientiert sich grob an den Ideen der geographischen Kartographie. So bestehen die Karten aus einem Kartengrund, auf den in mehreren Schichten Gestaltungsmittel aufgebracht werden (siehe Abbildung 1), die untereinander in Beziehung gesetzt werden können. Durch die Verwendung von Schichten lassen sich Informationen gruppieren und je nach Bedarf ein und ausblenden, um die Übersichtlichkeit der Karten zu steigern. Für verschiedene Interessengruppen spielt in der Regel nur ein bestimmter Ausschnitt der IT-Landschaft eine Rolle. Das Schichtenkonzept kann die selektive Darstellung verschiedener Sichten auf eine IT-Landschaft durch Filterung der darzustellenden Daten unterstützen.

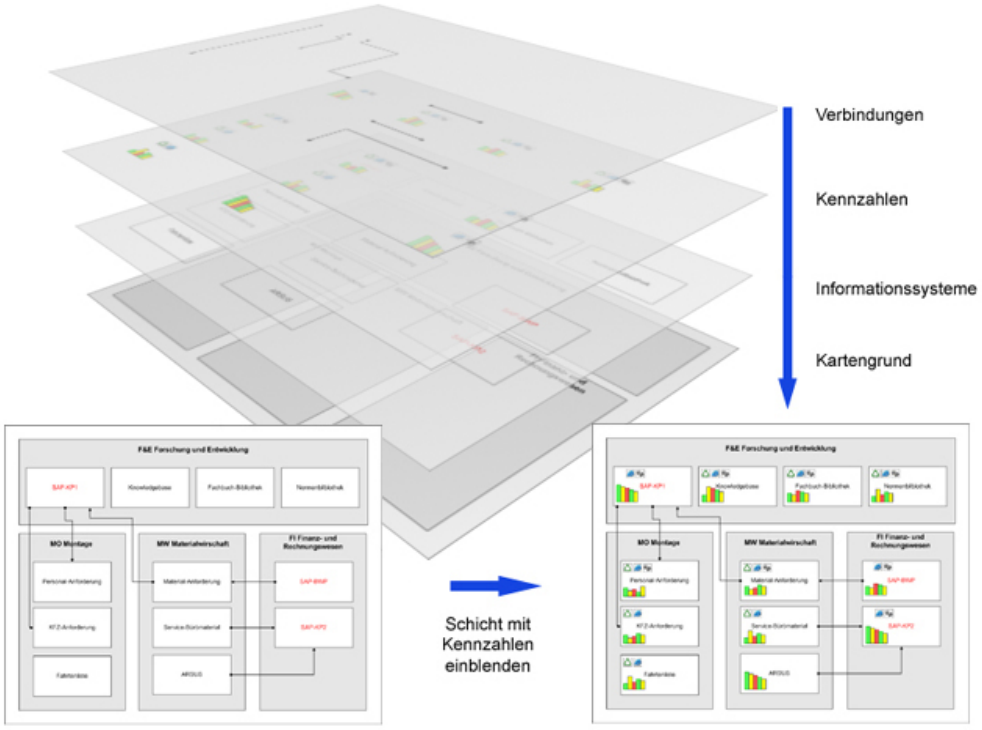


Abbildung 1 Das Schichtenkonzept

Die dreidimensionale Darstellung in Abbildung 1 dient lediglich dazu, das Schichtenkonzept zu verdeutlichen. Um Missverständnissen vorzubeugen, sei angemerkt, dass Softwarekarten zweidimensional sind. Das Beispiel zeigt eine Clusterkarte, die aus den vier Schichten *Kartengrund*, *Informationssysteme*, *Kennzahlen* und *Verbindungen* besteht. Bei Fragestellungen, zu deren Beantwortung Kennzahlen der Informationssysteme keinen Beitrag leisten, könnte zum Beispiel die Schicht *Kennzahlen* ausgeblendet werden, um die Übersichtlichkeit zu erhöhen.
In jedem Unternehmen lassen sich verschiedene Aufgabenfelder identifizieren, in denen Softwarekarten einen Beitrag leisten können. Hierzu beschreibt [MW04a] fünf Aspekte der Softwarekartographie:

Fachliche Aspekte	Fachliche Aspekte lassen sich in organisatorische und prozessorientierte Aspekte unterteilen. Organisatorische und prozessorientierte Aspekte sind stark miteinander verwoben, da Organisationseinheiten Prozessschritte durchführen. Beispiele: • Zuordnung von betrieblichen Anwendungssystemen zu logischen Einheiten • Anzahl der Nutzer pro betriebliches Anwendungssystem • unterstützte Geschäftsprozesse pro betriebliches Anwendungssystem
Operative Aspekte	Operative Aspekte beziehen sich auf den unmittelbaren Betrieb betrieblicher Anwendungssysteme. Beispiele: • Ausfall und Betriebszeiten pro betrieblichem Anwendungssystem • Geographische Position des Betriebsortes des betrieblichen Anwendungssystems • Abhängigkeiten zwischen betrieblichen Anwendungssystemen
Planerische Aspekte	Planerische Aspekte beziehen sich auf die Evolution von IT-Landschaften unter Einbeziehung der zeitlichen Komponente. Es sind Ist-, Soll- und Plananwendungslandschaften von Interesse. Unter einer Istanwendungslandschaft versteht man dabei die Anwendungslandschaft zum momentanen Zeitpunkt. Plan- und Solllandschaften beziehen sich auf angestrebte Anwendungslandschaften. Beispiele: • Programme und Projekte auf der Anwendungslandschaft • Lebenszyklus eines betrieblichen Anwendungssystems • Zeitliche Veränderung der Anwendungslandschaft • Zielarchitektur der Anwendungslandschaft

Technische Aspekte	Technische Aspekte beziehen sich auf alle technischen Eigenschaften der IT-Landschaft Beispiele: • Transaktionsraten pro betrieblichem Anwendungssystem • Datenvolumen pro betrieblichem Anwendungssystem • Schnittstellen-Beziehungen und Konnektoren • Implementierungssprache pro betrieblichem Anwendungssystem • Genutzte Datenbankmanagementsysteme pro betrieblichem Anwendungssystem • Genutztes Betriebssystem pro betrieblichem Anwendungssystems
Wirtschaftliche Aspekte	Wirtschaftliche Aspekte beziehen sich auf jegliche Kosten, die im Zusammenhang mit Informationssystemen auftreten. Beispiele: • Kosten pro betrieblichem Anwendungssystem • Kapitalwert des betrieblichen Anwendungssystems

Tabelle 1 Aspekte der Softwarekartographie, nach [MW04a]

2.1 Kartentypen

Die Softwarekartographie adressiert die Interessen verschiedener Interessengruppen bezüglich der Anwendungslandschaft. Das Management eines Unternehmens hat beispielsweise ein großes Interesse an der Planbarkeit der Evolution der IT-Landschaft des Unternehmens und damit verbunden an Kostenreduktion, legt aber weniger Wert auf technisch detaillierte Darstellungen, die aber wiederum für Mitarbeiter anderer Bereiche von großem Interesse sind. Dies macht, wie auch schon die Aspekte der Softwarekartographie erkennen lassen, verschiedene Darstellungen empfehlenswert. [LMW05] stellt die vier in der Praxis gebräuchliche Kartentypen Clusterkarten, Prozessunterstützungskarten, Intervallkarten und Karten ohne Kartengrund zur Verortung vor, die unterschiedliche Visualisierungsmuster bieten, um den Anforderungen der verschiedenen Interessengruppen entsprechen zu können.

2.1.1 Clusterkarten

Clusterkarten erlauben es, Anwendungssysteme logischen Einheiten zuzuordnen. Hierbei gibt der Kartengrund eine Clusterung vor. In Schichten lassen sich neue Cluster und Anwendungen aufbringen, die bei Bedarf miteinander verbunden werden können, um sie in Beziehung zu setzen. Bei Clusterkarten spielt die Position der dargestellten Elemente eine entscheidende Rolle. Man spricht hierbei, in Anlehnung an die herkömmliche Kartographie, von Verortung. Anwendungssysteme, die sich in einem bestimmten Cluster befinden sind diesem zugeordnet. Die Bedeutung von Zuordnungen durch Schachtelungsbeziehungen ist in

einer Legende genau festzulegen. Die Position der einzelnen Cluster auf dem Kartengrund und die Anordnung von Anwendungssystemen unterliegen zunächst keinen Regeln. Hier lassen sich jedoch Konventionen treffen, die nach bestimmten Kriterien die Position der Cluster festlegen, um die Lesbarkeit der Karten weiter zu erhöhen.
Mit Hilfe von Clusterkarten lassen sich zum Beispiel Anwendungen Organisationseinheiten zuordnen. Durch entsprechende Konnektoren und Verbindungen könnte der Datenaustausch zwischen den Anwendungen dargestellt werden.

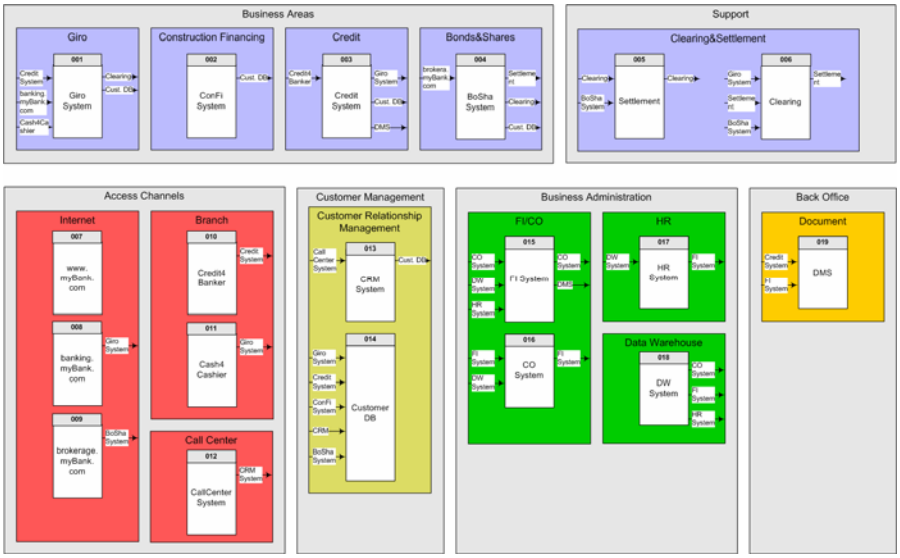


Abbildung 2 Clusterkarte, nach [LMW05]

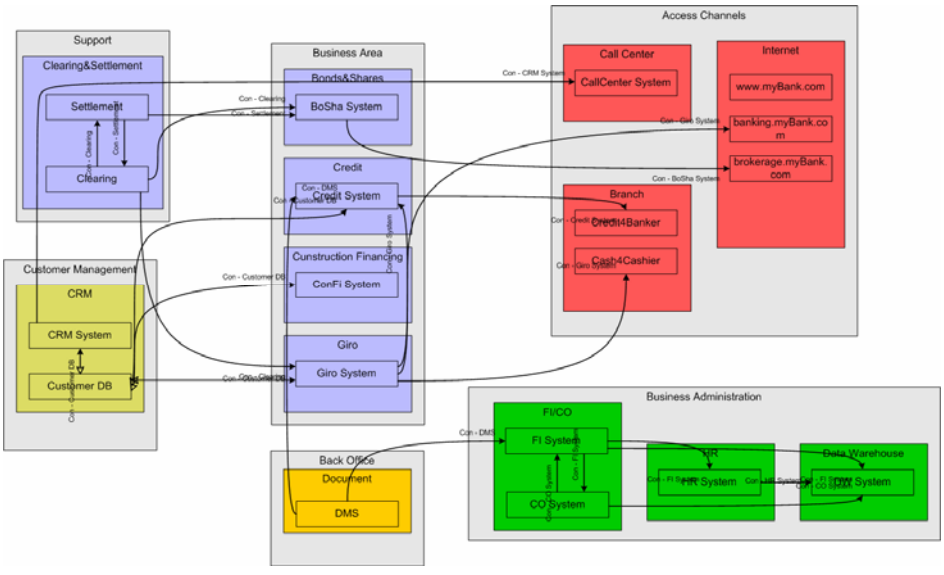


Abbildung 3 Clusterkarte mit Verbindungen, nach [LMW05]

2.1.2 Prozessunterstützungskarten

Prozessunterstützungskarten ermöglichen es, Anwendungen Prozessen, sowie Ausprägungen eines Merkmals oder Entitäten, wie zum Beispiel Organisationseinheiten, zuzuordnen. Dabei werden auf der Horizontalen Prozesse, bzw. Prozessschritte der Wertschöpfungsketten aufgebracht, während das zu visualisierende Merkmal, bzw. Entitäten denen Anwendungssysteme zugeordnet werden sollen, die Vertikale bilden. Die Verortung eines Anwendungssystems transportiert die Information, welche Prozesse das Anwendungssystem unterstützt und welche Ausprägung das visualisierte Merkmal annimmt, bzw. welcher Entität des auf der Vertikalen visualisierten Typs das Anwendungssystem zugeordnet ist. Welcher Natur diese Beziehung genau ist, muss in einer Legende festgelegt werden.
Eine solche Darstellung erlaubt es dem Betrachter beispielsweise Anwendungen zu identifizieren, die bereichsübergreifend Verwendung finden, oder bei der Durchführung mehrere Prozessschritte benötigt werden. Anwendungssysteme werden als Kästchen dargestellt. Wenn sich nun beispielsweise die Breite eines Anwendungssystems über mehrere, der auf der Horizontalen aufgebrachten Prozessschritte erstreckt, wird dieses Anwendungssystem bei diesen Prozessschritten verwendet. Das gleiche gilt für die Höhe des Kästchens und den auf der Vertikalen aufgebrachten Bereichen, falls auf dieser Achse Bereiche gezeigt werden.
Prozessunterstützungskarten erlauben es also bestimmte fachliche Aspekte zu visualisieren.

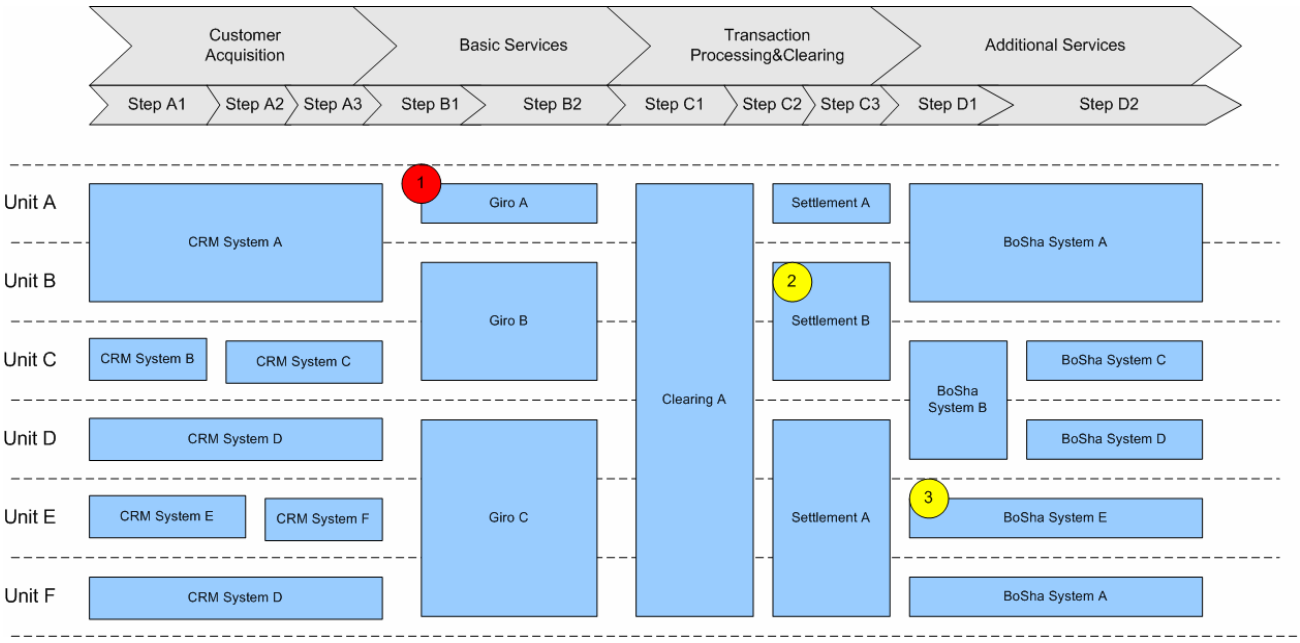


Abbildung 4 Prozessunterstützungskarte, nach [LMW05]

2.1.3 Intervallkarten

Intervallkarten beziehen zeitliche Informationen mit ein und können zur Beschreibung planerischer Aspekte dienen. Die Horizontale stellt eine Zeitachse dar, während auf der Vertikalen zum Beispiel Anwendungssysteme aufgebracht werden können. Diese Darstellung

erinnert stark an Gantt-Diagramme und könnte einem Betrachter zum Beispiel Informationen über den Einsatzzeitraum von Anwendungssystemen und deren geplante Ablösung liefern.

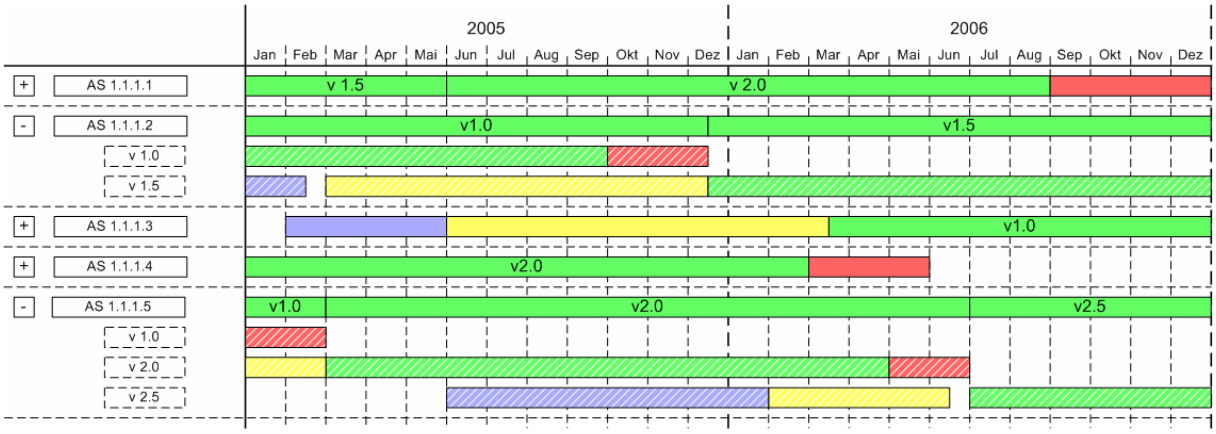


Abbildung 5 Intervallkarte nach [LMW05]

2.1.4 Karten ohne Kartengrund zur Verortung

Bei Karten ohne Kartengrund zur Verortung transportiert die Positionierung der Gestaltungsmittel keine explizite Information. Anders als beispielsweise bei Clusterkarten spielt die Position bei der Interpretation also keine Rolle. Es liegt jedoch nahe der Übersichtlichkeit wegen dennoch einige Regeln zu beachten, die allerdings nicht zwingend befolgt werden müssen. Mögliche Regeln könnten zum Beispiel sein, dass Überschneidungen von Kanten minimiert werden, Kästchen gleichmäßig verteilt sind und Verbindungen in etwa die gleiche Länge haben [Pu01].
Diese Karten dienen hauptsächlich dazu speziell für eine Problemstellung gedachte Visualisierungen automatisch generieren zu können.

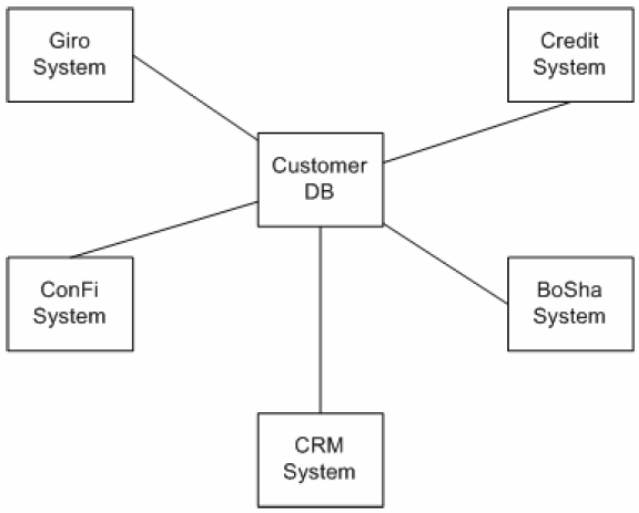


Abbildung 6 Karte ohne Kartengrund, nach [LMW05]

2.2 Der Standard IEEE 1471

Um ein definiertes Vokabular für den Kontext, in dem sich die Softwarekartographie als Verfahren zur Beschreibung von Anwendungslandschaften bewegt, zu verwenden, bietet sich zum Beispiel das von [LMW05b] vorgeschlagene, auf dem Standard IEEE 1471 [IE00], welcher die Dokumentation von Softwarearchitekturen zum Thema hat, aufbauende konzeptuelle Modell an. Dieses Modell definiert Objekte und Akteure, die für die Softwarekartographie von grundlegender Bedeutung sind, setzt diese zueinander in Beziehung und liefert so einen definierten Begriffsapparat für die Softwarekartographie. Nachfolgend sollen die wichtigsten Begriffe näher erörtert werden.

