Softwarekartographie als Beitrag zum Architekturmanagement

Josef Lankes, Florian Matthes, André Wittenburg

Software Engineering betrieblicher Informationssysteme (sebis)
Ernst Denert-Stiftungslehrstuhl
Lehrstuhl für Informatik 19, Institut für Informatik
Technische Universität München

Boltzmannstraße 3, 85748 Garching

Zusammenfassung: Unternehmen betrachten ihre Anwendungslandschaft und ihre Unternehmensarchitektur im Kontext der Planung, Umsetzung und Kontrolle von Unternehmenszielen und -strategien zunehmend als ganzheitliches vernetztes System. Die Softwarekartographie bietet – unter Rückgriff auf Methoden der Kartographie – graphische Darstellungen in Form von Softwarekarten, die eine ganzheitliche Betrachtung der Anwendungslandschaft in Verbindung mit Geschäftsprozessen, Organisationseinheiten, Projekten etc. ermöglicht. Dieser Beitrag stellt die Softwarekartographie und verschiedene Softwarekartentypen vor, diskutiert die notwendigen Modelle zur Kartographie der Unternehmensarchitektur und ordnet die Softwarekartographie in den Bereich des Architekturmanagements ein.

Schlüsselworte: Softwarekartographie, Architekturmanagement, Softwarekarten, Symbolisches Modell, Semantisches Modell

1 Einleitung

Das Architekturmanagement in Unternehmen beschränkt sich nicht ausschließlich auf das Gebiet der Softwarearchitekturen[1], welches sich mit der Gestaltung und Beschreibung einzelner Softwaresysteme beschäftigt, sondern betrachtet verschiedenste Aspekte einer Unternehmung, die sich von den Strategien und Zielen über die Geschäftsprozesse und betrieblichen Informationssysteme bis hin zu Infrastrukturkomponenten erstrecken.

Das *Enterprise Architecture Management*, als ein Prozess zum Explizieren der Kern-Geschäfts- und IT-Strategien und ihres Einflusses auf Geschäftsprozesse und -funktionen, soll bei dieser ganzheitlichen Betrachtung die Rolle einer Brücke einnehmen, die eine Ausrichtung der IT an dem Geschäft des Unternehmens ermöglicht.[2]

In unserem Forschungsprojekt Softwarekartographie entwickeln wir in Zusammenarbeit mit unseren Projektpartnern (Allianz, AMB Generali Informatik Service, BMW, Deutsche Börse Systems, Deutsche Post, HVB Systems, Kühne + Nagel, Krones, Münchener Rück, Siemens, T-Com) Methoden und Modelle zur Beschreibung, Bewertung und Gestaltung von Anwendungslandschaften, die verschiedene Betrachtungsebenen bedienen und damit eine ganzheitliche Sicht auf die Anwendungslandschaft ermöglichen.

1.1 Betrachtungsebenen der Softwarekartographie

Die drei Betrachtungsebenen der Softwarekartographie, die in Abbildung 1 dargestellt sind, sollen analog zum *Enterprise Architecture Management* eine ganzheitliche Betrachtung der Anwendungslandschaft ermöglichen und nicht nur die Ebene der technischen Realisierung („Wie?“) mit Informationssystemen, Schnittstellen etc. einbeziehen.

---

[1] Vgl. Clements et al. (2003).
[2] Vgl. META Group (2002), S. 4.

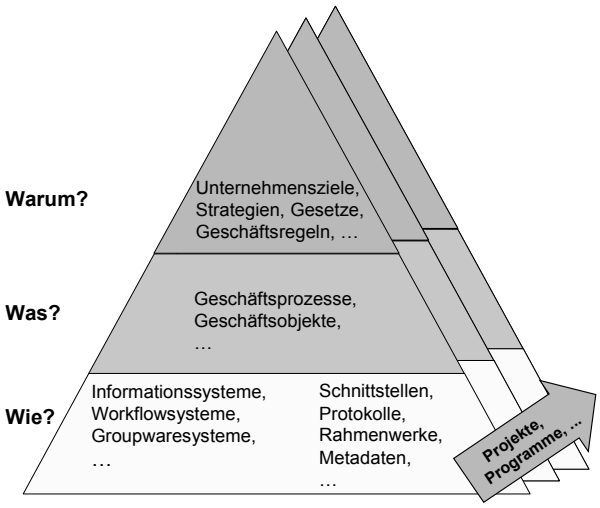

Abbildung 1 Betrachtungsebenen der Softwarekartographie

Den unternehmerischen und strategischen Zielen eines Unternehmens wird auf der obersten Ebene („Warum?") Rechnung getragen. IT-Strategien, die z. B. eine Reduzierung von Individualsoftware vorsehen, oder neue gesetzliche Regelungen, die Veränderungen oder neue Geschäftsprozesse fordern (z. B. MaK oder Basel II im Finanzsektor), haben Auswirkungen auf die Anwendungslandschaft und ändern Investitionsplanungen für IT-Projekte in den kommenden Planungsphasen.

Änderungen und Neuerungen bei operativen Geschäftsprozessen und Geschäftsobjekten, haben direkte Auswirkungen auf die unterstützenden Informationssysteme und werden auf der mittleren Ebene („Was?") in die Betrachtung der Anwendungslandschaft einbezogen.

Auf der untersten Ebene („Wie?") werden die Geschäftsprozesse/-objekte implementiert bzw. durch betriebliche Informationssysteme unterstützt, die durch unter-schiedliche Technologien realisiert sind, verschiedene Softwarearchitekturen benutzen, über verschiedene Schnittstellen[3] verfügen etc.

[3] Vgl. Matthes/Wittenburg (2005b).

Die Fragestellungen „Warum?“, „Was?“ und „Wie?“ haben sich bereits im Business Process Reengineering für eine grundlegende Betrachtung von Geschäftsprozessen bewährt.[4]

Die drei Betrachtungsebenen ermöglichen in der Summe eine statische Analyse der Anwendungslandschaft, die um eine dynamische Betrachtung ergänzt werden muss, um die Evolution der Anwendungslandschaft zu berücksichtigen. Informationssysteme werden aufgrund von sich ändernden Rahmenbedingungen, Geschäftsprozessen oder Zielen etc. neu gebaut, adaptiert oder abgelöst. Dieser dynamische Wandel wird durch Programme und/oder Projekte vollzogen, die möglicherweise parallel verschiedene Informationssysteme modifizieren.

1.2 Probleme in der Praxis

Bei der Ist-Aufnahme des Status quo zur Darstellung von Anwendungslandschaften wurden eine Kategorisierung relevanter Aspekte[5] und eine Klassifikation der Darstellungen, so genannter Softwarekarten[6], entwickelt. Die wesentlichen, bei der Ist-Aufnahme identifizierten Problemstellungen, sind:

- Hoher Aufwand bei der Erstellung von Softwarekarten
- Hoher Aufwand sowohl bei der initialen Datenaufnahmen als auch bei der Datenpflege
- Geringe Anzahl vordefinierter Viewpoints (siehe Abschnitt 4) zur Analyse von Anwendungslandschaften
- Hohe Begriffserosion im Bereich des Enterprise Architecture Managements
- Geringe Unterstützung beim Aufbau eines geeigneten Informationsmodells

Abbildung 2 zeigt eine exemplarische Softwarekarte, die als Kartengrund die Funktionsbereiche (z. B. Sparten oder Geschäftsbereiche) verwendet und die Informationssysteme mit Schnittstellen den Funktionsbereichen zuordnet.

