Fachhochschule Wiener Neustadt
Wirtschaft • Technik • Gesundheit • Sicherheit • Sport

Softwarearchitektur und Design
Block 4 – Muster(Patterns)

SS2014

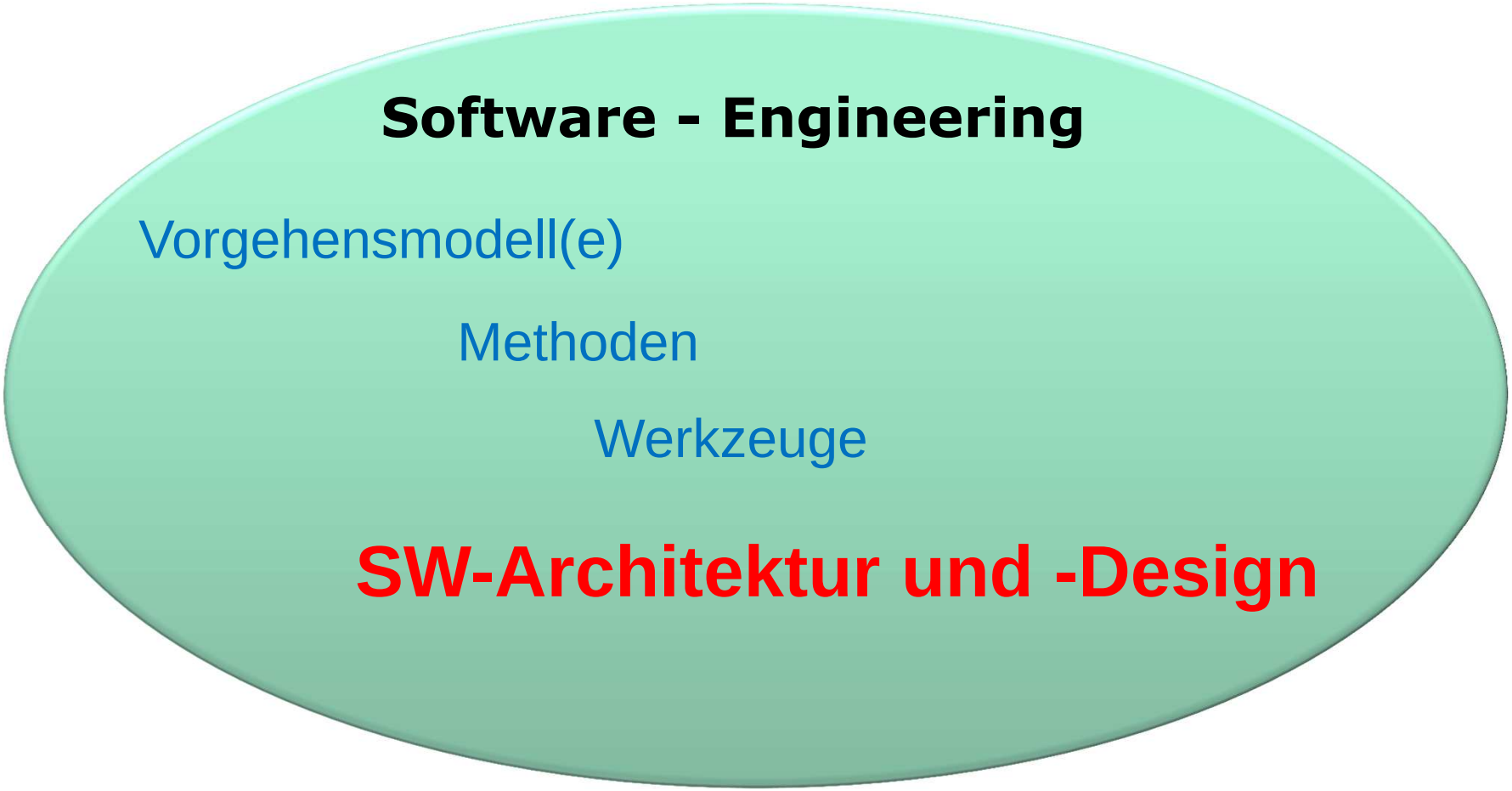
DI Dr. Gottfried Bauer

LV-Typ: VO, UE
Semester: 2
LV-Nummer: **S 2012 ILV**
LV-Bezeichnung: Softwarearchitektur und Design