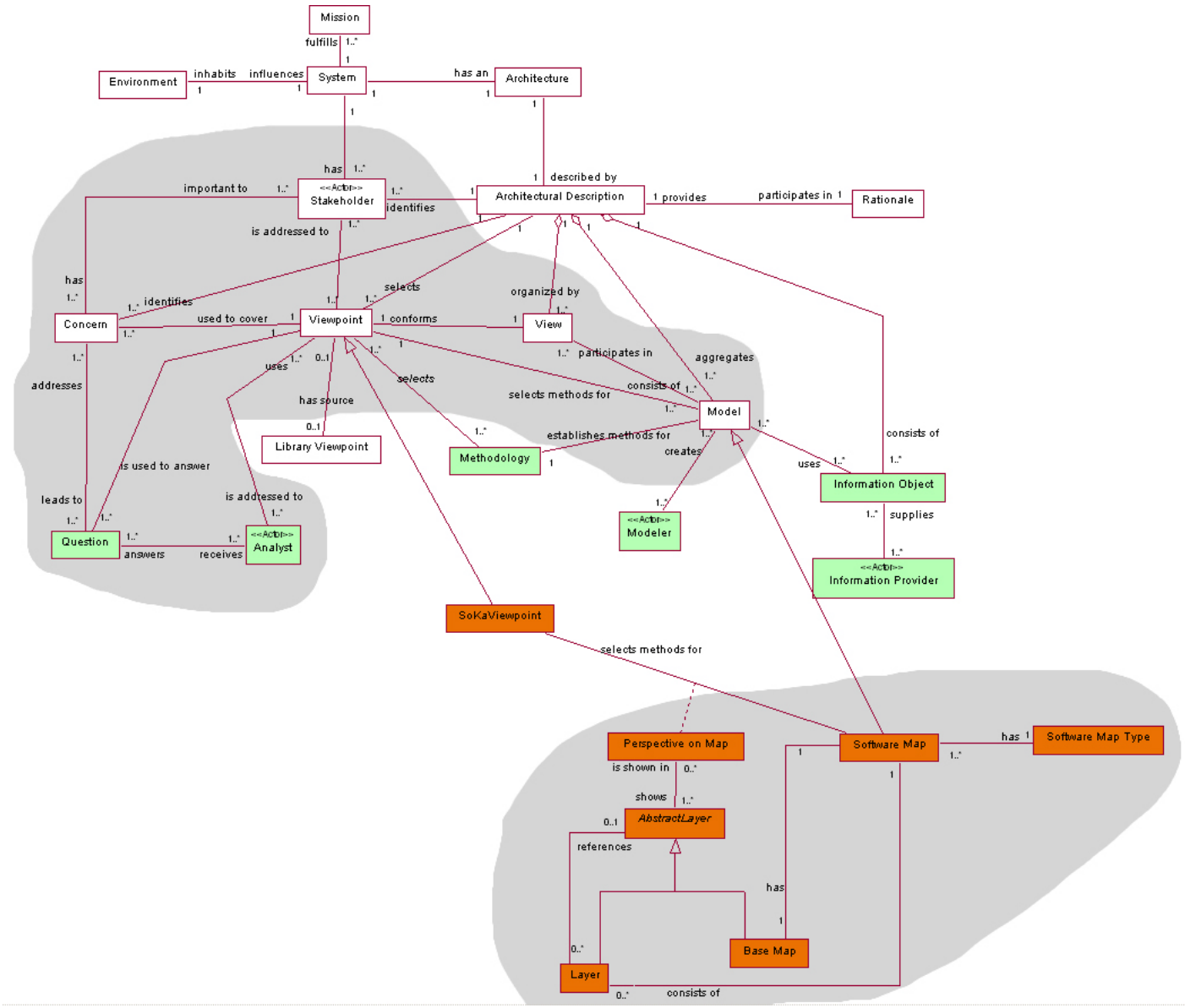


Abbildung 7 Klassendiagramm des Standards IEEE 1471 und Verfeinerung (farbig dargestellte Klassen), nach [LMW05b]

In Abbildung 7 weiß dargestellte Klassen sind Teil des Standards IEEE 1471 [IE00], während gefärbte Klassen in [LMW05b] hinzugefügt wurden, um Anforderungen der Softwarekartographie genügen zu können. Klassen der graphisch hinterlegten Bereiche sind für diese Arbeit von besonderer Bedeutung und werden anschließend aus der Sicht der Softwarekartographie näher beschrieben.

Stakeholder sind Einzelpersonen, Personengruppen oder Organisationen, die bestimmte *Concerns* (Interessen) an der zu dokumentierenden Anwendungslandschaft haben. Zu jedem *Concern* formulieren *Stakeholder* oder *Analysten* konkrete *Questions* (Fragen), welche letztlich durch die Softwarekarten beantwortet werden sollen. Nicht in allen Fällen sind die Stakeholder auch die Nutzer der Karten. So könnte es zum Beispiel sein, dass das Topmanagement nicht direkt mit den Karten arbeitet, sondern andere Personen oder Teams beauftragt dies zu tun. Da Views den Kenntnisstand und andere Eigenschaften ihrer Nutzer mit einbeziehen müssen, ist es erforderlich, zwischen Nutzern und Stakeholdern unterscheiden zu können. Diese Unterscheidung schlägt sich in der Klasse *Analyst* nieder, *Analysten* sind Personen, die mit den Karten arbeiten und sich dabei mit den Concerns von Stakeholdern beschäftigen. Natürlich können *Stakeholder* auch die Rolle des *Analysten* einnehmen, wenn sie selber die Karten nutzen.
Der Begriff des *Stakeholders* ist für die Softwarekartographie insofern von großem Interesse, da in der Regel Anforderungen an zu erstellenden Softwarekarten erst in Gesprächen mit den Stakeholdern und Analysten erhoben werden müssen (siehe Kapitel 3.1.2).
Auf Grund der oft sehr unterschiedlichen Concerns, die die verschiedenen Stakeholder einer Anwendungslandschaft haben, ist es erforderlich, die Beschreibung einer Anwendungslandschaft in *Views* (Sichten) zu unterteilen, die aus einer oder mehreren *Software Maps* (Softwarekarten) bestehen. Jede *Software Map* basiert auf dem der *View* zugeordneten *Viewpoint* (Blickwinkel), in dem ein Darstellungsregelwerk, die Notation der Karte, definiert ist. *Software Maps* (Softwarekarten) sind eine Spezialisierung der Klasse *Model* und stellen die zu dokumentierende Anwendungslandschaft graphisch dar, sind also ein Modell selbiger.
Jede Softwarekarte ist von einem *Software Map Type* (Kartentyp). Vier unterschiedliche Kartentypen wurden bereits in dem Kapitel 2.1 vorgestellt. Jede Softwarekarte kann, wie oben beschrieben (siehe Abbildung 1), aus mehreren *Layers* (Schichten) bestehen, durch die Informationen gruppiert ein- und ausgeblendet werden können.
Eine View beschreibt unter Verwendung ggf. mehrerer Karten die Anwendungslandschaft aus einem, durch die dem *Viewpoint* zugeordneten *Concerns* bestimmten, Blickwinkel. Dabei ist jedem View genau ein Viewpoint zugeordnet. Der Viewpoint beschreibt den ihr zugehörigen View abstrakt. In diesem Sinn verhält sich der View zu seinem Viewpoint wie ein Objekt zu dessen Klasse.
Viewpoints spielen eine zentrale Rolle in der Softwarekartographie, da hier die Gestaltungsmittel und deren Beziehungen untereinander definiert werden. In Viewpoints werden die Legenden zu den entsprechenden Karten festgelegt.

3 Phasen bei der Entwicklung von Softwarekarten

In diesem Abschnitt werden Phasen und damit verbundene Aktivitäten beschrieben, die bei der Erstellung von Softwarekarten durchlaufen werden sollten.
Die Konstruktion von Softwarekarten lässt sich in Anlehnung an einen Softwareentwicklungsprozess in fünf Phasen gliedern, die in den folgenden Unterkapiteln vorgestellt werden.

Analyse-Phase	In der Analyse-Phase werden die Anforderungen (*Requirements*) an die zu erstellenden Softwarekarten erhoben.
Design-Phase	In der Design-Phase wird ein Informationsmodell der Anwendungslandschaft inklusive wichtiger Teile der Infrastruktur, sowie des organisatorischen Umfelds erstellt, welches als Verankerung für die Visualisierung und als Grundlage für die Datenerhebung dient, die ebenfalls in der Design-Phase durchgeführt werden kann. Außerdem werden in der Design-Phase die Viewpoints spezifiziert.
Konstruktions-Phase	Da zurzeit oft keine ausreichende Werkzeugunterstützung zum automatisierten Erstellen von Softwarekarten zur Verfügung steht, müssen diese manuell „gezeichnet“ werden, was in der Konstruktions-Phase geschieht.
Test-Phase	Neu erstellte Softwarekarten sollten getestet werden, um sicherstellen zu können, dass die Karten den erhobenen Anforderungen auch gerecht werden. Dies geschieht in der Test-Phase.
Pflege- und Nutzungs-Phase	Neben der eigentlichen Nutzung der Karten, müssen diese aufgrund fehlender Werkzeugunterstützung manuell gepflegt werden, was in der Pflege- und Nutzungs-Phase geschieht.

Tabelle 2 Phasen bei der Entwicklung von Softwarekarten

3.1 Analyse-Phase

Das Hauptziel der Analyse-Phase besteht darin die Anforderungen an die Softwarekarten zu erheben. Hier muss in enger Zusammenarbeit mit den Stakeholdern geklärt werden, bei welchen Problematiken ihrer Arbeit mit der Anwendungslandschaft sie sich Unterstützung von den Karten erhoffen und zu welchen Fragestellungen die Karten Antworten liefern sollen. Im Vordergrund steht also die Erhebung der Concerns und Questions (entsprechend der in Kapitel 2.2 vorgestellten Definition). Die Analyse-Phase läst sich grob in die folgenden Aktivitäten unterteilen:

- Identifizieren der Stakeholder
- Grobe Terminplanung
- Erhebung der Concerns und Questions
- Validation der Concerns und Questions
- Abschlussbesprechung

3.1.1 Identifikation der Stakeholder

Die Auswahl der richtigen Stakeholder ist eine wesentliche Voraussetzung für die Erstellung nutzbringender Karten. Das Fehlen wichtiger Stakeholder führt zu unvollständigen Concerns und Questions, deren Adressierung durch die Karten unter Umständen nur durch späte Änderungen sichergestellt werden kann. Deshalb muss vor der Erhebung der Anforderungen an die Karten gewährleistet sein, dass alle wichtigen Interessengruppen vertreten sind.
Das späte außerplanmäßige Ändern von Softwarekarten kann mit sehr hohem Aufwand verbunden sein, da es keine ausreichende Unterstützung zur automatischen Generierung von Softwarekarten gibt. Im besten Fall müssen lediglich neue Merkmale berücksichtigt werden, was dazu führt, dass neue Gestaltungsmittel aufgebracht werden müssen. Dies kann aber auch dazu führen, dass die Karten aus Platzgründen komplett neu gezeichnet werden müssen und der Aufwand der Datenerhebung ggf. beträchtlich steigt. Im schlechtesten Fall wurden wichtige Aspekte übersehen, die einen notwendigen Kontext für die darzustellenden Informationen bilden. In diesem Fall ist es möglich, dass das Informationsmodell (siehe Kapitel 3.2) vollständig überarbeitet werden muss, was weit reichende Folgen für die Datenerhebung und die Legenden der Karten haben kann und letztlich das gesamte Projekt in die Design-Phase zurückwerfen könnte.
Wahrscheinlich wird das Unternehmen, dessen IT-Landschaft analysiert und dokumentiert werden soll, Stakeholder vorschlagen. Um diese Vorauswahl rechtzeitig vervollständigen zu können, sollte zu Beginn des Projektes eine genaue Zieldefinition erarbeitet werden, die festlegt, was für einen erfolgreichen Projektabschluss realisiert werden muss. Aus einer guten Zieldefinition wird ersichtlich sein, welche Teile der Anwendungslandschaft des Unternehmens durch die Karten beschrieben werden sollen und damit letztlich auch welche Stakeholder vertreten sein müssen. Als Orientierung können hierbei die Aspekte der Softwarekartographie (siehe Kapitel 2) herangezogen werden. Aus der Zieldefinition muss hervorgehen, welche Typen von Stakeholdern vertreten sein müssen und welche Bereiche des Unternehmens analysiert und dargestellt werden sollen. Aus jedem dieser Bereiche sollte mindestens ein Stakeholder bestimmt werden, der nach Möglichkeit nicht nur zur Erhebung der Anforderungen an die Karten, sondern auch für die Dauer des gesamten Projektes für Fragen und zu Testterminen zur Verfügung steht.
Auch bei einer gewissenhaften Selektion der Stakeholder ist in der Praxis immer damit zu rechnen, dass neue, zunächst nicht berücksichtigte, Stakeholder in das Projektteam mit aufzunehmen sind, was im Rahmen der Anforderungserhebung zu einem Mehraufwand führen kann, der bei der Projektplanung berücksichtigt werden sollte.
Auch wenn im Vordergrund die Modellierung der Anwendungslandschaft steht, sollte erwogen werden, auch Personen mit einzubeziehen, die Infrastrukturdienste betreuen, da Kenntnisse über die Infrastruktur des Unternehmens eine wichtige Rolle bezüglich Aspekten spielen können, die für die Anwendungslandschaft von Bedeutung sind. In der Zieldefinition sollte jedoch genau festgelegt werden, in welchem Detailgrad Infrastrukturdienste miteinbezogen werden sollen, da die Gefahr besteht, dass die Analyse und Dokumentation der Infrastruktur den Projektaufwand explodieren läst. Ähnliches gilt beispielsweise auch für Prozesse.

Bei der Durchführung des praktischen Teils der Arbeit sah die Zieldefinition des Projektes die Erstellung von Softwarekarten für die Bereiche Betrieb, Applikationen, Prozessmanagement, und IM Leitung vor, aus denen jeweils Stakeholder vertreten waren.
Der Bereich Betrieb hat die Aufgabe die Funktionsfähigkeit der Hardwarelandschaft und der Infrastrukturdienste sicherzustellen und ist deshalb an einer Darstellung der Serverlandschaft

der Krones AG interessiert, aus der hervorgeht, welche Anwendungssysteme und Infrastrukturdienste auf welchen Maschinen betrieben werden.
Mitarbeiter des Bereichs Applikationen sind mit der Verwaltung und Pflege der Anwendungssysteme beauftragt. Hier liegt ein besonderes Interesse an der Dokumentation der Abhängigkeiten der Anwendungssysteme untereinander und damit verbunden auf einer detaillierten Schnittstellenbeschreibung.
Das Prozessmanagement versucht eine möglichst geradlinige Abarbeitung der Prozesse in den einzelnen Organisationseinheiten zu erreichen, indem unter anderem Datenpfade optimiert und redundante Anwendungssysteme eliminiert werden. Darüber hinaus hat das Prozessmanagement ein besonderes Interesse daran zu wissen, welche Anwendungssysteme in welchen Organisationseinheiten benötigt werden, um bestimmte Prozessschritte durchzuführen.
Die IM Leitung ist mit übergeordneten Managementaufgaben beauftragt. Hier stehen planerische Aspekte und wirtschaftliche Aspekte im Vordergrund. Das Ziel der IM Leitung bezüglich des Softwarekartographieprojektes ist die Erstellung von Softwarekarten, die den momentanen Ist-Zustand der Anwendungslandschaft dokumentieren und solchen, die eine angestrebte Soll-IT-Landschaft beschreiben. Die IM Leitung erhofft sich von der Softwarekartographie Kosten einsparen und langfristige IT-Investitionen besser planen zu können.

3.1.2 Erhebung von Concerns und Questions

Wie oben beschrieben ist eine wesentliche Voraussetzung für die Qualität der erhobenen Concerns und Questions die richtige Wahl der Stakeholder. Sind alle Stakeholder identifiziert kann mit der Erhebung der Concerns begonnen werden, welche den Charakter von Anforderungen haben und beschreiben, bei welchen Problemen die Softwarekarten unterstützen sollen. Das Requirements-Engineering hat verschiedene Methoden, zum Beispiel Fragebögen, computergestützte Dialoge [Ac97], oder Easy-Win-Win [BG02] hervorgebracht um Anforderungen zu erheben, die vor allem dann Verwendung finden, wenn die Zahl der Stakeholder sehr groß ist, und natürlich auch zur Erhebung von Anforderungen an Softwarekarten eingesetzt werden können, wenn das Projekt dies erforderlich macht. Die einfachste Methode Anforderungen zu erheben ist der moderierte Dialog mit den Stakeholdern in Meetings. Diese Methode bringt den Vorteil mit sich, dass die Stakeholder ein hohes Maß an kreativen Freiräumen haben, die beispielsweise durch Fragebögen oder computergestützte Dialoge genommen würden und auf Fragen direkt eingegangen werden kann. Der moderierte Dialog ist bei überschaubarer Stakeholderzahl vorzuziehen, da andere Methoden ggf. mit einem nicht vernachlässigbaren Mehraufwand verbunden sind. Darüber hinaus bringt das Gespräch mit den Stakeholdern psychologische Vorteile mit sich, da sich diese besser mit dem Projekt identifizieren können und anders, als zum Beispiel bei computergestützten Dialogen, das Gefühl haben, ein nicht unwesentlicher Teil des Projektes zu sein. Außerdem kann ein guter Moderator im Dialog mit den Stakeholdern unmittelbar auf Fragen und Probleme eingehen und die Anwesenden entsprechend anspornen und motivieren. Auf Grund der genannten Vorteile ist es nicht ratsam auf den moderierten Dialog mit den Stakeholdern zu verzichten, sofern es das Projekt zulässt.
Selbstverständlich können aber auch die oben genannten Methoden zum Einsatz gebracht und natürlich auch kombiniert werden.

Der Moderator sollte beim ersten Meeting zunächst ein Grundverständnis der Softwarekartographie vermitteln, damit alle Beteiligten eine Vorstellung davon haben, wie das angestrebte Projektergebnis aussieht. Die Verwendung des auf dem Standard IEEE 1471

[IE00] aufbauenden Modells [LMW05b] (siehe auch Kapitel 2.2) ist eine Möglichkeit, einen grundlegenden Begriffsapparat zur Verfügung zu stellen. Wird dieses Modell herangezogen, muss, wie Erfahrungen im praktischen Teil gezeigt haben, vor allem auch detailliert auf die Begriffe Concern und Question und deren Unterschied eingegangen werden, da diese auf den ersten Blick nicht unbedingt trennscharf sind. Um von vorneherein Unklarheiten und Diskussionen ausräumen zu können ist es hilfreich den Stakeholdern einfache Regeln an die Hand zu geben:

- Concerns sprechen Systemeigenschaften wie Performanz, Stabilität, Sicherheit, etc. an.
- Concerns beziehen sich auf das betrachtete System (die Anwendungslandschaft) und nicht auf Architekturbeschreibungen, d.h. die Softwarekarten.
- Concerns sollten keine speziellen Kennzahlen nennen.

In den meisten Fällen werden sich Stakeholder vor den entsprechenden Meetings nicht genau überlegt haben, was sie sich von den Softwarekarten erhoffen. Deshalb ist es die Aufgabe des Moderators die Stakeholder zum Nachdenken zu animieren. Dies kann zum Beispiel durch gezieltes Nachfragen, oder der Präsentation von Beispielkarten und den Aspekten der Softwarekartographie geschehen.

Das Erheben von Anforderungen (Requirements) ist keine triviale Aufgabe, die sich oft mühsam und zeitaufwendig gestaltet und einige Gefahren birgt. Fehler während des Requirement-Engineerings sind in der Regel schwer zu korrigieren und deshalb teuer, da wie oben beschrieben, ggf. das komplette Informationsmodell (siehe hierzu Kapitel 3.2.1) überarbeitet werden muss, was sich negativ auf die Datenerhebung (siehe Kapitel 3.2.2) und bereits gezeichnete Karten auswirken kann. Eine ganze Reihe von Faktoren erschwert das Erheben von Anforderungen, zum Beispiel:

- Unklare Zielvorstellungen
- Hohe Komplexität des Projektes und seiner Aufgabenstellung
- Kommunikationsprobleme oder Sprachbarrieren zwischen den Projektbeteiligten
- Sich ständig ändernde Ziele und Anforderungen
- Schlechte Qualität von Anforderungen
    - Unnötige Concerns oder Questions
    - Fehlen wichtiger Concerns und Questions
- Verlieren in Details

Unklare Zielvorstellungen führen letztlich zu schlechten Anforderungen an die Softwarekarten. Nur eine genaue Zieldefinition ermöglicht es gezielt Anforderungen an die Softwarekarten zu erheben. Unklare Zielvorstellungen können zum Beispiel dazu führen, dass wichtige Stakeholder nicht mit einbezogen oder dass die Besprechungen nicht zielorientiert abgehalten werden. Die daraus resultierenden Probleme spricht Kaptitel 3.1.1 an.

Die Erfolgswahrscheinlichkeit eines Projekts sinkt mit steigender Komplexität des Projektes, da diese beherrscht werden muss. Dieser Zusammenhang gilt natürlich auch für die Erstellung von Softwarekarten. Gerade in großen Unternehmen, die über tausende Anwendungssysteme verfügen können, welche über unterschiedliche Technologien miteinander vernetzt sind, gestaltet sich die reale IT-Landschaft dermaßen komplex, dass das Risiko besteht, dass aufgrund der Komplexität wichtige Aspekte von den Stakeholdern übersehen und somit nicht im Rahmen der Anforderungsanalyse erhoben werden. Eine große, unüberschaubare IT-

Landschaft birgt darüber hinaus auch Risiken hinsichtlich der Datenerhebung und einer sinnvollen Darstellung der Sachverhalte.

Im Vordergrund des Requirements-Engineerings steht immer eine Form von Kommunikation, da Stakeholder ihre Interessen mitteilen müssen. Kommunikationsprobleme können die Anforderungserhebung erschweren. Darüber hinaus ist die natürliche Sprache, auch bei gewissenhafter Anwendung, nicht eindeutig, missverständlich und nicht widerspruchsfrei [Ru02]. Außerdem sind die Stakeholder in der Regel in den verschiedensten Bereichen des Unternehmens tätig und wenden ggf. einen berufsspezifischen Wortschatz an. Vor diesem Hintergrund ist es wichtig, Begrifflichkeiten genau zu klären, um Missverständnisse ausräumen zu können.

