

Patterns im Softwareentwicklungsprozeß

ADV Arbeitsgemeinschaft für Datenverarbeitung

Dr. Michael Hahsler <hahsler@ai.wu-wien.ac.at>

Abteilung für Informationswirtschaft, Wirtschaftsuniversität Wien

20. September 2001

Aufbau des Vortrags

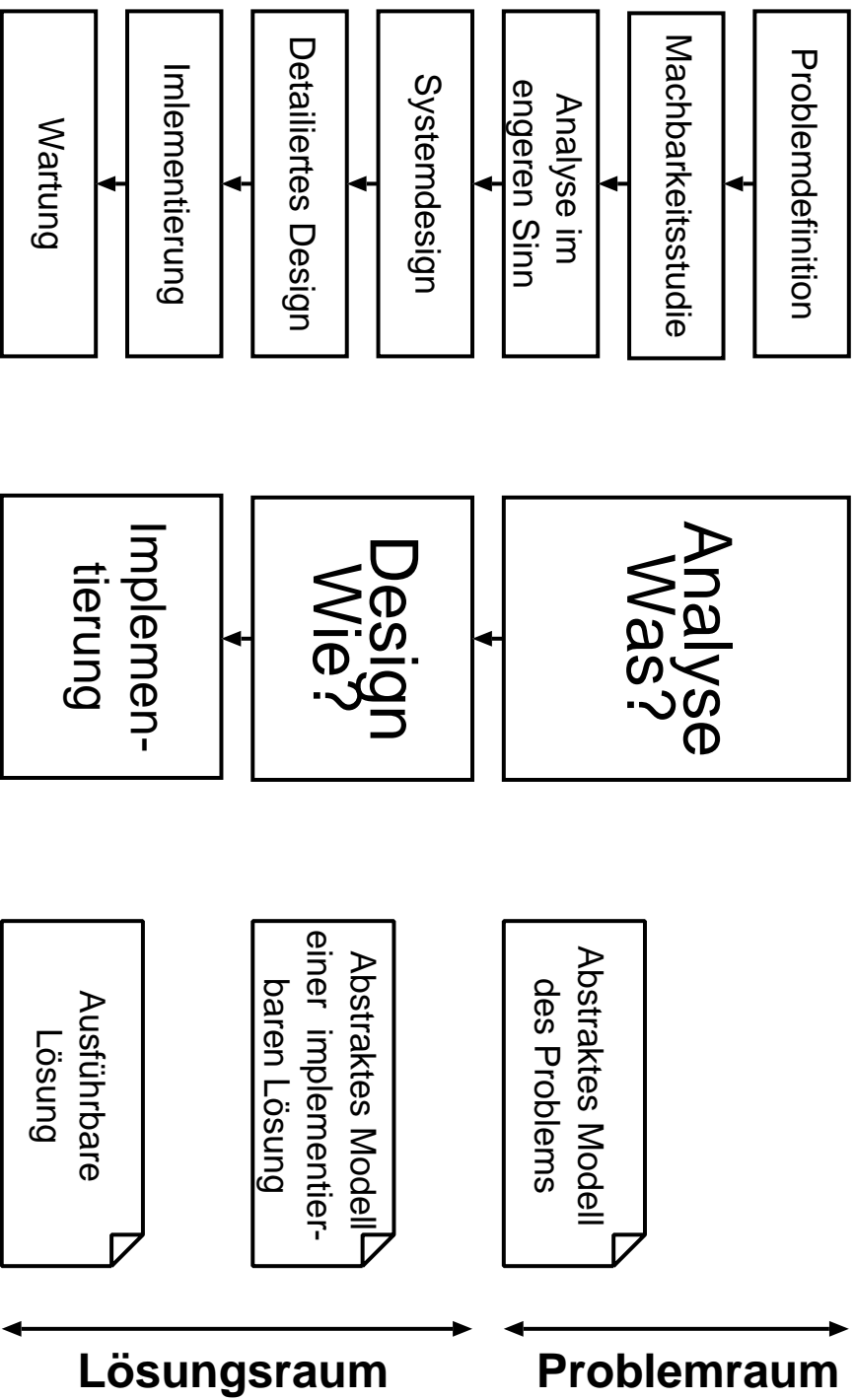
1. Der Software Lebenszyklus
 - (a) Analysephase
 - (b) Designphase
2. Der Pattern Ansatz
 - (a) Design Patterns
 - (b) Analyse Patterns
3. Beispiele

