Softwarekartographie: Systematische Darstellung von Anwendungslandschaften

Josef Lankes, Florian Matthes, André Wittenburg
Technische Universität München

Zusammenfassung: Ziel der Softwarekartographie ist es, unter Rückgriff auf Erkenntnisse und Methoden der Kartographie, komplexe Anwendungslandschaften in Unternehmen systematisch darzustellen und damit die Beschreibung, Bewertung und Gestaltung von Anwendungslandschaften zu verbessern. In Zusammenarbeit mit mehreren großen Unternehmen haben wir verschiedene Darstellungen – so genannte Softwarekarten – untersucht, die im Gegensatz zu Karten in der Kartographie keine geographische Verortung besitzen. In dieser Arbeit stellen wir verschiedene Kartentypen und ihre Sichten vor, die von unseren Projektpartnern und uns entwickelt wurden und bereits in der Praxis im Einsatz sind. Da eine manuelle Erstellung dieser Softwarekarten zu einem schlechten Kosten-Nutzen-Verhältnis führt, diskutieren wir im zweiten Teil der Arbeit eine adäquate Werkzeugunterstützung, die auf bestehenden Repository-gestützten Anwendungen aufbaut.

Schlüsselworte: Softwarekartographie, Softwarekarten, Anwendungslandschaften, Kartentypen

1 Einleitung

Die Anwendungslandschaft in einem Unternehmen besteht aus einer Vielzahl von Informationssystemen, die über verschiedene Technologien miteinander gekoppelt sind, unterschiedliche betriebliche Geschäftsprozesse unterstützen und durch zahlreiche Projekte einem steten Wandel unterliegen.

In Zusammenarbeit mit verschiedenen Unternehmen (u.a. Allianz, AXA Service, BMW, HVB Systems, T-Com) haben wir den Status quo bei der Beschreibung von Anwendungslandschaften untersucht und Anforderungen an adäquate Darstellungen erhoben. Parallel haben wir die verschiedenen existierenden Werkzeuge, die einen Bezug zum Management von Anwendungslandschaften besitzen, analysiert und Defizite bei der Erstellung und der Pflege von Softwarekarten identifiziert.

Das Ziel unseres Forschungsprojektes Softwarekartographie ist es in Zusammenarbeit mit unserem Projektpartnern

- einen Begriffsapparat zur Beschreibung von Anwendungslandschaften zu definieren,
- ein Modell für Softwarekarten zu entwickeln,
- das Modell adäquat in graphische Repräsentationen umzusetzen und
- den Projektpartnern Entscheidungshilfen zur Bewertung und Planung ihrer Anwendungslandschaften zu geben.

In Abschnitt 2 stellen wir unser Forschungsprojekt *Softwarekartographie* vor, zu welchen Ergebnissen wir bisher gelangt sind und wie Softwarekarten aufgebaut sind.

Unsere Projektpartner haben bereits eigene Softwarekarten für Anwendungslandschaften entwickelt, um für ihre individuellen Fragestellungen adäquate Darstellungen mit den relevanten Informationen zu erhalten. Abschnitt 3 kategorisiert diese Darstellungen, definiert verschiedene Typen von Softwarekarten und diskutiert die besondere Thematik des Kartengrundes.

Die Werkzeuge, die von unseren Projektpartnern eingesetzt werden, erfüllen deren Anforderungen an die Softwarekartographie nicht, da u.a. die Erstellung und Pflege der Softwarekarten mit einem hohen manuellen Aufwand verbunden sind. In Abschnitt 4 geben wir einen Überblick über die speziellen Anforderungen an ein Werkzeug zur Softwarekartographie und diskutieren einen Prototypen, der sich derzeit in der Entwicklung befindet.

2 Softwarekartographie

Die Softwarekartographie soll Unternehmen bei der Beschreibung, langfristigen Gestaltung und Bewertung ihrer Anwendungslandschaften unterstützen. Softwarekarten (Beispiel in Abbildung 1) sollen als Kommunikationsmittel dienen, Zusammenhänge zwischen relevanten Aspekten hervorheben und spezielle Fragestellungen, wie beispielsweise das Erkennen von Redundanzen, die Koordination von Projekten oder das Identifizieren von Abhängigkeiten, beantworten[1].

Die Anwendungslandschaften unserer Projektpartner variieren zwischen hundert und mehreren tausend Informationssystemen und sind für diese ein Investitionsgut, das nachhaltig geschützt werden muss. Ein erstes Ergebnis unserer Anforde-

[1] Die verschiedenen Softwarekarten in dieser Arbeit sind vereinfachte Darstellungen. Die Originale unserer Projektpartner sind wesentlich komplexer und enthalten vertrauliche Informationen.

rungsanalyse ist, dass eine ganzheitliche Betrachtung der Anwendungslandschaft auf verschiedenen Ebenen erfolgen muss [MaWi04a].

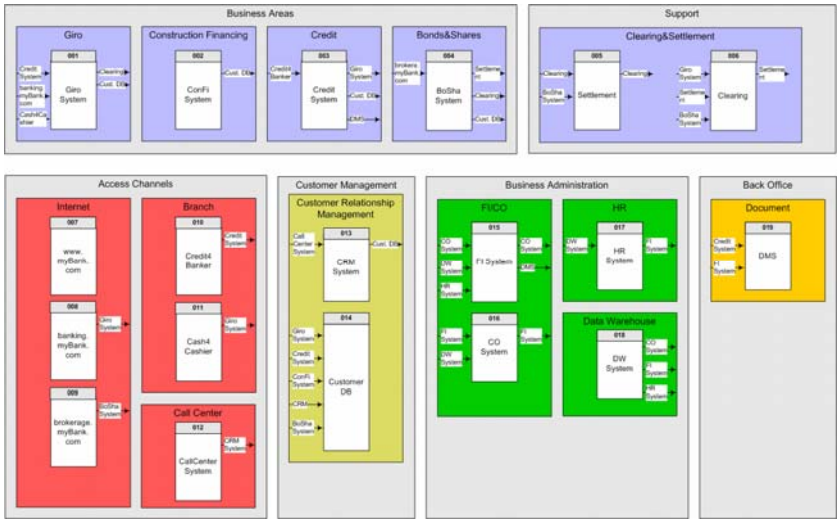

Abbildung 1 - Softwarekarte mit logischen Einheiten und Verbindungen

2.1 Betrachtungsebenen der Softwarekartographie

Die drei Betrachtungsebenen der Softwarekartographie sind in Abbildung 2 dargestellt. Die Softwarekartographie soll eine ganzheitliche Betrachtung der Anwendungslandschaft ermöglichen und nicht nur die Ebene „Wie?“ mit Informationssystemen, Schnittstellen etc. einbeziehen.