Während des Projektverlaufes ist es ein normaler Prozess, dass beteiligte Personen dazulernen und neue Perspektiven auf die zu behandelnde Problematik erlangen. Dies kann zu neuen Anforderungen führen und andere überflüssig machen. Eine Änderung bereits erhobener Anforderungen während des Requirements-Engineerings kann dazu führen, dass andere, ggf. validierte Anforderungen überarbeitet werden müssen, was mit einem Aufwand verbunden ist und weshalb vor der Änderung genau geprüft werden muss, ob diese sinnvoll und notwendig ist. Da sich Änderungen während des Requirements-Engineerings, oder solche die sich später ergeben, sukzessive auf das gesamte Projekt auswirken und mit hohem Aufwand verbunden sein können, sollten sie mit entsprechender Vorsicht behandelt werden. Das Vorgehen bei der Projektdurchführung sollte auf der anderen Seite aber auch flexibel genug sein, um derartige Änderungen berücksichtigen zu können.

Das Team welches mit der Erstellung der Softwarekarten beauftragt ist, ist darauf angewiesen sich an den Anforderungen an die Karten zu orientieren und wird versuchen diese nach Möglichkeit zu erfüllen. Anforderungen schlechter Qualität führen mit großer Wahrscheinlichkeit zu schlechten Karten. Unnötige Concerns und Questions können die Prozessdurchführung bremsen und unnötige Kosten verursachen, wenn zum Beispiel für sie ein hoher Aufwand bei der Datenerhebung betrieben worden ist.
Auf das Fehlen wichtiger Concerns und Questions und die damit verbundenen Auswirkung wurde in Kapitel 3.1.1 bereits näher eingegangen.

Bei der Anforderungsanalyse ist stets ein Auge auf die Machbarkeit und die Ziele des gesamten Projektes zu werfen. Das unnötige verlieren in Details führt zu Zeitverlusten und ist somit zu vermeiden. Es ist deshalb ein realisierbarer Rahmen zu finden, in dem sich das Projekt bewegen soll. Ein hoher Detailgrad bei der Einbeziehung von Infrastrukturdiensten oder aber bei der Beschreibung von Prozessen kann zu einer rapiden Aufwandsexplosion führen, da entsprechende Daten erhoben werden müssen und zur Zeit mit geeigneten Werkzeugen zur automatischen Erstellung von Softwarekarten nicht in jeder Situation gerechnet werden kann. Mit dem Einsatz entsprechender Werkzeuge in der Zukunft, sinkt der Aufwand für die Entwicklung von Softwarekarten, wodurch ein höherer Detailgrad mit tragbaren Kosten erreicht werden könnte.

Neben der hier vorgestellten Auswahl allgemeingültiger Faktoren, die keinen Vollständigkeitsanspruch erhebt, gibt es natürlich auch immer projektspezifische Risiken im Rahmen des Requirements-Engineerings denen man sich bewusst werden sollte.

Stakeholder haben oftmals keine genaue Vorstellung davon, was Softwarekarten leisten und wozu sie nützlich sind. Eine genaue Zieldefinition und eine hohe Qualität von Concerns und Questions ist aber, wie oben erläutert, eine Voraussetzung für einen zufrieden stellenden

Projektabschluss. Deshalb ist der häufige und enge Dialog mit den Stakeholdern unbedingt erforderlich. Allen Beteiligten sollte klar sein, dass das Requirements-Engineering kein linearer, sondern ein inkrementeller Prozess ist, bei dem die Qualität und Vollständigkeit der erhobenen Anforderungen allmählich zunimmt.
An den ersten Meetings sollten nach Möglichkeit alle Stakeholder beteiligt sein, da rechtzeitig ein Konsens über Begrifflichkeiten getroffen werden sollte. Ein Glossar, an das sich alle Beteiligten halten und das während der Gespräche ständig erweitert wird ist empfehlenswert. Die bereichsübergreifenden Diskussionen können Concerns und Questions zu Tage fördern, die in Einzelgesprächen verloren gehen könnten und sollten regelmäßig wiederholt werden.
Nach der Erhebung der Concerns werden diese in weiteren Meetings verfeinert. Dies kann zum Beispiel in Anlehnung an [LMW05b] geschehen, indem die Questions zu den einzelnen Concerns erhoben werden. Dadurch wird der für die jeweiligen Concerns benötigte Informationsbedarf festgelegt.
In der Regel wird es jedoch kaum gelingen, Concerns und Questions getrennt zu erheben. Realistisch betrachtet wird sich eine Fülle von Informationen ansammeln, aus denen Concerns und Questions identifiziert werden müssen.
Selbstverständlich müssen die erhobenen Concerns und Questions in angemessener Form dokumentiert und gepflegt werden. Dabei sollten sie näher beschrieben, priorisiert und Stakeholdern zugeordnet werden.
Die folgenden Tabellen verdeutlichen beispielhaft, wie ein Dokument aussehen könnte, welches erhobene Concerns und Questions dokumentiert. Dabei handelt es sich um Anforderungen, die im Rahmen des praktischen Teils erhoben wurden. Die aufgeführten Stakeholder sind in Kapitel 3.1.1 näher beschrieben.

Concerns				
Name	**Beschreibung**	**Stakeholder**	**Priorität**	**Id**
Sicherheit	Die Karten sollen dabei unterstützen Sicherheitslücken zu lokalisieren. Das Interesse richtet sich hierbei auf die Aktualität von Sicherheitssoftware und Patches. Ziel der Sicherheitsbemühungen ist es, alle unbefugten Aktionen zu unterbinden, zum Beispiel Datenzugriff oder -Modifikation durch Unbefugte, etc.	Betrieb	Hoch	1
Verfügbarkeit	Die Karten sollen dabei helfen Ausfallzeiten zu minimieren, in dem sie Aufschluss darüber geben welche Systeme bei einem Ausfall eines Systems betroffen sind und wer in einer derartigen Situation zu informieren ist. Unter Systemen werden hier sowohl Hardwaresysteme als auch Anwendungssysteme verstanden.	Betrieb, Applikationen	Hoch	2
Redundanz	Die Karten sollen dabei helfen, Redundanz bei der Abarbeitung von Geschäftsprozessen in der Anwendungslandschaft zu identifizieren Man verfolgt das Ziel, eine möglichst gradlinige, optimale Prozessabarbeitung zu erreichen. Der Fokus liegt dabei auf redundanten Anwendungssystemen.	Prozess-management	Mittel	3

Tabelle 3 Beispieldokument zur Erfassung von Concerns

Questions					
Name	**Beschreibung**	**Stakeholder**	**Concern**	**Priorität**	**Id**
Patchaktualität	Welche Server-Betriebssysteme verfügen nicht über aktuelle Patches?	Betrieb	Sicherheit	Hoch	1
Server_AWS	Auf welchen Servern laufen welche Anwendungssysteme	Betrieb, Applikationen	Verfügbarkeit	Hoch	2
Schnittstellen	Welche Applikationen kommunizieren über welche Schnittstellen miteinander? Die Karten sollen die Vernetzung der AWS darstellen. Dabei sollen neben dem eigentlichen Datenfluss auch Schnittstellen über Konnektoren dargestellt werden. Es sollte ersichtlich sein, welche Schnittstellentypen verwendet werden und welche Daten zwischen Schnittstellen fließen.	Applikationen Prozess-management	Verfügbarkeit	Hoch	3
Redundante-AWS/Daten	Welche AWS/Daten sind redundant? Es soll ersichtlich sein, welche Anwendungs-systeme oder Datenströme redundant sind.	Prozess-management	Redundanz	Mittel	4

Tabelle 4 Beispieldokument zur Erfassung von Questions

3.1.3 Validation der Concerns und Questions

Auf jeden Fall sollten vor Abschluss der Analyse-Phase die Concerns und Questions bewertet werden. Neben der Priorisierung sollte untersucht werden, ob Concerns oder Questions hinsichtlich des Projektes gänzlich überflüssig, widersprüchlich, oder nicht realisierbar sind. Concerns und Questions können beispielsweise überflüssig werden, wenn sich herausstellt, dass in absehbarer Zeit neue Technologien eingeführt werden sollen. So könnte ein vorläufiges Ergebnis der Erhebung der Anforderungen sein, dass Karten den aktuellen Patchstand von Systemen zeigen und Informationen darüber liefern sollen, welche Systeme aktuelle Patches benötigen. Durch die anstehende Einführung eines Patch-Management-Systems werden diese Anforderungen, bzw. entsprechende Visualisierungen durch Software-karten, unter Umständen überflüssig.
Es ist auch möglich, dass Concerns und Questions bei näherer Betrachtung keine Rolle spielen, da bei der Erhebung nicht alle Aspekte der Realität ausreichend berücksichtigt wurden. Denkbar wären beispielsweise, dass Softwarekarten zeigen sollen, ob ein bestimmtes Anwendungssystem von Hardwaresystemen unterstützt werden kann oder nicht. Bei näherer Betrachtung ist diese Information nicht relevant, da im betrachteten Unternehmen Hardware und Software von einem Hersteller im Paket erworben werden und sich somit die Frage nicht stellt, ob Anwendungssysteme unterstützt werden.
Bei der Bewertung der Concerns und Questions sollte eine Aufwandsabschätzung mit einbezogen werden. Dabei kann sich herausstellen, dass einige Wünsche der Stakeholder mit den vorhandenen Mitteln im geplanten Projektrahmen nicht machbar sind. Es ist zum Beispiel möglich, dass Softwarekarten die Auslastung von Systemen zeigen sollen. Es macht jedoch

keinen Sinn, die Auslastung über einen in der Vergangenheit liegenden Zeitraum zu mitteln. Um jedoch ständig aktuelle Informationen liefern zu können, bedürfte es einer Vielzahl an Monitorsystemen und entsprechender Werkzeuge, die Karten automatisch generieren. Wenn diese Werkzeuge und Infrastrukturdienste nicht zur Verfügung stehen müssen diese Anforderungen als nicht realisierbar eingestuft werden.
Ein wesentlicher Aspekt bei der Realisierbarkeit ist auch die Tatsache, dass zur Darstellung der IT-Landschaft entsprechende Daten erhoben werden müssen. Alle Concerns und Questions sollten darauf hin überprüft werden, ob genügend Zeit und Mittel zur Verfügung stehen, um die Datenerhebung in einem sinnvollen Rahmen durchführen zu können.
Concerns und Questions, die sich auf Attribute der IT-Landschaft beziehen, welche sich stetig ändern, oder die enorm komplex sind sollten also mit Blick auf die Datenerhebung und Pflege der Karten hinsichtlich der Machbarkeit speziell überdacht werden.

3.1.4 Abschlussbesprechung der Analyse-Phase

Der Abschluss der Analyse-Phase ist der erste wichtige Meilenstein, mit dessen Erreichen das Erheben der Concerns und Questions, soweit im Rahmen des verwendeten Prozessmodells gefordert, abgeschlossen sein sollte. An diesem Punkt ist zu prüfen, ob eine ausreichende Qualität der erhobenen Anforderungen erreicht wurde. Das heißt insbesondere, dass sie im Rahmen des verwendeten Vorgehensmodells eine tragbare Basis bilden. Es ist sicherzustellen, dass Ergebnisartefakte, welche die Concerns und Questions in geeigneter Form auflisten vorhanden sind und eine brauchbare Grundlage für die nächste Phase darstellen. Gegebenenfalls müssen im Dialog mit entsprechenden Stakeholdern Concerns und Questions nacherhoben oder verfeinert, bzw. genauer definiert werden.

3.2 Design-Phase

Nachdem in der Analyse-Phase die Anforderungen an die zu zeichnenden Softwarekarten in Form von Concerns und Questions erhoben wurden, werden in der Design-Phase Karten spezifiziert. Dabei wird festgelegt, welche Karten erstellt werden sollen, welche Concerns diese adressieren, welche Questions die Karten beantworten sollen und wie Informationen auf den Karten darzustellen sind.
Ein Ergebnis der Design-Phase ist das Informationsmodell des Unternehmens für die auf den Karten zu visualisierende Information, welches als Grundlage für die Datenerhebung und als Verankerung der Visualisierung dient.

Die Design-Phase lässt sich in folgende Aktivitäten gliedern:

- Erstellung des Informationsmodells
- Erhebung der zum Zeichnen der Karten benötigten Daten
- Spezifikation von Modellen und Viewpoints
- Abschlussbesprechung

3.2.1 Das Informationsmodell

Ein Informationsmodell für die Softwarekartographie modelliert die Informationsobjekte (Entitäten) eines Unternehmens, die dargestellt werden müssen, um die erhobenen Anforderungen an die Karten adäquat adressieren zu können, mit Entitätstypen und deren Beziehungen (Relationen) untereinander mit Relationstypen.
Ein Anwendungssystem „Finanzbuchhaltung“ würde in einem Informationsmodell selber also nicht auftauchen, sondern wäre eine Instanz des Entitätstyps „Informationssystem“. Dabei beschreiben die Attribute der Entitätstypen die Eigenschaften der modellierten Entitäten. Im objektorientierten Paradigma entspricht ein Entitätstyp einer Klasse und Relationstypen Assoziationen.

Welche Entitäten und Relationen modelliert werden ergibt sich mittelbar aus den erhobenen Anforderungen.
Wie bei jeder Datenmodellierung bietet es sich auch bei der Erstellung des Informationsmodells in der Softwarekartographie an, zunächst ein konzeptionelles Modell zu entwickeln, das später technisch umgesetzt wird. Das Informationsmodell ist also konzeptioneller Natur und sollte objektorientiert entwickelt werden, da die Objektorientierung mächtige Konzepte wie Vererbung und n:m-Beziehungen zur Verfügung stellt und entsprechende Werkzeuge vorhanden sind, welche die Visualisierung des Modells ermöglichen und so auch ihren Beitrag bei dessen Kommunizierbarkeit leisten, wenn allen Beteiligten die graphischen Notationen bekannt sind. Ein objektorientiertes Modell kann darüber hinaus für die Datenerhebung mit wenig Aufwand in ein relationales Datenbankschema transformiert werden. Bei guten objektorientierten Werkzeugen kann dies vollautomatisch geschehen.
Es ist davon abzuraten, auf ein Informationsmodell auf konzeptioneller Ebene zu verzichten und lediglich eine Datenbank in einem Datenbanksystem zu entwickeln, da hier viele Vorteile der Objektorientierung zur Modellierung auf konzeptioneller Ebene nicht zur Verfügung stehen und Visualisierungsmöglichkeiten ggf. erst beschafft werden müssen. Wie bereits oben erwähnt hat das Informationsmodell auch direkten Einfluss auf die Visualisierung in Softwarekarten, da es die entsprechenden Entitätstypen aufführt und diese vor allem in Beziehung zu einander setzt.

Obwohl das zu entwickelnde Informationsmodell stark unternehmensspezifisch ist, können allgemeingültige Referenzmodelle, wie zum Beispiel das Common Information Model [DMT99] herangezogen werden, um den Modellierungsprozess zu beschleunigen. Da diese Informationsmodelle für ein hohes Maß an Wiederverwendbarkeit entworfen wurden und ggf. stark ins Detail gehen, um Vollständigkeitsansprüchen gerecht werden zu können, ist mit einem Anpassungsaufwand zu rechnen, da das zu entwickelnde Informationsmodell auf das Projekt zurechtgeschnitten werden sollte. Deshalb sollte genau geprüft werden, welche übernommene Teile tatsächlich nützlich bzw. realisierbar sind.
Der Anpassungsprozess kann neue noch nicht berücksichtigte Aspekte zu Tage fördern, was hinsichtlich der Realisierbarkeit der Datenerhebung auch mit Gefahren verbunden sein kann, da ein zu hoher Detailgrad den Aufwand der Datenerhebung schnell explodieren lässt. Da die Darstellung der Anwendungslandschaft im Vordergrund steht, sollte wie oben bereits erwähnt, nur so weit wie nötig auf Infrastrukturdienste eingegangen werden.
Das Informationsmodell sollte in enger Zusammenarbeit mit den Stakeholdern, bzw. Fachmännern aus den zu modellierenden Bereichen entwickelt werden. Dabei muss eine Form gefunden werden, in der das Modell kommuniziert werden kann, da unter Umständen Modellierungssprachen wie UML [OMG04] wenig bekannt sind. Eine zusätzlich textuelle Beschreibung aller Entitäten und Beziehungen ist nicht nur für die Kommunizierbarkeit des

Modells sinnvoll, sondern leistet auch einen wichtigen Beitrag zum allgemeinen Verständnis. Das Informationsmodell sollte nach Abschluss des Projektes im Rahmen der Kartenpflege weiter angepasst werden, damit es Änderungen der IT-Landschaft, wie zum Beispiel die Einführung neuer Technologien, gerecht werden kann. Im diesem Sinn kann die Lebensdauer des Informationsmodells lang sein, was letztendlich genaue Beschreibungen der Entitäten und Beziehungen notwenig macht, da damit gerechnet werden muss, dass sich das bearbeitende Personal ändert und nicht davon ausgegangen werden kann, dass das Informationsmodell an sich vollständig selbsterklärend ist.
Der besseren Kommunizierbarkeit und Übersichtlichkeit wegen sollte das Informationsmodell geeignet in Module aufgeteilt werden. Dabei kann ein *Core-Modul* den Einstiegspunkt darstellen, in dem die wesentlichen Entitäten zusammengefasst sind und über den zu anderen Modulen navigiert werden kann.

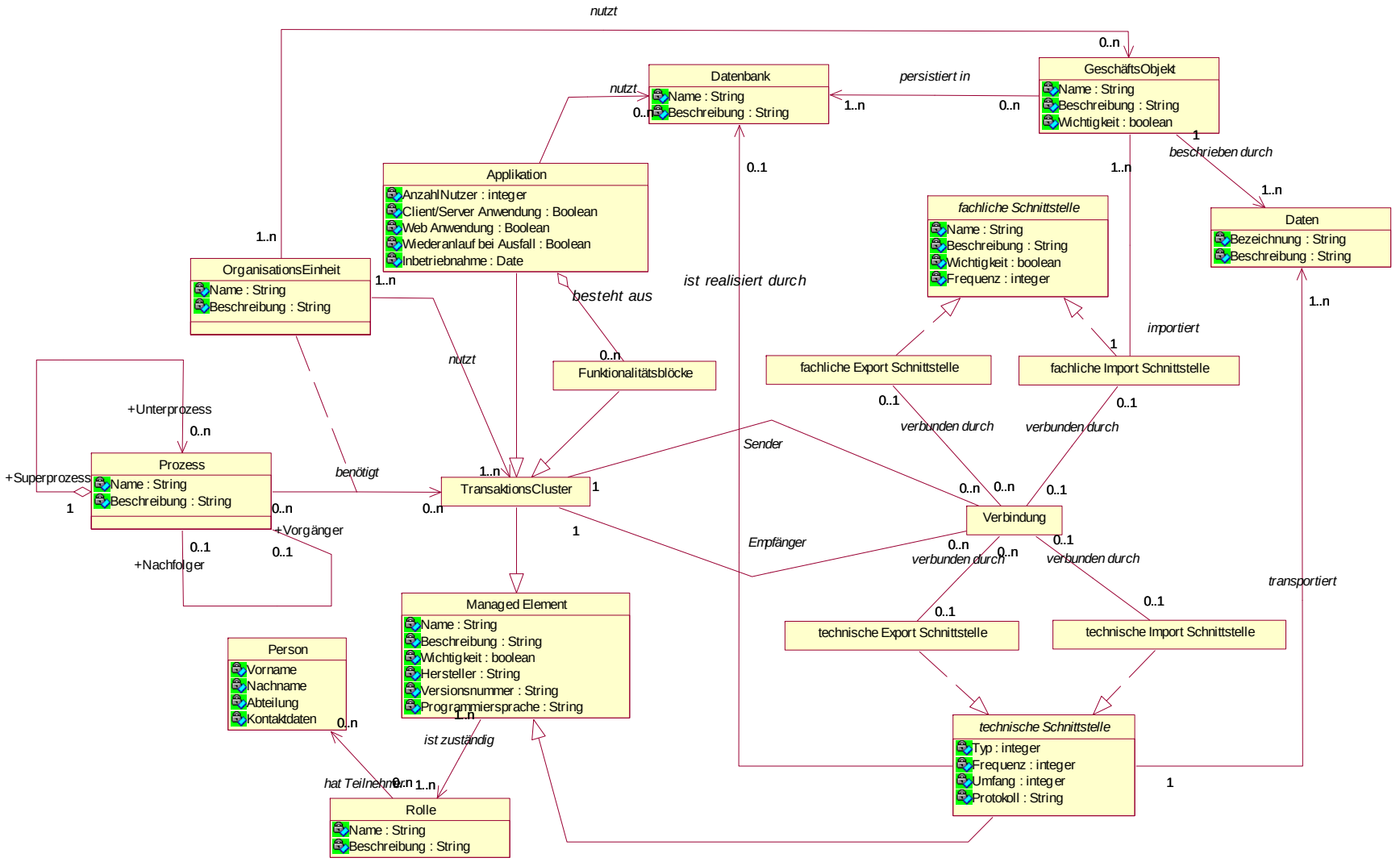


Abbildung 8 Vereinfachtes Informationsmodell der Krones AG (Applikationen)