---

[4] Vgl. Hammer/Champy (1993), S. 32 f.
[5] Vgl. Matthes/Wittenburg (2004a).
[6] Vgl. Lankes/Matthes/Wittenburg (2005a).

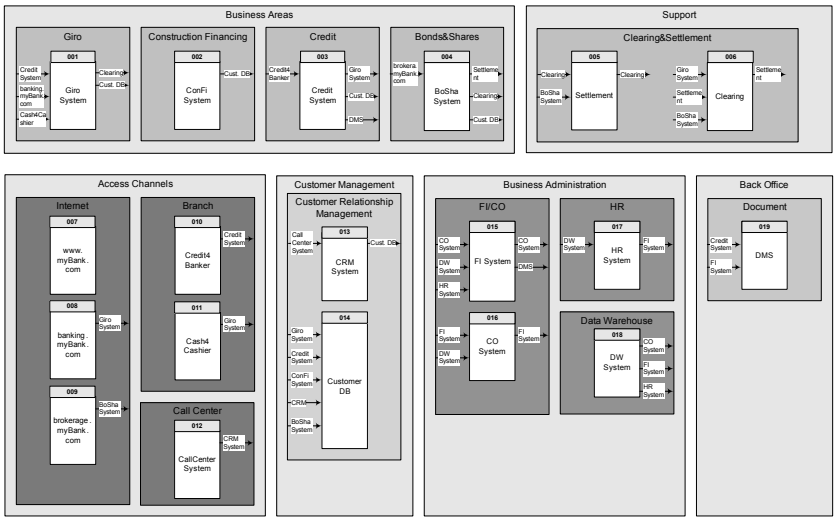

Abbildung 2 Beispiel für eine Softwarekarte von Typ Clusterkarte (1)
Alle Softwarekarten sind anonymisiert und stark vereinfacht; die Originale der Projektpartner zeigen bis zu 200 Anwendungssysteme auf einer Karte.

Bei der Erstellung von Softwarekarten kommen sowohl Repository-gestützte Anwendungen (z. B. Corporate Modeler von Casewise oder ARIS Toolset von IDS Scheer) als auch nicht Repository-gestützte Anwendungen (z. B. Microsoft Visio oder Microsoft Powerpoint) zum Einsatz. Beiden Ansätzen ist gemein, dass die Erstellung der Softwarekarten nur manuell bis semi-automatisiert erfolgt, da die verwendeten Werkzeuge die Darstellungen nur begrenzt unterstützen. Insbesondere die vollautomatisierte regelbasierte Erstellung von Softwarekarten wird nicht unterstützt. Im folgenden Abschnitt 2 wird das Zusammenspiel von Konzepten, die bei der Adressierung und Lösung der obigen Probleme helfen.

2 Semantische und symbolische Modelle

Bei der Konzeption einer regelbasierten und automatisierten Erstellung von Softwarekarten hilft eine Trennung von semantischen und symbolischen Modell, wie sie Abbildung 3 zeigt.

Das semantische Model befasst sich mit den eigentlichen Informationen, die über die Anwendungslandschaft verfügbar sind, also mit den Informationsobjekten (siehe Abschnitt 4), wobei nur die Inhalte selbst gemeint sind, unabhängig von einer ihrer Darstellungsformen. Diese Trennung zwischen Information und ihrer Repräsentation oder Visualisierung verwenden auch Torre et al.[7] Es gilt somit, dass ein semantisches Modell ohne mindestens ein dazugehöriges symbolisches Modell sinnlos ist, da die Informationsobjekte in diesem Fall weder kommunizierbar noch erfahrbar sind.

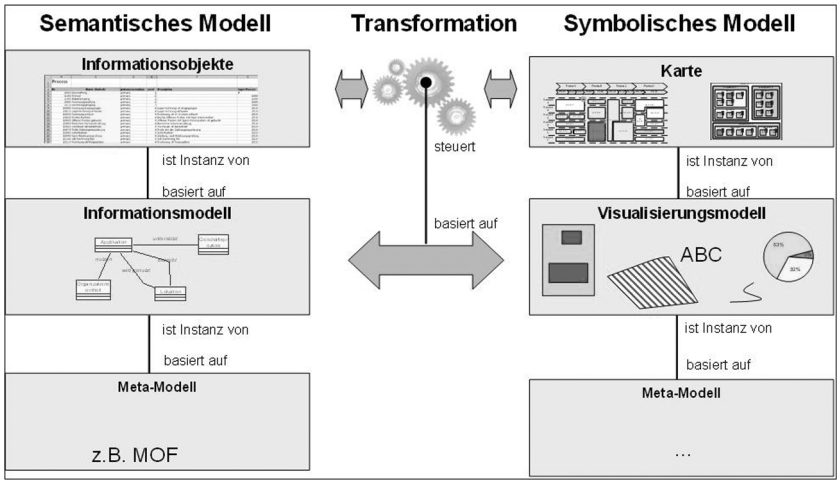

Abbildung 3 Zusammenhang von semantischem und symbolischem Modell

Auf der Seite des symbolischen Modells werden die Informationsobjekte des semantischen Modells über ein geeignetes Visualisierungsmodell dargestellt, um eine Kommunizierbarkeit der Informationen zu erhalten. Die Kommunikation muss hierbei nicht ausschließlich, wie in Abbildung 3 dargestellt, über Karten erfolgen, sondern kann auch tabellarische Reports etc. umfassen.

[7] Vgl. van der Torre et al. (2004).

2.1 Semantisches Modell

Das instantiierte Modell einer bestimmten Architektur wird durch ein geeignetes Informationsmodell formalisiert. Der Zusammenhang zwischen Informationsobjekt und einem Entitätstyp im Informationsmodell ist analog zu der Beziehung Objekt zu Klasse in der objektorientierten Programmierung.

Ein Informationsmodell für das *Enterprise Architecture (EA) Management* muss die relevanten Informationsobjekte, ihre Attribute und die Beziehungen zwischen Informationsobjekten in einem geschlossenen Modell abbilden. Als Modellierungssprache wird die Unified Modeling Language der OMG[8] verwendet. Ein zusätzliches Glossar muss die Begrifflichkeiten aufnehmen und erläutern.

Abbildung 4 visualisiert die zu betrachtenden Ebenen (*Layers*) und die auf diese einwirkenden Querschnittsfunktionen (*Cross Functions*). Eine Modularisierung in verschiedene Pakete (z. B. CoreModule, ApplicationArchitectureModule, ApplicationModule, BusinessModule, InterconnectionModule, LifecycleModule) ermöglicht es, zusätzliche Module inkrementell einzusetzen. Einzelne Module setzen sich aus Artefakten von einem oder mehreren Blöcken der Ebenen und Querschnittsfunktionen zusammen.