Analog zur thematischen Kartographie, bei der auf topographischen Kartengründen thematische Inhalte visualisiert werden (z.B. die Bevölkerungsdichte), kommen bei der Analyse von Anwendungslandschaften verschiedene Sichten zum Einsatz. Die Anwendungszwecke der einzelnen Sichten variieren und stellen je nach Anforderung unterschiedliche Aspekte bzw. Kennzahlen dar (siehe Abschnitt 2.2), die aus den Informationen der drei Betrachtungsebenen stammen.

Insbesondere die Rolle des Betrachters ist hierbei hervorzuheben, da unterschiedliche *Stakeholder* (vgl. [IEEE00; Clem+02]) verschiedene Sichten benötigen, um ihre spezifischen Fragestellungen zu beantworten.

Den unternehmerischen und strategischen Zielen eines Unternehmens wird auf der obersten Ebene („Warum?“) Rechnung getragen. IT-Strategien, die 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, die durch immer kürzere Produktzyklen und sich wandelnde Kundenanforderungen entstehen, 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 unterschiedliche Technologien realisiert sind, verschiedene Softwarearchitekturen benutzen etc. Die Vernetzung der Anwendungslandschaft, die durch zahlreiche Verbindungen über Schnittstellen zwischen den Informationssystemen realisiert ist, die unterstützenden Middleware-Systeme etc. komplettieren die Sicht auf die Anwendungslandschaft auf der technischen Ebene.

Die Fragestellungen „Warum?“, „Was?“ und „Wie?“ haben sich mit einem anderen Betrachtungswinkel bereits im *Business Process Reengineering*, für eine grundlegende Betrachtung von Geschäftsprozessen bewährt [HaCh93, S. 32-33].

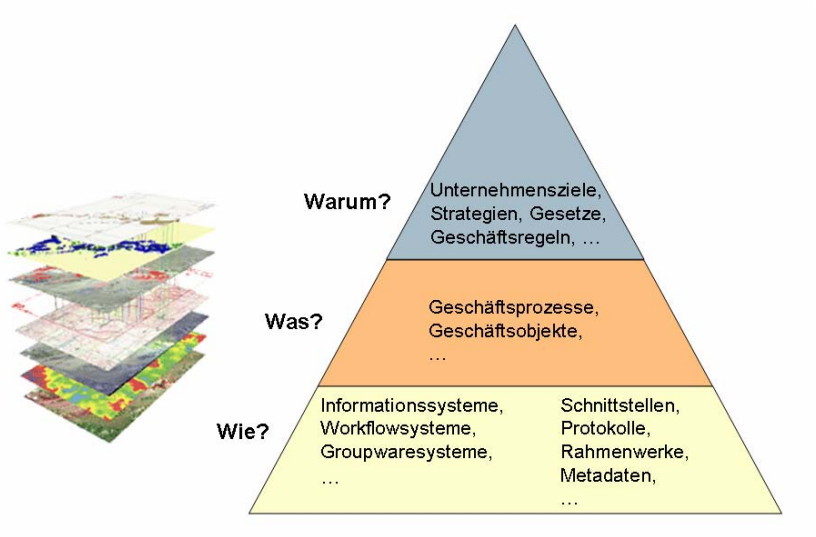

Abbildung 2 – Betrachtungsebenen der Softwarekartographie

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 dy-

namische Wandel wird durch Programme und/oder Projekte vollzogen, die möglicherweise parallel verschiedene Informationssysteme modifizieren.

2.2 Relevante Aspekte von Informationssystemen

In zahlreichen Gesprächen mit unseren Projektpartnern haben wir die relevanten Aspekte von Informationssystemen, die für eine Analyse der Anwendungslandschaft relevant sind und auf Softwarekarten visualisiert werden sollen, gesammelt und die folgende Kategorisierung in fünf Gruppen herausgearbeitet:

Planerische Aspekte umfassen die Evolution der Anwendungslandschaft über die Zeit. Programme und Projekte verändern Informationssysteme, schaffen neue oder lösen existierende ab. Dies führt zu einer Betrachtung von Ist-, Soll- und Plan-Anwendungslandschaften, die verschiedene Zyklen der Evolution analysieren. Ist-Anwendungslandschaften erfassen den Status quo und sind die Basis für die Abstimmung und Priorisierung von Projekten und Programmen im nächsten Planungsintervall.

Soll-Anwendungslandschaften aggregieren die Programme und Projekte eines Planungszeitraums und führen zu einer Anwendungslandschaft in der Zukunft. Unsere Projektpartner betrachten mehrere Soll-Anwendungslandschaften, die typischerweise halbjährliche oder jährliche Evolutionsschritte darstellen. Jede Soll-Anwendungslandschaft ergibt hierbei die Basis für den nächsten Planungszeitraum, um sicherzustellen, dass die Basis für den Planungszeitraum x+1 nicht die Ist-Anwendungslandschaft sondern der Planungszeitraum x ist.

Plan-Anwendungslandschaften sind zukunftsgerichtete Betrachtungen, die nicht durch geplante oder konkrete Projekte oder Programme entstehen, sondern strategische Ziele der Anwendungslandschaften aufnehmen. Projektpartner stützen die Unterscheidung zwischen Soll- und Plan-Anwendungslandschaften beispielsweise durch die Festlegung, dass Soll-Anwendungslandschaften ein Budget besitzen, Plan-Anwendungslandschaften hingegen nicht.

Wirtschaftliche Aspekte betrachten Kosten, die im Lebenszyklus eines Informationssystems entstehen bzw. entstehen werden. Verschiedene Kostenarten (Anschaffung, Wartung, Betrieb etc.), IT-Kennzahlen [Kütz03] und Balanced Scorecards [KaNo91] sollen visualisiert werden.

Prozesse, Organisationseinheiten, Geschäftsobjekte etc. bilden die Kategorie der *fachlichen Aspekte*. Die Anzahl von Nutzern oder der quantifizierte Nutzen (z.B. mittels Function Points [IFPU01]) zählen ebenso zu den fachlichen Aspekten.

Die Implementierungssprachen, die Verbindungen über Schnittstellen, die genutzten Middleware-Systeme oder die Softwarearchitekturen sind Beispiele für *technische Aspekte* von Informationssystemen. Unterscheidungen wie Individual- vs.

Standardsoftware in Kombination mit Eigen- oder Fremdentwicklung zählen gleichfalls zu den technischen Aspekten.