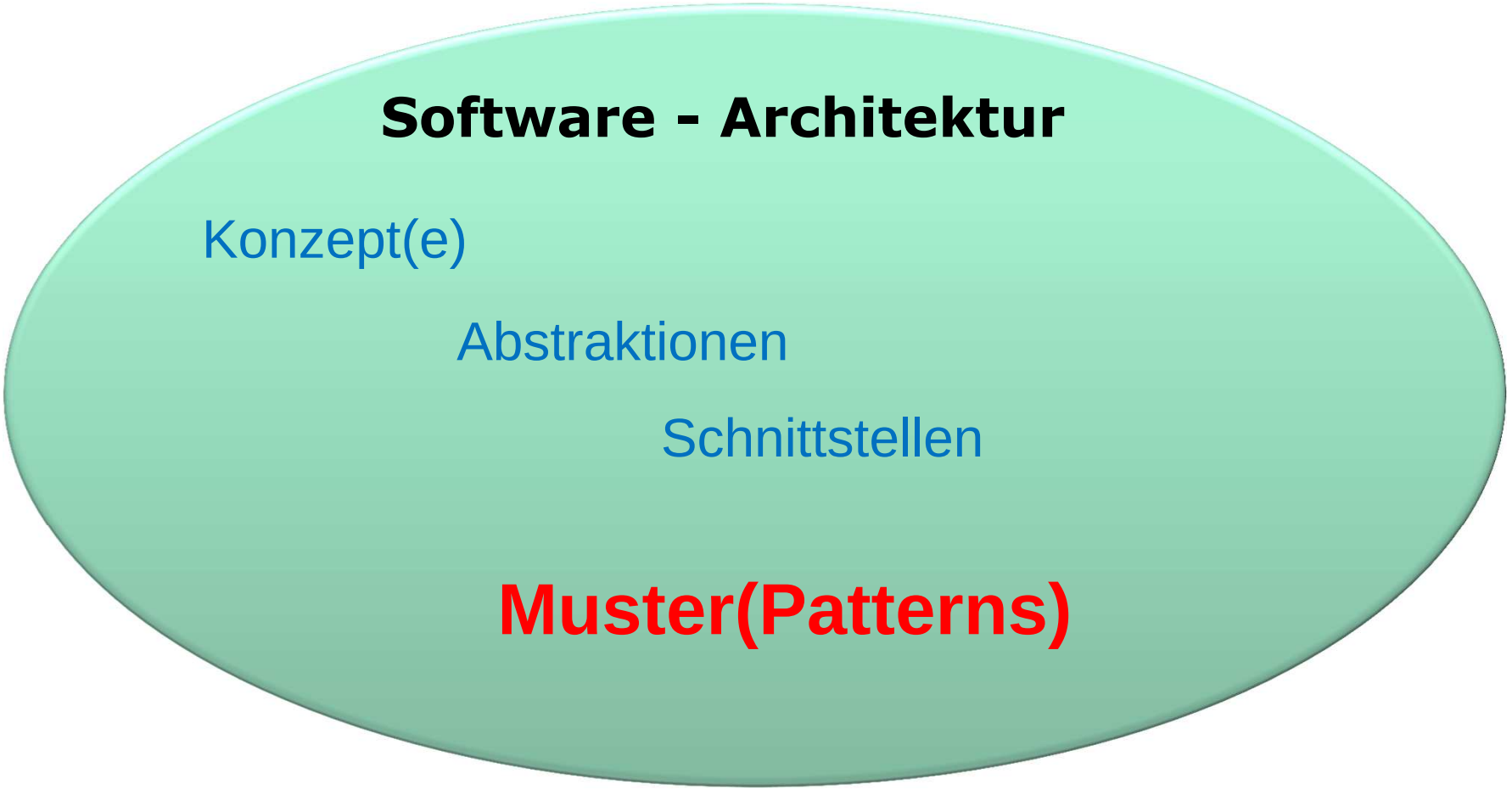
Inhalt – Block 4

- **Spezialthema: Muster(Patterns)**
  Einführung und Überblick

Software Engineering



Software - Architektur



Definitionen von Mustern

Definitionen von Mustern (Patterns)