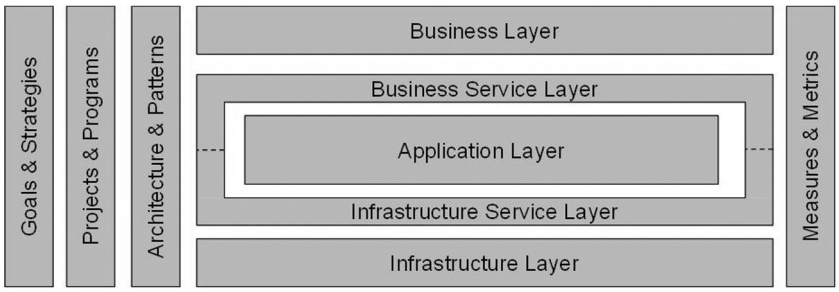

Abbildung 4 Aufbau eines Informationsmodells für das EA Management

Die fünf Ebenen *Business Layer*, *Business Service Layer*, *Application Layer*, *Infrastructure Service Layer* und *Infrastructure Layer* kapseln die Informationsobjekte der *Enterprise Architecture*, die durch die Querschnittsfunktionen beeinflusst werden, die weitere Informationsobjekte enthalten.

[8] Vgl. OMG (2004).

Der *Business Layer* enthält Produkte, Geschäftsprozesse, Organisationseinheiten etc., die über den *Business Service Layer* mit den Artefakten des *Application Layers* verbunden sind. Der *Business Service Layer* kapselt *Business Objects*, *Business Services* und zugehörige *Business Service Level Agreements* (SLAs), die mittels der Applikationen und Komponenten des *Application Layers* implementiert werden.

Der *Infrastucture Service Layer* stellt die Verbindung zwischen den Artefakten des *Application Layers* und des *Infrastructure Layers* her. Analog zum *Business Service Layer* kapselt der *Infrastructure Service Layer* die *Infrastructure Services* und die zugehörigen SLAs[9]. Die Infrastrukturkomponenten (z. B. Middleware- und Hardwaresysteme) des *Infrastructure Layers* erbringen die notwendigen *Services* auf die Artefakte des *Application Layers* zurückgreifen.

Die Querschnittsfunktionen *Strategies & Goals, Projects & Programs, Architecture & Patterns* und *Measures & Metrics* beeinflussen die Artefakte der unterschiedlichen Ebenen. Strategien und Ziele (*Strategies & Goals*) generieren Handlungsbedarfe, die in Form von Maßnahmen in Projekten und Programmen (*Projects & Programs*) umgesetzt werden. Bei der Umsetzung der Maßnahmen werden zusätzlich Architekturen und Muster (*Archictectures & Patterns*) betrachtet, um z. B. eine Konformität und Standardisierung im Unternehmen zu erreichen.

Die Kennzahlen und Metriken (*Measures & Metrics*) bilden die Steuerungskomponente der Architektur und ermöglichen eine quantifizierbare Bewertung von Artefakten bzw. deren Kombinationen hinsichtlich bestimmter Qualitätsmerkmale.

2.2 Symbolisches Modell

Das symbolische Modell ermöglicht die Kommunizierbarkeit und Erfahrbarkeit der Informationsobjekte. Während Informationsobjekte einem Informationsmodell gehorchen, richten sich Visualisierung, wie z. B. Softwarekarten, an einem Visualisierungsmodell aus.

Die Anforderungen an ein Visualisierungsmodell für die Softwarekartographie, ergeben sich aus der Kategorisierung von Softwarekartentypen und

[9] Vgl. OGC (2000).

ihren spezifischen Anforderungen (siehe Abschnitt 3). Im Folgenden werden die allgemeingültigen Regeln zum Aufbau von Softwarekarten vorgestellt (siehe Abschnitt 2.2.1), die in einem Visualisierungsmodell (siehe Abschnitt 2.2.2) adäquat zusammengefasst werden müssen.

2.2.1 Aufbau von Softwarekarten

Eine Softwarekarte ist eine graphische Repräsentation der Anwendungslandschaft oder von Ausschnitten selbiger, wobei sich eine Softwarekarte aus einem Kartengrund und den auf dem Kartengrund aufbauenden Schichten zusammensetzt, die verschiedene Informationen bzgl. der Anwendungslandschaft transportieren.

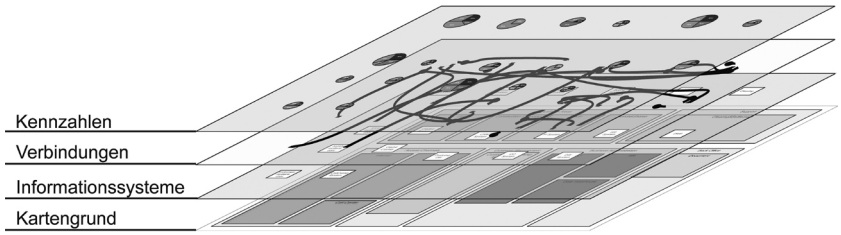

Abbildung 5 Schichtenprinzip von Softwarekarten

Die Abbildung 5 zeigt beispielhaft den Schichtenaufbau einer Softwarekarte. Auf den Kartengrund, der je nach Kartentyp (siehe Abschnitt 3) variiert, werden die unterschiedlichen Schichten aufgetragen, wobei zu jeder Schicht eine Referenzschicht existiert, auf die sich die Elemente dieser Schicht beziehen. Durch das Ein-/Ausblenden und das Zoom-In/Out können die angezeigten Informationen gefiltert und die Informationsdichte variiert werden, um für einen bestimmten Anwendungsfall die gewünschte Darstellung zu erhalten.

Existierende Visualisierungssprachen im Bereich des Software Engineering fokussieren auf der Darstellung eines einzelnen Softwaresystems und berücksichtigen dabei nur Teile der Betrachtungsebenen und Aspekte, die die Softwarekartographie adressieren soll. UML[10] definiert zwar unterschiedliche Diagrammtypen, die für unterschiedliche Fragestellungen bei der Analyse,

dem Design etc. von Softwaresystemen geeignet sind, verbindet jedoch beispielsweise nicht mehrere Informationssysteme mit den unterstützten Geschäftsprozessen. Ebenso besitzt UML keine Semantik für Position, Größe oder Farbe von Objekten auf den Diagrammen.[11] Ein Schichtenaufbau eines Diagramms, um für unterschiedliche Fragestellungen bei gleich bleibendem Kartengrund die entsprechenden zusätzlich einzublendenden Informationen zu selektieren, ist in UML ebenso nicht möglich.

2.2.2 Visualisierungsmodell für Softwarekarten

Ein Visualisierungsmodell gibt vor, wie Darstellungen organisiert sind, d. h. welche Darstellungen möglich sind. Es liefert Konzepte, die eine Beschreibung dieser Darstellungen ermöglichen, wobei die Beschreibung dabei in einer Art und Weise erfolgen sollte, die einerseits eindeutig angibt, was auf der Darstellung gezeigt werden soll andererseits auch eindeutig erkennen lässt, welche Freiräume für die Erstellung der Darstellung die Beschreibung bietet.