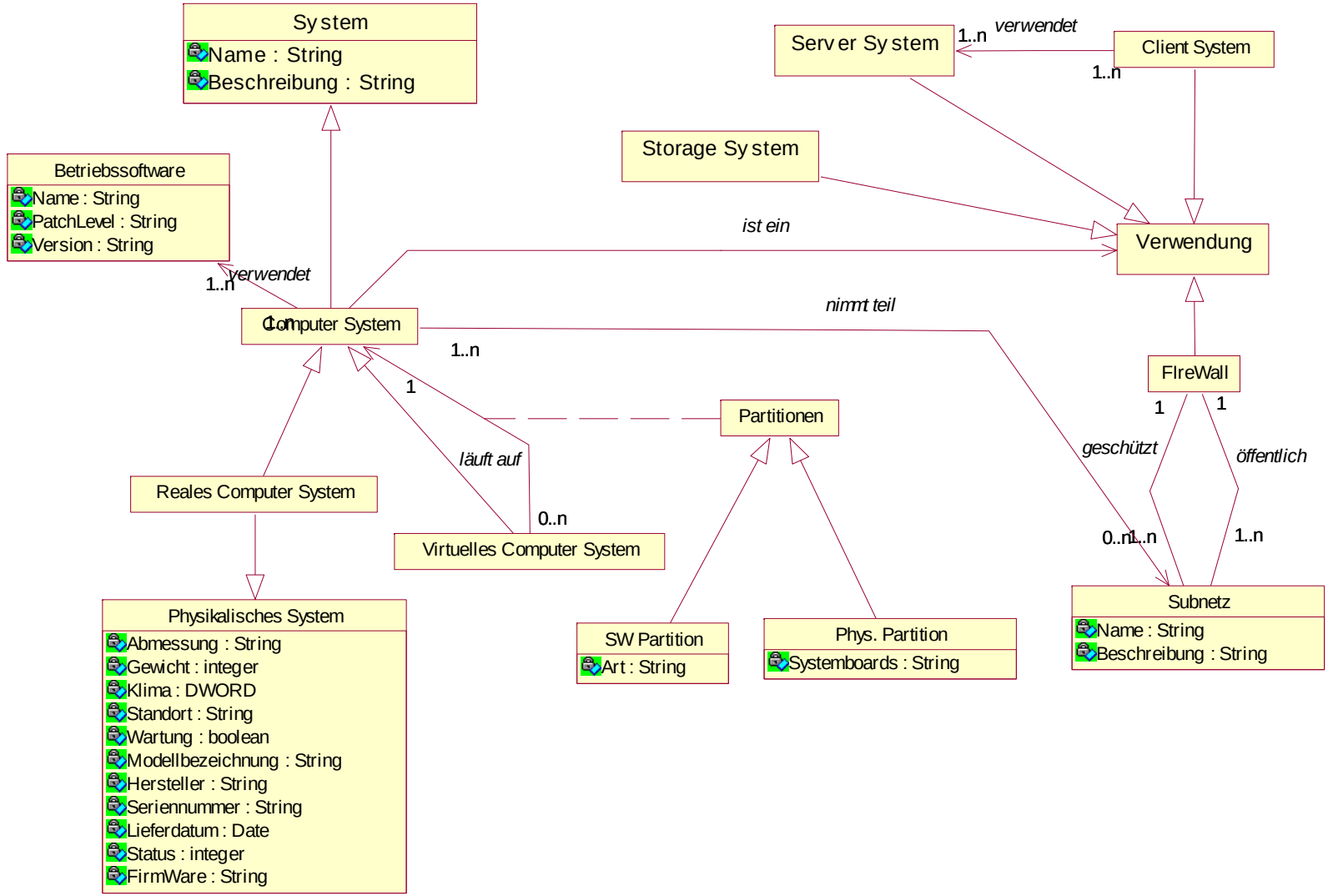


Abbildung 9 Vereinfachtes Informationsmodell der Krones AG (Betrieb)

Die vereinfachten Beispiele in den Abbildungen 8 und 9 sind dem Informationsmodell entnommen, das im Rahmen des praktischen Teils der Arbeit für die Krones AG erstellt wurde.
Abbildung 8 zeigt Entitätstypen, die mit Applikationen in Zusammenhang stehen, d.h. hauptsächlich zur Adressierung der Concerns und Questions des Applikationsstakeholders eingeführt wurden. *Applikationen* bestehen aus *Funktionalitätsblöcken*, Elementen der entsprechenden Applikation, die Funktionalitäten kapseln. *Applikationen* und *Funktionalitätsblöcke* sind von der Klasse *TransaktionsCluster* abgeleitet. Der Begriff *TransaktionsCluster* entstand während der Modellierungsbemühungen in Zusammenarbeit mit der Krones AG und hat seinen Ursprung in der SAP-Welt, in der man im Zusammenhang von SAP-Funktionalitäten von Transaktionen spricht, die in verschiedenen Detailgraden betrachtet zu Modulen zusammengefasst werden. Ein *TransaktionsCluster* modelliert also eine Menge von Funktionalitäten einer *Applikation*. In diesem Sinne ist eine *Applikation* selber ein *TransaktionsCluster* und kann, im Rahmen einer detaillierteren Betrachtung, aus weiteren *Funktionalitätsblöcken* bestehen. In Zusammenhang von SAP spricht man bei den *Funktionalitätsblöcken* auch von Modulen. Diese Modellierung erlaubt es zum Beispiel SAP nicht als Black-Box betrachten zu müssen, was für das praktische Projekt notwendig war, da insbesondere Schnittstellen zwischen SAP-Modulen und anderen Applikationen dargestellt werden sollten.
Ein *TransaktionsCluster* ist von der Klasse *ManagedElement* abgeleitet. Mit *Managed-Element* sind Rollen assoziiert. Ein *ManagedElement* ist also ein Softwareobjekt, dem zum Beispiel ein Betreuer zugeordnet ist.
TransaktionsCluster können über Schnittstellen mit anderen *TransaktionsClustern* kommunizieren. In diesem Fall besteht eine *Verbindung* zwischen *TransaktionsClustern*, bei der ein *TransaktionsCluster* als Sender und einer als Empfänger agiert. Eine Verbindung besteht aus *Import-* und *Exportschnittstelllen*. Dabei ist die Exportschnittstelle am Sender-*TransaktionsCluster* und die Importschnittstelle am Empfänger-*TransaktionsCluster*.
Auf eine Schnittstelle gibt es eine *technische* und eine *fachliche* Sicht. Während die technische Sicht die technischen Eigenschaften der Schnittstelle, wie zum Beispiel die verwendete Technologie, beschreibt, kann durch die fachliche Sicht beschrieben werden über welche fachlichen Eigenschaften die Schnittstelle verfügt. Fachlich betrachtet transportiert eine Schnittstelle Geschäftsobjekte, die technisch betrachtet durch Daten beschrieben werden, welche wiederum über die technische Schnittstelle transportiert werden.
Einer technischen Schnittstelle können verschiedene Technologien zugrunde liegen. So kann eine technische Schnittstelle zum Beispiel durch eine *Datenbank* realisiert sein. Dabei schreibt der Sender-*TransaktionsCluster* Daten in die *Datenbank*, die dann von Empfänger-*TransaktionsClustern* ausgelesen werden können. In der Realität werden technische Schnittstellen durch Schnittstellensoftware realisiert, die in der Regel betreut werden muss. Aus diesem Grund ist die Klasse *technische Schnittstelle* von *ManagedElement* abgeleitet.
Die Modellierung der Verbindung mit *fachlichen* und *technischen Export-* und *Import-schnittstellen* orientiert sich weitestgehend an [MW04b].
Zur Durchführung von *Prozessen* kommen *Applikationen* bzw. *Funktionalitätsblöcke* zum Einsatz. Prozesse bestehen aus Prozessschritten, die wiederum als *Prozesse* modelliert sind. Außerdem sind Prozesse bzw. Prozessschritte geordnet. Deshalb haben *Prozesse Nachfolger* und *Vorgänger*. *Prozesse* werden von *Organisationseinheiten* durchgeführt, die als Assoziationsklasse modelliert wurden, welche an der Assoziation zwischen *Prozess* und *TransaktionsCluster* „hängt“. Diese Modellierung macht es möglich zu beschreiben, dass ein *TransaktionsCluster* in einer *Organisationseinheit* zur Durchführung eines *Prozessschrittes* benötigt wird.

Dieser Teil des Informationsmodells adressiert Concerns und Questions der Stakeholder des Bereichs Applikationen. Die detaillierte Modellierung der Schnittstellen hat beispielsweise seinen Ursprung in der Forderung nach einer hohen Verfügbarkeit der Systeme. Durch einen Schnittstellenplan wird schnell deutlich welche Applikationen im Fehlerfall, oder bei geplanten Änderungen an der Anwendungslandschaft betroffen sind. Die mit Applikationen assoziierten Rollen machen es möglich, verantwortliches Personal zu hinterlegen, das bei Problemen mit der entsprechenden Applikation so, durch Hilfe der Karten, schnell informiert werden kann.
Die Dokumentation der Schnittstellen ist hinsichtlich der Datenpfade auch für das Prozessmanagement von großem Interesse, wie auch die Information, welche Applikationen in welchen Organisationseinheiten Verwendung finden, um bestimmte Prozessschritte durchführen zu können, da diese Stakeholder an einer geradlinigen Abarbeitung der Prozesse interessiert sind.

Abbildung 9 zeigt den Teil des Informationsmodells, der für den Stakeholder „Betrieb“ entwickelt wurde. Hier steht die Modellierung der Hardwarelandschaft der Krones AG im Vordergrund, wobei die Herausforderung darin lag, *Computer Systeme* so zu modellieren, dass sie sowohl real als auch virtuell sein können. *Reale Computer Systeme* sind physikalische Maschinen, die physikalische Eigenschaften haben. Deshalb erbt die Klasse *Reales Computer System* sowohl von *Computer System*, als auch von *Physikalisches System*. *Virtuelle Computer Systeme* haben keine physikalische Ausprägung und laufen auf *Partitionen* von *Realen Computer Systemen* oder *Virtuellen Computer Systemen*. *Computer Systemen* sind verschiedene Verwendungszwecke zugeordnet. So können *Computer Systeme* als *Server System*, *Client System*, oder *Storage System* verwendet werden.
Auch im Bereich Betrieb spielen Aspekte der Verfügbarkeit eine wichtige Rolle. Bei Ausfällen eines *Server Systems* muss schnell reagiert werden. Dabei müssen betroffene Anwendungssysteme und Infrastrukturdienste identifiziert und Verantwortliche informiert werden. Dieser Teil des Informationsmodells adressiert zum Beispiel die Question nach den Abhängigkeiten zwischen Hardwaresystemen und Anwendungssystemen bei der Krones AG.

3.2.2 Datenerhebung

Die Datenerhebung ist einer der aufwendigsten Aktivitäten bei der Erstellung von Softwarekarten und birgt deshalb große Projektrisiken in sich. Aufgrund der hohen, mit der Datenerhebung verbundenen Kosten spielt die Datenerhebung bei der Kosten-Nutzen-Schätzung für Softwarekarten eine große Rolle. In diesem Kontext sind auch zur Datenerhebung eingesetzte Werkzeuge, wie zum Beispiel Systems Management- oder Monitoring-Tools relevant.
Der Aufwand der Datenerhebung kann schnell unwirtschaftlich hoch werden, was bei der Erstellung des Informationsmodells berücksichtigt werden sollte. Es ist empfehlenswert frühzeitig für alle Bereiche des Datenmodells Daten beispielhaft zu erheben, um dessen Umsetzbarkeit sicherstellen und die Machbarkeit der Datenerhebung besser einschätzen zu können, wobei kritische Teile des Datenmodells bevorzugt behandelt werden sollten.
Die Bereiche des Unternehmens, in denen der Aufwand für die Datenerhebung anfällt, unterscheiden sich in der Regel von denen, die direkt von den Karten profitieren. Personell betrachtet heißt das, dass Personen, die mit der Datenerhebung betraut sind keinen unmittelbaren Nutzen aus ihrer „undankbaren“ Aufgabe ziehen und ggf. nicht einmal über das Projekt informiert sind, was aus psychologischen Gründen zu Motivationsdefiziten und somit zu einer schlechteren Datenqualität führen kann.

Diese Problematik kommt insbesondere bei einer dezentralen Datenerhebung zum Tragen, bei der beispielsweise durch entsprechende Fragebögen Daten erhoben werden. Bei einer zentralisierten Datenerhebung ist eine Person oder Personengruppe mit der Erhebung der Daten beauftragt, die beispielsweise Daten über persönliche Gespräche und gezielte Befragungen einsammelt. Die zentrale Datenerhebung ermöglicht eine frühe Kontrolle der Daten, wodurch deren Güte sichergestellt werden kann. Die Daten werden früh in das eigentliche Datenschema eingepflegt und es kann schnell auf Probleme reagiert werden. Bei komplexen Projekten ist jedoch eine zentrale Datenerhebung mit hohen Kosten verbunden. Die Durchführung der Datenerhebung lag bei dem praktischen Teil der Arbeit in den Händen der Krones AG und wurde zumindest teilweise mit Hilfe von Fragebögen realisiert.
Wird eine dezentrale Datenerhebung durchgeführt ist unbedingt genau zu beschreiben, welche Daten erhoben werden sollen. Eine solche Beschreibung schließt vor allem eine genaue textuelle Dokumentation der zu erhebenden Entitäten und deren Attribute mit ein. Dabei muss festgelegt werden worum es sich bei den Entitäten handelt und welche Werte die Attribute annehmen können, was besonders bei nichtnumerischen Attributen eine große Rolle spielt. Um Rückfragen und Konvertierungsaufwand zu minimieren sollten alle Attribute mit Formatangaben versehen werden. Wenn den Mitarbeitern der Umgang mit Datenbank-systemen zuzutrauen ist, ist das Einpflegen der Daten in zur Verfügung gestellte Datenbanken denkbar, um sicherzustellen, dass die Information in der vom Informationsmodell geforderten Weise geliefert wird. Hierbei können natürlich auch Eingabemasken zur Verwendung kommen, die den Umgang mit Datenbanken erheblich vereinfachen können. Wird bei der Erhebung der Daten mit Datenbanken gearbeitet, gestaltet sich natürlich auch das Einpflegen der Daten in die eigentliche Datenhaltung einfacher oder wird gänzlich überflüssig.

Im Idealfall erfolgt die Datenerhebung nach Erstellung des Informationsmodells bzw. des daraus abgeleiteten Datenmodells. Außerdem können relevante Daten auch schon vor Beginn des Projektes vorhanden sein. Es ist in beiden Fällen unbedingt erforderlich die Daten in das Datenschema einzupflegen, da ihre Integrität und Konsistenz sichergestellt werden muss. Es ist ratsam, vorhandene Daten hinsichtlich der Aufwandsschätzung genau auf ihre Qualität zu prüfen. Besonders Daten, die im Rahmen anderer Projekte erhoben wurden und sich nicht in einer Datenbank befinden, sondern beispielsweise in Form einfacher Tabellen gehalten werden, sind diesbezüglich genau zu untersuchen.

3.2.3 Spezifikation von Modellen und Viewpoints

Die Spezifikation der Modelle und der Viewpoints kann zeitgleich mit der Datenerhebung durchgeführt werden, da hierzu keine Daten benötigt werden. Zunächst muss festgelegt werden welche Karten erstellt werden sollen, von welchem Typ diese Karten sein sollen, welche Concerns und Questions diese Karten beantworten und welche Stakeholder, bzw. Analysten mit ihnen arbeiten.
Die folgenden Tabellen zeigen beispielhaft, wie Dokumente aussehen könnten, welche Viewpoints und Modelle spezifizieren.

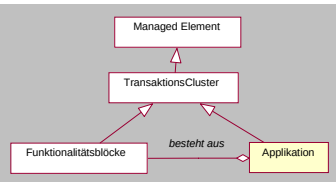
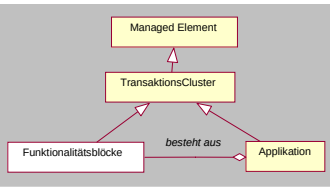
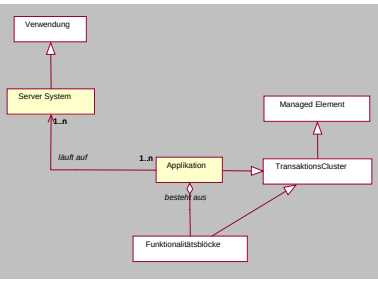
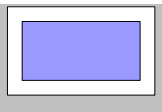
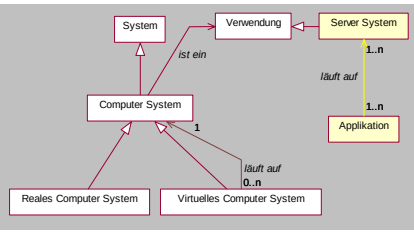
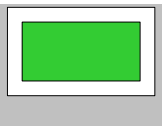
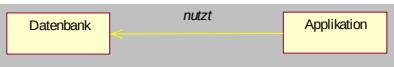
Viewpoint	
VP-Nummer	0
Name	VP_Applikationen
Ersteller	Lauschke
Stakeholder	Applikationen
Analysten	Applikationen
Concerns	Verfügbarkeit von Applikationen
Questions	• Von welchen HW-Systemen hängen bestimmte Applikationen ab? • Von welchen anderen Applikationen hängen bestimmte Applikationen ab? • Wer ist für bestimmte Applikationen zuständig? • Über welche technischen und fachlichen Schnittstellen kommunizieren Applikationen miteinander?
Beschreibung	Dieser Viewpoint legt eine Sicht auf die IT-Landschaft fest, welche Informationen beinhaltet, die für die Stakeholder „*Applikationen*“ von Interesse sind. Aus diesem Blickwinkel sollen nur Applikationen, deren Abhängigkeiten untereinander und deren Schnittstellen sichtbar sein.
Begründung	Für die Stakeholder „*Applikationen*“ ist nur die Sicht auf die Anwendungslandschaft von Interesse. Andere Aspekte, die zum Beispiel für die Stakeholder „*Betrieb*“ eine Rolle spielen sind hier nicht wichtig. Die Fragen der Stakeholder „*Applikationen*“ lassen sich gut in einem Viewpoint bearbeiten, da es kaum Überschneidungen mit den Interessen anderer Stakeholder gibt.
Dokumente	Concerns und Questions; Informationsmodell
View	V_Applikationen
Legenden	L_Applikationen

Tabelle 5 Beispieldokument zur Beschreibung eines Viewpoints

Alle Spezifikationsdokumente sollten einen eindeutigen Namen und eine eindeutige ID enthalten. Die ID ist numerisch und könnte bei sehr komplexen Projekten in Zusammenhang mit Datenbanksystemen Verwendung finden oder lediglich zu einer einfachen Referenzierung dienen. Namen sollten nach einem festgelegten Muster vergeben werden. Der Name eines Dokumentes steht dabei in direktem Bezug zu dem Inhalt des Dokumentes. Bei den Beispielen charakterisieren Großbuchstaben das spezifizierte Objekt gefolgt von einem „_“ und dem eigentlichen Namen. „VP“ steht beispielsweise für Viewpoint, während „L“ für Legende steht.

Die Spezifikation eines Viewpoints, wie zum Beispiel in Tabelle 5 gezeigt, listet die zugehörigen Concerns und Questions, eine Beschreibung des Viewpoints, eine Begründung, warum der Viewpoint genau in dieser Art gebildet wird, mit dem Viewpoint in Verbindung stehende Dokumente und die dem Viewpoint zugeordnete View. Außerdem ist es ratsam den Ersteller des Viewpoints und die Stakeholder zu vermerken, um Rückfragen zu erleichtern. Hier ist es denkbar, entsprechende Kontaktinformationen zu hinterlegen.
Unter „Legenden“ werden die Legenden gelistet, für die eine dem Viewpoint entsprechende View eine Softwarekarte enthalten muss.

Legende Applikationen	
L-Nr	0
Name	L_Applikationen
Ersteller	Lauschke
Modell	M_Applikationen
Begründung	In dieser Karte werden Informationen über Applikationen dargestellt. Der Kartengrund soll aus Organisationseinheiten bestehen, auf denen Applikationen, die dort verwendet werden aufgebracht werden. Über den Applikationen sollen in einer Schicht die Verbindungen zwischen den Applikationen aufgebracht werden. In einer weiteren Schicht sollen Informationen über Schnittstellen untergebracht werden.

Name	**Darstellung/ Schicht**	**Beschreibung**	**Enthält**	**Informationsmodell**
Applikation	AppXYZ Kartengrund(0)	Repräsentiert eine Applikation	Name Anzahl Nutzer Server Datenbank	
Anzahl Nutzer	 Kartengrund(0)	Muss in einer Applikation enthalten sein. Gibt die Anzahl der Nutzer der Applikation an	Zahl	
Server	Name Kartengrund(0)	Muss in einer Applikation enthalten sein. Repräsentiert den Server, auf dem die Applikation läuft	Name	
Datenbank	Name Kartengrund(0)	Muss in einer Applikation enthalten sein. Repräsentiert eine von der Applikation verwendete Datenbank	Name	
Verschachtelungsregeln				
Server in Applikation		Stellt dar, dass die *Applikation* auf dem *Server System* läuft.		
Datenbank in Applikation		Stellt dar, dass die *Applikation* die *Datenbank* verwendet.		

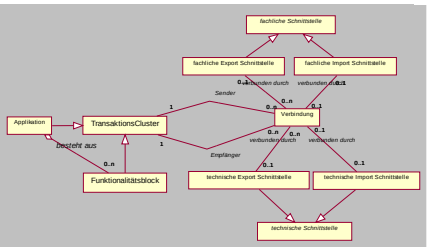
Verbinder				
Verbindung	Verbinder (1)	Repräsentiert die Assoziationen „*Sender*“ und „*Empfänger*“ zwischen *Transaktions-Cluster* und *Verbindung*.		
	Verbinder (1)	Repräsentiert die Assoziation „*Sender*“ zwischen *Transaktions-Cluster* und *Verbindung*.		
	Verbinder (1)	Repräsentiert die Assoziation „*Empfänger*“ zwischen *Transaktions-Cluster* und *Verbindung*.		

