H. Wandke, S. Kain & D. Struve (Hrsg.): Mensch & Computer 2009: Grenzenlos frei!?
München: Oldenbourg Verlag, 2009, S. 363-372

Musterbasiertes Usability Engineering am Beispiel eines Landkartenbrowsers

Markus Specker, Markus Düchting

Siemens AG, Siemens IT Solutions and Services, SIS D IEH C-LAB, Paderborn

Zusammenfassung

Musterbasierte Ansätze spielen in der Softwareentwicklung eine immer größere Rolle. Sie ermöglichen, Probleme mit Hilfe von wieder verwendbaren Konzepten zu analysieren und zu lösen. Zur Problemlösung hat sich das Konzept der Entwurfsmuster (*Design Patterns*) mittlerweile bewährt. Auch für die Analyse von Problemen existieren musterbasierte Konzepte (*Problem Frames*), die bisher weniger bekannt sind. In diesem Beitrag wird anhand des konkreten Beispiels der Konzeption eines interaktiven Landkartenbrowsers für eine Web 2.0-Plattform veranschaulicht, wie die musterbasierten Ansätze Problem Frames und MCI-Entwurfsmuster so miteinander kombiniert werden können, dass bereits während der Problemanalyse auf mögliche Lösungen in Form von Entwurfsmustern gestoßen werden kann und somit ein durchgehend musterbasierter Ansatz für das Usability-Engineering ermöglicht wird.

1 Einleitung und Motivation

In Entwurfsmustern werden Best Practices zur Lösung typischer und wiederkehrender Probleme beschrieben. Sie helfen dabei, gesammeltes Erfahrungswissen festzuhalten und bieten insbesondere Anfängern eine gute Möglichkeit, vom Erfahrungsschatz anderer zu profitieren. Im Gegensatz zu Komponenten können Entwurfsmuster immer und immer wieder *variiert* angewendet werden. Zu Style Guides (z. B. Apple 2007) unterscheiden sie sich durch ihre höhere Flexibilität und Unabhängigkeit von einer bestimmten Plattform. Ursprünglich für den Bereich des Städtebaus und der Architektur von (Alexander 1977) entwickelt, übertrugen (Beck & Cunningham 1987) die Idee der Entwurfsmuster auf die Software-Entwicklung. Mit (Gamma et al. 1995) wurde erstmals eine umfassende Sammlung verschiedener softwaretechnischer Entwurfsmuster geschaffen, welche auch heute noch von grundlegender Bedeutung ist. Im Laufe der Zeit erschienen eine Reihe verschiedener Sammlungen für weitere Einsatzbereiche. Im Bereich der Mensch-Computer-Interaktion (MCI) entstanden diverse Sammlungen, z. B. zur Gestaltung von User Interfaces, die sich allmählich weiter spezialisierten (Borchers 2001; Tidwell 2005; van Duyne et al. 2006; Welie 2008; Yahoo 2009).

Allerdings gehen die Entwurfsmustersammlungen in der Regel nur sehr begrenzt darauf ein, wie aus der Fülle der beschriebenen Entwurfsmuster in einer konkreten Problemsituation effizient das passende Entwurfsmuster gefunden werden kann. Zwar geht aus den Beschreibungstexten hervor, wofür das jeweilige Entwurfsmuster eingesetzt werden kann. Wenn man jedoch für ein aktuelles Problem passende Entwurfsmuster finden möchte, kann es sehr mühselig und langwierig sein, jedes Mal alle Beschreibungstexte zu sämtlichen Entwurfsmustern zu überprüfen, bis man geeignete Muster identifiziert hat. Der Erfolg stellt sich hierbei eher zufällig ein. Auch wenn in den bestehenden Sammlungen zum Teil auf andere Muster referenziert wird, werden die Kriterien für eine Mustersprache (*Pattern Language*) nicht immer erfüllt (vgl. Todd et al. 2004), was z. B. (Tidwell 2005, xiv) für ihre Sammlung bestätigt. Noch weniger findet man in den vorhandenen Sammlungen Hinweise, um *übergreifend* Entwurfsmuster aus verschiedenen Sammlungen kombinieren zu können. Stattdessen ähneln sich die Sammlungen inhaltlich zum Teil und wiederholen eher bekannte Konzepte, haben andererseits aber auch jeweils sehr unterschiedliche Granularitätsgrade, was deren Verbindung zusätzlich erschwert. Dabei könnten insbesondere neue Sammlungen hier einen echten Mehrwert bieten. Eine zusätzliche Hürde stellt dabei dar, dass die Muster in den verschiedenen Sammlungen sehr unterschiedlich aufgebaut sind. Mit PLML gibt es zwar Bemühungen (Fincher et al. 2003), die Beschreibungsstruktur auf ein XML-basiertes Format zu vereinheitlichen, dieses lässt den Autoren jedoch einerseits nach wie vor einen relativ großen Spielraum, da die meisten Textfelder freiwillig und frei formatierbar sind, und hat andererseits bei den bestehenden Sammlungen bisher kaum eine Verbreitung gefunden.