Der Software Lebenszyklus

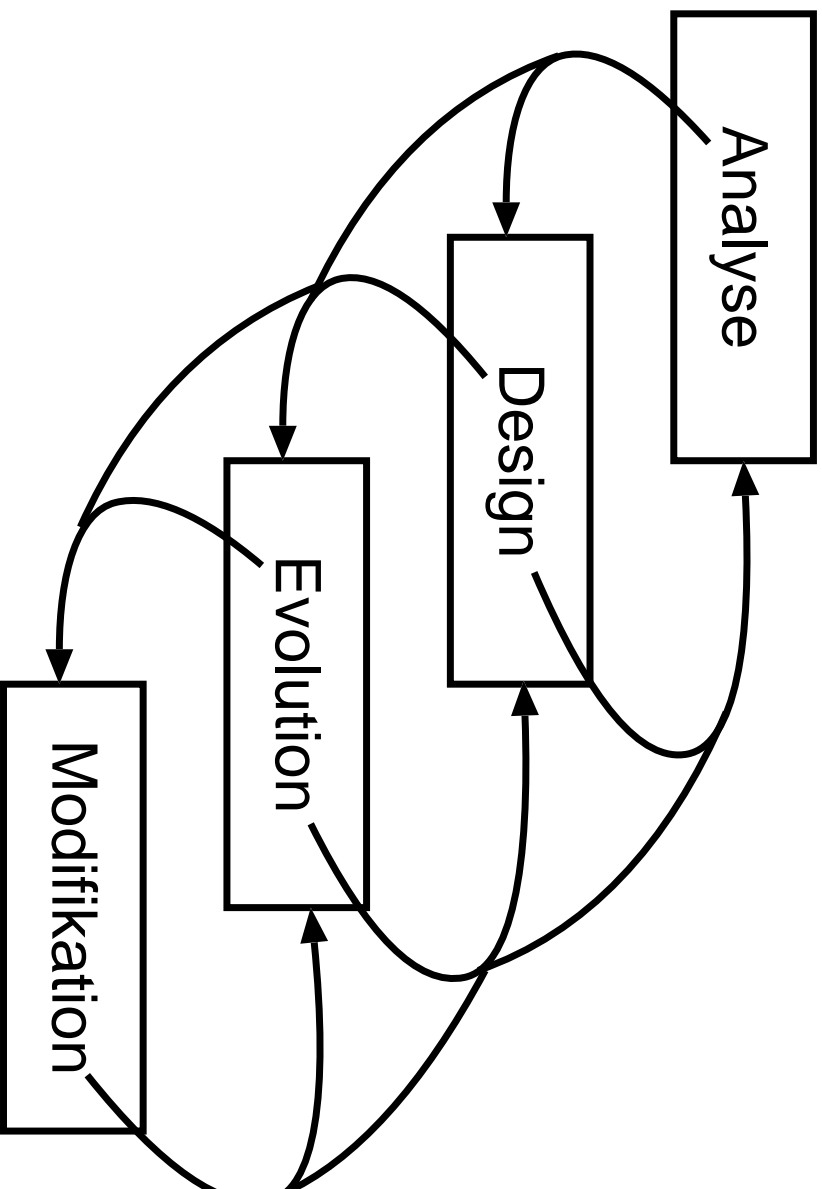
Der gängige Ansatz, um ein großes Projekt zu planen und zu steuern ist die Zerlegung des Gesamtprojekts in mehrere Teilschritte, die nacheinander durchlaufen werden.

Das Wasserfallmodell

Phasen Ergebnis der Phasen



OO-Lebenszyklus nach Booch



OOD - Object Oriented Design, 1994

Analysisphase

Tom DeMarco (Structured Analysis, 1979):

Analysis is the **study of a problem, prior to taking some action**. In the specific domain of computer systems development, analysis refers to the study of some business area or application, **usually leading to the specification of a new system**.

James Rumbaugh et al. (OMT, 1991):

The purpose of object-oriented analysis is to **model the real-world system so that it can be understood**. The successful analysis model states what must be done, without restricting how it is done, and avoids implementation decisions.

Ziele und Probleme

1. Optimales Ziel wählen.
2. Detaillierte Spezifikation des Ziels.
3. Genaue Schätzung der wichtigen Parameter.
4. Erreichung einer Übereinstimmung der ersten drei Punkte bei allen Beteiligten.

Probleme:

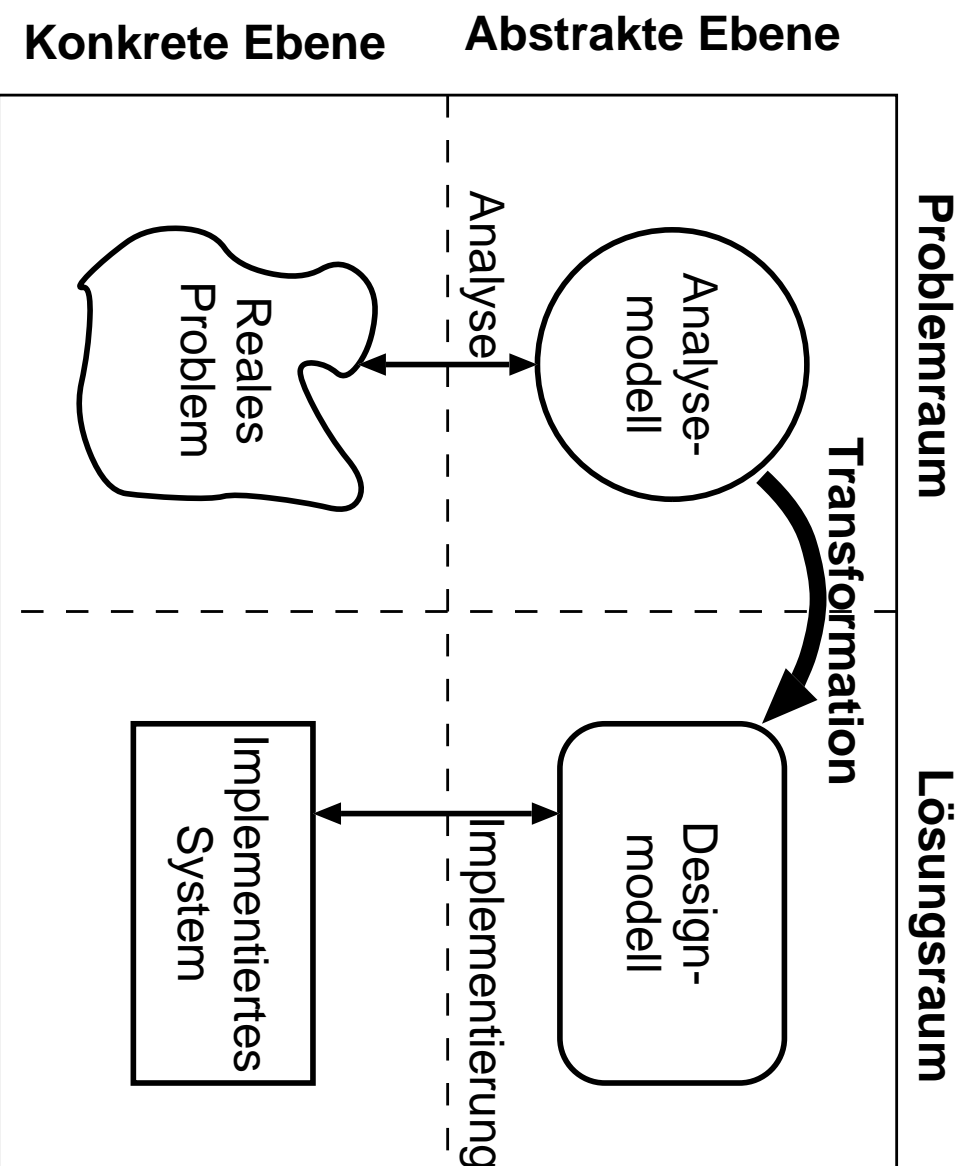
- Kommunikation
- Sich ständig ändernde Anforderungen an das System
- Dokumentation des Ziels
- Politik