Die *operativen Aspekte* fokussieren auf den Betrieb von Informationssystemen und die verbundenen Ereignisse (zeitgesteuerte Abläufe, Batch-Jobs etc.). Datenflüsse und Kontrollflüsse mit bestimmten Abhängigkeiten bei Verarbeitungszyklen oder Domino-Effekte bei Ausfällen werden von den operativen Aspekten berücksichtigt. Das Problem-Management, wie es in der IT Infrastructure Library diskutiert wird, detailliert die operativen Aspekte um beispielsweise „mean time between failure“ oder „number of incidents“ [OGC00].

Bei der Umsetzung der verschiedenen relevanten Aspekte in ein Modell muss berücksichtigt werden, dass Unternehmen nicht ad hoc über alle Informationen verfügen, sondern einzelne Aspekte ggf. nachträglich aufnehmen. Des Weiteren ist die Beziehung zwischen erfassbaren und pflegbaren Aspekten sowie dem Nutzen zu beachten, da mit dem entstehenden Aufwand nicht immer ein adäquater Nutzen einhergeht.

2.3 Aufbau von Softwarekarten

Softwarekarten als graphische Repräsentationen von Anwendungslandschaften sollen Informationssysteme, die verschiedenen relevanten Aspekte und die Beziehungen zwischen ihnen visualisieren.

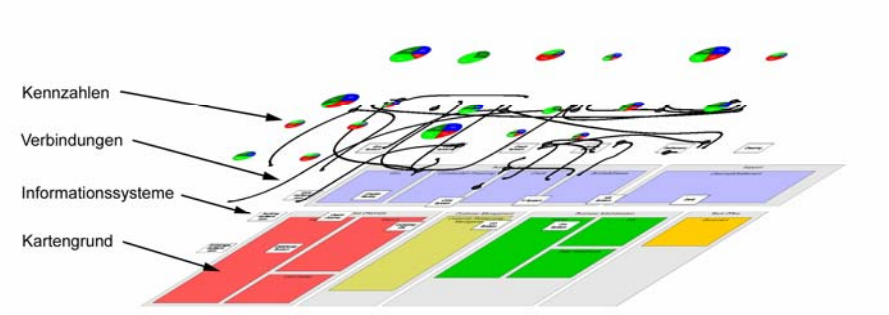

Abbildung 3 – Schichtenaufbau von Softwarekarten

Abbildung 3 zeigt beispielhaft den Schichtenaufbau einer Softwarekarte, die aus einem Kartengrund und mehren Schichten besteht. Auf den Kartengrund, der je nach Kartentyp (siehe Abschnitt 3) variiert, werden die unterschiedlichen Schich-

ten aufgetragen, wobei zu jeder Schicht eine Referenzschicht existiert, auf die sich die Elemente dieser Schicht beziehen[2].

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 Softwarekarte zu erhalten.

Existierende Visualisierungssprachen im Bereich des Software-Engineering fokussieren auf der Darstellung eines einzelnen Softwaresystems und berücksichtigen dabei nur Teile der oben genannten Betrachtungsebenen und relevanten Aspekte.

UML [OMG03] 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 (vgl. [MaWi04a]). Ein Schichtenaufbau eines Diagramms, um für unterschiedliche Fragestellungen bei gleich bleibendem Kartengrund die entsprechenden Informationen zu selektieren, ist in UML ebenso nicht möglich.

3 Typen von Softwarekarten

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 manipulieren [Robi+95, S. 9-10].

Damit bietet 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 ist immer ein topogra-

[2] Beispielsweise würde eine Schicht mit Verbindungen, die Verbindungen zwischen Informationssystemen visualisiert, auf die Schicht mit Informationssystemen referenzieren.

phischer, 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.“ [Robi+95, S. 9]

Informationssysteme als Elemente einer Anwendungslandschaft lassen sich anhand verschiedenster Merkmale beschreiben (unterstützte Prozessschritte, Zeitraum des aktiven Betriebs, betreibende 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.

Durch Auswahl bestimmter Merkmale als Dimensionen, die dann zur Bildung des Kartengrunds (zur Verortung) zum Einsatz kommen, erhält der „Softwarekartograph“ zusätzlichen Freiraum bei der Gestaltung der Karte.

Bei den Projektpartnern, mit denen wir im Rahmen des Projekts „Softwarekartographie“ zusammenarbeiten, haben wir Karten identifiziert, die sich anhand der Merkmale, auf denen die Verortung der Elemente basiert, in vier Kartentypen einteilen lassen.

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[3].

Durch den gleichartigen Aufbau und eine gleich bleibende Positionierung von Elementen auf der Softwarekarte entsteht zusätzlich ein Wiedererkennungswert, der es dem Betrachter erleichtert, sich schnell in der Karte zu orientieren.

3.1.1 Clusterkarte

Bei diesem Kartentyp bilden logische Einheiten, die in der Organisation, deren Anwendungslandschaft untersucht wird, existieren, den Kartengrund. Als logische Einheiten einsetzbar sind z.B. Funktionsbereiche, Organisationseinheiten oder sogar geographische Einheiten wie z.B. Standorte, Städte oder Regionen. Eine Mög-

[3] Bei anderen graphischen Darstellungen, wie z.B. UML-Klassendiagrammen ist dies nicht der Fall. Einen auf diesem Visualisierungskonzept aufbauenden Kartentyp beschreibt Abschnitt 3.2.

lichkeit, die verschiedenen logischen Einheiten zu unterscheiden besteht z.B., wie in Abbildung 4 dargestellt, in der Verwendung eines Farbcodes[4].

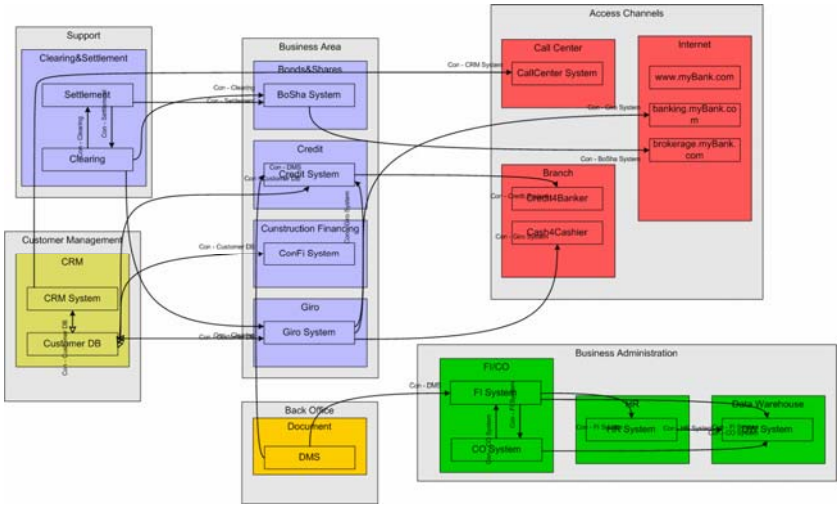

Abbildung 4 – Softwarekarte vom Typ Clusterkarte

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 (mit den Kartenelementen, die best. Attribute dieses Systems visualisieren) darzustellen ist. Es besteht die Möglichkeit, dass ein System, das zu mehreren der in der Darstellung verwendeten logischen Einheiten gehört, auf der Karte mehrmals erscheint.