Ein anderer Aspekt ist, dass MCI-Entwurfsmuster bisher nicht fest im Softwareentwicklungsprozess verankert sind, obwohl ihre Berücksichtigung die Softwarequalität verbessern könnte. Daher stellt sich die Frage, ob sich das Konzept der MCI-Entwurfsmuster systematisch in den Softwareentwicklungsprozess integrieren lässt. Hierzu wird eine strukturierte Vorgehensweise benötigt, aus der hervorgeht, wie die Entwurfsmuster miteinander kombiniert werden können. Dieser Beitrag stellt anhand des praktischen Beispiels eines interaktiven Landkartenbrowsers einen musterbasierten Modellierungsansatz vor, in dem Problem Frames mit MCI-Entwurfsmustern kombiniert werden, um zu den modellierten Problemen passende Muster zu finden und so musterbasiertes Usability Engineering durchzuführen.

2 Fallbeispiel Landkartenbrowser

Im Rahmen des Projektes THESEUS[1] des Bundesministeriums für Wirtschaft und Technologie wird im Anwendungsfall ALEXANDRIA angestrebt, eine Web 2.0-Wissensplattform zu entwickeln, die die Veröffentlichung, Erstellung und Suche nach Inhalten durch eine intuitive Nutzung ermöglichen soll. In diesem Zusammenhang stehen Forschungsarbeiten zur einfachen und intuitiven Nutzbarkeit einer solchen Plattform im Vordergrund.

[1] Die dargestellten Forschungsarbeiten werden im Rahmen des THESEUS-Forschungsprogramms vom Bundesministerium für Wirtschaft und Technologie (BMWi) unter dem Förderkennzeichen 01MQ07014 gefördert.

Konkret wird an der Umsetzung einer Plattform zur Veröffentlichung von Artikeln, Bildern etc. zu historischen Ereignissen und Personen gearbeitet. In diesem Kontext entsteht unter anderem ein Konzept, das es ermöglichen soll, die Inhalte der Plattform über eine Landkarte darzustellen, um sie interaktiv zu explorieren und für den Nutzer zugänglich zu machen. Die geografische Darstellung der Inhalte der Plattform ALEXANDRIA soll ein exploratives „Durchstöbern" der verfügbaren Inhalte fördern und neue inhaltliche Beziehungen durch eine entsprechende Visualisierung für den Nutzer zugänglich machen. Des Weiteren soll diese Darstellung der Inhalte zu einer neuartigen User Experience führen, im Gegensatz zum Zugriff auf Inhalte durch eine SERP (Serach Engine Result Page) oder bspw. einer inhaltlichen Kategorisierung.

Für den Landkartenbrowser wurden folgende Aufgaben (*Tasks*) identifiziert, welche der Nutzer auf dieser Plattform durchführen kann (verkürzte Skizze):

1. Artikel und Bilder sollen interaktiv über die Landkarte exploriert werden können.
2. Inhalte sollen aus verschiedenen Blickrichtungen zugänglich gemacht werden.
3. Historische Grenzveränderungen sollen nachvollziehbar werden.
4. Die Landkarten sollen bei der Erstellung von Artikeln benutzt werden können.
5. Man soll beliebige Verknüpfungen über die Landkarte browsen können (z. B. Land X, Stadt Y → Welche Beziehungen liegen vor?).
6. Es sollen Darstellungstechniken von komplexen Zusammenhängen erarbeitet werden, z. B. Landkarten nach Größen wie Einkommen proportionalisieren.

Um unsere Vorgehensweise zu veranschaulichen, beschränkt sich dieser Beitrag auf die Realisierung der ersten Aufgabe.