Was sind Muster(Patterns)

Definitionen ursprünglich aus anderen Kontexten (als SW):

- (Bau)Architekt C.Alexander (*), A pattern Language, Towns Buildings, Construction, 1977. Übernahme in andere Bereiche (Kontexte) wie objektorientierte Softwareentwicklung und Mensch-Computer-Interaktion (Human-Computer Interaction)

  (*) „Each pattern describes a problem which occurs over and over again … 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 it the same way twice"

  V. Gruhn, 2006, MDA: Effektives Softwareengineering mit UML2 und Eclipse, Xpert.press,

Allgemeine Definitionen:

- Muster sind wiederkehrende Problemstellungen und bewährte Lösungsvorschläge, die sich bereits (mehrfach und/oder in mehreren Systemen) bewährt haben.
- Muster sind Schablonen
- „Ein Muster ist eine Idee , die sich in einem praktischen Kontext als nützlich erwiesen hat und es wahrscheinlich auch in anderen sein wird"
  M. Fowler, Analysis Patterns: Reusable Object Models, Addison-Wesley, 1996

Was macht ein Muster aus

Muster unterliegen einem dreiteiligen Schema:

- **Kontext:**

  Eine Situation, in der ein Problem auftritt

- **Problem:**

  Im Kontext häufig auftretende Probleme

- **Lösung:**

  Eine erprobte Lösung des Problems

Muster(Patterns) – Trends-5

Einsatz und Verbreitung(erung) von Patterns:

- Patterns als Mittel des Wissenstransfers
- Patterns und Agile Software Development
- Patterns in Analyse -> Patterns zu Requirementsengin.
- Patterns für Re-Engineering
- Patterns für Dokumentation
- Patterns für Testen
- Patterns für Konfigurations- und Versionsmanagement
- Security and Safety Patterns
- Software Management Patterns
- Software Quality Patterns
- Software Performance Patterns
- Organisational Patterns

Analyse-Muster

Analyse-Muster
(Analysis Patterns)

Analyse-Muster

Analyse-Muster:

Analyse von **Geschäftsprozessen** und **Geschäftsfällen** in OOA von SW – immer wiederkehrende Problemstellungen („generische" Problemstellungen).

Beispiele: Listen, Baugruppe, Rollen, Organisationen, Verwaltungen, Hierarchien, Register, Rollen, Gruppen, Historien, Materialwirtschaft.
Party, Accountabilty, …