Dieser Kartentyp spezifiziert nicht, wie die logischen Einheiten auf der Karte platziert werden und wie sich die verschiedenen Elemente innerhalb der Darstellung einer logischen Einheit anordnen. Bezüglich dieser Fragestellungen wurden bei den Projektpartnern unterschiedliche Vorgehensweisen identifiziert:

- platzoptimierte Verortung
- konventionsgestützte Verortung

Bei der konventionsgestützten Verortung werden beispielsweise Einheiten mit Kundenkontakt rechts und Einheiten mit Lieferantenkontakt links auf der Karte angeordnet.

[4] Je nach Verwendung eines Farb- und/oder S/W-Drucks muss der Farbcode entsprechend optimiert werden.

Als Mittel zum Erhöhen der Lesbarkeit stehen bei diesem Kartentyp zur Verfügung:

- Verortung der logischen Einheiten bzw. der Elemente in den logischen Einheiten
- Selektive Informationspräsentation über Schichten

Für eine selektive Informationspräsentation nutzen Abbildung 1 und Abbildung 4 unterschiedliche Darstellungsweisen der Verbindungen zwischen Informationssystemen. Abbildung 1 visualisiert die Schnittstellen mit kurzen Pfeilen, die mit dem Quell- bzw. Zielsystem beschriftet sind; Abbildung 4 verbindet die Informationssysteme und verwendet unterschiedliche Pfeilenden für Kommunikationsarten. Mittels Schichten könnte je nach Bedarf diese Information ein- oder ausgeblendet werden. Für weitere Darstellungsmöglichkeiten von Anwendungslandschaften und Verbindungen zwischen Informationssystemen wird auf [MaWi04b] verwiesen.

3.1.2 Prozessunterstützungskarte

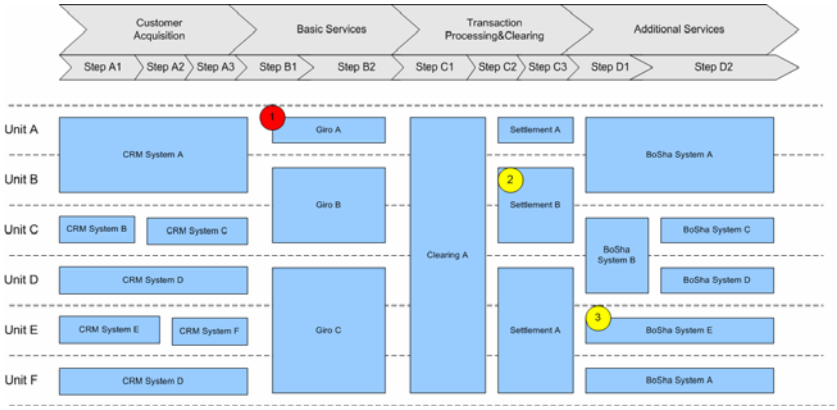

Abbildung 5 – Prozessunterstützungskarte mit Geschäftseinheiten

Im Rahmen der Prozessorientierung erhalten organisationseinheitsübergreifende Prozesse eine größere Bedeutung innerhalb von Unternehmen [Sche01, S. 7-11]. 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.

Damit bietet ein Prozess ähnliche Möglichkeiten im Rahmen der Verortung wie *eine* räumliche Dimension. Eine Voraussetzung für eine derartige Verwendung eines Prozesses ist, dass dieser linear abläuft. Deshalb kommen hier Prozessdarstellungen auf höheren Ebenen (üblicherweise Ebenen 0 bis maximal 3) zum Einsatz, deren Darstellung, wie in Abbildung 5 gezeigt, als Wertschöpfungsketten erfolgt. Dieser Prozess legt typischerweise die x-Achse der Softwarekarte fest.

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 in der Verortung der Kartenelemente gewählt werden. Diese Auswahl kann sich am erwarteten Einsatzzweck der Karte orientieren. So ist es z.B. ein wichtiger 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.

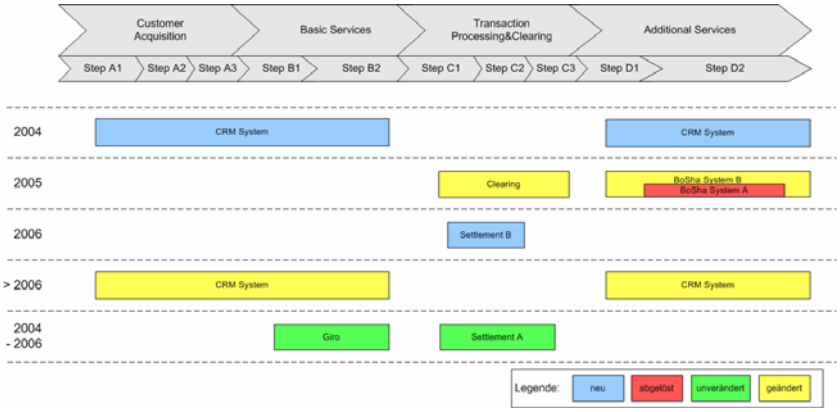

Abbildung 6 – Prozessunterstützungskarte mit Evolutionsschritten

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

- Geschäftseinheiten, zu denen ein System gehört (siehe z.B. Abbildung 5)
- Zeit, während der ein System besteht (siehe z.B. Abbildung 6)
- Systemtyp, z.B. dispositiv, operativ, administrativ (siehe z.B. Abbildung 7)

Die Karte in Abbildung 6 stellt die Evolution der einzelnen Informationssysteme, die den Prozess auf der x-Achse unterstützen, dar. In dem Betrachtungsintervall unveränderte Informationssysteme werden in der untersten Zeile (im Beispiel Zeitraum 2004 bis 2006) dargestellt. Die anderen Zeilen zeigen Neuerstellung, Ablösung und Veränderung von Informationssystemen in den jeweiligen Planungszeiträumen. Neue, abgelöste, unveränderte und geänderte Systeme werden durch eine Farbkodierung dargestellt.