Tabelle 6 Beispieldokument einer Legende für eine Softwarekarte

Neben dem verwendeten Symbol wird in Tabelle 6 in der Spalte „Darstellung/Schicht“ auch spezifiziert in welcher Schicht sich das Gestaltungsmittel befindet. Die Zahl in den Klammern hinter dem Namen der Schicht definiert eine Ordnung auf den Schichten. Eine Schicht, der eine kleinere Zahl zugeordnet ist befindet sich in der Karte unter einer, der eine größere zugeordnet wurde und kann demnach von dieser Schicht teilweise verdeckt werden.
Um sowohl eine exakte Legenden-Information als auch eine schnell zugängliche, für die Nutzung der Karten bestimmte Legende festzuhalten, ist es ratsam Legenden von Softwarekarten zweifach auszuführen. Neben der in Tabelle 6 abgebildeten ausführlichen Legende gibt es noch eine kürzere Legende, die auf der eigentlichen Karte abgebildet ist. In der ausführlichen Legende wird die Zuordnung von Gestaltungsmitteln zu Entitätstypen fixiert. Die Existenz eines Entitätstypen und dessen Attribute ergeben sich dabei aus dem Informationsmodell. Um dies zu verdeutlichen kann zu jeder Entität der entsprechende Ausschnitt des Informationsmodells dargestellt werden. In der beispielhaft gezeigten ausführlichen Legende in Tabelle 6 sind Klassen des Informationsmodells, die direkt mit dem spezifizierten Gestaltungsmittel in Verbindung stehen gelb eingefärbt. Klassen die lediglich einen Kontext darstellen sind dagegen weiß.
Aus der Legende sollte darüber hinaus ebenfalls hervorgehen welche Attribute in einer Entität dargestellt werden, wie Entitäten in Beziehung gesetzt werden dürfen und wie die Schachtelungs- und Ausrichtungsregeln sind.
Bei der Wahl der Gestaltungsmittel ist zu beachten, dass sie nach Möglichkeit in allen Karten durchgehend verwendet werden. Dies erhöht den Wiedererkennungswert. Vererbungsbeziehungen im Informationsmodell können auch bei der Darstellung aufgegriffen werden. So könnte eine ähnliche Farbgebung oder Form eine Verwandtschaft der Entitäten verkörpern. Selbstverständlich darf die Ähnlichkeit hier nicht zu groß sein.
Die Symbolik sollte wenn immer möglich, intuitiv gewählt werden. Es ist beispielsweise sehr verwirrend für den Benutzer wenn eine kritische Applikation durch einen grünen Schriftzug symbolisiert würde.

Die dargestellten Beispieldokumente sind aus Platzgründen nicht vollständig. Für eine vollständige, zusammenhängende Darstellung sei hier auf den Anhang der Arbeit verwiesen.

3.2.4 Abschlussbesprechung der Design-Phase

Der Abschluss der Design-Phase ist der zweite wichtige Meilenstein bei der Konstruktion von Softwarekarten. Nach Beendigung der Design-Phase (je nach gewählten Vorgehen kann diese Phase auch öfter durchlaufen werden) sollten das Informationsmodell erstellt, die Datenerhebung, zumindest der Kernbereiche, abgeschlossen und die Modelle und Viewpoints vollständig spezifiziert sein. Da die Datenerhebung in der Regel eine der langwierigsten Aufgabenfelder ist, kann sie parallel zu den anderen Aktivitäten weitergeführt werden. Können entsprechende Daten aus Zeitgründen nicht erhoben werden sollte versucht werden ausdrucksstarke Beispieldaten zu erstellen, damit die Durchführung des Projektes nicht ins Stocken gerät und rechtzeitig mit der Konstruktionsphase begonnen werden kann, bei der dann beispielhafte Karten erstellt werden können. Diese Beispielkarten sind wichtig, um eine termingerechte Validation (siehe Kapitel 3.4) garantieren zu können.

3.3 Konstruktions-Phase

In der Konstruktions-Phase werden die Softwarekarten nach den in der Design-Phase spezifizierten Regeln gezeichnet.

3.3.1 Zeichnen von Softwarekarten

Da Werkzeuge zum Zeichnen von Softwarekarten nicht durchgehend ausgereift zur Verfügung stehen kann zum jetzigen Zeitpunkt nicht immer von einer Tool-Unterstützung zur Automatisierung des Zeichenprozesses ausgegangen werden. Die Karten müssen eventuell manuell mit entsprechenden Zeichenprogrammen wie Microsoft Office Visio erstellt werden, was in vielen Unternehmen tatsächlich auch praktiziert wird und auch im praktischen Teil der Arbeit der Fall war. Das manuelle Erstellen von Softwarekarten ist mit einem hohen Aufwand und somit mit Kosten verbunden.
In Visio ist eine Automatisierung mit Hilfe von Makros teilweise möglich. Auf diese Weise kann eine Datenbankanbindung realisiert werden, die lästiges manuelles Kopieren aus der Datenbank überflüssig macht und mögliche Fehlerquellen minimiert.
Bei der Positionierung der grafischen Elemente sollte darauf geachtet werden, dass die Karten im Rahmen der Pflege erweiterbar sein müssen. Deshalb sollten nach Möglichkeit entsprechende Freiräume eingeplant werden.
Die Darstellung ist an die Vorgaben der Viewpoints mit deren Legenden gebunden. Dennoch bleiben gestalterische Freiräume, die genutzt werden sollten, um ein Höchstmaß an Übersichtlichkeit zu erlangen. Bei der Erstellung der Karten kann man sich an einfachen Regeln orientieren, die die Ästhetik der Karten und deren Lesbarkeit erhöhen. So sollten sich Verbinder nach Möglichkeit nicht kreuzen und in etwa gleiche Länge haben [Pu01].

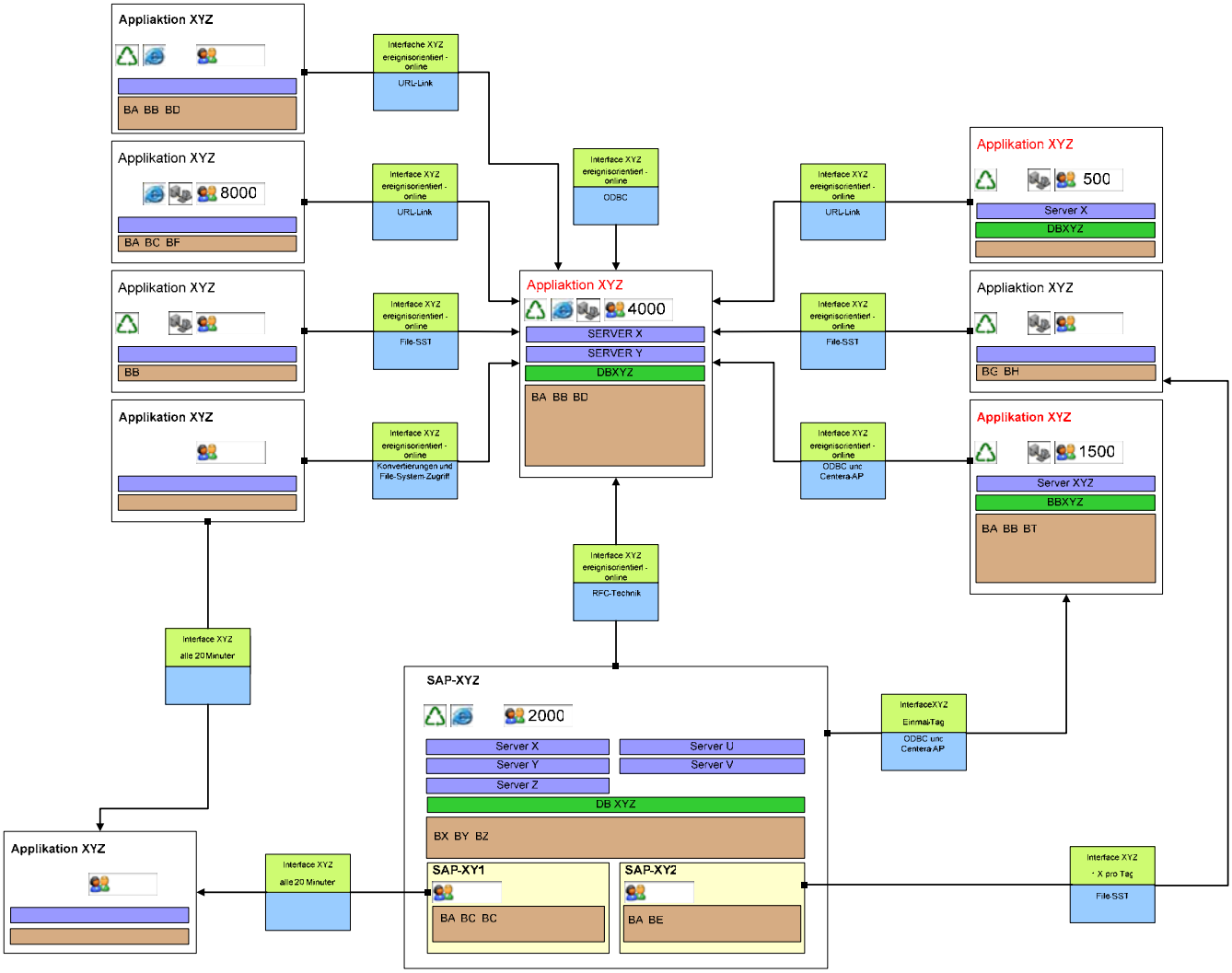


Abbildung 10 Anonymisierte Karte aus dem Bereich Applikationen

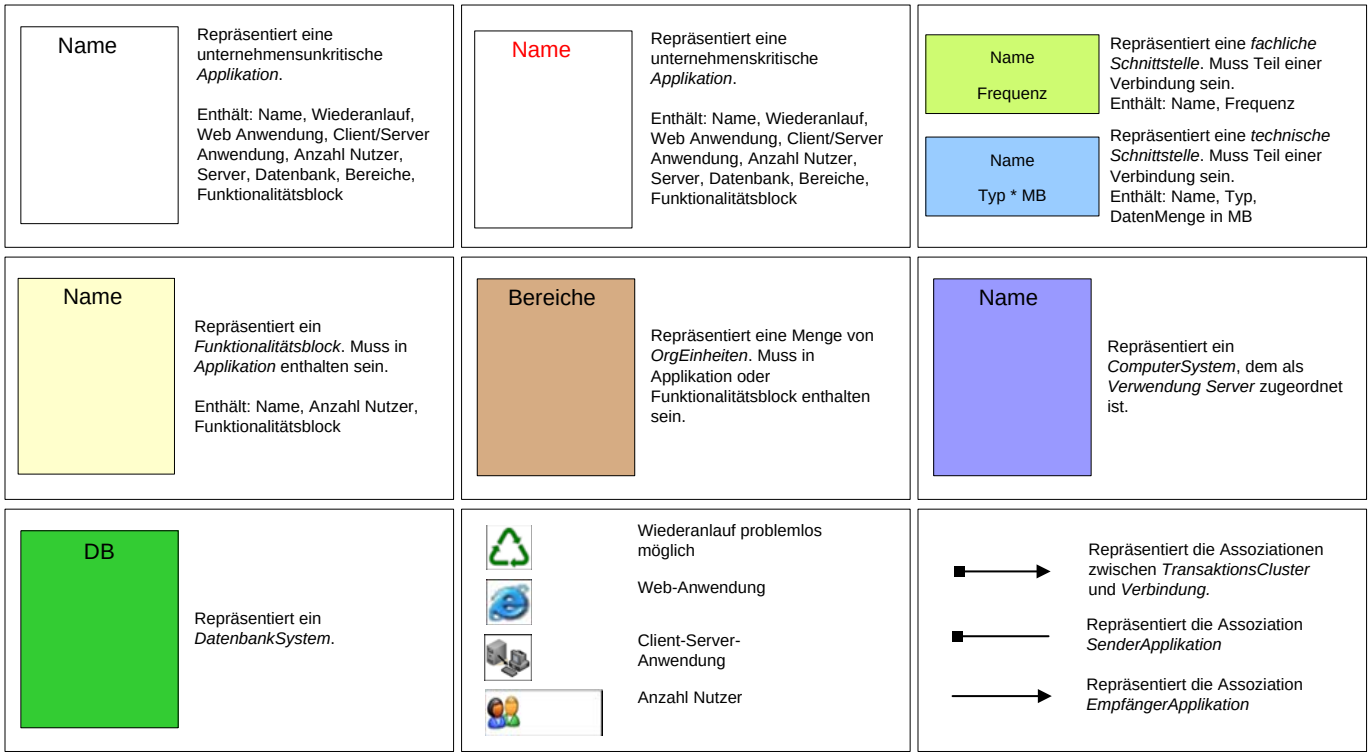


Abbildung 11 Kurzlegende der Softwarekarte Applikationen

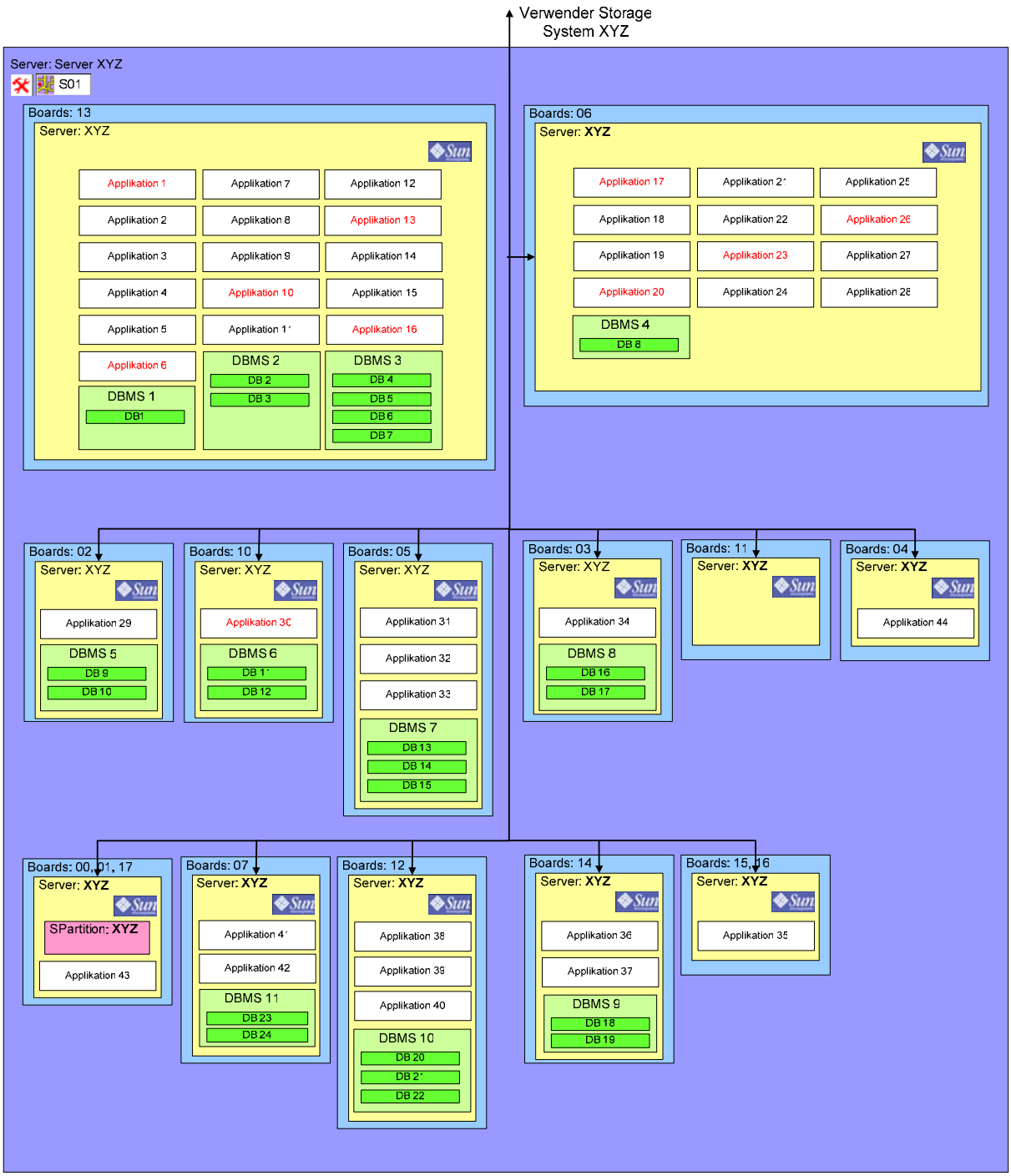


Abbildung 12 Anonymisierte Darstellung eines Servers aus der Softwarekarte Betrieb

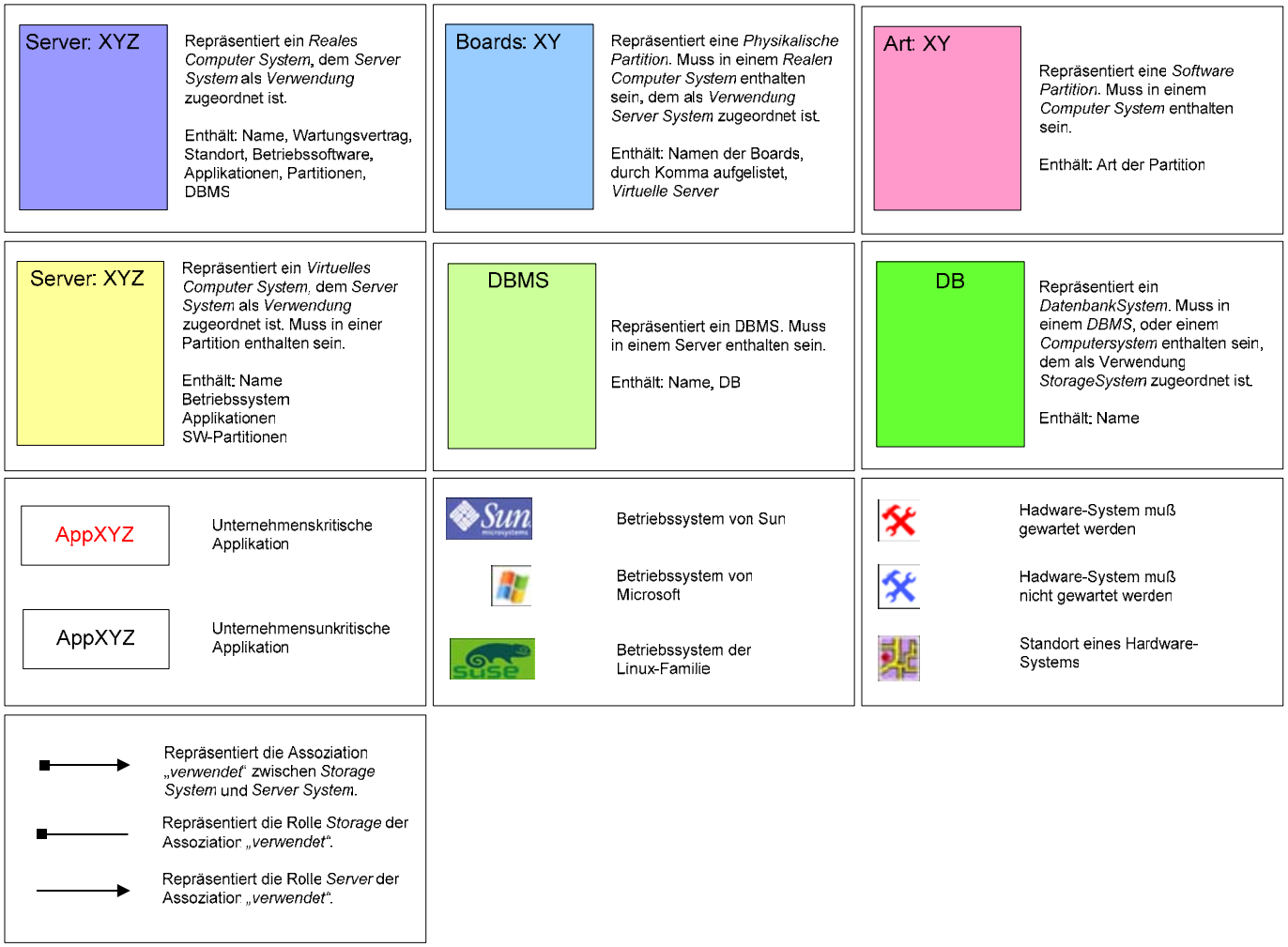


Abbildung 13 Kurzlegende der Softwarekarte Betrieb

Abbildung 10 zeigt beispielhaft eine anonymisierte Karte aus dem Bereich „Applikationen“. Eine ausführliche Legende zu der Applikations-Karte findet sich im Anhang. Weiße Kästchen repräsentieren Anwendungssysteme, auf denen, wie aus der ausführlichen Legende entnommen werden kann, weitere Gestaltungsmittel aufgebracht werden können. Hierzu zählen Gestaltungsmittel, die folgende Informationen visualisieren: Die Serversysteme, auf denen die Anwendung läuft, die Datenbank, welche die Anwendung ggf. nutzt, die Bereiche in denen die Anwendung verwendet wird und die Anzahl der Nutzer. Weiterhin werden Symbole, die signalisieren, dass es sich um eine Web-Anwendung oder eine Client-Server-Anwendung handelt und dass ein Anwendungssystem bei einer Störung problemlos wieder in Betrieb genommen werden kann gezeigt. Komplexere Anwendungssysteme können darüber hinaus auch noch Gestaltungsmittel für Funktionalitätsblöcke enthalten, wie dies beispielhaft bei der SAP-Komponente dargestellt wurde.
Außerdem liefert die Karte darüber Aufschluss, welche Anwendungssysteme von anderen Anwendungssystemen Daten beziehen. Da für die Stakeholder eine Unterscheidung zwischen Export- und Importschnittstellen nicht notwendig ist, wurde die Schnittstellen-Dokumentation lediglich auf die technische und fachliche Sicht der Schnittstellen beschränkt, aus denen unter anderem der Name der Schnittstelle, die Frequenz, der Umfang der transportierten Daten und die verwendete Technologie ersichtlich ist.

Bei einer Nutzung der Karten direkt in Visio ist es möglich durch Doppelklick auf entsprechende Gestaltungsmittel weitere Detail-Informationen, wie zum Beispiel *Hersteller der Anwendung*, *Entwicklungssprache* oder *Betreuer* über Makros anzuzeigen.

In Abbildung 12 ist eine Karte aus dem Bereich „Betrieb“ dargestellt, die anonymisiert einen Großrechner der Krones AG dokumentiert, der aus mehreren Hardwarepartitionen besteht, auf denen wiederum Softwarepartitionen und virtuelle Server betrieben werden. Neben der Partitionierung kann entnommen werden, welche Anwendungssysteme und Datenbank-Management-Systeme auf den virtuellen Servern betrieben werden und welche Betriebssysteme diese nutzen.
Auch die Betriebskarte bietet Zugsang zu weiteren Detailinformationen durch Doppelklick auf entsprechende Gestaltungsmittel.