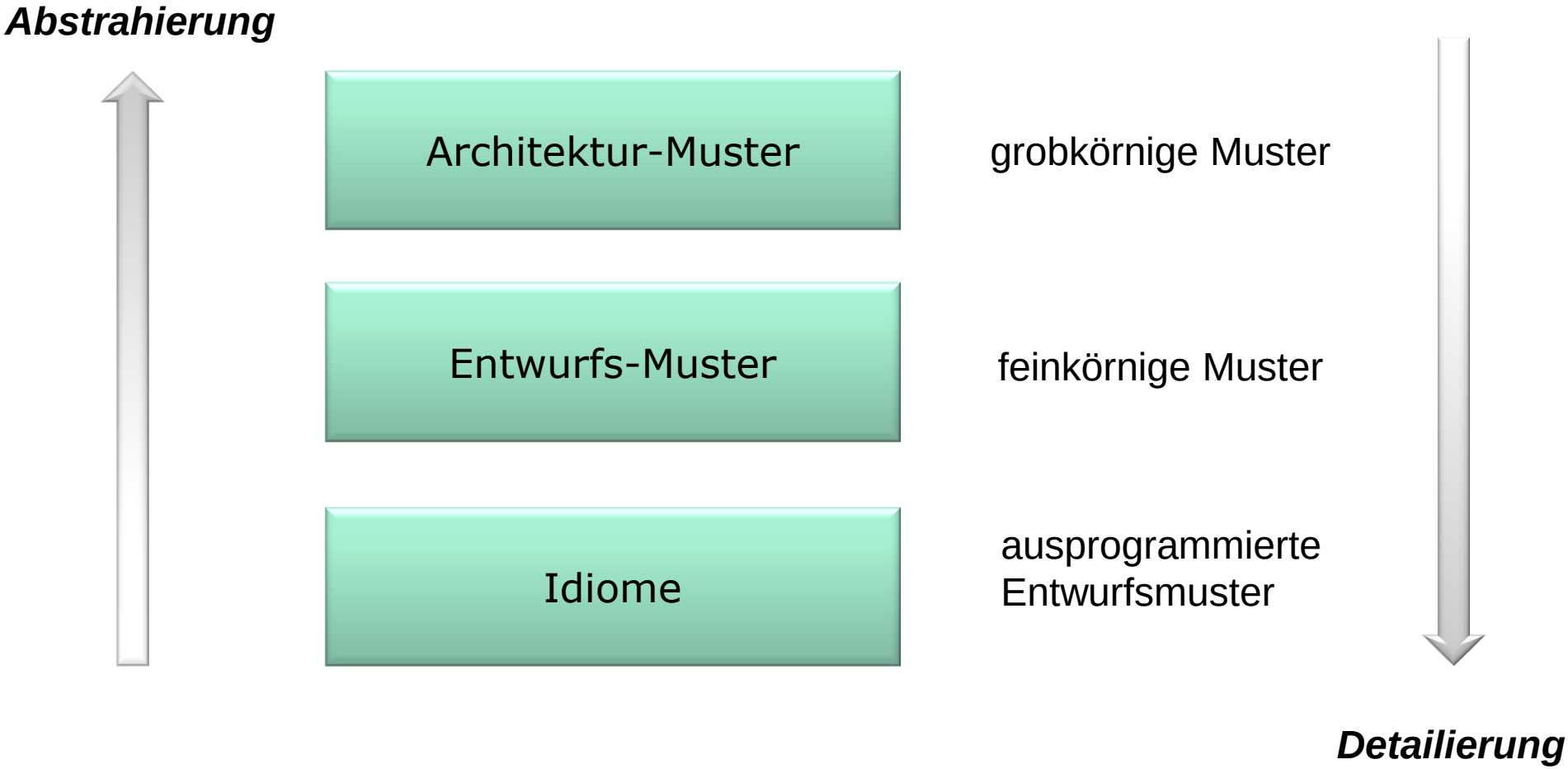
Archetyp-Muster:

Führt Idee der Analyse-Muster weiter – „domänenspezifische" (Analyse)Muster.

Analyse-Muster

Wichtige Muster - SW Kategorien

Kategorien von Mustern



Kategorien entsprechen (auch) wichtigen Phasen in der SW-Entwicklung.

Architekturmuster

Architektur-Muster
(Architectural Patterns)

Architekturmuster: Nutzen

„Architekturmuster helfen beim Zerlegen des Systems um die Komplexität zu beherrschen".

Architekturmuster:

- beeinflussen die Grundzüge der Architektur eines Systems
- helfen bei der Zerlegung eines Systems in Bausteine und Verteilung der Verantwortlichkeiten
- können verschiedene Entwurfsmuster enthalten
- …

Architektur-Muster: Beispiele

Muster	Thematiken
Model View Controller	Unterteilung einer interaktiven Anwendung in drei Komponenten
Pipes-and-Filters	Struktur für Systeme, die Datenströme verarbeiten
Layers	Strukturierung und Aufteilung von Anwendungen; Teilaufgaben auf verschiedenen Abstraktionsebenen
Blackboard	Hilfreich bei Problemen, für die es keine deterministische Lösungsstrategien gibt
Broker	Hilfreich bei Strukturierung verteilter Systeme, mit entkoppelten Komponenten (z.B. Diensten)
Reflection	Hilfreich bei bei dynamischen Veränderungen der Struktur und des Verhaltens von SW-Systemen
Presentation-Abstraction-Control	Struktur für interaktive SW-Systeme in Form einer Hierarchie kooperierender Agenten
Microkernel	Minimaler funktionaler Kern einer Anwendung und getrennt davon erweiterte Funktionalität und kundenspezifische Teile