3 Vorstellung der Vorgehensweise

Der Problem-Frames-Ansatz (Jackson 2001) dient dazu, Probleme musterbasiert zu analysieren und zu modellieren. Im Gegensatz zu anderen Problemanalyse-Ansätzen, wie z. B. UML-Use Cases, ist dieser Ansatz stringenter und strukturierter, sodass sich Problemsituationen insgesamt klarer darstellen lassen. Problem Frames ermöglichen es, die Komplexität von Problemen durch Problemzerlegung erheblich zu reduzieren und in Teilproblemen strukturiert darzustellen. Die Teilprobleme können sich überlagern, weswegen Jackson von einer Problemzerlegung durch *Projektion* spricht. Durch die Komplexitätsreduktion ist es relativ einfach möglich, die analysierten Probleme in den Entwurf zu überführen.

Der Problem Frames-Ansatz ist geleitet durch das Requirement-Engineering. Für jedes Requirement überlegt man sich den entsprechenden Problemzusammenhang. Ebenso wie andere Ansätze bieten auch Problem Frames die Möglichkeit, Probleme grafisch zu repräsentieren. (Jackson 2001) beschreibt fünf Basis-Problem-Frames welche durch Kombination miteinander erweitert und variiert werden können. Ein Problem Frame besteht aus wenigen

Basiselementen (siehe Abb. 3), die auf verschiedene Weise miteinander kombinierbar sind. Dieses ermöglicht eine insgesamt sehr flexible Handhabung der Probleme. Zu den Basiselementen im Problem Frame zählen *Domänen*, mit welchen Entitäten in einem Problem (ähnlich wie beim Entity-Relationship-Ansatz) beschrieben werden können (in Abb. 3 als Rechtecke im Diagramm dargestellt). Man unterscheidet zwischen verschiedenen Domänentypen. So wird beispielsweise die zu erstellende Anwendung durch zwei Längsstriche besonders markiert. Domänen sind durch Schnittstellen (*Interfaces*) miteinander verbunden (einfache Verbindungslinien zwischen den Domänen). An diesen werden die Gemeinsamkeiten, die die Domänen miteinander teilen, notiert (*gemeinsame Phänomene*). Jedes Problem Frame behandelt ein Teilproblem. Die Anforderungen innerhalb eines solchen Teilproblems werden in *Requirements* festgehalten (gestrichelte Ellipsen). Die Requirements beziehen sich auf bestimmte Domänen im Problem Frame, nach Jackson sind dieses *Referenzen* (gestrichelte ungerichtete Linien zwischen den Requirements und Domänen) oder *Einschränkungen* (gestrichelte gerichtete Linien zwischen den Requirements und Domänen).

Um (MCI-)Probleme mit Problem Frames zu modellieren, sind folgende Schritte erforderlich:

1. Am Anfang steht ein reales Problem, welches modelliert werden soll. Bei MCI-Problemen sollte zunächst eine Analyse des Nutzungskontextes durchgeführt werden. Danach wird das gesamte Problem in einem Kontextdiagramm (Abb. 2) dargestellt, welches alle Domänen sowie deren Beziehungen zueinander (Schnittstellen) enthält. An den Schnittstellen werden die gemeinsamen Phänomene notiert.

2. Das Kontextdiagramm bildet die Basis für die Zerlegung zu Problemdiagrammen, d. h. es wird analysiert nach darin enthaltenen Problem Frames. Auf diese Weise wird das ursprünglich komplexe Problem in kleine Teilprobleme zerlegt, welche sich einfacher lösen lassen als das Gesamtproblem. Für jedes Teilproblem wird angenommen, dass alle anderen Teilprobleme jeweils bereits gelöst sind, um die Abhängigkeiten innerhalb der Teilprobleme zu minimieren. Dieses Vorgehen ermöglicht eine strukturierte Problemanalyse. Als Ergebnis erhält man für jedes Teilproblem ein Problemdiagramm, welches eine konkrete Instanz eines abstrakten Problem Frames darstellt, instanziert durch die aktuelle Problembeschreibung.