Aus den Anforderungen zur Erstellung von Softwarekarten ergeben sich bereits erste Anforderungen an das Visualisierungsmodell, wie beispielsweise:

- Aspekte auf verschiedenen Schichten zu visualisieren
- Aspekte einer Schicht können Aspekte einer anderen Schicht referenzieren
- Schichten können ein- und ausgeblendet werden

Hinzukommen Anforderungen, die sich aus den Softwarekartentypen (siehe Abschnitt 3) extrahieren lassen.

2.3 Modelle und Transformationen

Um eine Verbindung zwischen dem semantischen Modell und dem symbolischen Modell herzustellen, soll eine (weitgehend automatisierte) Transfor-

---

[10] Vgl. OMG (2004).
[11] Vgl. Matthes/Wittenburg (2004a).

mation aus den über die Anwendungslandschaft verfügbaren Informationen (den Information Objects) zu einer Visualisierung erfolgen.

Dabei soll dem Modellierer eine gewisse Gestaltungsfreiheit erhalten bleiben. Kriterium dafür ist, dass bestimmte Einschränkungen (z. B. bezüglich Vorhandensein und Positionierung von Elementen) erfüllt sein müssen, um die entsprechende Information verfälschungsfrei transportieren zu können. Innerhalb dieser Einschränkungen soll der Modellierer aber manuelle Anpassungen vornehmen können, um eine optimierte (z. B. in Hinblick Lesbarkeit) Darstellung zu erhalten.

Eine geeignete Abbildung ist die Grundlage für obige Transformation sowie eine *korrekte* Interpretation der dabei entstehenden Visualisierungen. Sollen die Informationsobjekte auf der Seite des semantischen Modells in Objekte der Karte (bzw. Visualisierung) transformiert werden, so müssen die Transformationsregeln zwischen dem Informationsmodell und dem Visualisierungsmodell (siehe Abbildung 3) definiert werden.

Diese Abbildung zwischen dem semantischen Teil und dem symbolischen Teil legt fest, welche Dinge auf der Seite des semantischen Modells auf welche Dinge auf der Seite des symbolischen Modells abgebildet werden, d. h. zum Erscheinen oder zum Positionieren etc. welcher graphischen Elemente diese führen. In Abschnitt 5 wird ein kurzer Ausblick auf eine mögliche automatisierte Transformation auf Basis einer bidirektionalen Transformationssprache gegeben.

3 Softwarekartentypen

Ein Softwarekartentyp (oder kurz Kartentyp) bezeichnet eine grundlegende Möglichkeit zum Aufbau des Kartengrunds für eine Softwarekarte unter Nutzung des Visualisierungsmodells (siehe Abschnitt 2.2.2). Bei einer kartographischen Karte handelt es sich um ein Abbild eines geographischen Standorts, sie dient dazu, räumliche Beziehungen zu kommunizieren. Dazu liefert sie eine visuelle Repräsentation der räumlichen Beziehungen, die dem Kommunikationspartner mitgeteilt werden sollen. Die Kartographie stellt eine außerordentlich effiziente Möglichkeit dar, Ideen, Gedanken, Formen und Beziehungen auszudrücken, zu analysieren und zu manipuliere[12]n.

[12] Vgl. Robinson et al. (1995), S. 9 f.

Damit bietet es sich an, die von dieser Disziplin bereitgestellten Instrumente bei der Beherrschung der oben als komplex und mit hohen Investitionen verbunden dargestellten Anwendungslandschaften einzusetzen.

Beim Versuch, Techniken aus der Kartographie zur Erstellung von Repräsentationen einer Anwendungslandschaft zu nutzen, erweist sich allerdings eine grundsätzliche Begrenzung von Karten als hinderlich. Die Techniken der Kartographie eignen sich nur zur Abbildung von Gegebenheiten, die in einem zwei- oder dreidimensionalen Raum auftreten. Bei der Darstellung von geographischen Informationen stellt dies keine Beschränkung dar, der Kartengrund[13] ist immer ein topographischer, was unter anderem auch auf die enorme Bedeutung derartiger Informationen zurückgeht: „Our desire for spatial imagery of things in our environment is as normal as breathing."[14]

Informationssysteme als Elemente einer Anwendungslandschaft lassen sich anhand verschiedenster Merkmale beschreiben (unterstützte Geschäftsprozesse, nutzende Organisationseinheit, bearbeitende Projekte, verwendete Technologien etc.), darunter findet sich aber kein Satz von zwei oder drei Merkmalen, die als Dimensionen derartig prominent hervortreten wie die räumlichen Dimensionen in der Kartographie. Damit steht für Softwarekarten kein eindeutiger Kartengrund fest. Hierdurch entsteht ein gestalterischer Freiraum bei der Definition einer Softwarekarte. Bei den untersuchten Softwarekarten aus der Praxis ließen sich vier Softwarekartentypen identifizieren. Diese Unterteilen sich in Kartentypen, die den Kartengrund zur Verortung von Elementen verwenden (analog zur Kartographie), sowie in Kartentypen, die den Kartentypen nicht zur Verortung von Elementen verwenden.

3.1 Softwarekarten mit Kartengrund zur Verortung

Ein wichtiges Attribut eines Elements auf einer Karte in der Kartographie stellt die Position des Elements auf dem Kartengrund dar, die sich aus der geographischen Position des Objekts ableitet. Dies legt die Verortung von Elementen auf der Karte fest: Erscheint ein bestimmtes Element an einer anderen Stelle der Karte, ändert sich die von ihr transportierte Botschaft. Durch den gleichartigen Aufbau und eine gleich bleibende Positionierung

---

[13] Vgl. Hake/Grünreich/Meng (2002), S. 487.
[14] Robinson (1995), S. 9.

von Elementen auf der Softwarekarte entsteht zusätzlich ein Wiedererkennungswert, der es dem Betrachter erleichtert, sich schnell in der Karte zu orientieren.

Hier kann je nach zentral visualisierten Aspekten wie folgt unterschieden werden:

- Clusterkarte (siehe Abschnitt 3.1.1)
- Prozessunterstützungskarte (siehe Abschnitt 3.1.2)
- Intervallkarte (siehe Abschnitt 3.1.3)

3.1.1 Clusterkarte

Eine Clusterkarte gehört zu einem Softwarekartentyp, der sich durch die Verortung von Informationssystemen anhand von logischen Einheiten auszeichnet. Informationssysteme, die zur gleichen logischen Einheit gehören, werden in einen Cluster zusammengefasst.