Designphase

- System Design
- Detailed Design

Transformation des abstrakten Modells aus dem
Problemmraum (Analysephase) in ein abstraktes
Modell im Lösungsraum (Designphase)

Transformation Analyse-Design



Kosten von Softwareentwicklung

Kostenaufteilung (nach COCCOMO, Boehm, 1981):

Phase	Product Design	Detailed Design	Code	Integration
Aufwand	16%	25%	40%	19%

- Betreuung nach der Auslieferung? Refactoring?

Reduktion von Kosten:

- Fertigkomponenten (Clue-Code ist sehr teuer!)
- Wiederverwendung

Der Pattern Ansatz

- Ein Pattern ist die schriftlich festgehaltenes Expertenwissen für einen bestimmten Problemkreis.
- Das Pattern beschreibt ein Problem, das immer wieder in ähnlicher Form auftritt, und seine Lösung.

Geschichte der Design Patterns

Each pattern is a tree-part rule, which expresses a relation between a certain context, a problem, and a solution.

The Timeless Way of Building, Alexander 1979

*Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that **you can use this solution a million times over**, without ever doing the same way twice.*

A Pattern Language – Towns, Buildings, Construction, Alexander 1977

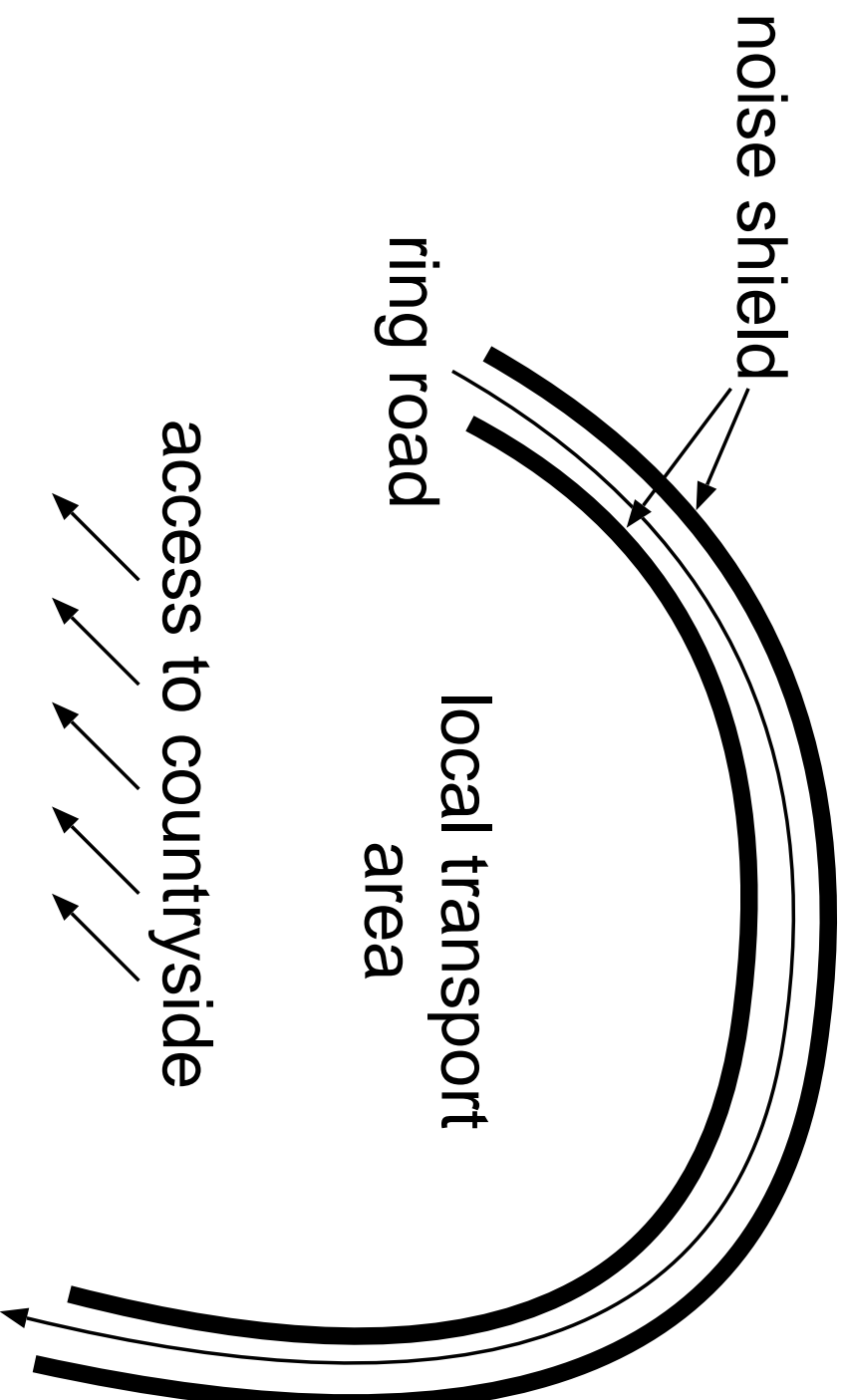
*A pattern lives when it allows its own **inner forces** to resolve themselves.*