3.1 Prinzipienanalyse in Entwurfsmustern

Jacksons Ansatz hat den Nachteil, dass in ihm nichtfunktionale Aspekte, wie etwa usability-spezifische Zusammenhänge nicht berücksichtigt werden. Aus diesem Grunde ist für die Darstellung von MCI-Problemen mit Problem Frames eine weitere Modifikation erforderlich. Außerdem beschränkt sich Jacksons Ansatz lediglich auf die Problemdarstellung, erläutert jedoch nicht die Lösung der Probleme. Hier kommen die Entwurfsmuster ins Spiel.

Um in einer konkreten Problemsituation passende Entwurfsmuster anbieten zu können, muss zunächst geklärt werden, welche Faktoren zur Auswahl eines Entwurfsmusters führen. Hinweise dazu befinden sich im Beschreibungstext der Muster, aus dem hervorgeht, wann ein Entwurfsmuster nach Ansicht des jeweiligen Autors zum Einsatz kommen sollte. Zum ande-

ren muss sich der User Interface (UI)-Designer des konkreten Nutzungskontextes, in welchem das Problem beschrieben ist, bewusst sein. Der Beschreibungstext eines in Frage kommenden Entwurfsmusters muss dem UI-Designer ermöglichen, zu entscheiden, ob das in Frage kommende Entwurfsmuster zu dem aktuellen Nutzungskontext passt oder nicht.

Eine Abstraktion der Entwurfsmusterbeschreibung hin zu vergleichbaren Faktoren, die beschreiben, in welchen Fällen das jeweilige Entwurfsmuster zum Einsatz kommt, hilft, um zu einer strukturierten Vorgehensweise zu gelangen. Solche Faktoren nennen wir *Entwurfsmusterprinzipien*. Abb. 1 links veranschaulicht unsere Vorgehensweise. Ähnlich wie (Smith & Williams 2002), welche ihre Performance-Muster als Umsetzung von einem oder mehrere Prinzipien beschreiben (S. 263), extrahieren wir Prinzipien aus MCI-Entwurfsmustern. Ein Beispiel hierfür ist das Prinzip „Focus on Context“. Dieses wird angewandt, um zu gewährleisten, dass ein Nutzer einen Zusammenhang überhaupt wahrnehmen kann, weil die Interaktion dann dort stattfindet, wo der Nutzer gerade seinen Fokus hat. Dieses Prinzip ist z. B. in den Entwurfsmustern „Same Page Error Messages“ und „Edit in Place“ (Tidwell 2005, S.239, 249) enthalten. Die extrahierten Prinzipien aus den Entwurfsmustern integrieren wir in den Problem Frames-Ansatz. Durch die Referenz der Prinzipien zu den Entwurfsmustern können dann in einer aktuellen Problemsituation passende Lösungsmuster selektiert werden.

Nach (Beu 2003) setzt sich der Nutzungskontext aus den Komponenten *Nutzer*, *Nutzungsumgebung*, *Hard- und Software* sowie *Aufgabe* zusammen. Die den Entwurfsmustern zugrunde liegenden Prinzipien können diesen Nutzungskontextkomponenten entsprechend zugeordnet werden, z. B. kann das Prinzip „Focus on Context“ dem Benutzer zugeordnet werden, da es sich auf dessen Eigenschaften (beschränkte Wahrnehmungsfähigkeit) bezieht. Für die Verknüpfung der Prinzipien mit Problem Frames kann dieser Zusammenhang ausgenutzt werden, denn eine Differenzierung der Prinzipien nach ihrem Nutzungskontext gibt Aufschluss darüber, an welcher Stelle im Problemdiagramm diese eingebettet werden können (siehe Abb. 1 rechts). So ist die Nutzungskontextkomponente *Aufgabe* den Requirements im Problem Frame zugeordnet, während die *Nutzungsumgebung* und der *Benutzer* den Domänen zugeteilt sind. Die Interaktion zwischen den Nutzungskontextelementen findet in den Problem Frames ihre Entsprechung bei den gemeinsamen Phänomenen.