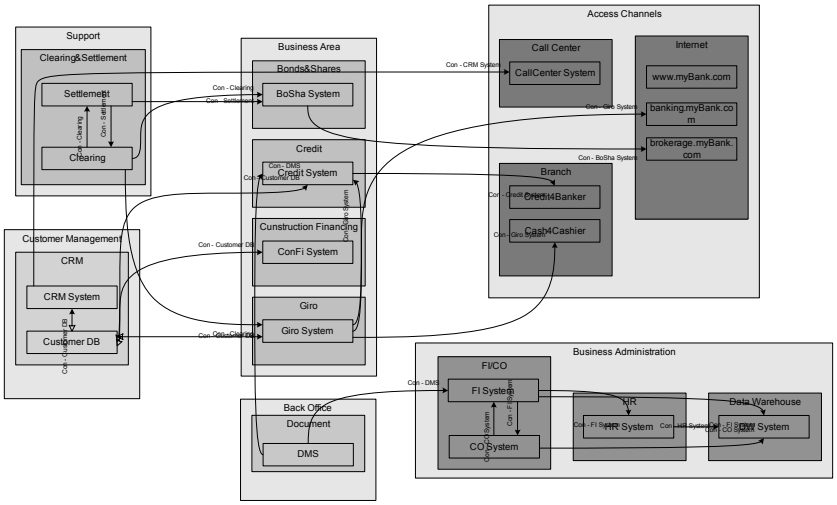

Abbildung 6 Beispiel für eine Softwarekarte von Typ Clusterkarte (2)

Bei diesem Kartentyp bilden logische Einheiten, die z. B. in der Organisation, deren Anwendungslandschaft untersucht wird, existieren, den Karten-

grund. Als logische Einheiten einsetzbar sind z. B. Funktionsbereiche, Organisationseinheiten oder auch geographische Einheiten wie Standorte, Städte oder Regionen. Eine Möglichkeit, die verschiedenen logischen Einheiten zu unterscheiden besteht beispielsweise, wie in der Abbildung 2 dargestellt, in der Verwendung eines Farbcodes.

Das Problem der Verortung von Elementen der Karte löst dieser Kartentyp, indem jedes Element in der logischen Einheit, zu der es in Beziehung steht, dargestellt wird. Damit stehen beim Erstellen einer derartigen Karte bereits grobe Regeln fest, wo z. B. ein bestimmtes Informationssystem darzustellen ist. Wenn zwischen den logischen Einheiten, die den Kartengrund bilden und den Informationssystemen im Informationsmodell eine n:m Beziehung besteht, existiert die Möglichkeit, dass ein System auf der Karte mehrmals erscheint.

Hervorzuheben ist, dass dieser Kartentyp nicht spezifiziert, wie die logischen Einheiten auf der Karte platziert werden und wie sich die verschiedenen Elemente innerhalb der Darstellung einer logischen Einheit anordnen. Dies bedeutet, dass die relative Position der Cluster zueinander, wenn diese nicht ineinander geschachtelt sind, keine formale Bedeutung besitzt.

3.1.2 Prozessunterstützungskarte

Eine Prozessunterstützungskarte ist ein Softwarekartentyp, der Anwendungssysteme anhand der von ihnen unterstützten (betrieblichen) Prozesse verortet.

Im Rahmen der Prozessorientierung und Unternehmensarchitekturen erhalten organisationseinheitsübergreifende Prozesse eine größere Bedeutung innerhalb von Unternehmen.[15] Obwohl zwischen den einzelnen Prozessschritten Schnittstellen existieren, besteht ein Ziel der Prozessorientierung (insb. auf den höheren Ebenen) darin, den kontinuierlichen Verlauf der Wertschöpfung im Prozess zu betonen. Die Unternehmensarchitektur gibt zusätzlich eine Gesamtsicht unter Berücksichtigung verschiedener Perspektiven auf das Unternehmen.

---

[15] Vgl. Scheer (2001), S. 7 ff.

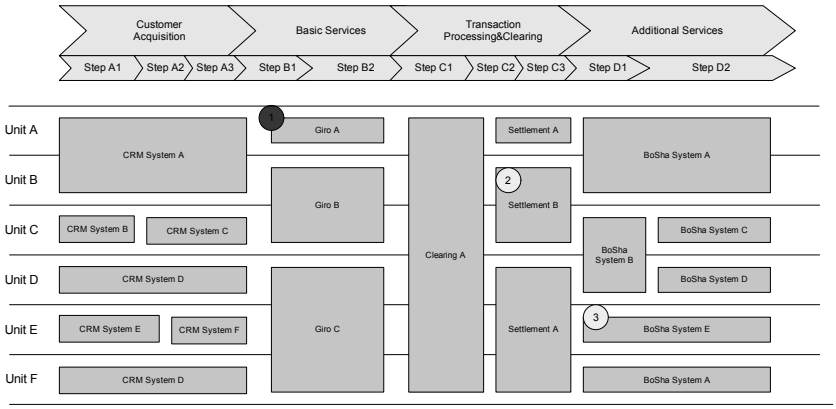

Abbildung 7 Beispiel für eine Softwarekarte von Typ Prozessunterstützungskarte

Durch die obige Einschränkung kommen Prozessdarstellungen auf höheren Ebenen (üblicherweise Ebenen 0 bis maximal 3) zum Einsatz, deren Darstellung, wie bei in Abbildung 7 gezeigt, als Wertschöpfungsketten erfolgt. Dieser Prozess legt typischerweise die x-Achse der Softwarekarte fest. Eine horizontale Ausdehnung eines Rechtecks, welches ein Informationssystem visualisiert, zeigt hierbei die Unterstützung eines Informationssystems für mehrere Prozesse an.

Da in einer Organisation meistens mehrere Prozesse existieren, die sich wie oben beschrieben, für die Verwendung in einer Softwarekarte eignen, muss ein bestimmter Prozess, dies kann der gesamte Primär-Prozess sein, zum Einsatz für die Verortung der Kartenelemente gewählt werden. Diese Auswahl kann sich am erwarteten Einsatzzweck der Karte orientieren. So ist es z. B. ein wesentlicher Einsatzzweck dieses Kartentyps, den Zusammenhang zwischen Prozessschritten und verwendeten Systemen zu untersuchen. Durch die gewachsene Anwendungslandschaft eines Unternehmens werden beispielsweise in gleichen Prozessschritten unterschiedliche Anwendungen mit gleicher Funktion eingesetzt, die aber selbst wiederum verschiedenste Prozessschritte bedienen können. Die Softwarekarte kann hier bei der Identifikation von Optimierungspotential und Redundanzen unterstützen.

Zur Verortung auf der y-Achse können bei diesem Kartentyp verschiedene Merkmale zum Einsatz kommen, wie z. B.:

- Systemtyp, z. B. dispositiv, operativ, administrativ
- Organisationseinheiten, die ein bestimmtes System nutzen
- Zeit, während der ein System besteht

Steht die Nutzung eines Informationssystems in einer bestimmten Organisationseinheit neben der Unterstützung im Vordergrund der Analyse, so werden die betroffenen Organisationseinheiten auf der y-Achse angeordnet (siehe Abbildung 7). In dieser Abbildung ergibt die Position und die vertikale Ausdehnung eines Rechtecks die Nutzung in einer bestimmten Organisationseinheit an.

3.1.3 Intervallkarte

Bei der Intervallkarte handelt es sich um einen Softwarekartentyp, der als charakterisierende Notationstechnik Balken einsetzt, deren Länge die Dauer von Vorgängen (z. B. Projekten) oder von Lebenszyklusphasen (z. B. eines Informationssystems) repräsentiert.