Die Prozessunterstützungskarte weist größere Ähnlichkeit zu den Darstellungen der Kartographie auf als die Verortung nach logischen Einheiten. Die logischen

Einheiten stellen eher ein nominales Merkmal dar und müssen deshalb erst unter Festlegung einer Vorgehensweise auf dem Kartengrund positioniert werden. Dagegen basiert die Verortung bei Karten vom Typ Prozessunterstützung auf der Auswahl von ein oder zwei Merkmalen[5] als Dimensionen zur Verortung. Dabei lässt sich mindestens eine Dimension, nämlich der Prozess, als ordinal betrachten, was diese besonders geeignet zur Verortung macht, wenn auch nicht der Eignungsgrad von Entfernungen, wie sie in der Kartographie verwendet werden, die ein metrisches Merkmal darstellen, erreicht wird.

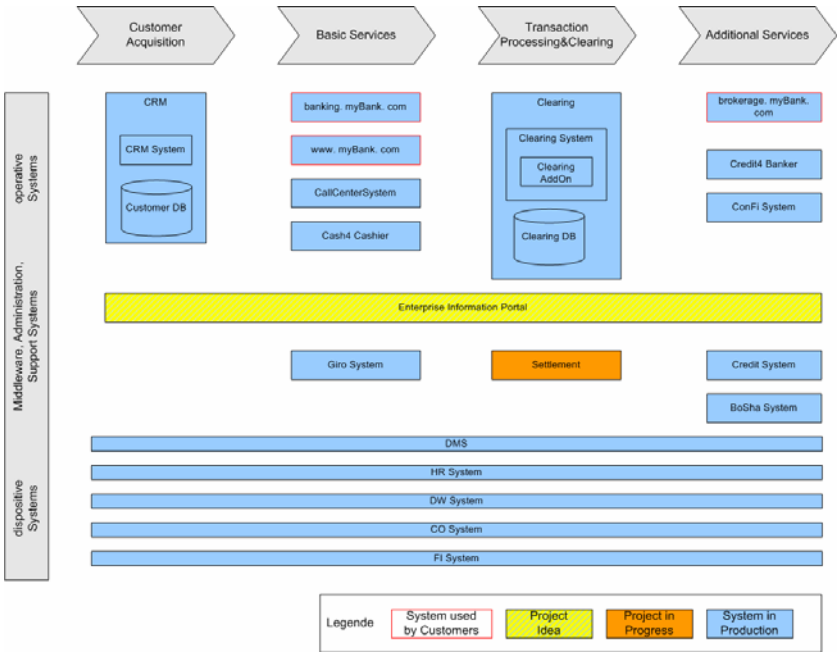

Abbildung 7 - Prozessunterstützungskarte mit Systemtypen

3.1.3 Intervallkarte

Als mögliche Dimension zur Verortung wurde in Abschnitt 3.1.2 die Zeit erwähnt, dieser Aspekt tritt bei der Intervallkarte in den Mittelpunkt. Sie verwendet als Dimension zur Verortung auf der x-Achse die Zeit, ein intervallskaliertes Merkmal.

Auf der y-Achse stellt dieser Kartentyp, wie in Abbildung 8 gezeigt, die verschiedenen Systeme dar. Damit besteht hier eine Nähe zu vorgangsbezogenen Gantt-Diagrammen [Balz98, S. 32-42]. Die Information, die die Karte bezüglich der Ent-

5 Die Verwendung der y-Achse ist optional, auch eine beliebige Anordnung entlang der y-Achse ist möglich.

wicklung der Informationssysteme über die Zeit hinweg enthält, lässt sich durch die Aufnahme von Versionsinformationen in die Darstellung, wie in Abbildung 8 geschehen, ergänzen.

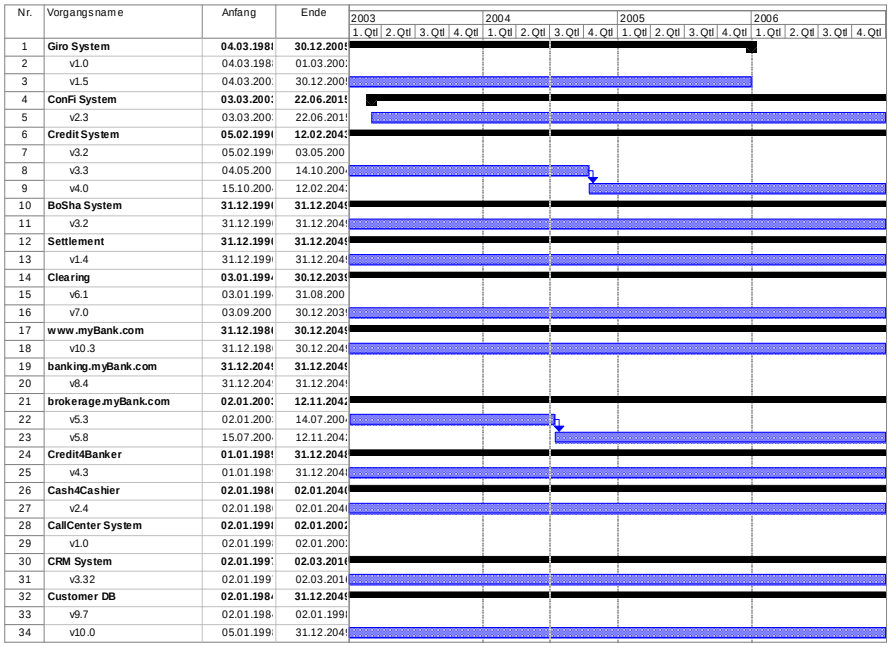

Abbildung 8 – Intervallkarte mit Versionsinformationen

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 wäre z.B. in den Balken, die ein Informationssystem/eine Version eines Informationssystems darstellen, mittels Farbcodierung anzugeben, in welchem Stadium ihres Lebenszyklus sich die Anwendung/Version zu den verschiedenen Zeitpunkten befindet.

3.2 Softwarekarten ohne Kartengrund zur Verortung

Neben den in Abschnitt 3.1 vorgestellten Kartentypen existieren auch Karten, bei denen die Verortung von Elementen auf der Karte keine festgelegte Bedeutung besitzt. Die Entscheidungen bezüglich 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 [OMG03] oder auch die ADLs ACME [Garl+97] oder RAPIDE [Luck96].

Meistens kommt dieser Kartentyp zum Einsatz, um speziell auf eine Problemstellung hin optimierte Visualisierungen einer Anwendungslandschaft oder von Ausschnitten daraus zu generieren.