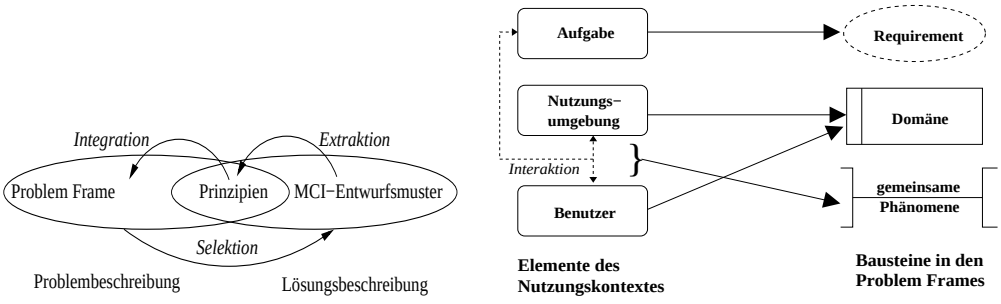

Abbildung 1: Links Verdeutlichung der Vorgehensweise zur Verknüpfung von Problem- und Lösungsmustern, rechts Mapping der Nutzungskontextelemente auf Elemente aus Problem Frames

3.1.1 Nutzungskontextanalyse

Für das Beispiel Landkartenbrowser haben wir zunächst eine Nutzungskontextanalyse durchgeführt (Tab. 1). Diese ist hilfreich, um die verschiedenen Aufgaben und Domänen innerhalb des Problems erfassen zu können und bildet die Grundlage für die Darstellung des Problems mit Hilfe eines Kontextdiagramms. In Tab. 1 wurde zunächst nur die erste Aufgabe aus der Beschreibung des Landkartenbrowsers berücksichtigt. Die Beschreibung der weiteren Aufgaben aus Kapitel 2 erfolgt analog.

Benutzer	*Aufgaben*	*Nutzungsumgebung*
Erfahrungsgrad „mittel“ Kennt andere Webapplikationen, z.B. wikipedia und google knol. Keine besonderen Einschränkungen. Allgemein: Menschliche Aufnahmefähigkeit ist begrenzt → *Prinzip Reduktion der Komplexität*	Artikel und Bilder sollen interaktiv über die Landkarte explorierbar sein. → *Prinzip Informationsvisualisierung*	Soll auf einem normalen PC sowie auf einem Laptop lauffähig sein (Begrenzter Bildschirmplatz). → *Prinzip Platz sparen* Aktuelles grafisches OS (Windows, Mac OS und Linux). Web-Zugang vorhanden (DSL darf vorausgesetzt werden). Nutzergenerierte Artikel und Bilder (enthalten Bezüge zu Jahreszahlen, Personen, Städten, Ereignissen, …). Landkartenmaterial vorhanden.

Tabelle 1: Nutzungskontextanalyse (vgl. Beu 2003)

Aus der dargestellten Aufgabe lassen sich folgende Requirements ableiten:

[R1] Die Daten, welche exploriert werden können sollen, müssen auf der Landkarte sichtbar gemacht werden.

[R2] Der Nutzer braucht Interaktionsmöglichkeiten, um das Lesen von Artikeln und das Betrachten von Bildern zu ermöglichen.

Diese Requirements sind für die Überführung des Kontextdiagrammes in Problemdiagramme von Bedeutung, welche durch die Anwendung von Problem Frames entstehen.

3.1.2 Darstellung des Problems durch ein Kontextdiagramm

Abb. 2 stellt das Problem in einem Kontextdiagramm dar. Entstehen soll ein Landkartenbrowser, welcher mit den Domänen Landkarte, Artikel und Bilder, Display und GUI sowie Nutzer interagiert. An den jeweiligen Interfaces sind die Beziehungen beschrieben, welche in diesem Zusammenhang eine Rolle spielen.

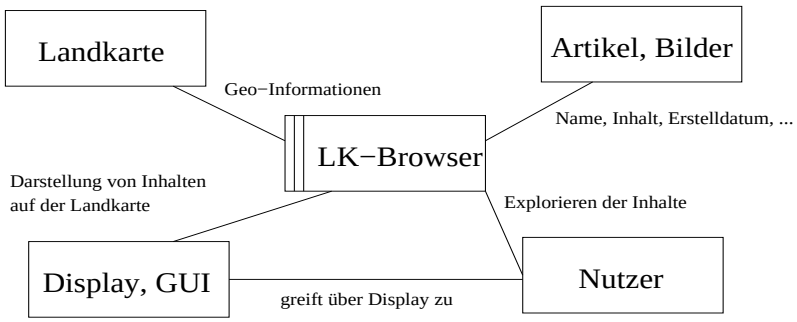

Abbildung 2: Kontextdiagramm zur Problemdarstellung

3.1.3 Problemmodellierung mittels Problem Frames

Nach der Problemmodellierung durch ein Kontextdiagramm erfolgt die Problemzerlegung durch Projektion in Problemdiagramme. Die Problemdiagramme sind dabei Instanzen von Jacksons Problem Frames. Abb. 3 zeigt das Teilproblem „Sichtbarmachen von Informationen auf der Landkarte“ `[R1]` als Instanz des Problem Frames *Information Display* (vgl. Jackson 2001, S.93). Da sowohl Artikel, Bilder und die Landkarte eine Form von Metadaten darstellen, wurden diese Domänen gemäß Jackson (2001) zu einer Metadomäne vereinigt.