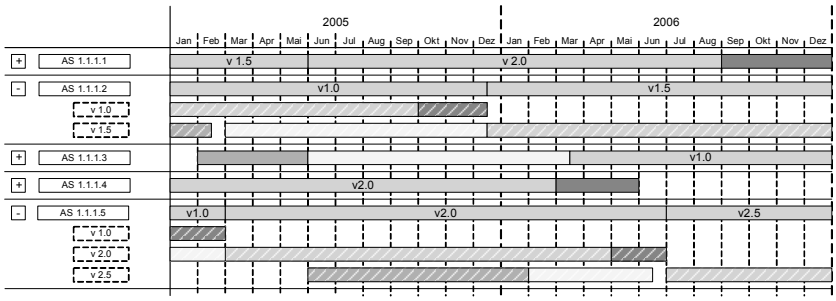

Abbildung 8 Beispiel einer Intervallkarte

Bei der Intervallkarte tritt der Aspekt Zeit, ein intervallskaliertes Merkmal, als Dimension zur Verortung in den Mittelpunkt. Auf der y-Achse stellt dieser Kartentyp, wie in der Abbildung 8 gezeigt, die verschiedenen Systeme dar. Damit besteht hier eine Nähe zu vorgangsbezogenen Gantt-Diagramm[16]en. Die Information, die die Karte bezüglich der Entwicklung der Infor-

mationssysteme über die Zeit hinweg enthält, lässt sich durch die Aufnahme von Versionsinformationen in die Darstellung ergänzen.

Auf dem eigentlichen Kartengrund stellt diese Karte mittels Balken dar, in welchen Zeiträumen sich die einzelnen Informationssysteme bzw. Versionen im Einsatz befinden. Auch hier können in konkreten Instanzen dieses Kartentyps Schichten zum Einsatz kommen, um weitere Informationen in die Darstellung zu integrieren. Eine Möglichkeit ist z. B. in den Balken, die ein Informationssystem/eine Version eines Informationssystems darstellen, mittels Farbcodierung anzugeben, in welchem Status sich das Informationssystem/eine Version zu den verschiedenen Zeitpunkten befindet (z. B. in Planung, in Entwicklung, in Betrieb, in Ablösung).

3.2 Softwarekarten ohne Kartengrund zur Verortung

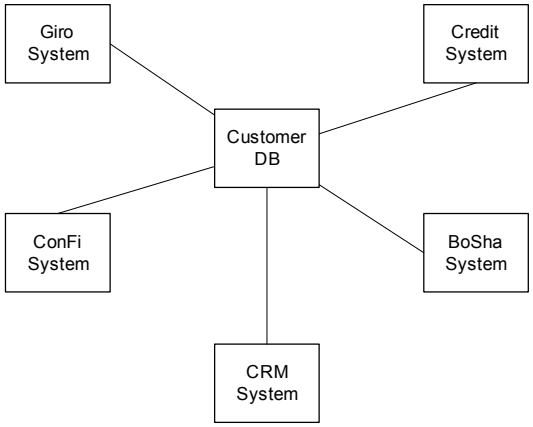

Abbildung 9 Beispiel für eine Softwarekarte ohne Kartengrund zur Verortung

Neben den oben genannten Typen existieren auch Karten, die den Kartengrund nicht zur Verortung von Elementen verwenden (siehe Abbildung 9). Die Verortung von Elementen auf einer derartigen Karte besitzt damit nicht mehr unbedingt eine festgelegte Bedeutung. Die Entscheidungen bezüglich

[16] Vgl. Balzert (1998).

der Positionierung der Elemente auf dem Kartengrund bleiben hier vollständig dem Ersteller der Karte überlassen. Damit rücken derartige Karten eher in die Nähe von graphischen Darstellungen, die nicht auf Verortung basieren, wie UML-Klassendiagramme[17] oder auch die ADLs ACME[18] oder RAPIDE[19]. Einschränkend muss hinzugefügt werden, dass bei Gestaltungsmitteln auf Schichten, die eine andere Schicht als den Kartengrund als Referenzschicht verwenden, die Verortung die Bedeutung des Gestaltungsmittels beeinflussen kann: Werden z. B. auf der Softwarekarte in Abbildung 9 zusätzliche Symbole genutzt, die weitere Aspekte eines Anwendungssystem repräsentieren, so spezifiziert die Verortung dieses Symbols die Zugehörigkeit zum Anwendungssystem.

4 Softwarekarten als Architekturbeschreibung

Einer der Hauptbeiträge der Softwarekartographie zum Architekturmanagement besteht in der Bereitstellung einer Methodik zur Dokumentation der Architektur von Anwendungslandschaften. Neben der Notation selbst erfordert eine nutzbringende Architekturdokumentation allerdings auch die Orientierung an *Best Practices* auf dem Gebiet der Architekturdokumentation, wie beispielsweise:

- Einsetzen unterschiedlicher Views, um verschiedene Aspekte des betrachteten Systems darzustellen[20]
- Trennen von abstrakter Viewdefintion (Viewpoint) und konkreten Views auf die Architektur eines Systems[21]
- Ausrichten der Beschreibung an Bedürfnissen von Stakeholdern

Zudem hat sich die Einigung auf gemeinsame Begriffsdefinitionen für die Architekturbeschreibung als schwierig erwiesen.22 Erschwerend bezüglich der Softwarekarten wirkt ebenso, dass auch für die aus der Kartographie übernommenen Konzepte Begrifflichkeiten gefunden werden müssen, um

---

[17] Vgl. OMG (2004).
[18] Vgl. Garlan/Monroe/Wile (1997).
[19] Vgl. Luckham (1996).
[20] Vgl. Clements et al. (2002); IEEE (2000).
[21] Vgl. IEEE (2000).

eine Kommunikation bezüglich des Einsatzes dieser Konzepte und damit deren sinnvollen Einsatz überhaupt erst zu ermöglichen.

Durch Begriffsdefinitionen sowie ein konzeptuelles Modell, die auf dem IEEE Std 1471-2000 (kurz IEEE 1471) aufbauen, können diese Anforderungen erfüllt werden.[22] Dadurch entstehen sowohl Begriffsdefinitionen als auch ein konzeptuelles Modell, die den Kontext von Architekturbeschreibungen für Anwendungslandschaften beschreiben und die Softwarekartographie im speziellen berücksichtigen.

Der IEEE 147122 ist auf Software-intensive Systeme anwendbar. Unter diese Bezeichnung fallen sämtliche Systeme, bei denen Software einen essentiellen Einfluss auf Design, Konstruktion, Einsatz (engl. *Deployment*) und Evolution des Systems in seiner Gesamtheit ausübt. Damit umfasst das Einsatzgebiet des IEEE 1471 auch Anwendungslandschaften.

Das Ziel des IEEE 1471 ist die Unterstützung des Dokumentierens, Explizierens und Kommunizierens von Architekturen. Im Gegensatz zu den verschiedenen Typen von Softwarekarten oder anderen Darstellungstechniken (wie z. B. UML), die graphische Notationen zur Beschreibung von Software-intensiven Systemen bereitstellen, adressiert der IEEE 1471 insbesondere die oben beschriebenen Probleme, die der Erstellung von eindeutigen, klaren und verständlichen Architekturbeschreibungen entgegenstehen. Er liefert klare Begriffsdefinitionen im Bereich der Architekturbeschreibungen.