The Timeless Way of Building, Alexander 1979

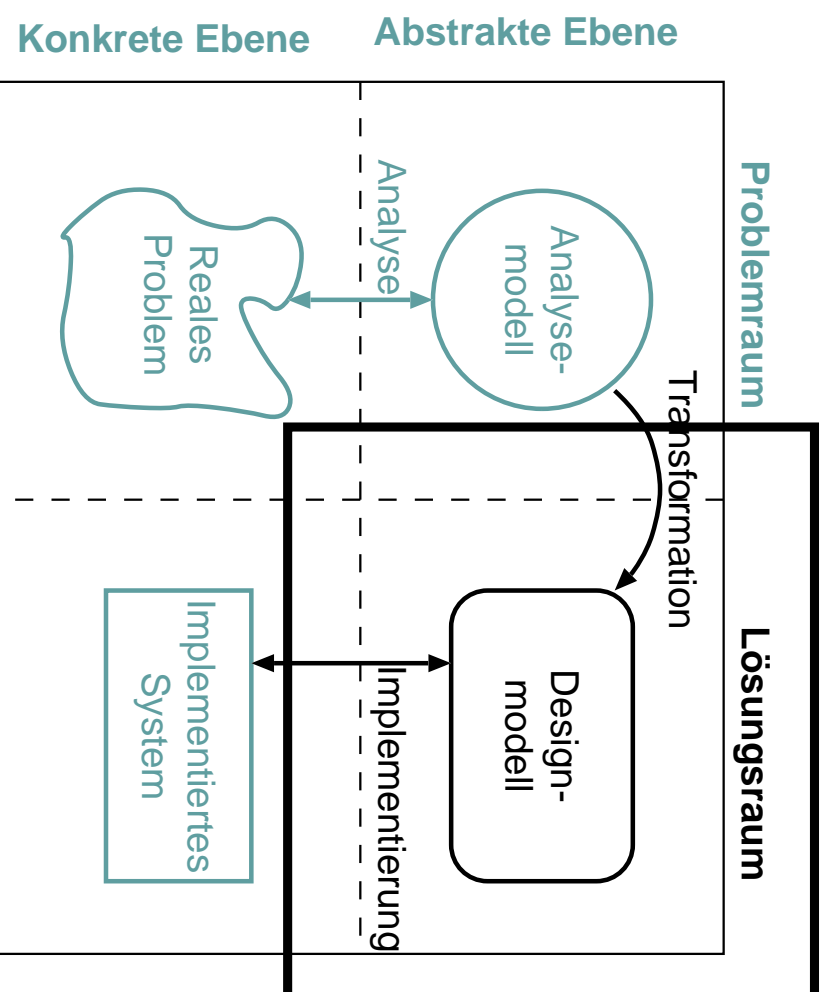
Beispiel: Ring Roads

It is not possible to avoid the need for **high speed roads** in modern society; but it is essential to place them and build them in such a way that they **do not destroy communities or countryside.**

1. At least one high speed road lies tangent to each local transport area.
2. Each local transport area has at least one side not bound by a high speed road, but directly open to the countryside.
3. The road is always sunken, or shielded along its length by berms, or earth, or industrial buildings, to protect the nearby neighborhoods from noise.



Design Patterns im SW-Lebenszyklus



Design Patterns

Gamma et al.: *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, NY, NY 1995.

Struktur

1. Name des Patterns
2. Problem - Wann kann das Pattern angewandt werden?
3. Lösung - Elemente der Lösung und ihre Beziehung zueinander
4. Konsequenzen - Ergebnisse und Abwägungen

Nutzen von Design Patterns

- Geordnete Beschreibung von effizienten Lösungen (Wiederverwendung)
- Gemeinsames Design Vokabular
- Dokumentations- und Lernhilfe
- Erweiterung zu existierenden Methoden
- Ziel für Refactoring

Problembereiche und Mythen

Problembereiche:

- Überbewertung des Ansatzes
- Einige Design Patterns sind unnötig schwer zu erlernen
- Noch unbrauchbare Kategorisierungen von Design Patterns

Mythen:

- Patterns benötigen Tools und Methoden, um effektiv zu sein
- Patterns garantieren wiederverwendbare Software, hohe Produktivität etc.
- Patterns sind nur für Design und Implementierung von Software verwendbar

Varianten

- **Idiome**
- **AntiPatterns (Brown et al., 1998)**
- **Application Frameworks, Cookbooks**
- **Analyse Patterns (Fowler, 1997)**

Analyse Patterns

Martin Fowler: *Analysis Patterns: Reusable Object Models*, Addison-Wesley, Reading, MA 1997:

Analysis patterns are groups of concepts that **represent a common construction in business modeling**. It may be relevant to only one domain, or it may span many domains.