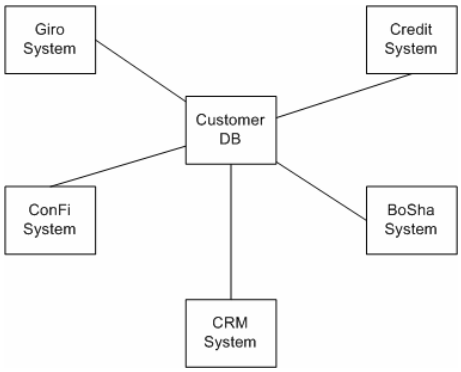

Abbildung 9 – Ad hoc generierte Softwarekarte

Aus der Domäne der Darstellung von Graphen sind verschiedene Positionierungsregeln bekannt, die zu übersichtlichen Darstellungen führen sollen [Purc+01]:

- Überschneidungen zwischen Verbindungslinien minimieren
- Kästchen (z.B. Klassen) gleichmäßig über das Diagramm verteilen
- Verbindungslinien zwischen Kästchen von ähnlicher Länge verwenden
- Gerichtete Verbindungen möglichst in die gleiche Richtung zeigen lassen

Solche Regeln können die Entwicklung von Layout-Algorithmen [Gutw+04, Wies+04] anleiten, die Softwarekarten automatisch aus einem Repository mit Daten zu einer Anwendungslandschaft generieren. Ein derartiger Algorithmus könnte z.B. die in Abbildung 9 gezeigte Karte erzeugen, indem er ein vom Benutzer auszuwählendes System in der Mitte der Karte positioniert und die Systeme, die Verbindungen zum ausgewählten System aufweisen, kreisförmig um dieses herum anordnet.

Stärker als die Befolgung von Regeln wie den oben genannten, die allgemein für die Darstellung von Graphen entwickelt wurden, wirkt sich die Berücksichtigung von inhaltlichen Aspekten, die nicht direkt in den Gestaltungselementen ausgedrückt werden können, auf die Übersichtlichkeit aus. So ist es z.B. möglich, über die Gruppierung bestimmter Elemente auszudrücken, dass sie im Rahmen des betrachteten Problems eng zusammen hängen [Purc+01]. In dieser Hinsicht erscheint es nützlich, Benutzern die Möglichkeit zu geben, die Positionierung der Elemente - wie z.B. bei UML-Werkzeugen üblich - manuell anzupassen.

Durch die relativ großen Freiheiten bei der Verortung, die dieser Kartentyp bietet, lassen sich, unter anderem durch Beachtung von Regelungen wie den oben erwähnten, Darstellungen erzeugen, die eine spezielle Problematik mit guter Übersichtlichkeit visualisieren. Die Möglichkeit der freien Positionierung der Elemente steht hier neben der selektiven Darstellung von Information (über Schichten) als weiteres Mittel zur Komplexitätsreduktion bereit.

Mangels eines Kartengrundes der auf Dimensionen zur Verortung basiert, fehlt bei diesen Karten die Möglichkeit, bestimmte Informationen mittels Positionierung von Elementen exakt auszudrücken. Beziehungen zwischen Informationssystemen und Prozessschritten müssten beispielsweise durch Assoziationslinien ausgedrückt werden.

4 Werkzeugunterstützung

Ein adäquates Werkzeug für die Softwarekartographie muss neben der Unterstützung der in Abschnitt 3 beschriebenen Softwarekartentypen weitere Anforderungen erfüllen. Der in Abschnitt 2.3 beschriebene Aufbau von Softwarekarten fordert beispielsweise einen mehrschichtigen Aufbau von Softwarekarten auf Basis eines zu definierenden Kartengrundes.

Der Kartengrund visualisiert je nach Kartentyp (mehrere) Instanzen unterschiedlicher Objekttypen (Prozessschritte, Funktionsbereiche etc.) und wird in Abhängigkeit vom Anwendungszweck aufgebaut. Dem Kartengrund und den Elementen werden auf einer darüber liegenden Schicht Informationssysteme zugeordnet, die mittels eines entsprechenden Filters aus allen Informationssystemen zur Visualisierung ausgewählt werden.

Die nächst höheren Schichten visualisieren relevante Aspekte mit einem Bezug zu den darunter liegenden Schichten (vgl. Referenzschicht in Abschnitt 2.3) und können je nach Bedarf ein- und ausgeblendet werden.

In unserer Anforderungsanalyse haben wir verschiedene Werkzeuge zur Softwarekartographie identifiziert, die wir in Abschnitt 4.1 betrachten. Beispiele für Objekte, die für ein Meta-Modell der Softwarekartographie relevant sind, werden in Abschnitt 4.2 vorgestellt, um anschließend unser Konzept für ein Werkzeug zur Softwarekartographie in Abschnitt 4.3 vorzustellen.

4.1 Repository-Unterstützung zur Softwarekartographie

Eine erste grobe Klassifikation, der von uns identifizierten Werkzeuge [MaWi04a], unterteilt diese in Repository-gestützte und nicht Repository-

gestützte, die verschiedene Vor- und Nachteile zur Erstellung und Pflege von Softwarekarten besitzen.

In der Klasse der nicht Repository-gestützten Werkzeuge finden sich u.a. Microsoft PowerPoint und Microsoft Visio, die von einigen Projektpartnern zur Erstellung ihrer Softwarekarten eingesetzt werden. Der Vorteil dieser Werkzeuge liegt in dem Freiheitsgrad, den der Kartenersteller beim Modellieren besitzt. Beispielsweise können Elemente auf der Karte beliebig positioniert werden oder Farben, Größen etc. frei gewählt werden.

Der Nachteil dieser nicht Repository-gestützten Werkzeuge liegt in dem hohen manuellen Aufwand und der Fehleranfälligkeit bei der Erstellung von Softwarekarten. Existierende Daten für die Softwarekarten über Informationssysteme, Schnittstellen etc. werden manuell in die Softwarekarte übertragen. Dieser Prozess der manuellen Datenübernahme führt ebenso zu Fehlern, da ein Abgleich der Daten zwischen dem Quell-Datenbestand und den Softwarekarten nicht automatisch möglich ist.

Der Vorteil der Repository-gestützten Werkzeuge liegt in der Erfassung und Pflege der Daten. Einige unserer Projektpartner setzen derartige Werkzeuge bereits zur Planung ihrer Anwendungslandschaft ein und pflegen Daten über Prozesse, Informationssysteme, Organisationseinheiten etc. in den Repositories. Allerdings sind nicht alle Meta-Modelle dieser Werkzeuge variabel genug, um die relevanten Aspekte in geeigneter Form im Repository zu speichern.