Dabei liegen dem Standard u. a. folgende Paradigmen zugrunde:

- Jedes System hat eine Architektur, unabhängig davon, ob diese in einer Architekturbeschreibung (*Architectural Description*) expliziert ist.
- Eine Architekturbeschreibung muss den Bedürfnissen von Stakeholdern folgen, um nützlich zu sein.
- Ein einzelnes Artefakt kann (in den meisten Fällen) den Bedürfnissen aller Stakeholder nicht gerecht werden. Daraus folgt die Notwendigkeit der Verwendung mehrerer *Views*.
- Die Spezifikation einer *View* (ein *Viewpoint*) sollte nicht für jede Architekturbeschreibung neu erfunden werden. Stattdessen ist eine Wiederverwendung von *Viewpoints*, mittels der *Library Viewpoints*, anzustreben.

---

[22] Vgl. Lankes/Matthes/Wittenburg (2005b).

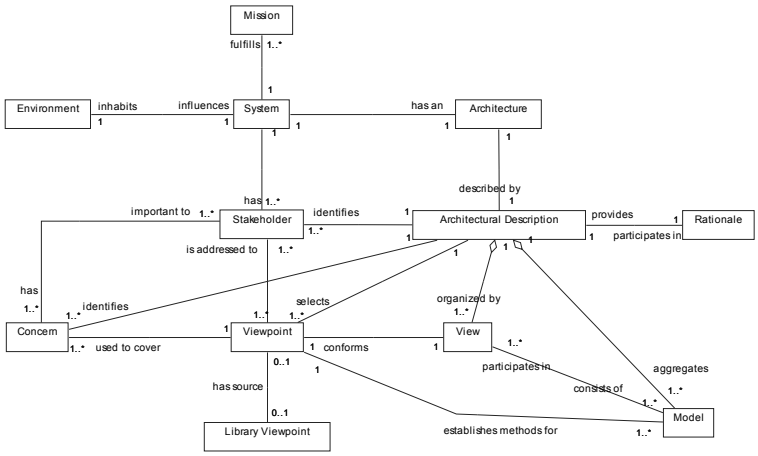

Abbildung 10 Konzeptuelles Modell des IEEE 1471

Das konzeptuelle Modell (engl. *Conceptual Model*) stellt die Zusammenhänge der Begriffe und der durch sie bezeichneten Konzepte dar. Die Notation dieses, in Abbildung 10 gezeigten, Modells ist an UML angelehnt, wobei im Gegensatz zum Originalmodell im IEEE 1471 die fehlenden Kardinalitäten ergänzt worden sind.

Für die in Abbildung 10 gezeigten zwölf Begriffe stellt der IEEE 1471 Definitionen bereit, die für das Management von Anwendungslandschaften und die Softwarekartographie erweitert werden müssen.[23]

Die Einführung des *Information Objects* (siehe Abbildung 11), das ein *Information Item* bezüglich der in einer Architekturbeschreibung zu beschreibenden Architektur darstellt, ist eine dieser Erweiterungen zur Architekturbeschreibung von Anwendungslandschaften. Im Gegensatz zu Einzelanwendungen, bei denen der Architekt oft lokal Zugriff auf die nötigen Informationen zur Dokumentation der Architektur des System bekommen kann, stellt die Gewinnung der in den Architekturmodellen abzubildenden Infor-

[23] Für eine ausführliche Diskussion der Begriffe und Konzepte im Kontext für das Management von Anwendungslandschaften und notwendige Erweiterungen wird auf Lankes/Matthes/Wittenburg (2005b) verwiesen.

mation bei Anwendungslandschaften eine der zentralen Herausforderung und einen bedeutenden Kostenfaktor bei der Erstellung der Dokumentation einer Anwendungslandschaft dar. Dieser Kostenfaktor ist sowohl in der Theorie[24] als auch in der Praxis bekannt und muss bei Entscheidungen zur Dokumentation der Architektur berücksichtigt werden. Die Problematik wird noch dadurch erschwert, dass die Kosten, die die Erhebung der Information auslöst, oft an anderen Stellen (z. B. in einzelnen Fachabteilungen) auftreten als der Nutzen durch die Dokumentation (z. B. in der IT- oder CIO-Abteilung).

Neben den Erweiterungen zur besseren Berücksichtigung des Managements von Anwendungslandschaften erscheint in Abbildung 11 das Konzept *Methodology*. Dieses wurde zusätzlich zum IEEE 1471 eingeführt, um einer Schwäche des Originalmodells (siehe Abbildung 10) entgegenzuwirken. Diese liegt darin, dass ein *Model* teil mehrerer *Views* sein kann. Da zu jedem *Model* genau ein *Viewpoint* existiert, können (indirekt) zu jedem *Model* mehrere *Viewpoints* vorhanden sein. Nach der Assoziation zwischen *Model* und *Viewpoint* legt allerdings ein *Viewpoint* die Methoden für ein *Model* fest. Damit stellt sich die Frage, welcher *Viewpoint* diese Funktion wahrnehmen soll.

Die oben erwähnte Problematik lässt sich auflösen, indem das abstrakte Gegenstück zum *Model*, entsprechend der Beziehung *Viewpoint* (abstrakt) zu *View* (konkret), eingeführt wird. Dieses besteht hier in der *Methodology*, die Methodiken und Techniken zur Konstruktion und Nutzung des entsprechenden Models liefert. Da die Etablierung von Methoden für ein *Model* jetzt von der *Methodology* übernommen wird, ändert sich die Beziehung *establishes methods for* zwischen *Viewpoint* und *Model* zu *selects methods for*.

In dieses konzeptuelle Modell lassen sich auch die Softwarekarten als eine Möglichkeit zur Dokumentation der Architektur von Anwendungslandschaften einordnen (siehe Abbildung 11).

---

[24] Vgl. Clements et al. (2003).

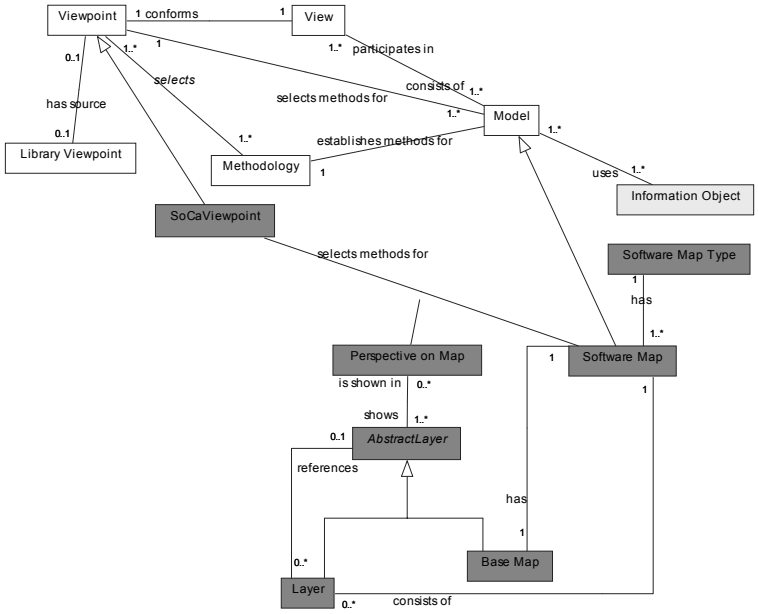

Abbildung 11 Erweiterung des konzeptuellen Modells des IEEE 1471

In diesem Kontext ist eine Softwarekarte eine spezielle Art eines *Models*, das bestimmte Aspekte einer Anwendungslandschaft entsprechend einer in einem *Viewpoint* festgelegten Notation zeigt. Da sich mit dem Wissen, dass ein *Viewpoint* als *Models* speziell Softwarekarten verwenden will einige, im normalen *Viewpoint* allgemein gehaltene, Dinge jetzt genauer spezifizieren lassen, wird als Spezialisierung des *Viewpoints* der *SoCaViewpoint* eingeführt.