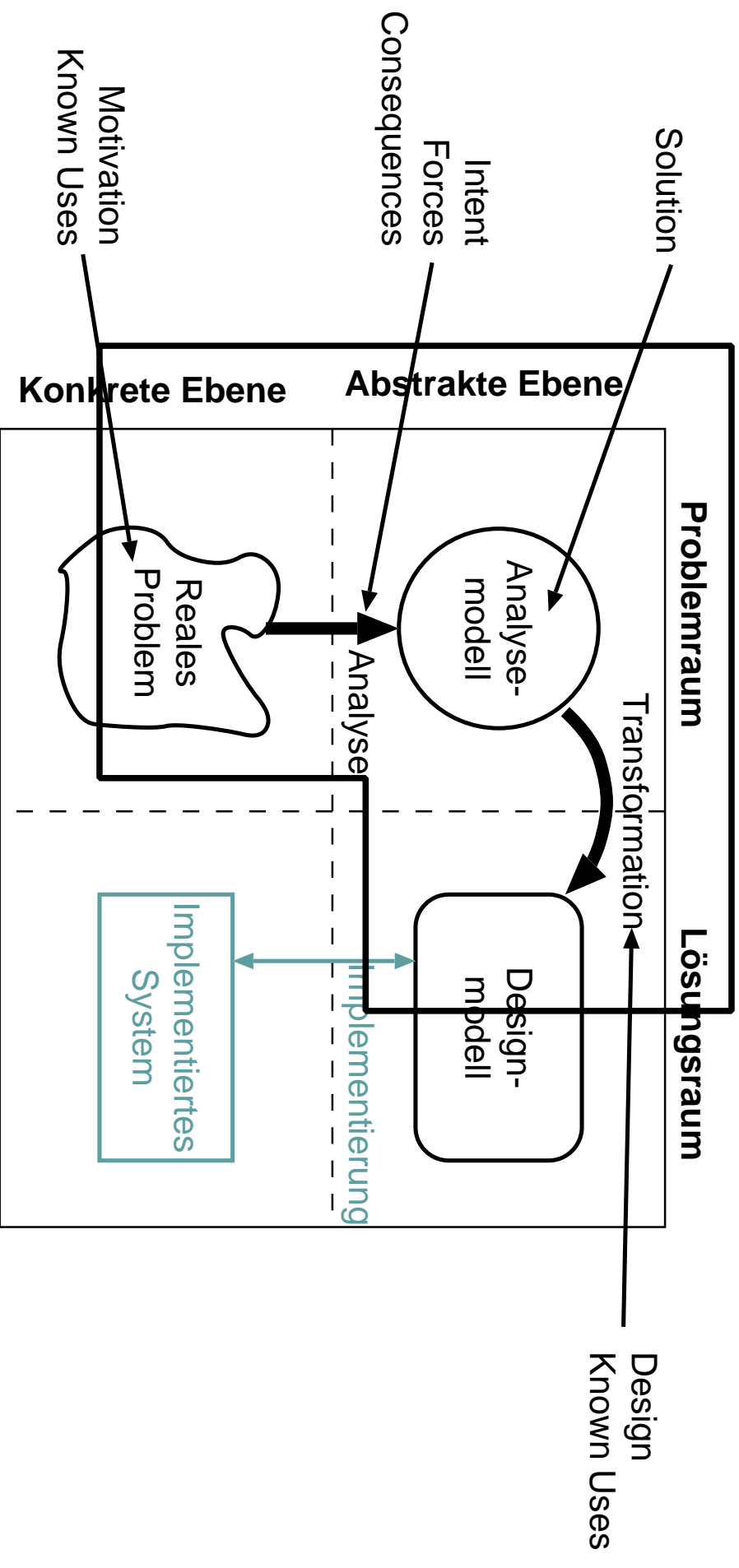
Ein Formular für Analyse Patterns

Fowler beschreibt Analyse Patterns ohne vorgegebene Struktur.

Vorschlag einer Struktur (Geyer-Schulz und Hahsler, 2001):

1. Name des Patterns - Pattern Name [Gamma, Buschmann]
2. Auch bekannt als - Also Known as [Gamma, Buschmann]
3. Zweck - Intent [Gamma]
4. Motivation [Gamma]
5. Kräfte - Forces [Alexander]
6. Lösung - Solution [Buschmann]
7. Konsequenzen - Consequences [Gamma, Buschmann]
8. Design [New]
9. Known Uses [Gamma, Buschmann]