Bei der Erstellung von Softwarekarten bietet keines der analysierten Repository-gestützten Werkzeuge die Möglichkeit verschiedene Aspekte mittels unterschiedlicher Schichten auf einem gemeinsamen Kartengrund aufzutragen und einzelne Schichten Ein- und Auszublenden. Auch die Nutzung von Gestaltungsmitteln (Linien, Signaturen, Schrift etc.) und -variablen (Farbe, Form, Größe etc.) in Abhängigkeit von Daten des Repositories genügt nicht den Anforderungen der Softwarekartographie.

Eine Auflistung der von uns für die Softwarekartographie untersuchten Werkzeuge findet sich in [MaWi04a].

4.2 Objekte eines Meta-Models für die Softwarekartographie

Durch die Analyse der relevanten Aspekte und der existierenden Softwarekarten wurden verschiedene Objekte und Assoziationen zwischen diesen Objekten identifiziert. Im Gegensatz zu Meta-Modellen, wie sie Werkzeugen zur Prozessmodellierung zugrunde liegen, fokussiert ein Modell für die Softwarekartographie auf die Informationssysteme der Anwendungslandschaft, die Beziehungen zwischen diesen und ihre relevanten Aspekte. Meta-Modelle zur Prozessmodellierung fokussieren hingegen auf Funktionen, Ereignissen, Daten etc. und deren Simulation.

Ein Meta-Modell für die Softwarekartographie muss des Weiteren zwischen optionalen und obligatorischen Objekten, Assoziationen und Attributen von Objekten unterscheiden (siehe Abschnitt 2.2).

Eine Softwarekarte vom Typ Prozessunterstützungskarte, die die Abhängigkeit zwischen Informationssystemen, den unterstützten Prozessen und den Einsatzbereichen visualisieren soll, benötigt beispielsweise neben den Prozessschritten und den Einsatzbereichen die Beziehungen zwischen diesen Objekten und den Informationssystemen, um die Objekte richtig platzieren zu können.

Unternehmen wollen zusätzlich die Möglichkeit besitzen, die für sie relevanten Softwarekartentypen sowie die darauf darzustellenden Elemente auszuwählen und das Repository inkrementell zu erweitern. Entscheidungen in diesem Zusammenhang sollen auf Basis von Kosten/Nutzen-Verhältnissen getroffen werden können.

Neben den oben angeführten Objekten Prozessschritte, Einsatzbereiche und Informationssysteme zählen beispielsweise Verbindungen zwischen Informationssystemen, Konnektoren, die Schnittstellen anbieten, Organisationseinheiten oder Projekte zu den Objekten des Meta-Modells zur Softwarekartographie.

4.3 Architektur eines Werkzeuges zur Softwarekartographie

Die wesentlichen Anforderungen an ein adäquates Werkzeug zur Softwarekartographie sind:

- Import von Datenbeständen aus Repository-gestützten Anwendungen
- Abbildung der Import-Daten auf ein generisches Meta-Modell für Anwendungslandschaften
- Abbildung der Objekte des Meta-Modells auf Gestaltungsmittel und -variablen
- Export von Softwarekarten in unterschiedlichen Daten- und Grafikformaten

Die existierenden Datenbestände in den Repository-gestützten Anwendungen (siehe Abschnitt 4.1) sollen für die Erstellung von Softwarekarten genutzt werden, wobei die Hoheit über diese Daten in den existierenden Repositories verbleibt.

Die Abbildung der Import-Daten auf ein geeignetes Meta-Modell erfordert, so lange kein einheitliches Datenformat für den Export aus den Repository-gestützten Anwendungen existiert, eine Konvertierung in das Meta-Modell für die Softwarekartographie, um eine Abbildung der Objekte auf die Gestaltungsmittel der Softwarekarten zu ermöglichen.

Der Modellierer für Softwarekarten filtert nach der Konvertierung in einem ersten Schritt die relevanten Daten für einen Kartengrund aus dem Datenmodell und ordnet diese ggf. manuell an, wenn keine Verortung aus den gefilterten Objekten abzuleiten ist.

Nach der Definition des Kartengrundes werden über weitere Filter die Informationssysteme und die relevanten Aspekte ausgewählt und schrittweise auf der Softwarekarte visualisiert. Je nach Art des relevanten Aspektes (z.B. quantifizierbar oder nicht quantifizierbar) werden unterschiedliche Gestaltungsmittel für die einzelnen Schichten gewählt.

Die Informationen über eine erstellte Softwarekarte müssen in einem Repository gespeichert werden. Persistente Informationen sind beispielsweise die Position von Elementen, die Filter der einzelnen Schichten etc. Zusätzlich müssen Versionsinformationen der einzelnen Objekte gespeichert werden, um bei einer erneuten Generierung den Modellierer auf neue, geänderte oder gelöschte Objekte hinweisen zu können und ggf. die Evolution der Anwendungslandschaft zu visualisieren.

Abbildung 10 stellt die Architektur eines Werkzeuges zur Softwarekartographie dar und insbesondere die Beziehungen zu der Repository-gestützten Anwendung und der Intranet-Applikation.

Die derzeitige Architektur bei einigen unserer Projektpartner nimmt die Daten aus dem Repository und setzt diese mittels einer *Template-Engine* in so genannte *Knowledge-Pages* (*K-Pages*) um. Die erzeugten K-Pages werden im Intranet bereitgestellt, um die Informationen über die Anwendungslandschaft möglichst vielen Benutzern zur Verfügung zu stellen. Derzeit enthalten die K-Pages manuell erstellte Softwarekarten und zusätzliche Informationen über Informationssysteme und die Anwendungslandschaft.

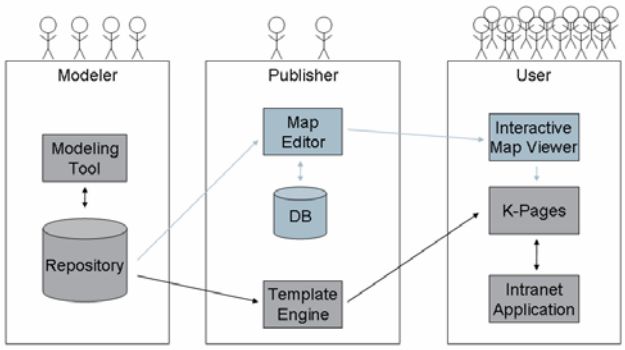

Abbildung 10 – Architektur eines Werkzeuges zur Softwarekartographie

Das Werkzeug zur Softwarekartographie soll diesen Workflow dahingehend ergänzen, dass die Daten aus dem Repository einem *Map Editor* zur Verfügung gestellt werden, mit dem Softwarekarten – wie oben beschrieben – erstellt werden können. Die von einem *Publisher* mit dem Map Editor erstellten Softwarekarten werden dann über eine Export-Funktion in die existierenden K-Pages eingebunden, um eine Integration in das existierende Intranet zu ermöglichen.