Die Softwarekartographie verwendet zur Beschreibung von Anwendungslandschaften an die Kartographie angelehnte Notationen. Damit ist in der Softwarekartographie ein *Model* eine einzelne Softwarekarte, deren Notation in einem *Viewpoint* definiert wird. Eine *View* beschreibt somit (unter Nutzung eines oder mehrerer *Models*) die Architektur der Anwendungslandschaft aus einem bestimmten, durch die zum *Viewpoint* gehörigen *Concerns* festgelegten, Blickwinkel. Die Softwarekarte (als *Model*) baut dabei auf einer Auswahl von *Information Objects* auf.

Als Mittel zur Komplexitätsreduktion von Darstellungen bieten Softwarekarten die Möglichkeit, bestimmte Schichten (Abstract Layers) ein- oder auszublenden. Eine derartige Kombination aus angezeigten und nicht angezeigten *Abstract Layers* lässt sich als *Perspective on Map* identifizieren, die diejenigen *Abstract Layers* bezeichnet, die angezeigt werden bzw. sichtbar sind.

Im Rahmen eines bestimmten (SoCa)Viewpoints wird u.U. nicht die Information aller Schichten einer Softwarekarte benötigt, so dass ein *SoCaViewpoint* mehrere Perspektiven (*Perspectives on Map*) einsetzen kann.

5 Ausblick

Die konzeptuelle Trennung von semantischen und symbolischen Modell, die in Abschnitt 2 beschrieben wird, eröffnet die Möglichkeit, zwischen den Modellen eine regelbasierte Transformation zu etablieren. Die „Bidirectional Object Oriented Transformation Language“[25] (BOTL) ermöglicht bereits eine *partiell bijektive* Abbildung zwischen zwei Modellen die auf einem gemeinsamen Meta-Modell basieren. Im Forschungsprojekt Softwarekartographie wird ein Visualisierungsmodell inklusive eines passenden Meta-Modells erarbeitet, um die Nutzung von BOTL für die Softwarekartographie zu ermöglichen.

Ziel des Einsatzes dieser regelbasierten Abbildung zwischen dem semantischen und symbolischen Modell ist es, die verschiedenen Views (siehe Abschnitt 4) für Anwendungslandschaften, die auf Viewpoints mit Softwarekarten (siehe Abschnitt 3) basieren, automatisch zu generieren.

Des Weiteren werden in einem verwandten Projekt der Softwarekartographie Metriken für Softwarekarten entwickelt, um eine Bewertung von Handlungsalternativen etc. zu ermöglichen. Insbesondere die Analyse der Auswirkungen von Komplexität als Eigenschaften von Anwendungslandschaften bildet einen Fokus der Untersuchung. Diese Metriken ergänzen wiederum die Querschnittsfunktion *Measures & Metrics* des Informationsmodells (siehe Abschnitt 2.1) für Unternehmensarchitekturen.

---

[25] Vgl. Braun/Marschall (2003).

Literatur

Balzert, H. (1998): Lehrbuch der Software-Technik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung. 1. Auflage, Heidelberg, Berlin: Spektrum.

Braun, P./Marschall, F. (2003): BOTL–The Bidirectional Object Oriented Transformation Language. Technische Universität München, Lehrstuhl für Informatik 4, TUM-I0307, 2003.

Clements, P./Kazman, R./Klein, M. (2002): Evaluating Software Architectures. 1. Auflage, Reihe: The SEI Series in Software Engineering: Addison-Wesley.

Clements, P./Bachmann, F./Bass, L. et al. (2003): Documenting Software Architectures: Views and Beyond. 1. Auflage, Reihe: The SEI Series in Software Engineering, Boston: Addison-Wesley.

Garlan, D./Monroe, R./Wile, D. (1997): Acme: An Architecture Description Interchange Language. CASCON'97, Toronto, Ontario, Canada: IBM Press.

Hake, G./Grünreich, D./Meng, L. (2002): Kartographie. 8. Auflage, Berlin, New York: de Gruyter.

Hammer, M./Champy, J. (1993): Reengineering the Corporation: A Manifesto for Business Revolution. 1. Auflage, New York: HarperCollins Publishers.

IEEE (2000): IEEE Std 1471-2000 for Recommended Practice for Architectural Description of Software-Intensive Systems. IEEE Computer Society.

Lankes, J./Matthes, F./Wittenburg, A. (2005a): Softwarekartographie: Systematische Darstellung von Anwendungslandschaften. 7. Internationale Tagung Wirtschaftsinformatik 2005, Bamberg, Germany: Physica-Verlag.

Lankes, J./Matthes, F./Wittenburg, A. (2005 b): Architekturbeschreibung von Anwendungslandschaften: Softwarekartographie und IEEE Std 1471-2000. Software Engineering 2005, Essen,: Köllen Druck+Verlag.

Luckham, D. C. (1996): Rapide: A Language and Toolset for Simulation of Distributed Systems by Partial Orderings of Events. DIMACS Partial Order Methods Workshop IV, Princeton University: Princeton Press.

Maier, M. W./Emery, D./Hilliard, R. (2001): Software Architecture: Introducing IEEE Standard 1471. In: IEEE Computer, 34. Jg., Nr. 4, April 2001, S. 107–109.

Matthes, F./Wittenburg, A. (2004 a): Softwarekarten zur Visualisierung von Anwendungslandschaften und ihrer Aspekte. München, Deutschland: Technische Universität München, Lehrstuhl für Informatik 19 (sebis), Technischer Bericht TB0402.

Matthes, F./Wittenburg, A. (2004 b): Softwarekartographie: Visualisierung von Anwendungslandschaften und ihrer Schnittstellen. Informatik 2004–Informatik verbindet, 34. Jahrestagung der GI, Ulm, Deutschland: Köllen Druck+Verlag.

META Group (2002): Enterprise Architecture Desk Reference. META Group, Inc, 2002.

OGC (2000): ITIL–Service Delivery.: IT Infrastructure Library (ITIL), Norwich, UK: Office of Government Commerce, The Stationery Office.

OMG (2004): UML 2.0 Superstructure Specification (Revised Final Adopted Specification). Object Management Group.

Robinson, A. H./Morrison, J. L./Muehrcke, P. C. et al. (1995): Elements of Cartography. 6. Auflage, Hoboken (USA): John Wiley & Sons, Inc.

Scheer, A.-W. (2001): ARIS–Modellierungsmethoden, Metamodelle, Anwendungen. 4. Auflage, Berlin, Heidelberg, New York: Springer-Verlag.

van der Torre, L./Lankhorst, M. M./ter Doest, H. et al. (2004): Landscape Maps for Enterprise Architectures. Information Centre of Telematica Instituut TI/RS/2004/016.