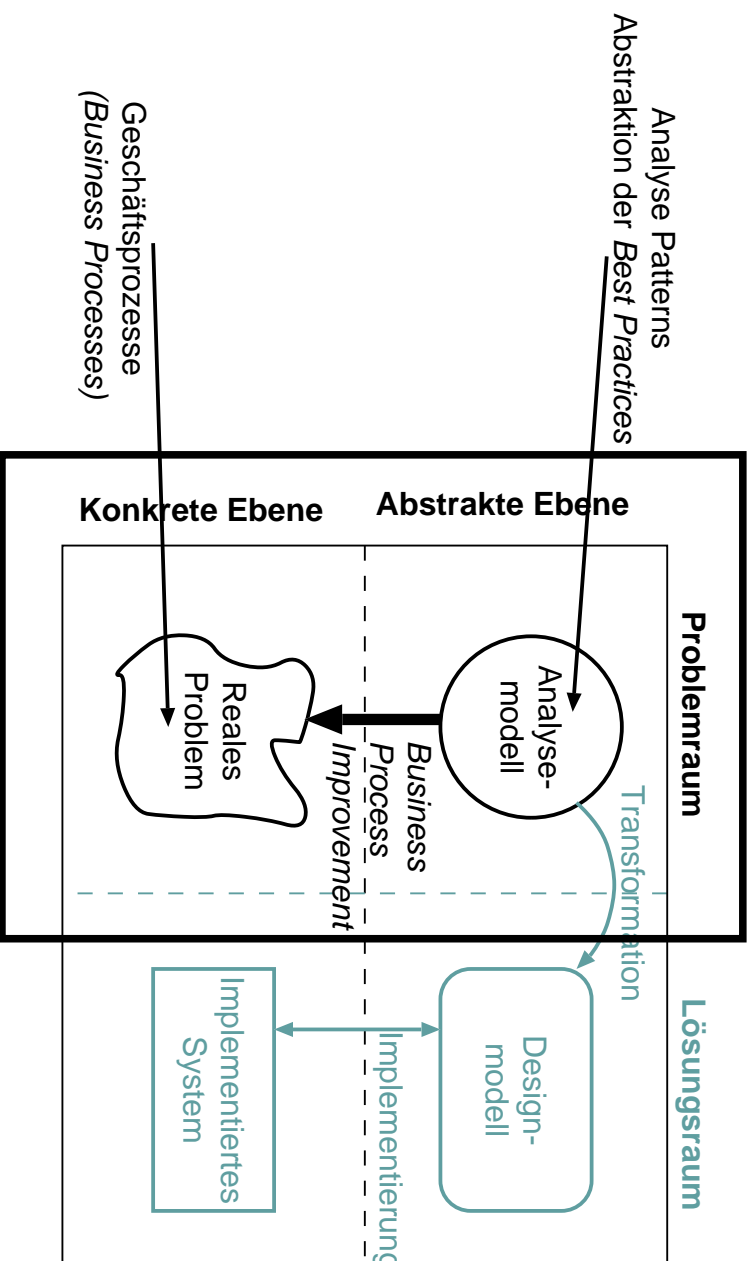
Wirkung von Analyse Patterns



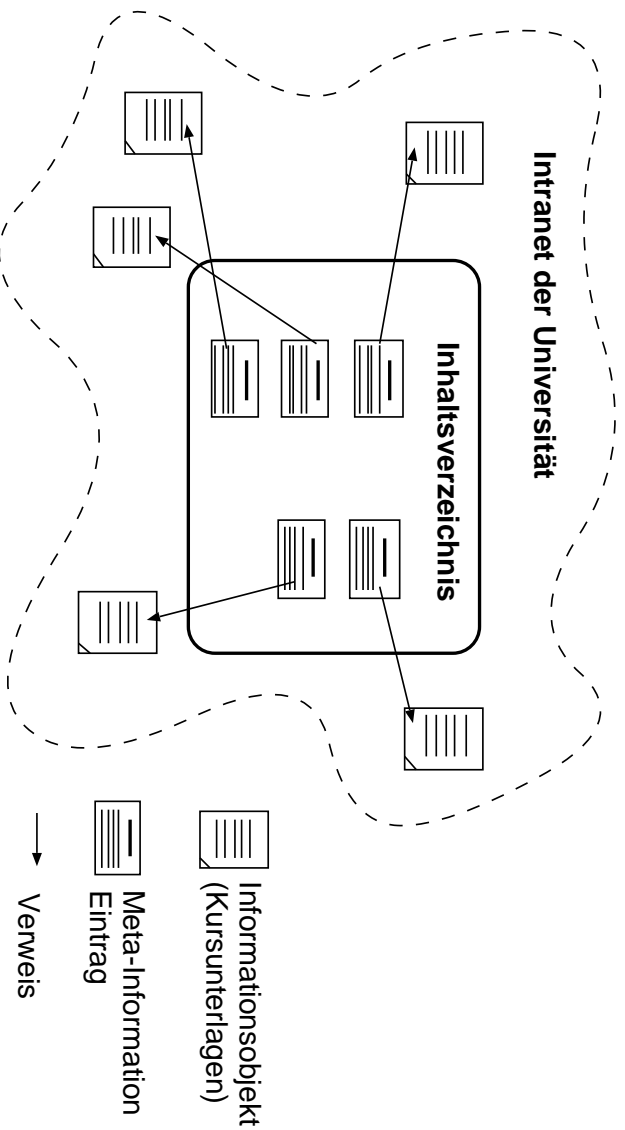
Nutzen von Analyse Patterns

- Geordnete Beschreibung von effizienten Lösungen (Wiederverwendung)
- Unterstützung bei der Transformation von Analyse zu Design.
- Ein gemeinsames Vokabular für die Analysephase.
- Eine Dokumentations- und Lernhilfe.
- Eine Erweiterung zu existierenden Methoden.
- Ein Ziel für die Verbesserung von Geschäftsprozessen.

Business Process Improvement / Process Innovation



Beispiel: IIS für Forschung und Lehre einer Universität



Sammlung von Lehrveranstaltungsunterlagen, Forschungsinformationen, Hilfetexten... mit Verwaltung, Recommendersystem usw.