3.4 Test-Phase

Explizit sollten die entstandenen Karten nach Abschluss jeder Konstruktionsphase (wie oft dies geschieht ist abhängig vom gewählten Vorgehen) getestet werden. Dabei werden die Karten den Nutzern vorgelegt und beurteilt. Bewertet werden dabei beispielsweise Korrektheit, Vollständigkeit, Eindeutigkeit, Übersichtlichkeit, Handhabbarkeit, Zugänglichkeit der Symbolik und die Qualität der Legende. Hierbei macht es Sinn Personen hinzuzuziehen, die an der eigentlichen Projektdurchführung nicht beteiligt waren, bzw. sind.
Erfahrungsgemäß werden nicht offensichtliche Fehler und Probleme erst bei der Handhabung deutlich. Außerdem empfinden beteiligte Personen Reviews und Test in der Regel als überflüssig und lästig, was schnell dazu führen kann, dass die vorgestellten Karten unrechtmäßig als gut befunden werden.
Dieser Problematik kann begegnet werden, indem die Tester dazu gezwungen werden sich mit den Karten auseinanderzusetzen. Dies kann durch das Durchspielen von Nutzungsszenarien erreicht werden, die aus den erhobenen Anforderungen an die Softwarekarten abgeleitet werden können. Diese Methode wird in der Softwaretechnik mit *„Active Design Review (ADR)*“ [CKK02] bezeichnet.
Zu jeder Question sollte den Testern ein Szenario präsentiert werden, welches sie unter Zuhilfenahme der Karten lösen müssen. Auf diese Weise können Fragen vermieden werden, die sich allzu leicht mit einem „Ja“ oder „Nein“ beantworten lassen, ohne dass sich der Befragte mit allen Aspekten beschäftigt, die mit der Frage in Verbindung stehen.

3.5 Pflege- und Nutzungs-Phase

Da zum jetzigen Zeitpunkt nicht immer ausgereifte Werkzeuge zur Verfügung stehen, die in der Lage sind Softwarekarten automatisch zu generieren, ist die Pflege der Karten und damit verbunden die Sicherstellung der Konsistenz der Karten mit den zugrunde liegenden Daten mit manuellem Aufwand verbunden.
Da Daten aus den unterschiedlichsten Bereichen des Unternehmens von verschiedenen Personen in die Datenbasis der Karten eingepflegt werden und diese Personen in der Regel nicht mit der Pflege der Karten beauftragt sein dürften, müssen auf Datenbankebene Mechanismen integriert werden, welche die Pflege der Karten erleichtern. Für die mit der Pflege der Karten beauftragten Personen muss schnell und einfach ersichtlich sein, welche

Veränderungen am Datenbestand vorgenommen wurden. Am einfachsten lässt sich dieses Problem mit dem Führen von Dirty-Flags in der Datenbank realisieren, die geänderte Datenfelder markieren. Solch ein Flag könnte beschreiben, ob ein Datensatz neu angelegt, geändert oder gelöscht worden oder mit den Karteninhalten konsistent ist.
Personen, die die Datenbank pflegen, würden die Flags auf die entsprechenden Werte setzen. Der Kartenpfleger aktualisiert die Karten und setzt die entsprechenden Flags, um zu signalisieren, dass die Datensätze übernommen wurden. Natürlich darf nur der jenige, der auch die Karten pflegt signalisieren, dass Datensätze mit den Karten konsistent sind.
Es ist darüber hinaus sinnvoll, dass nur bestimmte Personen die Karten verändern dürfen. Reine Nutzer der Karten sollten deren Inhalte natürlich nicht manipulieren können. Die Pflege der Karten sollte außerdem ein definierter Auftrag sein, um sicherstellen zu können, dass sie vorgenommen wird.
Über die eigentliche Nutzung der im praktischen Teil erstellten Softwarekarten durch die Krones AG konnten zum Zeitpunkt der Erstellung dieser Arbeit keine Informationen erhoben werden.

4 Prozessmodelle

Bei der Erstellung von Softwarekarten sollte ein Prozessmodell zugrunde gelegt werden, um ein strukturiertes Vorgehen erreichen zu können. Hierbei liegt es nahe sich an der Softwaretechnik zu orientieren, da die hier vorgestellten Phasen der Erstellung von Softwarekarten starke Ähnlichkeiten mit Phasen der Vorgehensmodelle in der Softwaretechnik haben.

Nach [Ba98] sollte ein Prozessmodell folgendes festlegen:

- Reihenfolge des Arbeitsablaufes (Entwicklungsstufen, Phasenkonzepte)
- In den Phasen jeweils durchzuführende Aktivitäten
- Definition der Teilprodukte einschließlich Layout und Inhalt
- Fertigstellungskriterien (Wann ist ein Teilprodukt fertig gestellt?)
- Notwendige Mitarbeiterqualifikationen
- Verantwortlichkeiten und Kompetenzen
- Anzuwendende Standards, Richtlinien, Methoden und Werkzeuge

Im Folgenden werden gebräuchliche Prozessmodelle der Softwaretechnik mit ihren Vor- und Nachteilen vorgestellt und untersucht, welche Prozessmodelle für die Anwendung in der Softwarekartographie adaptiert werden können.

4.1 Prozessmodelle der Softwaretechnik

Die vorgestellten Prozessmodelle orientieren sich an [Ba98].

4.1.1 Das Wasserfallmodell

Das Wasserfallmodell legt fest, dass Software in Phasen zu entwickeln ist, die sukzessive durchlaufen werden. Die Ergebnisse einer Phase „fallen“, wie bei einem Wasserfall, in die nächste. Aufgrund dieser Analogie erhielt das Modell seinen Namen.
Die Phasen sind mit Rückkopplungsschleifen versehen, die das Zurückkehren in eine frühere Phase erlauben, falls in der aktuellen Probleme aufgetreten sind, die Änderungen in früheren Phasen erforderlich machen. Das Wasserfallmodell in seiner ursprünglichen Form sieht allerdings keine Validationsaktivitäten in den einzelnen Phasen explizit vor.

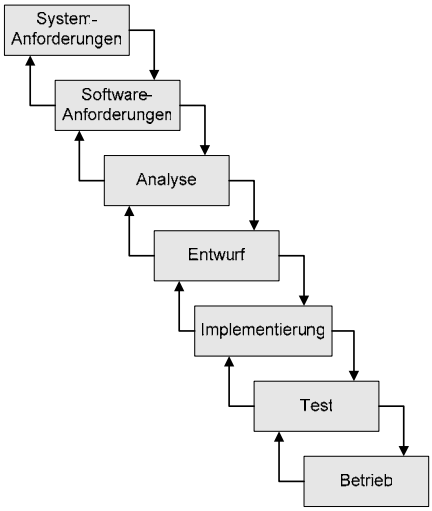


Abbildung 14 Das Wasserfallmodell

Die Aktivitäten des Wasserfallmodells müssen sequentiell in der vorgegebenen Reihenfolge und vollständig durchlaufen werden. Dabei wird erst mit einer neuen Aktivität begonnen, wenn die aktuelle abgeschlossen ist. Als Ergebnis jeder Aktivität existiert mindestens ein fertig gestelltes Dokument, welches als Grundlage für die nächste Phase dient.

[Ba98] nennt folgende Vor- und Nachteile des Wasserfallmodells

Vorteile des Wasserfallmodells:

- Das Wasserfallmodell ist einfach, verständlich und benötigt nur wenig Managementaufwand.

Nachteile des Wasserfallmodells:

- Es ist nicht immer sinnvoll alle Entwicklungsschritte vollständig durchzuführen.
- Es ist nicht immer sinnvoll alle Entwicklungsschritte strikt sequentiell abzuarbeiten.
- Es besteht die Gefahr, dass die Dokumentation wichtiger wird als das eigentliche System.
- Risikofaktoren werden unter Umständen nicht genügend berücksichtigt, da immer der einmal festgelegte Entwicklungsablauf durchlaufen wird.

Ein starker Nachteil des Wasserfallmodells bezüglich der Erstellung von Softwarekarten ist, dass die Kommunikation mit den Auftraggebern ausschließlich in der Definitionsphase stattfinden soll. Eine enge Kommunikation mit den Stakeholdern ist für das Erstellen von Softwarekarten aber beinahe essentiell. Das gilt nicht nur für das Erheben der Anforderungen, sondern zum Beispiel auch für die Definition der Gestaltungsmittel, da hier große gestalterische Freiräume vorhanden sind. Die praktischen Erfahrungen bei der Krones AG haben über dies auch gezeigt, dass Stakeholder nicht genau wissen, was sie brauchen und sich wichtige Anforderungen erst während der Projektdurchführung ergeben können. Es wäre jedoch tatsächlich denkbar ein wasserfallartiges Vorgehen zu wählen, wenn bereits Anforderungen erhoben wurden, die einem Vollständigkeitsanspruch genügen.

4.1.2 Das V-Modell

Das V-Modell erweitert das Wasserfallmodell um die Qualitätssicherung. Es schreibt die Verifikation und Validation der Teilprodukte als Aktivitäten vor und ist besonders gut zur Entwicklung eingebetteter Systeme geeignet, da es einen Systemkontext und Integrationstests mit einbezieht. Dieses Modell lässt sich in Form eines V darstellen. Diesem Sachverhalt verdankt das V-Modell seinen Namen.

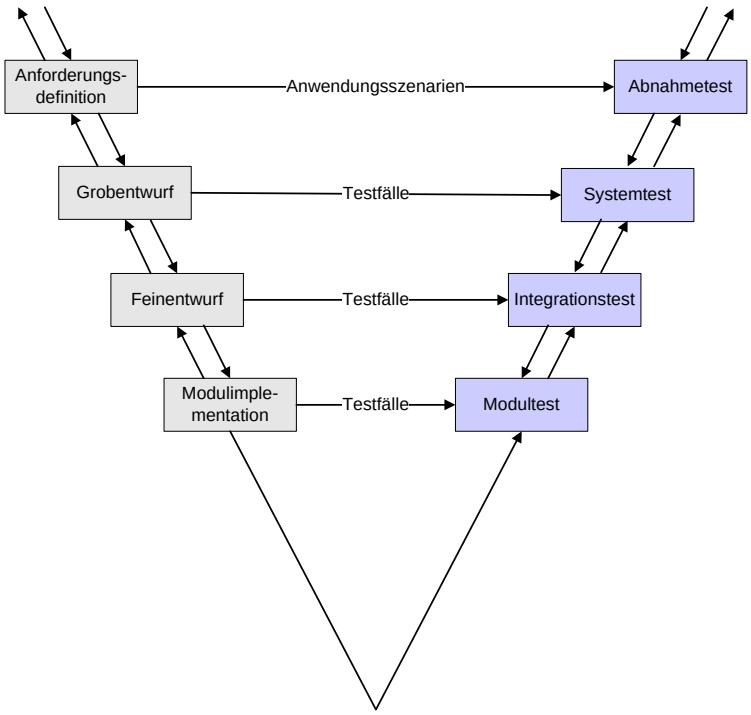


Abbildung 15 Vereinfachte Darstellung des V-Modells

Vor- und Nachteile nach [Ba98]

Vorteile des V-Modells:

- Das V-Modell unterstützt eine integrierte, detaillierte Darstellung der Systemerstellung, der Qulitätssicherung, des Konfigurationsmanagements und des Projektmanagements.
- Das V-Modell ist ein generisches Vorgehensmodell mit definierten Möglichkeiten zum Maßschneidern auf projektspezifische Anforderungen.
- Das V-Modell ermöglicht eine standardisierte Abwicklung von Systemerstellungsprojekten.
- Das V-Modell ist gut geeignet für große Projekte, insbesondere für eingebettete Systeme.

Nachteile des V-Modells:

- Die Vorgehenskonzepte, die für große eingebettete Systeme sinnvoll sind, werden unkritisch auf andere Anwendungstypen übertragen.
- Für kleine und mittlere Softwareentwicklungen führt das V-Modell zu einer unnötigen Produktvielfalt und Software-Bürokratie.
- Ohne geeignete CASE-Unterstützung ist das V-Modell nicht handhabbar.
- Die 25 definierten Rollen sind, außer für Großprojekte, unrealistisch.

Für die Erstellung von Softwarekarten scheint das V-Modell eher ungeeignet zu sein, da dessen Vorteile eher bei großen Projekten zum Tragen kommen. Außerdem liegen die Stärken des V-Modells bei der Entwicklung eingebetteter Systeme, die bei der Softwarekartographie keine Entsprechung haben. Somit zahlen sich die Vorteile des V-Modells für die Softwarekartographie kaum aus. Der Mehraufwand, welcher dieses Modell mit sich bringt ist aus der Perspektive der Softwarekartographie genau zu betrachten und es ist zu klären, ob er wirklich den gewünschten Nutzen bringt.

4.1.3 Das Prototypenmodell

Die Entwicklung größerer Softwareprodukte bringt Probleme mit sich, die weder vom Wasserfallmodell noch vom V-Modell adressiert werden. Auftraggeber sind oftmals nicht in der Lage genau zu definieren, was ihre Anforderungen an das zu entwickelnde Produkt sind, was zu Problemen bei traditionellen Prozessmodellen führt, da diese in der Definitionsphase eine vollständige Spezifikation der Anforderungen vorsehen. Bei diesen Modellen beschränkt sich der Dialog zwischen Entwicklern und Auftraggebern auf die Erhebung der Anforderungen. Das Entwickler-Team arbeitet dann selbstständig weiter und präsentiert fertige Ergebnisse. Bei vielen Projekten ist ständiger Austausch und ein gemeinsames Gestalten des Projektes jedoch sinnvoll, da dann sichergestellt ist, dass die Projektergebnisse im Sinne des Auftraggebers sind. Bei manchen Projekten sind unterschiedliche Lösungsansätze denkbar, von denen in der Definitionsphase nicht alle ausgeschlossen werden können. Experimentelle Entwürfe können helfen die beste Lösung zu finden und eröffnen zugleich die Möglichkeit die Realisierbarkeit zu prüfen.
Das Prototypenmodell sieht vor auf systematische Weise frühzeitig Lösungsvorschläge zu produzieren, die dann als Prototypen bezeichnet werden.
Prototypen haben verschiedene Beziehungen zum fertigen Produkt. Sie können als Studie, als Teil der Produktdefinition oder als Grundbaustein zu inkrementellen Weiterentwicklung dienen.

Vorteile des Prototypenmodells nach [Ba98]

- Reduzierung des Entwicklungsrisikos durch frühzeitigen Einsatz von Prototypen.
- Prototypen können sinnvoll in andere Prozessmodelle integriert werden.
- Prototypen leisten einen Beitrag bei der Planung.
- Prototypen fördern die Kreativität und führen zur Entwicklung von Lösungsalternativen.
- Prototypen erlauben eine starke Rückkopplung mit den Stakeholdern.

Nachteile des Prototypenmodells nach [Ba98]

- Höherer Entwicklungsaufwand, da Prototypen oftmals zusätzlich erstellt werden.
- Es besteht die Gefahr, dass aus Zeitgründen ein Prototyp Teil des Endproduktes wird.
- Die Beschränkungen und Grenzen von Prototypen sind oft nicht bekannt.

Die Verwendung von Prototypen ist bei der Erstellung von Softwarekarten denkbar. Prototypen in der Softwarekartographie könnten Beispielkarten sein, die zu Testzwecken aus aussagekräftigen Beispieldaten oder Daten von Teilen der Anwendungslandschaft erstellt werden. Auf diese Weise könnte zum Beispiel frühzeitig untersucht werden, ob die geplanten Softwarekarten wahrscheinlich den gewünschten Nutzen mit sich bringen oder die gewählten Gestaltungsmittel sinnvoll sind.
Konkrete Beispielkarten können die Kreativität der Stakeholder und Kartenersteller fördern und frühzeitig die Aufmerksamkeit auf Probleme, zum Beispiel bei der Visualisierung, lenken.
Das Prototypenmodell könnte bei der Entwicklung von Softwarekarten zum Beispiel mit iterativen Modellen kombiniert werden.

4.1.4 Das evolutionäre/inkrementelle Modell

Das evolutionäre und inkrementelle Modell hat starke Übereinstimmungen mit dem Prototypenmodell und ist aus einer ähnlichen Motivation heraus entwickelt worden. Beim Wasserfallmodell und V-Modell wird das Produkt zunächst vollständig spezifiziert und dann in voller Breite ohne Rückkopplung mit dem Auftraggeber entwickelt. In der Praxis sind Auftraggeber jedoch oftmals nicht in der Lage vollständige Produktanforderungen zu nennen.

Beim evolutionären Modell werden die Kernanforderungen erfasst und umgesetzt. Nach einer Präsentation der Ergebnisse werden mit den Auftraggebern neue Anforderungen erhoben und das Produkt weiterentwickelt. Dies wird wiederholt bis das Produkt einen zufrieden stellenden Status erreicht hat.

Vorteile des evolutionären Modells nach [Ba98]:

- Der Auftraggeber erhält in kurzen Zeitabständen Ergebnisse.
- Die Modelle erlauben eine gute Kombinierbarkeit mit dem Prototypenmodell.
- Ein Frühzeitiger Einsatz von Teilprodukten ist möglich.
- Ein Produkt entsteht in kleinen, überschaubaren Abschnitten. Die Richtung der Entwicklung kann leicht beeinflusst werden.
- Die Entwicklung ist nicht nur auf einen einzigen in der Ferne liegenden Abgabetermin ausgerichtet.

Nachteile des evolutionären Modells nach [Ba98]:

- Es besteht die Gefahr, dass die Kernversion nicht genügend erweiterbar ist, weil wichtige Anforderungen nicht berücksichtigt wurden.
- Die Kernversion ist nicht flexibel genug, um sich einem Evolutionspfad anzupassen.

Das inkrementelle Modell adressiert die Nachteile des evolutionären Modells, indem die Anforderungen möglichst vollständig erhoben werden. Es wird ansonsten wie beim evolutionären Modell vorgegangen, wobei bei jeder Iteration Teile der Anforderungen umgesetzt werden, bzw. falls nötig die erhobenen Anforderungen korrigiert werden.

Das inkrementelle bzw. evolutionäre Modell eignet sich gut für die Verwendung bei der Erstellung von Softwarekarten, da hier eng mit den Stakeholdern zusammengearbeitet werden muss. Die Vorteile kommen vor allem zum Tragen, da Aspekte wie die Ästhetik und Übersichtlichkeit der Karten schlecht in Anforderungen formuliert werden können und die Kartenersteller hier teilweise große Freiräume und von den Kartennutzern abweichende Einschätzungen haben.
Durch eine regelmäßige Absprache, welche bei diesen Modellen durch ihre iterative Natur leicht integrierbar ist, kann sichergestellt werden, dass ein für alle Beteiligten akzeptabler Konsens gefunden wird.
Ein iteratives Vorgehen adressiert darüber hinaus die Problematik, dass sich die Datenerhebung sehr aufwendig gestalten kann, indem zunächst weniger aufwendige Bereiche des Unternehmens kartographiert werden können und während der Projektdurchführung entschieden werden kann, ob weitere aufwendigere Bereiche realisiert werden sollen oder nicht.

4.2 Für die Softwarekartographie geeignete Prozessmodelle

Wie oben beschrieben ist das V-Modell für die Softwarekartographie eher ungeeignet. Ähnliches dürfte in den meisten Fällen auch für das Wasserfallmodell gelten, da dieses in seiner ursprünglichen Form nicht flexibel genug ist, es sei denn es kann auf entsprechende Vorarbeiten zurückgegriffen werden. Wenn dies nicht der Fall ist, kann aus den oben genannten Gründen ein evolutionäres bzw. inkrementelles Vorgehensmodell gewählt werden. Vorzugsweise sollten zunächst alle Anforderungen an die Karten in der Analyse-Phase erhoben und später inkrementell umgesetzt werden. Dazu ist nach der Analyse-Phase ein Ausschnitt der IT-Landschaft zu definieren, der auf den ersten Karten dargestellt werden soll. Auf diese Weise können schnell Karten erstellt und mit den Stakeholdern diskutiert werden, was frühe Korrekturen erlaubt und ggf. neue Ideen zu Tage fördert. Außerdem kann so den Risiken begegnet werden, welche die Datenerhebung mit sich bringt. Die erstellten Karten können dann in folgenden Iterationen um weitere Bereiche der IT-Landschaft verfeinert werden.