Entwurfs-Muster

Entwurfs-Muster
(Design Patterns)

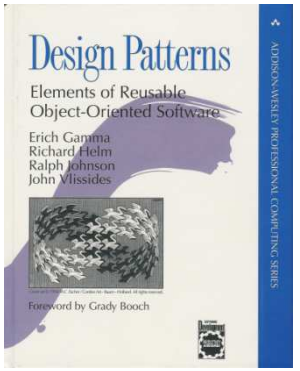
Entwurfs-Muster und Historie

„Entwurfsmuster sind bewährte Lösungsvorschläge für bestimmte Problemstellungen, die sich bereits in mehreren Systemen bewährt haben".

Historie:

- 70er C. Alexander - Architekt
- 1987 Beck / Cunningham – GUI
- 1995 Standardwerk über Entwurfsmuster

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



Entwurfsmuster (Design Patterns):

- sind plattformunabhängig
- sind nicht auf bestimmte Sprache beschränkt
- sind erprobte Standardmodelle für das OOD
- sind eine Möglichkeit Erkenntnisse wiederzuverwenden

Entwurfs-Muster - Beschreibungen

Entwurfsmuster sind:

- Entwurfsmuster sind abstrakte Beschreibungen von Problemen und deren Lösung beim Entwurf und der Implementierung von Softwaresystemen. Das Verständnis von Entwurfsmustern erleichtert die Analyse von Anwendungsproblemen.

Entwurfsmuster unterstützen:

- Anhand erkannter Muster in der Problemstellung lassen sich:
  - relevante Objekte und deren Zusammenwirken vorhergesagen,
  - verbreiterte Probleme beim Design wiederverwertbarer und erweiterbarer Softwaresysteme vermeiden.

Entwurfsmuster als „Sprache“:

- Entwurfsmuster führen klar definierte Begriffe zur Beschreibung des Verhaltens komplexer Softwaresysteme ein und vereinfachen so die Kommunikation zwischen Software-Entwicklern bei der Problemanalyse und des Design erweiterbarer Lösungen.

Entwurfsmuster – weitere Tabelle

SAD Muster

Erzeugungsmuster (Creational Design Patterns)	Strukturmuster (Structural Design Patterns)	Verhaltensmuster (Behavioral Design Patterns)	Muster für Objektrelationale Abbildung	Weitere Muster
Abstrakte Fabrik (abstract factory pattern)	Adapter (adapter pattern)	Beobachter (observer pattern)	Data Mapper	Business Delegate
Einzelstück (singleton pattern)	Brücke (bridge pattern)	Besucher (visitor pattern)	Data Access Object	Dependency Injection
Erbauer (builder pattern)	Dekorierer (decorator pattern)	Interceptor (interceptor pattern)	Datentransferobjekt (data transfer object)	Extension Interface
Fabrikmethode (factory method pattern)	Fassade (façade pattern)	Interpreter (interpreter pattern)	Table Data Gateway	Fluent Interface
Multiton (multiton pattern)	Fliegengewicht (flyweight pattern)	Iterator (iterator pattern)	Row Data Gateway	Inversion of Control
Prototyp (prototype pattern)	Kompositum (composite pattern)	Kommando (command pattern)	Active Record	Model View Controller (MVC)
	Stellvertreter (proxy pattern)	Memento (memento pattern)	Unit of Work	Model View Presenter (MVP)
	Half Object Plus Protocol	Nullobjekt (Null Object pattern)	Identity Map	Model View ViewModel (MVVM)
		Schablonenmethode (template method pattern)	Lazy Load	Transaction Script
		Strategie (strategy pattern)	Identity Field	Domain Model
		Vermittler (mediator pattern)	Dependent Mapping	Table Module
		Zustand (state pattern)	Embedded Value	Service Layer
		Zuständigkeitskette (chain of responsibility)	Serialized LOB	Page Controller
		State/Event	Inheritance Mapper	Template View
		Consequences	Metadata Mapping	Transform View
		Accumulator	Query Object	Two-Step View
		MapReduce		Application Controller
		Reduce/Combin		Remote Facade
		Rekursive Erweiterung		Locks
		Software Pipelining		Session States
		Prozess Wrapper		Repository oder Registry
		Token		Value Object