Analyse Pattern: Die Virtuelle Bibliothek

- **Pattern Name: Virtuelle Bibliothek**
- **Zweck:** Einheitlicher und effizienter Zugang zu verteilten Informationsquellen und Diensten mit zentraler Suchmöglichkeit.
- **Kräfte:** Zentraler Zugangspunkt, Information ändert sich oft, Besitz von Information problematisch, verschiedene Formate,...
- **Lösung:** Verteiltes System...
- **Konsequenzen:** Organisatorisch, sozial, technisch.
- **Design:** Model-View-Controller Pattern (Buschmann), Command Processor Pattern (Sommerlad), Facade Pattern (Gamma), Dublin Core Standard und Resource Description Framework...
- **Bekannte Verwendungen:** WWW Virtual Library, Yahoo!, Open Directory Project (Google)

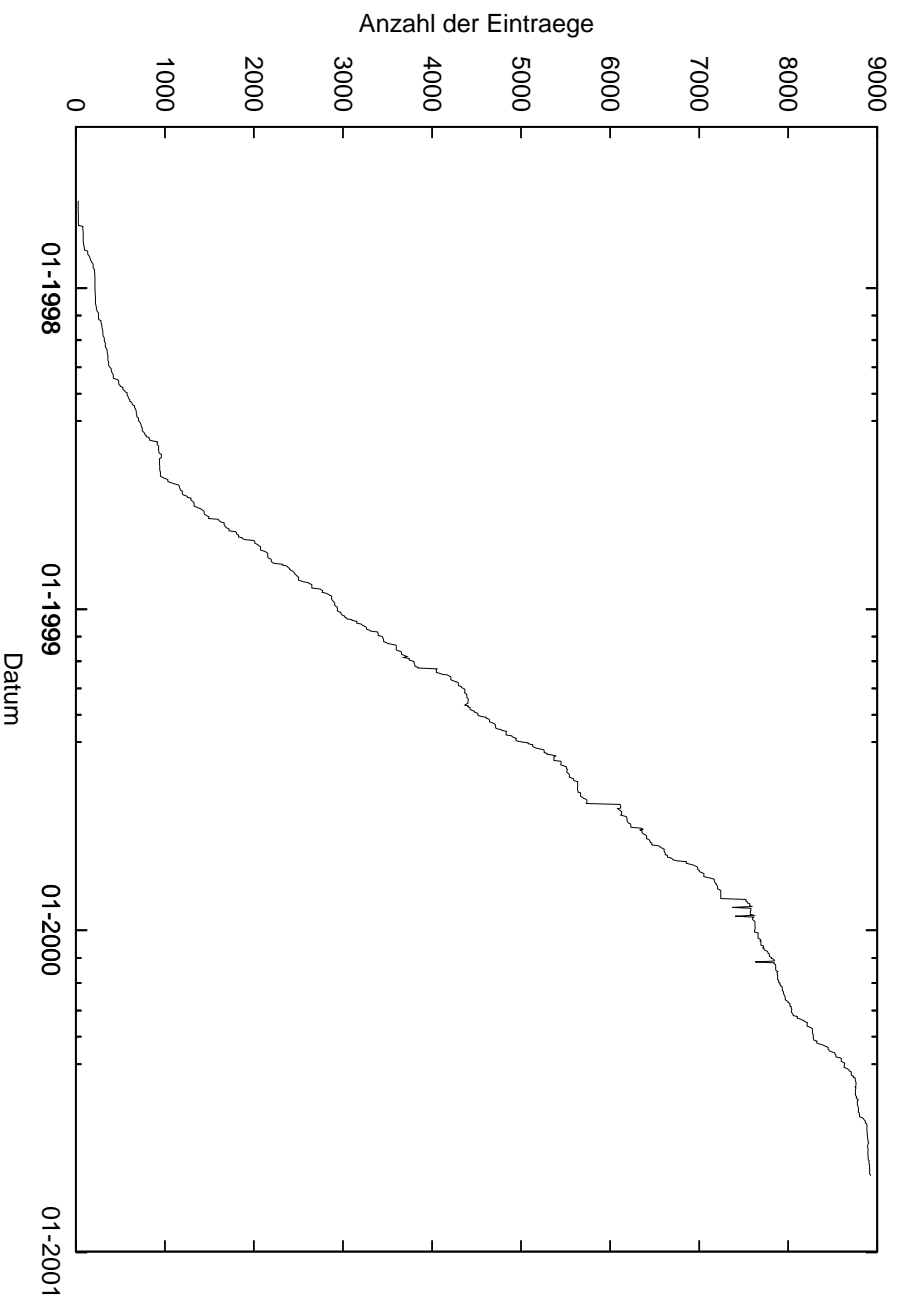
Untersuchung des Nutzens

Welchen Nutzen bringt der Einsatz des Analyse Patterns?