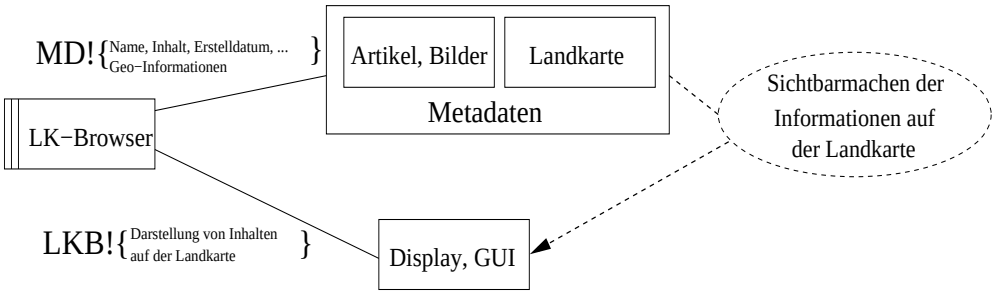

Abbildung 3: Darstellung des Requirements R1 als Problemdiagramm durch Mapping mit dem Problem Frame Information Display von Jackson (2001)

In Abb. 3 sind noch keine kontextrelevanten Informationen berücksichtigt. Dementsprechend können an dieser Stelle auch noch keine Angaben über den Einsatz von MCI-Entwurfsmustern gemacht werden. Um solche Angaben machen zu können, bedienen wir uns so genannter Beschreibungsreferenzen (*description references*), was uns ermöglicht, nutzungskontextbezogene Informationen in den Problem Frames-Ansatz zu integrieren und gleichzeitig in der Notation von (Jackson 2001, S. 209) zu bleiben. Diese werden durch durchgezogene Ellipsen an die die bereits bekannten Elemente gehängt.

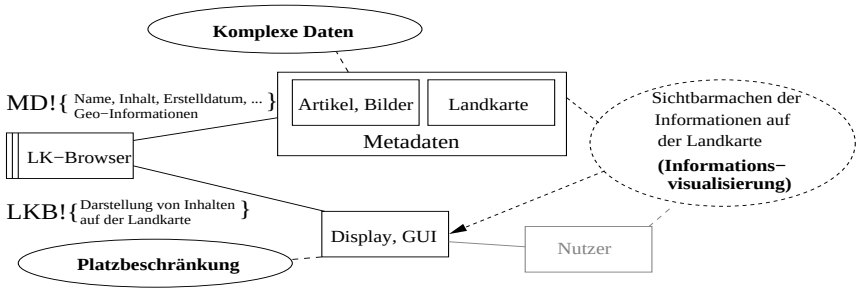

Abbildung 4: Modifiziertes Problemdiagramm zu Requirement R1 unter Berücksichtigung des Nutzungskontexts

Durch die Berücksichtigung des Nutzungskontextes (Tab 1.) in den Problem Frames können die Prinzipien, welche aus den Entwurfsmustern extrahiert wurden, unmittelbar in die Problemmodellierung einfließen. Im Beispiel von Abb. 4 sind die resultierenden Prinzipien: *Reduzierung der Komplexität*, *Platz sparen* und *Informationsvisualisierung*. Entsprechend sind solche Entwurfsmuster zur Lösung des Problems gefragt, die diese Prinzipien beinhalten. In Tab. 2 haben wir zu diesen Prinzipien in Frage kommende Entwurfsmuster aufgelistet. Sie stammen aus einer Prinzipienanalyse, welche wir zuvor für den Entwurfsmusterkatalog von Tidwell (2005) durchgeführt haben. Einige Entwurfsmuster enthalten gleichzeitig mehrere der gesuchten Prinzipien. Diese sind daher für die Problemlösung besonders relevant. Die Entwurfsmuster „Overview + Detail“ und „Data Tips“ enthalten alle drei Prinzipien, bei den „Dynamic Queries“ und „Cascading Lists“ gibt es je zwei Übereinstimmungen.