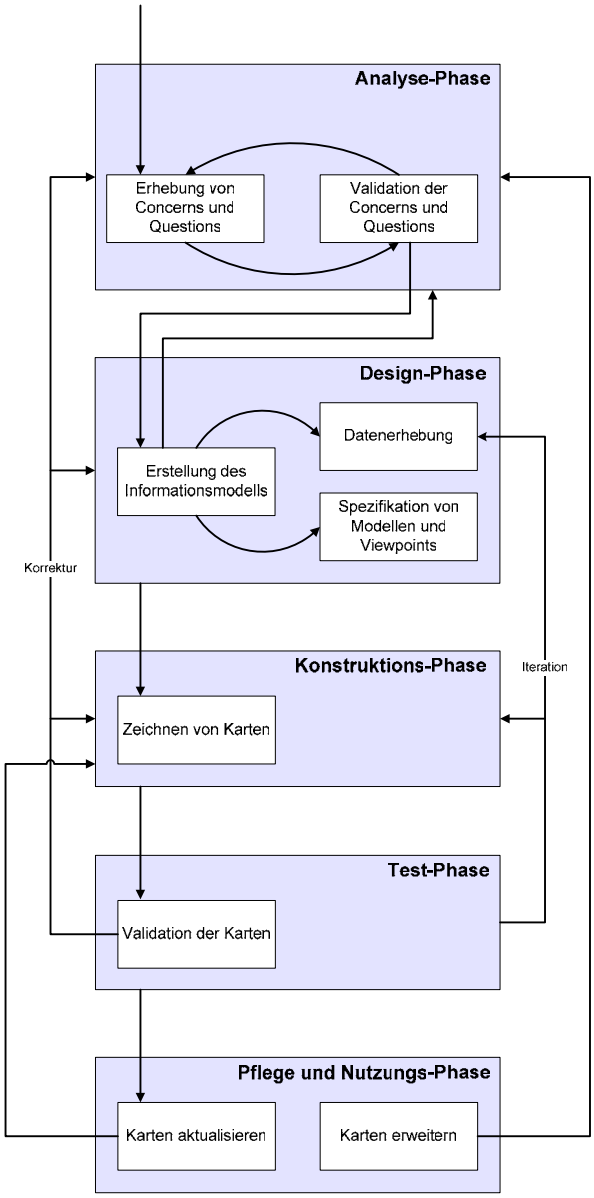


Abbildung 16 Prozessmodell zur Erstellung von Softwarekarten

Ein Beispiel für ein derartiges auf die Erstellung von Softwarekarten angepasstes inkrementelles Modell zeigt Abbildung 16, in der Phasen mit blauen Kästchen, deren Aktivitäten mit weißen Kästchen und Übergänge zwischen Phasen und Aktivitäten mit Pfeilen dargestellt sind.
Zu Beginn des Projekts werden die Anforderungen an die Karten in Form von Concerns und Questions iterativ erhoben und validiert.
Sind die Anforderungen möglichst vollständig erhoben und validiert kann in die Design-Phase übergegangen und mit der Erstellung eines Informationsmodells begonnen werden, das als Grundlage für die Spezifikation der Modelle und Viewpoints und zur Datenerhebung dient.
Nach Abschluss der Design-Phase beginnt mit Eintritt in die Konstruktions-Phase das Zeichnen der Softwarekarten, die in der folgenden Test-Phase validiert werden. Die Validation der Karten kann Mängel offenbaren, die durch Wiederaufnahme einer der vorherigen Phasen beseitigt werden müssen. Ist dies nicht der Fall können die Karten in der

nächsten Iteration der Design- oder Konstruktions-Phase verfeinert werden. Dazu kann es auch notwendig sein die Datenerhebung weiter voran zu treiben, wenn man sich zuvor darauf geeinigt hat zunächst nur einen Ausschnitt der IT-Landschaft zu behandeln und weitere Daten noch nicht erhoben sind.
Sind die Karten ausreichend verfeinert, können sie zur Nutzung freigegeben werden. Da sich die IT-Landschaft ständig weiterentwickelt, müssen die Karten selbstverständlich auch entsprechend gepflegt werden. Die Pflege der Karten schließt auch eine Erweiterung mit ein, wenn beispielsweise neue Technologien eingeführt werden. Dann sollte für die betroffenen Bereiche der gesamte Prozess erneut durchlaufen werden.

4.3 Risikomanagement

Risiken sind potentielle Probleme, die als solche identifiziert wurden und eine erfolgreiche Durchführung des Projektes beeinträchtigen können. Was unter einem erfolgreichen Projektabschluss zu verstehen ist wird durch die erhobenen Anforderungen bestimmt. Probleme, insbesondere solche, die spät erkannt werden, sind in der Regel mit hohen Kosten verbunden. Diese Tatsache macht ein Risikomanagement erforderlich.
Im Rahmen des Risikomanagement sollen Risiken identifiziert, analysiert, priorisiert und letztlich beherrscht werden.
Ein gutes Risikomanagement sollte nach jedem Prozessschritt die Risiken neu erheben und beurteilen. Das bedeutet, dass untersucht wird, ob sich neue Risiken abzeichnen und wie alte zu bewerten sind. Dabei sollten vor allem auch die beteiligten Stakeholder in regelmäßigen Abständen zu Risiken befragt werden. Nach [Ba98] ergibt sich die Bewertung von Risiken aus dem Produkt der Eintrittswahrscheinlichkeit und dem zu erwartenden Schaden bei Eintritt des Problems. Dabei wird für den Schaden beispielsweise eine Zahl von 1 bis 10 eingesetzt. Dieses Produkt wird als Risikofaktor bezeichnet.
Es sollte also eine entsprechende Tabelle geführt werden, welche die Risiken und deren Risikofaktoren listet. Dazu kann auch für jedes Risiko ein Vorgehen bei Eintritt des Schadens, das dessen negative Auswirkungen minimieren soll, skizziert werden.
Mögliche Risiken bei der Softwarekartographie bestehen vor allem bei der Datenerhebung, aber natürlich auch bei allen anderen Schritten. Selbstverständlich sind die Risiken stark projekt- bzw. unternehmensbezogen. Einige allgemeingültige Risiken sind:

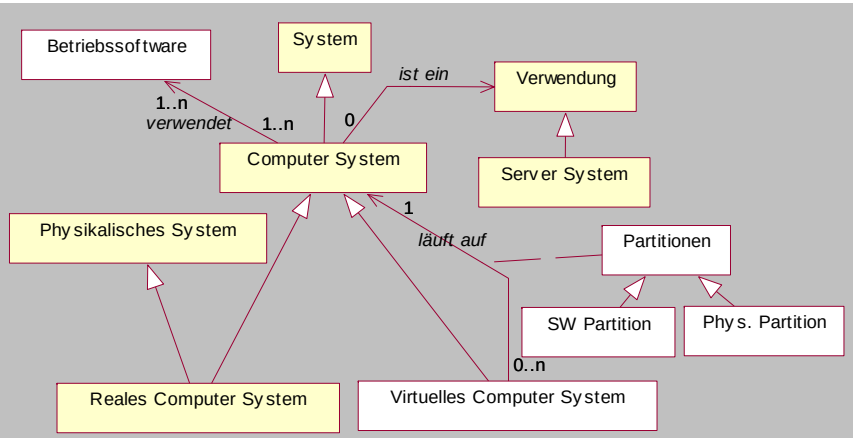
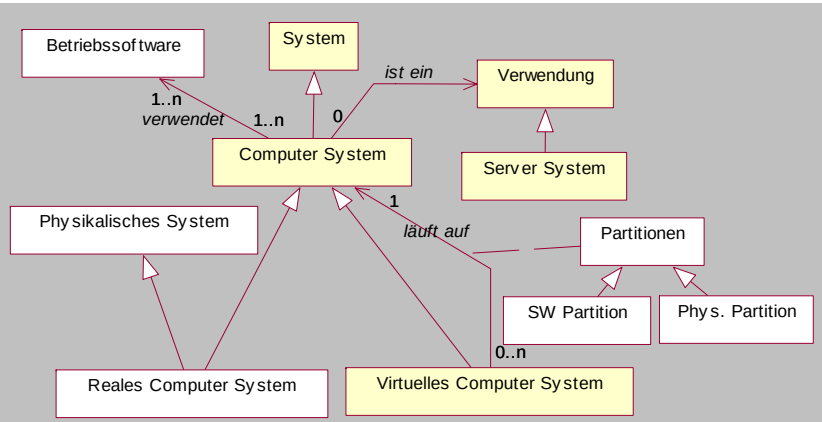
- Wichtige Stakeholder wurden nicht mit einbezogen
- Stakeholder sind nicht in der Lage ihre Anliegen zu kommunizieren
- Wichtige Concerns wurden vergessen
- Wichtige Questions wurden vergessen
- Das Informationsmodell modelliert nicht die Realität
- Wichtige Aspekte der realen Unternehmensstruktur wurden nicht modelliert
- Wichtige Daten sind nicht vorhanden
- Die Daten haben eine schlechte Qualität
- Der Aufwand der Datenerhebung wurde unterschätzt
- Die entwickelten Darstellungsformen sind ungeeignet
- Die Karten sind unübersichtlich
- Die Karten sind mehrdeutig

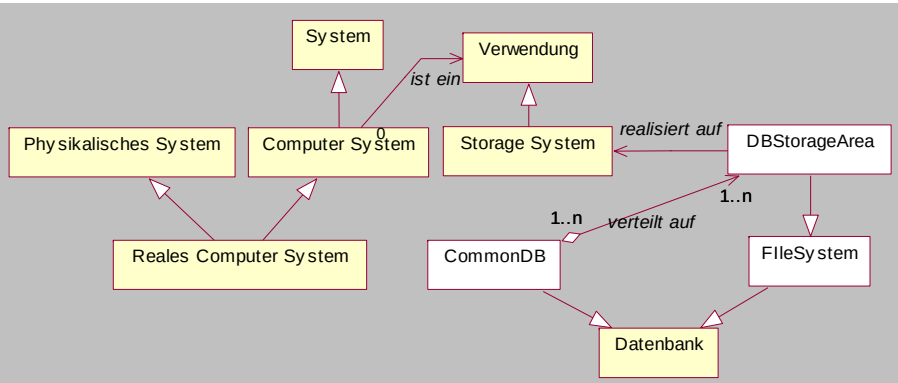
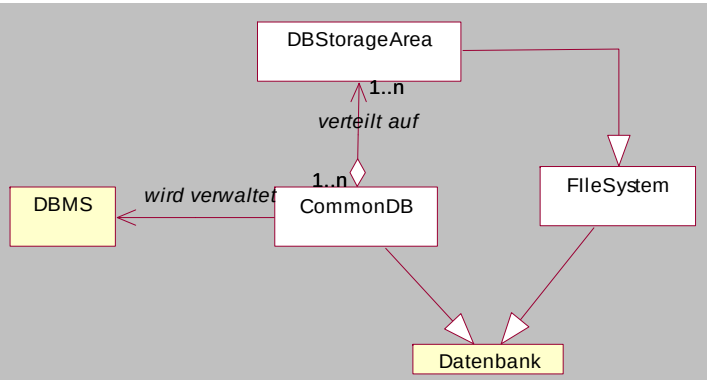
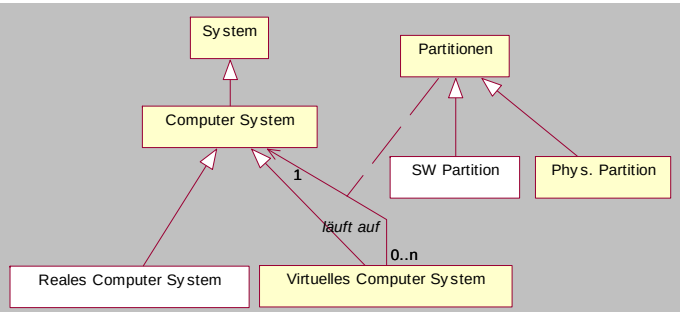
5 Schluss und Ausblick

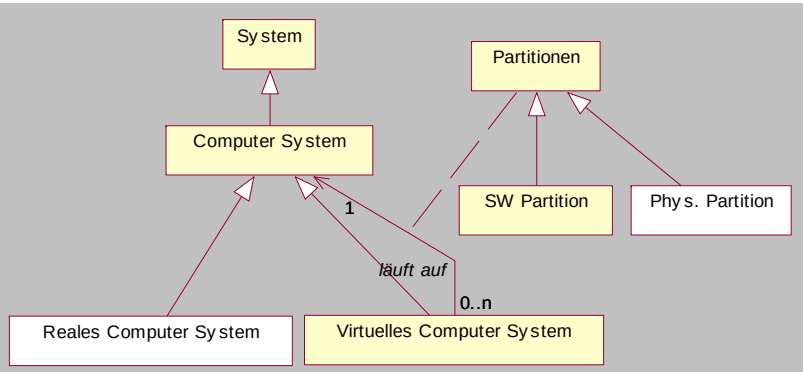
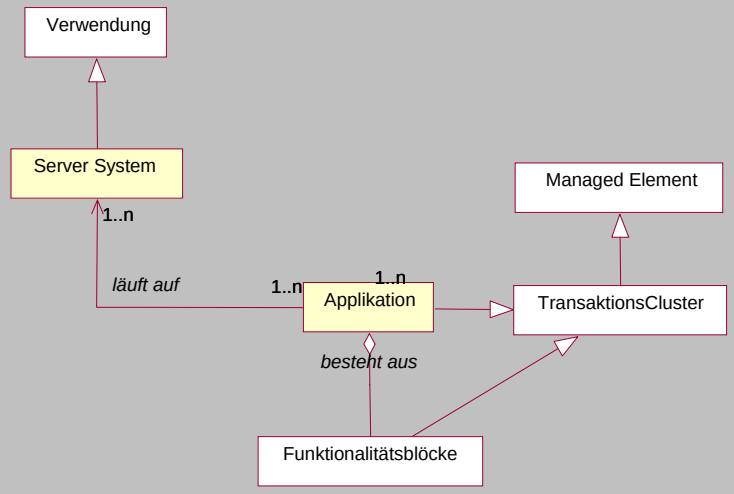
Für Unternehmen gestaltet sich die eigene IT-Landschaft zunehmend undurchsichtiger. Die starke Abhängigkeit von einer funktionierenden IT-Landschaft und hohe IT-Investitionen machen eine Dokumentation selbiger unumgänglich.
Die Softwarekartographie adressiert dieses Problemfeld. Mit Hilfe von Softwarekarten können IT-Landschaften beschrieben werden. Dabei liefern Softwarekarten Antworten auf spezifische Fragen.
In dieser Arbeit wurden Phasen und Aktivitäten zur Erstellung von Softwarekarten beschrieben und in ein Vorgehensmodell eingeordnet. Diese Phasen beinhalten Aktivitäten zur Anforderungsanalyse, zur Spezifikation, zur Konstruktion, zum Testen, zur Pflege und Nutzung von Softwarekarten.
Das große Manko der Softwarekartographie zum jetzigen Zeitpunkt besteht darin, dass die Karten oft manuell gezeichnet werden müssen, was nicht nur mit einem hohen Aufwand verbunden ist, sondern die Art der darstellbaren Informationen stark einschränkt.
Es wäre sehr wünschenswert über Werkzeuge zu verfügen, die Softwarekarten automatisch zeichnen können. Auf diese Weise würde der Pflegeaufwand stark reduziert. Durch entsprechende Datenbankanbindungen könnte es möglich gemacht werden Informationen auf den Karten darzustellen, die sich ständig ändern und deshalb, bei manueller Pflege, nicht berücksichtigt werden können. Entsprechende Monitore könnten Informationen aus dem laufenden Betrieb liefern und in Datenbanken ablegen, die dann von kartenerzeugenden Werkzeugen in regelmäßigen Abständen oder bei Bedarf ausgelesen werden könnten. Solche Werkzeuge stehen zurzeit leider noch nicht im gewünschten Maße zur Verfügung.
Auf die Darstellung vieler interessanter Aspekte der IT-Landschaft muss derzeit auf Grund des schlechten Kosten-Nutzen-Verhältnisses verzichtet werden, das durch die Datenerhebung und Erstellung der Karten negativ beeinflusst wird. Eine entsprechende Werkzeugunterstützung würde das Gebiet der Softwarekartographie für viele Unternehmen noch attraktiver machen.
Mit entsprechenden Werkzeugen wären außerdem Karten denkbar, die der statischen Natur einer Karte im herkömmlichen Sinn entsagen. Virtuelle Karten könnten Benutzer zur Navigation durch die IT-Landschaft des Unternehmens über verschiedene Abstraktionsebenen einladen und ihn schnell zu spezifischen Informationen geleiten. Komplexere Systeme könnten Simulationen erlauben, durch Monitore unterstützt auf Probleme hinweisen, verantwortliche Personen informieren und vielleicht sogar Lösungsvorschläge produzieren und visualisieren.

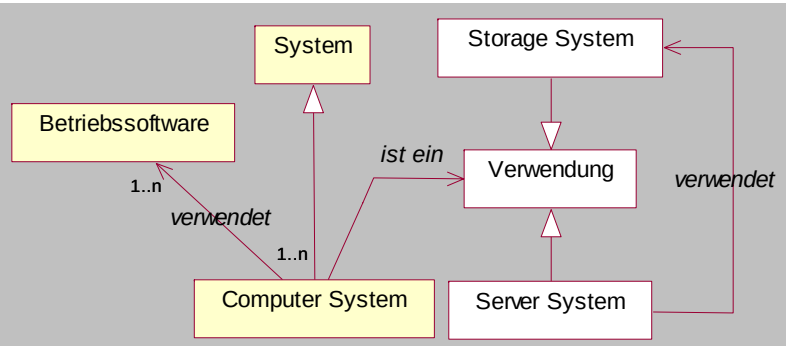
Anhang

Legende Softwarekarte Betrieb

Name	Darstellung	Beschreibung	Enthält	Informationsmodell
Objekte				
Server	Server: XYZ	Repräsentiert eine *Reales Computer System*, dem *Server System* als *Verwendung* zugeordnet ist.	Name Wartungsvertrag Standort Betriebssoftware Applikationen Partitionen	
Virtueller Server	Server: XYZ	Repräsentiert ein *Virtuelles Computer System,* dem *Server System* als *Verwendung* zugeordnet ist. Muss in einer Partition enthalten sein.	Name Betriebssystem Applikationen SW-Partitionen	

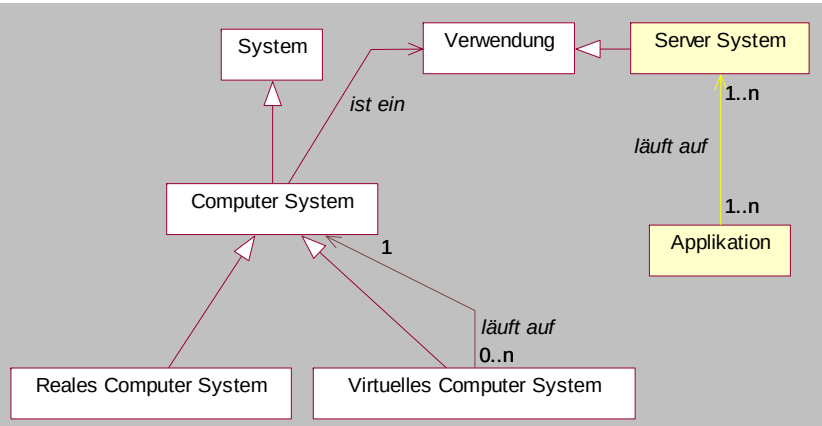
Storage	Storage: XY	Repräsentiert ein *Computer System,* dem *Storage System* als *Verwendung* zugeordnet ist. Es handelt sich dabei immer um *Reale Computer Systeme*.	Name Datenbank	
Datenbank-Man-agement-System	DBMS	Repräsentiert ein *DBMS*	Name Datenbank	
Datenbank	DB	Repräsentiert eine Datenbank	Name	
Physi-kalische Partition	Boards: XY	Repräsentiert eine *Physikalische Partition*. Muss in einem *Realen Computer System* enthalten sein, dem als *Verwendung Server System* zugeordnet ist.	Namen der Boards, durch Komma aufgelistet, *Virtuelle Server*	

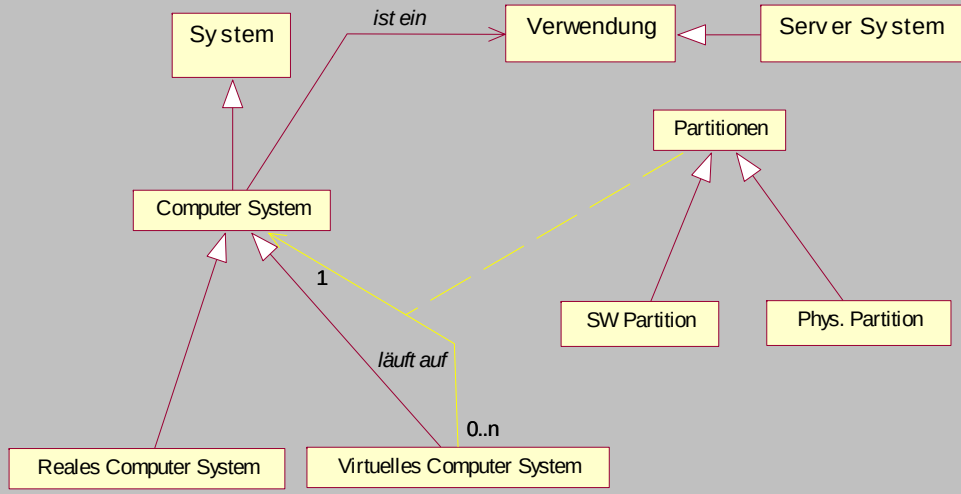
Software Partition	Art: XY	Repräsentiert eine *Software Partition*. Muss in einem *Computer System* enthalten sein	Art der Partition	
Applikation (kritisch)	AppXYZ	Repräsentiert eine unternehmens-kritische *Applikation*. Muss in einem *Computer System* enthalten sein, dem *Server* System als *Verwendung* zugeordnet ist.	Name	
Applikation (unkritisch)	AppXYZ	Repräsentiert eine unternehmensunkritische *Applikation*. Muss in einem *Computer System* enthalten sein, dem *Server System* als *Verwendung* zugeordnet ist.	Name	