Idiome

Idiome (ausprogrammierte Muster)

Idiome - Beispiele

Idiom	Kommentar
Singleton (z.B. für C++)	Von einer Klasse darf es nur eine einzige Instanz geben (siehe Entwurfsmuster)
Factory-Method	…
Template-Method	…
Counted Pointer (z.B. für C++) / Reference Counter	Erleichtert die Speicherverwaltung von dynamisch erzeugten, mehrfach referenzierten Objekten in C++
Typesafe Enumeration	…
Multiple Inheritance	…

Klassifizierung von Mustern

Klassifizierung von Mustern

Kriterien der Klassifizierung

Es gibt nicht nur EIN Klassifikationsschema ...

- Das bekannteste ist von der GoF(*) betreffend **Entwurfsmuster** mit Klassifizierungen nach „Erzeugung", „Struktur" und „Verhalten" (siehe folgende Folien).

  (*) Erich Gamma et.al., Design Patterns, Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995 – out from Foreword by Grady Booch; GoF – 23 Entwurfsmuster

- **Klassifizierung nach Buschmann et. al:**

  Kriterium 1 - Musterkategorien

  Bereits bekannt die Kategorisierung nach:
  Architektur-Muster, Entwurfs-Muster, Idiome

  Kriterium 2 – Problemkategorien

  Problemsicht; Gruppierung nach verwandten Problemen

  F.Buschmann, Pattern-orientierte Softwarearchitektur, Addison-Wesley, 1998

  Unter Verwendung der beiden **Kriterien 1+2** kann ein Klassifikations-schema bereitgestellt werden und die Muster der GoF in die Aufstellung eingeordnet werden (siehe folgende Tabellen – blaue Einträge).

Klassifikations-Schema – Teil 1

SAD Muster

Gruppe	Architekturmuster	Entwurfsmuster	Idiome
Vom Chaos zur Struktur	Layers, Pipes-and-Filters, Blackboard	*Interpreter*	
Verteilte Systeme	Broker Pipes-and-Filters Microkernel		
Interaktive Systeme	Model-View-Controller Presentation-Abstraction-Control		
Adaptierbare Systeme	Microkernel Reflection		
Erzeugung		*Abstract-Factory* *Prototype* *Builder*	*Singleton* *Factory-Method*
Strukturelle Zerlegung		Whole Part *Composite*	

GoF-Muster

Klassifikations-Schema – Teil 2

SAD Muster

Gruppe	Arch.muster	Entwurfsmuster	Idiome
Organisation von Arbeit		Master-Slave *Chain-of-Responsibility* *Command Mediator*	
Zugriffskontrolle		Proxy, *Facade, Iterator*	
Variation von Diensten		*Bridge, Strategy, State*	*Template-Method*
Erweiterung der Dienstleistung		*Decorator, Visitor*	
Management		Command Processor View Handler *Memento*	
Adaption		*Adapter*	
Kommunikation		Publisher-Subscriber Forwarder-Receiver Client-Dispatch.-Server	
Ressourcenverwaltung			Counted Pointer

GoF-Muster

Vorgangsweise bei der Auswahl*

Wichtige Schritte - Auswahlverfahren:

1. Problem spezifizieren

2. Musterkategorie auswählen
3. Problemkategorie auswählen
4. Prüfen der Auswahl – der aus den Kategorien ausgewählten Mustern - gegen die Problembeschreibung
5. Vorteile und Nachteile der Auswahl der Muster vergleichen
6. Auswahl der optimalen Variante der angedachten Muster
7. Wenn keine geeignete Kategorie bzw. Muster gefunden werden können, Auswahl anderer Kategorien
8. Muster gefunden – Realisierung mit Muster ODER
   Kein Muster gefunden – Realisierung ohne Muster

Beide Varianten (mit oder ohne Mustereinsatz) sind als finale (Architektur-)Entscheidung OK !