Prinzip Info.visualisierung	**Prinzip Platz sparen**	**Prinzip Komplexitätsred.**
Overview + Detail	One Window Drilldown	Wizard
Data Tips	Extras on Demand	***Overview + Detail***
Dynamic Queries	Clear Entry Points	Extras on Demand
Data Brushing	Hub and Spoke	Clear Entry Points
Local Zooming	Card Stack	Global Navigation
View Stripping	Closable Panels	Hub and Spoke
Sortable Table	Movable Panels	Pyramid
Cascading Lists	Liquid Layout	Modal Panel
Multi-Graph	***Overview + Detail***	Command History
Small Multiples	**Data Tips**	**Data Tips**
Treemap	New Item Row	*Dynamic Queries*
Alternative Views	*Cascading Lists*	Responsive Disclosure
Tree Table		Responsive Enabling
Deep Background		Button Groups
Few Hues, Many Values		Action Panel
Corner Treatments		Smart Menu Items

Tabelle 2: In Frage kommende Entwurfsmuster zur Lösung des Problems. Hervorgehoben: Muster, die mehrere der bedingt durch den Nutzungskontext geforderten Prinzipien enthalten

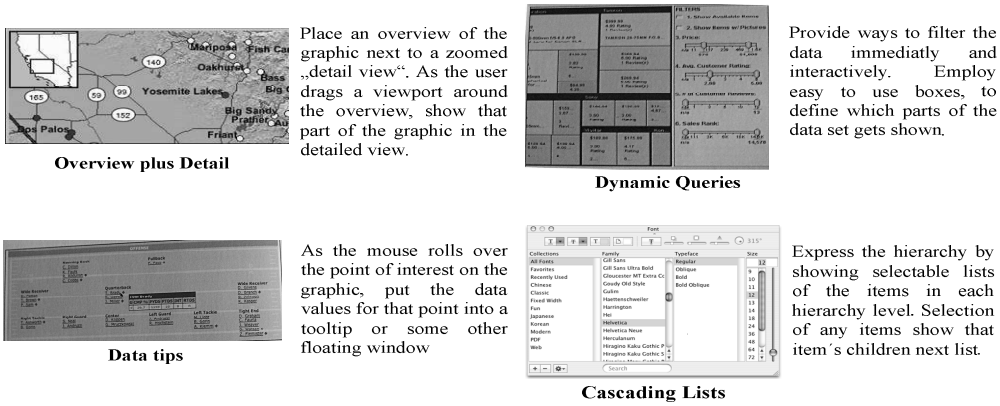

Abbildung 5: In Frage in kommende Entwurfsmuster (aus Tidwell 2005)

Die gefundenen Entwurfsmuster (Abb. 5) dienen als Ausgangsbasis für die Entwicklung des Landkartenbrowsers. Entsprechend dieser Vorgehensweise lassen sich auch die anderen Aufgaben mit Problem Frames analysieren, um zu passenden Entwurfsmustern zu gelangen. Somit erzielt man eine musterbasierte Problemspezifizierung mittels Problem Frames und erhält gleichzeitig bereits in dieser frühen Phase mögliche Lösungsideen in Form von Entwurfsmustern unter Ausnutzung der darin enthaltenen Prinzipien. Durch die Konzentration auf mehrfach auftretende Prinzipien kann der Design Space passend eingeschränkt werden.