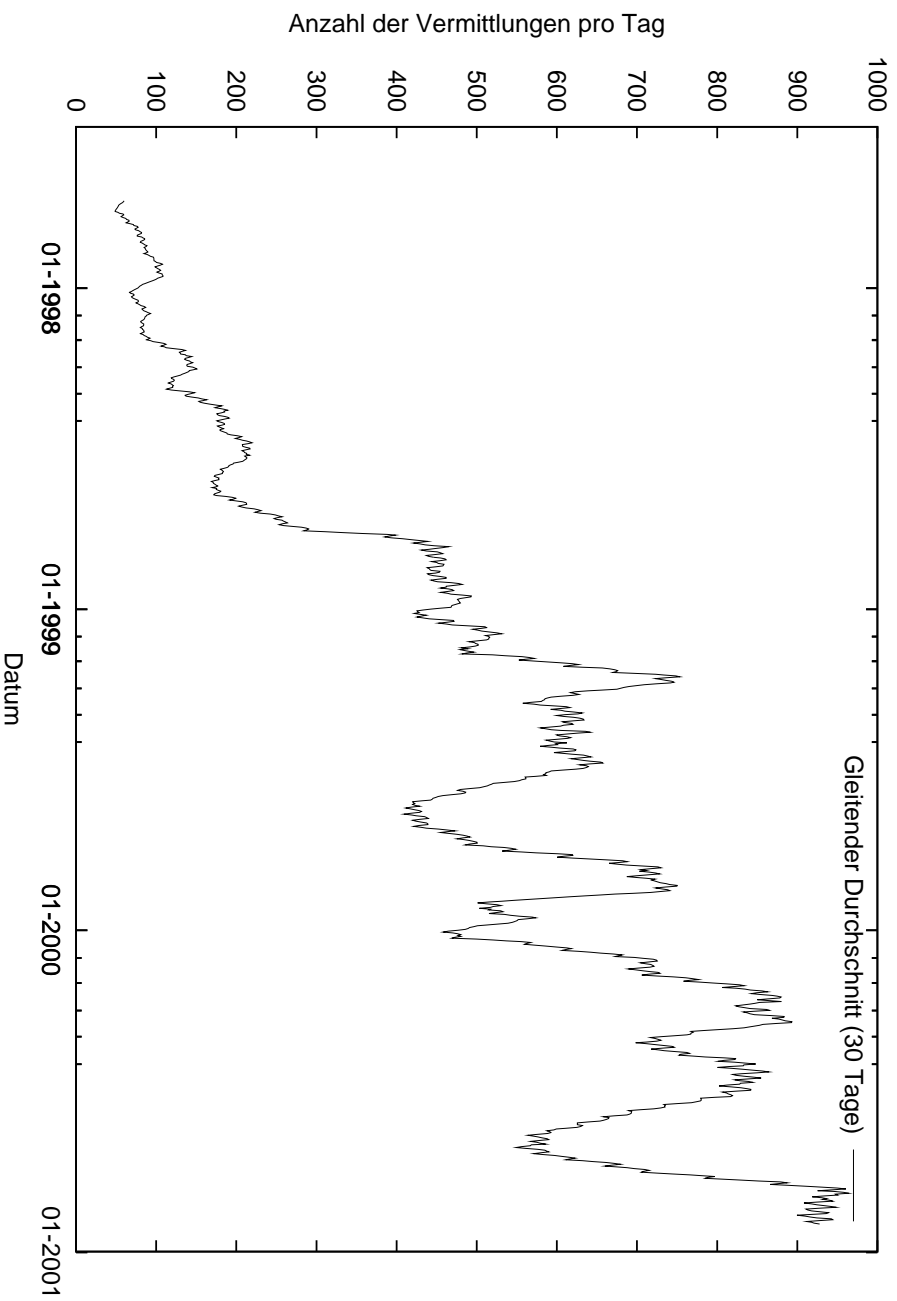
Wie kann dieser Nutzen gemessen werden?

1. Sozialer- , Organisatorischer Nutzen: Das IIS wird von den Benutzern angenommen.
2. Kosteneinsparungen in der Entwicklung durch Wiederverwendung von Analyse, Design und Code.

Wachstum der Einträge



Wachstum der Verwendung



Wiederverwendung durch das Analyse Pattern I

Eckdaten des untersuchten Projektes:

Projekt	Größe in SLOC	Teamgröße	Aufwand in MM
Neuer Katalog	4021	1	0,5

Wiederverwendung:

	Einheit	Total	Black-Box Reuse	Code Reuse
PERL	SLOC	3743	3502	93,56%
HTML	SLOC	278	192	69,06%
Summe	SLOC	4021	3694	91,67%
Summe	MM	10,35	9,46	91,40%

Wiederverwendung durch das Analyse Pattern II

Eckdaten des untersuchten Projektes:

Projekt	Größe in SLOC	Teamgröße	Aufwand in MM
Digitale Bibliothek	7616	2	6

Wiederverwendung:

	Einheit	Total	Black-Box Reuse	Code Reuse
PERL	SLOC	6954	3687	53,02%
HTML	SLOC	662	224	33,84%
Summe	SLOC	7616	3911	51,35%
Summe	MM	20,23	10,05	49,68%

Zusammenfassung

Kostenaufteilung (nach COCOMO, Boehm, 1981):

Phase	Product Design	Detailed Design	Code	Integration
Aufwand	16%	25%	40%	19%

- Patterns reduzieren Kosten in allen Phasen durch
 - Standardisierung (Kommunikation, Wartung)
 - Wiederverwendbarkeit
 - Schnellere Entwicklung
 - Reduktion von Fehlern
 - Flexibilität der Lösung
- Die Herstellung von Patterns kann nur durch Experten geschehen
- Viele Patterns sind relativ komplex (Lernaufwand, Vorkenntnisse)
- Erlernen von Patterns ist eine langfristige Investition und wird in der Zukunft Teil jeder Informatikausbildung

Weitere Informationen

Die Vortragsunterlage sowie weitere Informationen finden Sie unter *Software Engineering: Analysis- and Design Patterns* auf der Internetseite:

<http://www.wai.wu-wien.ac.at/~hahsler/research/>