5 Zusammenfassung und Ausblick

In dieser Arbeit haben wir unser Forschungsprojekt Softwarekartographie vorgestellt, in dem wir mit verschiedenen Unternehmen Modelle und Methoden zur Beschreibung, Bewertung und Gestaltung von Anwendungslandschaften entwickeln.

Aus den existierenden Softwarekarten und neu entwickelten haben wir vier Typen von Softwarekarten abgeleitet. Diese unterscheiden sich in dem zugrunde liegenden Aufbau des Kartengrundes, der mit ihnen verfolgten Zielsetzung und dem Redaktionsprozess (automatisch vs. semi-automatisch vs. manuell).

Bedingt durch den hohen Aufwand, den die Erstellung und Pflege von Softwarekarten verursacht, haben wir ein Konzept für ein Werkzeug zur Softwarekartographie entwickelt. Dieses Konzept sieht hierbei eine Integration in existierende Systemumgebungen vor, die bereits über Repositories mit Daten der Anwendungslandschaft und Intranet-Anwendungen zur Publikation von Softwarekarten verfügen.

Um eine Interoperabilität mit einer großen Anzahl Repository-gestützter Anwendungen zu erreichen – ohne zusätzliche Aufwände für Konverter –, ist ein gemeinsames Austauschformat nötig. Den derzeit entstehenden Begriffsapparat und das Meta-Modell zur Softwarekartographie werden wir in den kommenden Projektschritten mit unseren Projektpartner abstimmen und konkretisieren. Dieser Begriffsapparat und das Meta-Modell bilden die Basis für ein mögliches Austauschformat.

Das parallel entstehende Werkzeug zur Softwarekartographie wird die Machbarkeit zur semi-automatischen Erstellung von Softwarekarten zeigen und entsteht in einer Systemumgebung mit einem Modellierungswerkzeug inkl. Repository und einer entsprechenden Template-Engine zur Publikation der Softwarekarten.

Literatur

[Balz98] Balzert, H.: Lehrbuch der Software-Technik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung. 1. Auflage, Heidelberg, Berlin: Spektrum, 1998. ISBN 3-8274-0065-1.

[Clem+02] Clements, P.; Bachmann, F.; Bass, L.: Documenting Software Architectures views and beyond.1. Auflage, Boston: Addison-Wesley, 2002. ISBN 0-201-70372-6.

[Garl+97] Garlan, D.; Monroe, R.; Wile, D.: Acme: An Architecture Description Interchange Language. In CASCON'97, Toronto, Ontario, Canada, IBM Press, 1997.

[Gutw+04] Gutwenger, C.; Jünger, M.; Klein, K.; Kupke, J.; Leipert, S.; Mutzel, P.: GoVisual - A Diagramming Software for UML Class Diagrams. In Graph Drawing Software,

Jünger, M.; Mutzel, P.(Eds). Berlin, Heidelberg: Springer-Verlag, 2004. ISBN 3-540-00881-0.

[HaCh93] Hammer, M.; Champy, J.: Reengineering the Corporation: A Manifesto for Business Revolution. 1. Auflage, New York: HarperCollins Publishers, 1993. ISBN 0-88730-640-3.

[IEEE00] IEEE: IEEE Std 1471-2000 for Recommended Practice for Architectural Description of Software-Intensive Systems. IEEE Computer Society, 2000.

[IFPU01] IFPUG. Function Point Counting Practices Manual. Version 4.1.1: International Function Point Users Group, 2001.

[KaNo91] Kaplan, R.; Norton, D.: The Balanced Scorecard - Measures That Drive Performance. Harvard Business Review, vol. 70, pp. S. 71-79, 1991.

[Kütz03] Kütz, M.: Kennzahlen in der IT : Werkzeuge für Controlling und Management, 1. Auflage, dpunkt.Verlag, 2003. ISBN 3-89864-225-9.

[Luck96] Luckham, D.C.: Rapide: A Language and Toolset for Simulation of Distributed Systems by Partial Orderings of Events. In DIMACS Partial Order Methods Workshop IV, Princeton University, 1996.

[MaWi04a] Matthes, F.; Wittenburg, A.: Softwarekarten zur Visualisierung von Anwendungslandschaften und ihrer Aspekte. Technische Universität München, Lehrstuhl für Informatik 19 (sebis), Technischer Bericht 02/04, 2004. http://wwwmatthes.in.tum.de/de/main.htm?t=document/1xeim9muktl5m.htm (Abruf am 2004-07-01)

[MaWi04b] Matthes, F.; Wittenburg, A.: Softwarekartographie: Visualisierung von Anwendungslandschaften und ihrer Schnittstellen. In: Informatik 2004 - Informatik verbindet, 34. Jahrestagung der GI, Ulm, Deutschland, 2004. ISBN 3-88579-380-6.

[OGC00] Office of Government Commerce (OGC): ITIL - Service Support. Norwich, UK: The Stationery Office, 2000. ISBN 0-11-330015-8.

[OMG03] OMG: Unified Modeling Language Specification, Version 1.5. Object Management Group, 2003.

[Purc+01] Purchase, H.; McGill, M.; Colpoys, L.; Carrington, D.: Graph drawing aesthetics and the comprehension of UML class diagrams: an empirical study. In Australian symposium on Information visualisation - Volume 9, Sydney, Australia, Australian Computer Society, Inc., 2001.

[Robi+95] Robinson, A.; Morrison, J.; Muehrcke, P.; Kimerling, A.; Guptill, S.: Elements of Cartography. 6. Auflage, Wiley, 1995. ISBN 0471555797.

[Sche01] Scheer, A.-W.: ARIS - Modellierungsmethoden, Metamodelle, Anwendungen. 4. Auflage, Berlin, Heidelberg, New York: Springer-Verlag, 2001. ISBN 3-540-41601-3.

[Wies+04] Wiese, R.; Eiglsperger, M.; Kaufmann, M.: yFiles - Visualization and Automatic Layout of Graphs. In Graph Drawing Software, Jünger, M.; Mutzel, P. (Eds.). Berlin, Heidelberg: Springer-Verlag, 2004. ISBN 3-540-00881-0.