4 Zusammenfassung und Ausblick

Die Vorgehensweise in diesem Beitrag hat gezeigt, wie Problem Frames zur Problemanalyse von MCI-Problemen genutzt werden können. Es wurde dargestellt, wie die beiden Ansätze Problem Frames und MCI-Entwurfsmuster miteinander kombiniert werden können, um für die mittels Problem Frames analysierten Probleme auf Basis von Prinzipien passende Lösungsideen in Form von Entwurfsmustern zu finden. Durch den Einsatz von Entwurfsmustern bereits in der Problemanalyse können Designfragen frühzeitig und zielgerichtet erkannt und geklärt werden. Gleichzeitig hilft der Ansatz beim Aufbau einer übergreifenden Mustersprache, da über Prinzipien die Verbindung der Entwurfsmuster ausgenutzt werden. Außerdem leistet er einen Beitrag zur systematischen Verbesserung der Gebrauchstauglichkeit von Software, denn durch die Verwendung der MCI-Entwurfsmuster werden einerseits Best Practices angewandt, andererseits können durch die Prinzipienanalyse konkrete Beziehungen zu den entsprechenden Usability-Normen nachgewiesen werden (Specker & Wentzlaff 2007). Der Ansatz ist zu vielen Seiten flexibel. So ist es sowohl denkbar, dass der Ansatz mit Entwurfsmustern aus anderen Domänen ähnlich durchgeführt werden kann, als auch, dass andere Problemanalysetechniken als Problem Frames zum Einsatz kommen, wobei Problem Frames von Vornherein den Vorteil einer strukturierten Problemanalyse bieten. Ebenso können jederzeit neue Entwurfsmuster integriert werden, es ist lediglich notwendig, für diese eine entsprechende Prinzipienanalyse durchzuführen. Derzeit wird der Ansatz zur Komplet-

tierung des Landkartenbrowsers verwendet. Des Weiteren wird geprüft, ob zur weiteren Einschränkung des Design Spaces künftig formale Methoden wie eine Clusteranalyse eingesetzt werden können, um auch bei einer umfangreichen Musterbasis präzise Lösungen zu liefern.

Literaturverzeichnis

Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., Angel, S. (1977). *A Pattern Language*. New York: Oxford University Press.

Apple Computer Inc. (2007). *Apple Human Interface Guidelines*. http://developer.apple.com/ documentation/UserExperience/Conceptual/AppleHIGuidelines, Zugriff am 26.05.2009.

Borchers, J. (2001). *A Pattern Approach to Interaction Design.* Chichester: John Wiley & Sons.

Beck, K. & Cunningham, W. (1987). *Using Pattern Languages for Object-Oriented Programs*. In: OOPSLA-87 Workshop on the Specification and Design for Object-Oriented Programming.

Beu, A. (2003). Analyse des Nutzungskontextes. In Machate, J., Burmester, M. (Hrsg.): *User Interface Tuning - Benutzungsschnittstellen menschlich gestalten.* Frankf.: Software&Support Verl. S. 68-83.

Duyne, van, D. K. ,Landay, J. A., Hong, J. I. (2006). *The Design of Sites*. Boston: Addison-Wesley.

Fincher, S., Finlay, J., Greene, S., Jones, L., Matchen, P., Thomas, J., Molina, P. J. (2003). Perspectives on HCI patterns: concepts and tools. In *CHI '03 extended abstracts on Human factors in computing systems.* New York: ACM Press.

Gamma, E., Helm, R., Johnson, R.E., Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston: Addison-Wesley Professional.

Jackson, M. (2001). *Problem Frames. Analyzing and structuring software development problems*. Boston: Addison-Wesley.

Smith, C. U. & Williams, L. G. (2002). *Performance Solutions - A Practical Guide to Creating Responsive, Scalable Software*. New York: Addison-Wesley Professional.

Specker, M. & Wentzlaff, I. (2007). Exploring Usability Needs by Human Computer Interaction Patterns. In Winkler, M., Johnson, H., Palanque, P. (Eds.): *Tasks, Models and Diagrams for User Interface Design: 6th International Workshop*. Heidelberg, New York: Springer Verlag. S. 254-260.

Tidwell, J. (2005). *Designing Interfaces: Patterns for Effect. Interaction Design*. Sebastopol: O'Reilly.

Todd, E., Kemp, E., Phillips, C. (2004). What makes a good user interface pattern language? In CRPIT '04: *Proceedings of the fifth conference on Australasian user interface*. Austr. Comp. Sc. S. 91-100.

Welie, van M. (2008). *Patterns in Interaction Design*. http://www.welie.com, Zugriff am 26.05.2009.

Yahoo (2009). *Yahoo Design Pattern Library.* http://developer.yahoo.com/ypatterns, Zugriff am 26.05.2009.

Kontaktinformationen

{Markus.Specker | Markus.Duechting}@c-lab.de, Siemens AG, Siemens IT Solutions and Services (SIS), C-LAB, Fürstenallee 11, 33102 Paderborn, Tel.: +49 5251/60-6058