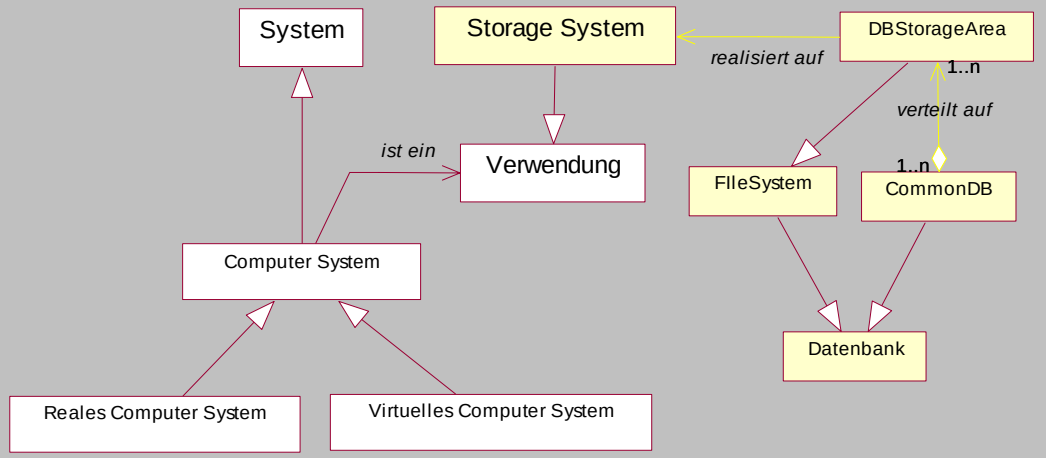
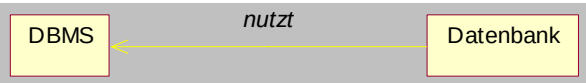
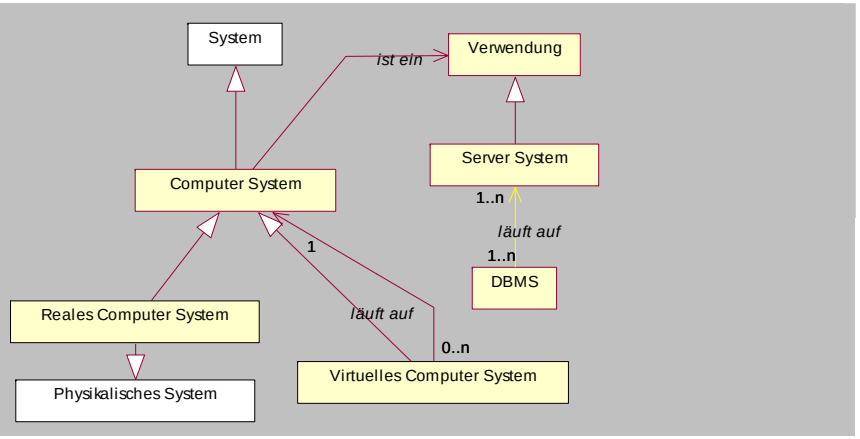
Betriebs-software		Betriebssystem von Sun (Solaris). Muss in einem *Computer System* enthalten sein.		
		Betriebssystem von Microsoft. Muss in einem *Computer System* enthalten sein.		
		Suse Betriebssystem. Muss in einem *Computer System* enthalten sein.		
Kein Wartungs-vertrag vorhanden		Es besteht kein Wartungsvertrag. Muss in einem *Realen Computer System* enthalten sein.		Attribut von *Reales Computer System*
Wartungs-vertrag vorhanden		Es besteht ein Wartungsvertrag. Muss in einem *Realen Computer System* enthalten sein.		
Standort		Gibt den Standort eines *Realen Computer Systems* an. Muss in einem *Realen Computer System* enthalten sein.		Attribut von *Reales Computer System*

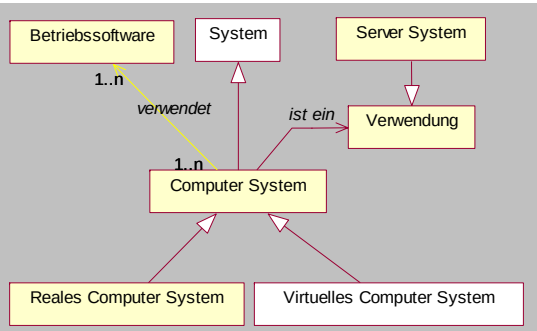
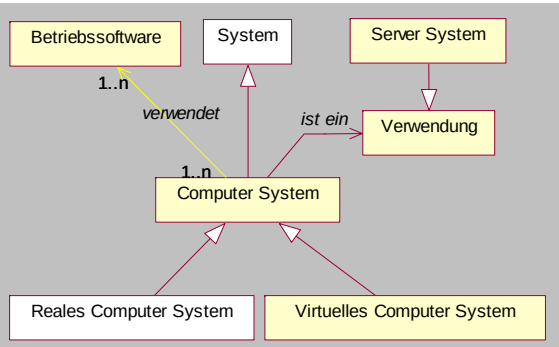
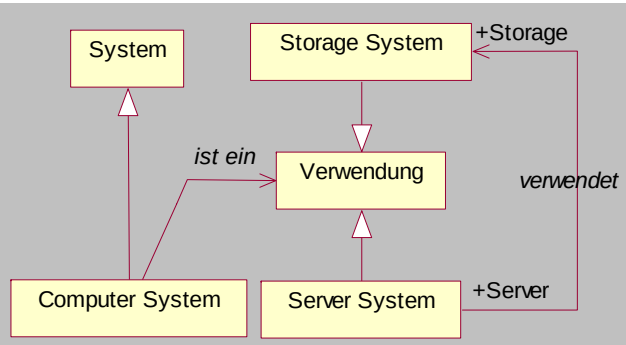
Verschachtelungsregeln

Applikation in Server	AppXYZ	Stellt dar, dass die *Applikation* auf einem *Realem Computer System*, dem die *Verwendung Server System* zugeordnet ist, läuft.	
Applikation in virtuellem Server	AppXYZ	Stellt dar, dass die *Applikation* auf einem *Virtuellem Computer System*, dem die *Verwendung Server System* zugeordnet ist, läuft.	
Physi-kalische Partition in Server		Stellt dar, dass die *Physikalische Partition* Teil genau eines *Realen Computer Systems* ist, dem die *Verwendung Server System* zugeordnet ist.	
Software Partition in Server		Stellt dar, dass die *Software Partition* auf einem *Realem Computer System* läuft, dem als *Verwendung Server System* zugeordnet ist.	
Software Partition in Virtuellem Server		Stellt dar, dass die *Software Partition* auf einem *Virtuellem Computer System*	

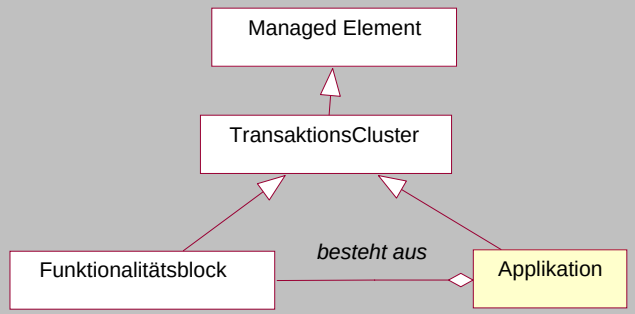


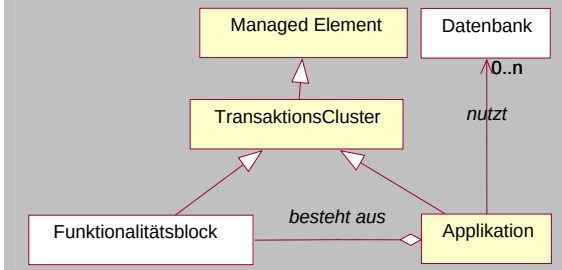
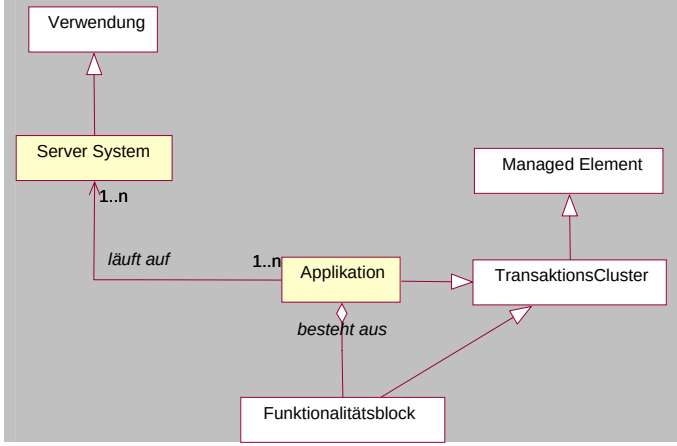


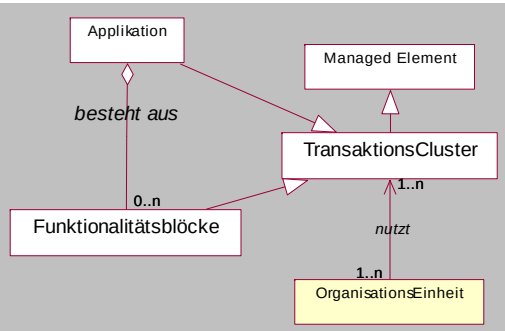
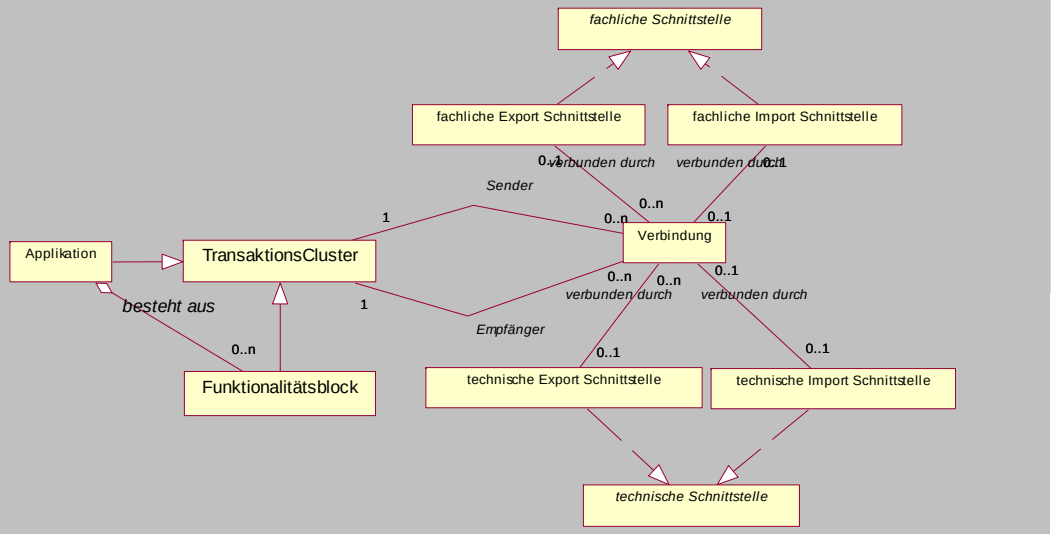
		läuft, dem als *Verwendung Server System* zugeordnet ist.		
Datenbank in Storage		Stellt dar, dass mindestens ein Teil der *Datenbank* auf einem *Realen Computer System* läuft, dem als *Verwendung Storage System* zugeordnet ist.		
Datenbank in DBMS		Stellt dar, dass eine *Datenbank* von einem *DBMS* verwaltet wird.		
DBMS in Server		Stellt dar, dass ein *DBMS* auf einem *Realem Computer System* läuft, dem als *Verwendung Server System* zugeordnet ist.		
DBMS in virtuellem Server		Stellt dar, dass ein *DBMS* auf einem *Virtuellem Computer System* läuft, dem als *Verwendung Server System* zugeordnet ist.		

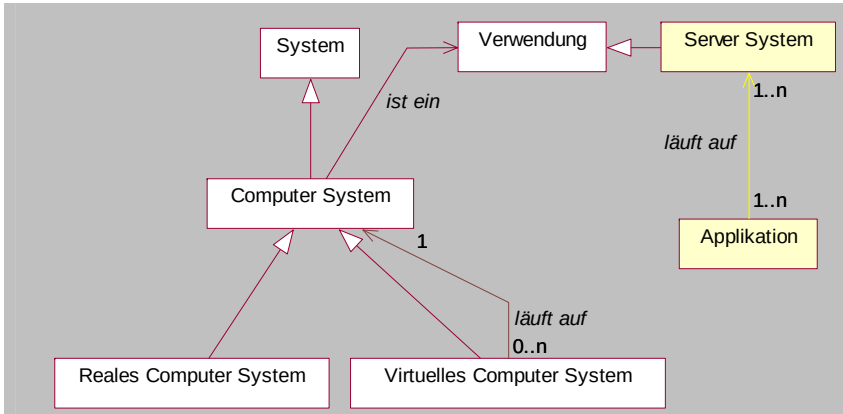
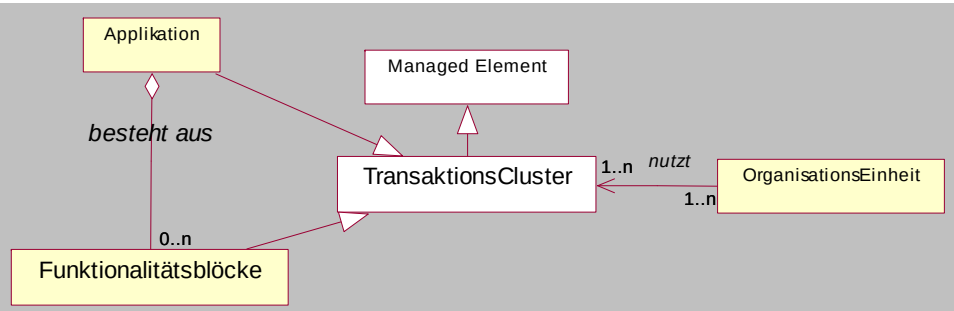
Betriebs-software in Server		Stellt dar, dass ein *Reales Computer System*, dem die *Verwendung Server System* zugeordnet ist eine bestimmte *Betriebssoftware* (hier Solaris) verwendet.		
Betriebs-software in Virtuellem Server		Stellt dar, dass ein *Virtuelles Computer System*, dem die *Verwendung Server System* zugeordnet ist eine bestimmte *Betriebssoftware* (hier Solaris) verwendet.		
Verbinder				
Storage Ver-wendung		Repräsentiert die Assoziation „*verwendet*“ zwischen *Storage System* und *Server System.*		
		Repräsentiert die Rolle *Storage* der Assoziation *„verwendet“.*		
		Repräsentiert die Rolle *Server* der Assoziation *„verwendet“.*		

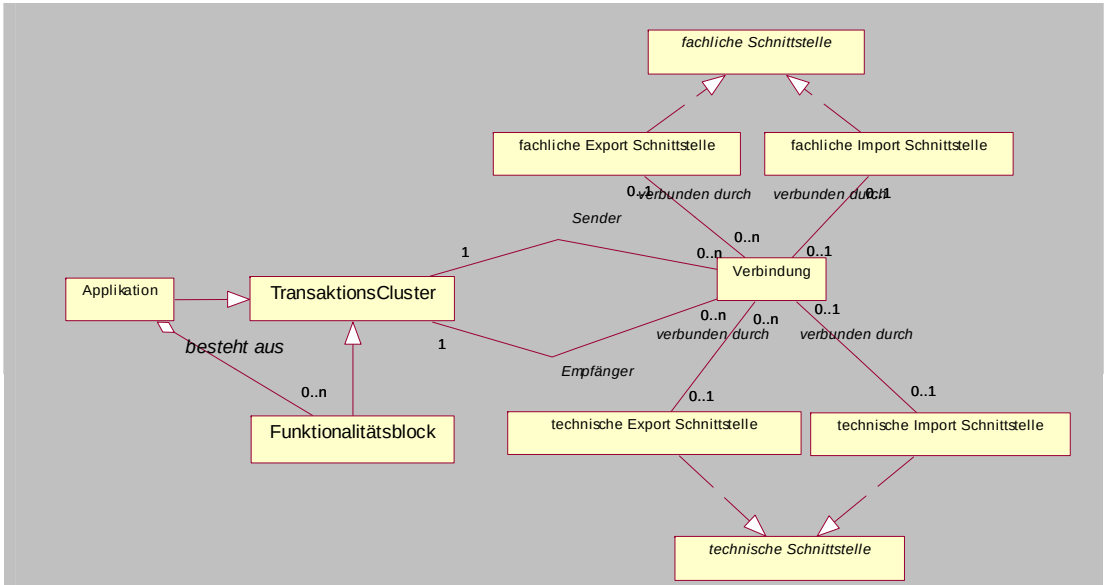
Legende Softwarekarte Applikationen

Name	Darstellung	Beschreibung	Enthält	Informationsmodell
Applikation (kritisch)	AppXYZ	Repräsentiert eine unternehmens-kritische *Applikation*	Name Wiederanlauf Web Anwendung Client/Server Anw. Anzahl Nutzer Server Datenbank	
Applikation (unkritisch)	AppXYZ	Repräsentiert eine unternehmens-unkritische *Applikation*	Name Wiederanlauf Web Anwendung Client/Server Anw. Anzahl Nutzer Server Datenbank	
Funktional-itätsblock	Name	Repräsentiert einen *Funktionalitätsblock* einer *Applikation*. Muss in einer *Applikation* enthalten sein.		
Wieder-anlauf		Muss in einer *Applikation* enthalten sein. Gibt an, dass die *Applikation* bei einem Ausfall problemlos wieder gestartet werden kann		Attribut von *Applikation*

Web Anwen-dung		Muss in einer *Applikation* enthalten sein. Gibt an, dass es sich um eine Web Anwendung handelt		Attribut von *Applikation*
Client/ Server Anwen-dung		Muss in einer *Applikation* enthalten sein. Gibt an, dass es sich um eine Client/Server Anwendung handelt		Attribut von *Applikation*
Anzahl Nutzer		Muss in einer *Applikation* enthalten sein. Gibt die Anzahl der Nutzer der Applikation an	Zahl	
Server		Muss in einer *Applikation* enthalten sein. Repräsentiert ein *Computer System*, dem als *Verwendung Server System* zugeordnet ist und auf dem die *Applikation* läuft	Name	

Datenbank	Name	Muss in einer *Applikation* enthalten sein. Repräsentiert eine von der *Applikation* verwendete Datenbank	Name	
Org-einheiten	Bereiche	Muss in einer *Applikation* oder einem *Funktionalitätsblock* enthalten sein. Listet alle Organisations-einheiten, in denen die *Applikation* verwendet wird.	Liste der Organisations-einheiten	
Schnitt-stelle	Name Frequenz Technologie Frequenz * Umfang	Repräsentiert eine Schnittstelle. Besteht aus *Technische Schnittstelle* und *Fachliche Schnittstelle*		
Technische Schnitt-stelle	Technologie Frequenz * Umfang	Repräsentiert eine *technische Schnittstelle.*	Name Frequenz Technologie Umfang	
Fachliche Schnitt-stelle	Name Frequenz	Repräsentiert eine *fachliche Schnittstelle.*	Name Frequenz	

Verschachtelungsregeln				
Server in Applikation		Stellt dar, das die *Applikation* auf dem *Server System* läuft.		
Datenbank in Applikation		Stellt dar, dass die *Applikation* die *Datenbank* verwendet.		
Org-Einheiten in Applikation		Stellt dar, dass die *Applikation* in den aufgelisteten *Bereichen* verwendet wird.		
Org-Einheiten in Funktional-itätsblock		Stellt dar, dass der *Funktionalitätsblock* in den gelisteten *Bereichen* verwendet wird..		
Funktional-itätsblock in Applikation		Stellt dar, dass der *Funktionalitätsblock* ein Teil der *Applikation* ist.		

Verbinder				
Verbindung		Repräsentiert die Assoziationen „*Sender*“ und „*Empfänger*“ zwischen *TransaktionsCluster* und *Verbindung*.		
		Repräsentiert die Assoziation „*Sender*“ zwischen *TransaktionsCluster* und *Verbindung*.		
		Repräsentiert die Assoziation „*Empfänger*“ zwischen *TransaktionsCluster* und *Verbindung*.		

Literaturverzeichnis

[Ac97] Achour, B.: Linguistic instruments for the integration of scenarios in requirement engineering. In: Proceedings of the Third International Workshop on Requirements Engineering, Foundation for Software Quality (REFSQ'07), Barcelona, Spain, Juni 1997.

[Ba98] Balzert, H.: Lehrbuch der Software-Technik: Softwaremanagement, Software-Qualitätssicherung, Unternehmensmodellierung. Heiderlberg: Spektrum Akad. Verl., 1998. ISBN 3-8274-0065-1.

[BG02] Briggs, R.; Grünbacher P.: Easy win win: Managing complexity in requirements negotiation with gss. In: Proceedings of 35th Hawaii International Conference on System Sciences, Hawaii, 2002.

[CKK02] Clements, P; Kazman R., Klein M.: Evaluating Software Architectures. Boston, San Fransisco: Addison-Wesley, 2002. ISBN 0-201-70482-X.

[DMT99] DMTF: Common Information Model (CIM) Specification – Version 2.2. Distributed Management Task Force, Inc. (DMTF), 1999. http://www.dmtf.org/standards/documents/CIM/DSP0004.pdf, [abgerufen 2004-10-02]

[IE00] IEEE: IEEE Std 1471-2000 for Recommended Practice for Architectural Description of Software-Intensive Systems. IEEE Computer Society, 2000.

[LMW05a] Lankes, J.; Matthes, F.; Wittenburg, A.: Softwarekartographie: Systematische Darstellung von Anwendungslandschaften. In: 7. Internationale Tagung Wirtschaftsinformatik 2005, Bamberg, Deutschland, 2005.

[LMW05b] Lankes, J.; Matthes, F.; Wittenburg, A.: Architekturbeschreibung von Anwendungslandschaften: Softwarekartographie und IEEE Std 1471-2000. In: Software Engineering 2005, Essen, 2005.

[MW04a] Matthes F.; Wittenburg, A.: Softwarekarten zur Visualisierung von Anwendungslandschaften und ihrer Aspekte. Technische Universität München, Lehrstuhl für Informatik 19 (sebis), München, Deutschland, TB0402, 2004. http://wwwmatthes.in.tum.de/de/main.htm?t=document/1xeim9mukt15m.htm [abgerufen 2004-07-01].

[MW04b] Matthes, F.; Wittenburg, A.: Softwarekartographie: Visualisierung von Anwendungslandschaften und ihrer Schnittstellen. In: Informatik 2004 – Informatik verbindet, 34. Jahrestag der GI, Ulm, Deutschland, 2004. ISBN 3-88579-380-6.

[OMG03] OMG: Unified Modelling Language Specification, Version 1.5. OMG, 2003.

[Pu01] Purchase, C. et al.: Graph drawing aesthetics and the comprehension of UML class diagrams: an empirical study. In: Australian symposium on Information Visualisation. Sidney, Australian, 2001.

[Ru02] Rupp, C. Requirements-Engineering und –Management – Professionelle, iterative Anforderungsanalyse für die Praxis. 2. Auflage, München, Wien: Hanser-Verlag, 2002. ISBN 3-446-21960-9.