* *zugrundeliegend kann das zuvor genannte Klassifikationsschema verwendet werden*

SW-Architektur und Muster

Einsatz von Mustern in SW-Architektur und -Design

Muster in SW-Archit. und -Design

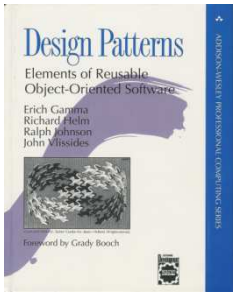
„Muster sollen immer dann angewandt werden, wenn sie auch sinnvoll sind und Vorteile bringen."

„Used in the wrong place, the best patterns will fail."

Muster (Patterns) als Merkmale für:

- „Best Practice"
- „State of the art"
- Qualität

von (OO)-SW (siehe auch nachfolgendes Zitat)**.**



„All well-structured object-oriented architectures are full of patterns. Indeed, one of the ways that I measure the quality of an object-oriented system is to judge wheter or not ist developers have paid careful attention to the common collaborations among it's objects.

Focusing on such mechanisms during a system's development can yield an architecture that is smaller, simpler, and far more understandable than if these patterns are ignored."

Muster in SW-Archit. und -Design

Vorteile der Verwendung von Mustern:

- Risiko von Entwurfsfehlern senken
- Gut geeignet für Kommunikation über Entwurfsentscheidungen
- Helfen Entwürfe flexibler zu gestalten
- …

Nachteile der Verwendung von Mustern:

- Es können zusätzliche Klassen oder Interfaces entstehen
- Mehrwert des Einsatzes immer prüfen:
  Mehrwert an Flexibilität, Performance, Wiederverwendbarkeit, …
- Letztentscheidung: Finale Architektur = einfachster Weg

Techniken der SW-Architektur - 1

SAD Muster

Funktionales:

Prinzip/Basistechnik	Aspekte	Muster (Beispiele)
Abstraktion		Layers, Abstract-Factory
Kapselung		Forwarder-Receiver
Information Hiding		Whole-Part
Modularisierung		Layers, Pipes-and-Filters, Whole-Part
Trennung der Belange		Model-View-Controller
Kopplung und Kohäsion		Client-Dispatcher, Publisher-Subscriber

Techniken der SW-Architektur - 2

SAD Muster

Funktionales:

Prinzip/Basistechnik	Aspekte	Muster (Beispiele)
Angemessenheit, Vollständigkeit und Einfachheit		Strategy
Trennung von Strategie und Implementierung		Strategy
Trennung von Schnittstelle und Implementierung		Bridge
Single Point of Reference	jedes Element nur einmal deklariert und definiert	
Divide and Conquer		Whole-Part; Microkernel

Techniken der SW-Architektur - 3

SAD Muster

Nichtfunktionales*:

Prinzip/Basistechnik	Aspekte	Muster (Beispiele *)
Änderbarkeit	Wartbarkeit, Erweiterbarkeit, Restrukturierbarkeit, Portierbarkeit	Reflection, Bridge
Interoperabilität		Broker
Effizienz		Forwarder-Receiver
Zuverlässigkeit	Fehlertoleranz, Robustheit	Master-Slave
Testbarkeit	Implementierung nach Broker machts aber komplexer	Broker, Command-Processor
Wiederverwendbarkeit	Nutzung von/bewusste Entwicklung Wiederverwendbarem	Model-View-Controller

* **Muster unterstützen/fördern Einhaltung von nichtfunktionalen Anforderungen, gewährleisten diese aber meist nicht 1:1**