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

Angewandtes Softwaredesign
Übungen Teil-2

WS2014

DI Dr. Gottfried Bauer

LV-Typ: VO, UE
Semester: 3
LV-Nummer: **D 1933 ILV**
LV-Bezeichnung: Angewandtes Softwaredesign

Abgaben und Termine

- **4 Teams mit jeweils 4-5 Team-Mitgliedern**
- **Übungen:**
  - **Aufgabenstellungen** siehe Informationen zu Übungen
  - **Projektbeschreibung** siehe Informationen zu Übungen
  - **Abgaben und Termine:**

    Abgabe-Einstieg: 24.10.2014
    Einfache Musterbeispiele - Entwurfsmuster
    (inklusive Einarbeitung der Diskussionsergebnisse)

    Abgabe-Teil-1: 19.12.2014
    Überlegungen – potentielle Muster für Übungsprojekt
    (inklusive Recherchen und Analysen)

    Abgabe-Teil-2: 12.01.2014
    Auswahl – Muster für Übungsprojekt

    Abgabe-Teil-3: 23.01.2015
    Lösung (Architektur) mit Mustern für Übungsprojekt
- **Präsentation - Lösung mit Mustern: 28.01.2015**

Ablauf der Übungen

- **Teil-Einstieg:** Entwurfsmuster verstehen. Aufbereitung und Präsentation eines zugeteilten Entwurfsmusters (Einzelpräsentation).
- **Teil-1:** Basierend auf der Definition einer zu realisierenden SW und deren Anforderungen (funktionale und nichtfunktionale) sollen Muster identifiziert werden, die entsprechend den Basiskonzepten für „gute" Software-Architektur geeignet sind, diese SW zu designen. Die Aufgabenverteilung auf die einzelnen Teams/Gruppen erfolgt entsprechend einzelner Basiskonzepte und Prinzipien. Konzentration auf Ausprägung von 2-4 frei wählbaren nichtfunktionalen Anforderungen bzw. Qualitätsmerkmalen.
- **Teil-2:** (Gemeinsame) Durchsicht der ausgewählten Muster aus Teil B und Auswahl der vielversprechensten bzw. wichtigsten Muster nach Priorisierung. Erarbeitung von Ansätzen zur Modellierung und dem Einsatz der (final) ausgewählten Muster für einen Entwurf zur Software.
- **Teil-3:** Betrachtung der Ergebnisse in Bezug auf die angepeilte Gesamtsoftware. Analyse der vorgeschlagenen Lösung auf die gegenläufige Einhaltung von Prinzipien und mögliche Konsequenzen (z.B. performancesteigernde versus performancehemmende Ansätze). Abrundung der Ergebnisse der Teams bzw. Gruppen entsprechend den Analyseergebnissen.
- **Präsentation:** alle Teams präsentieren finale Lösungen zur Aufgabenstellung – inklusive Diskussionen

Übungsprojekt - MedDev(ices)

Sammlung, Aufbereitung und Darstellung von medizinischen Daten auf unterschiedlichen Devices (Mobile Devices, Desktop, MS Surface, ...)

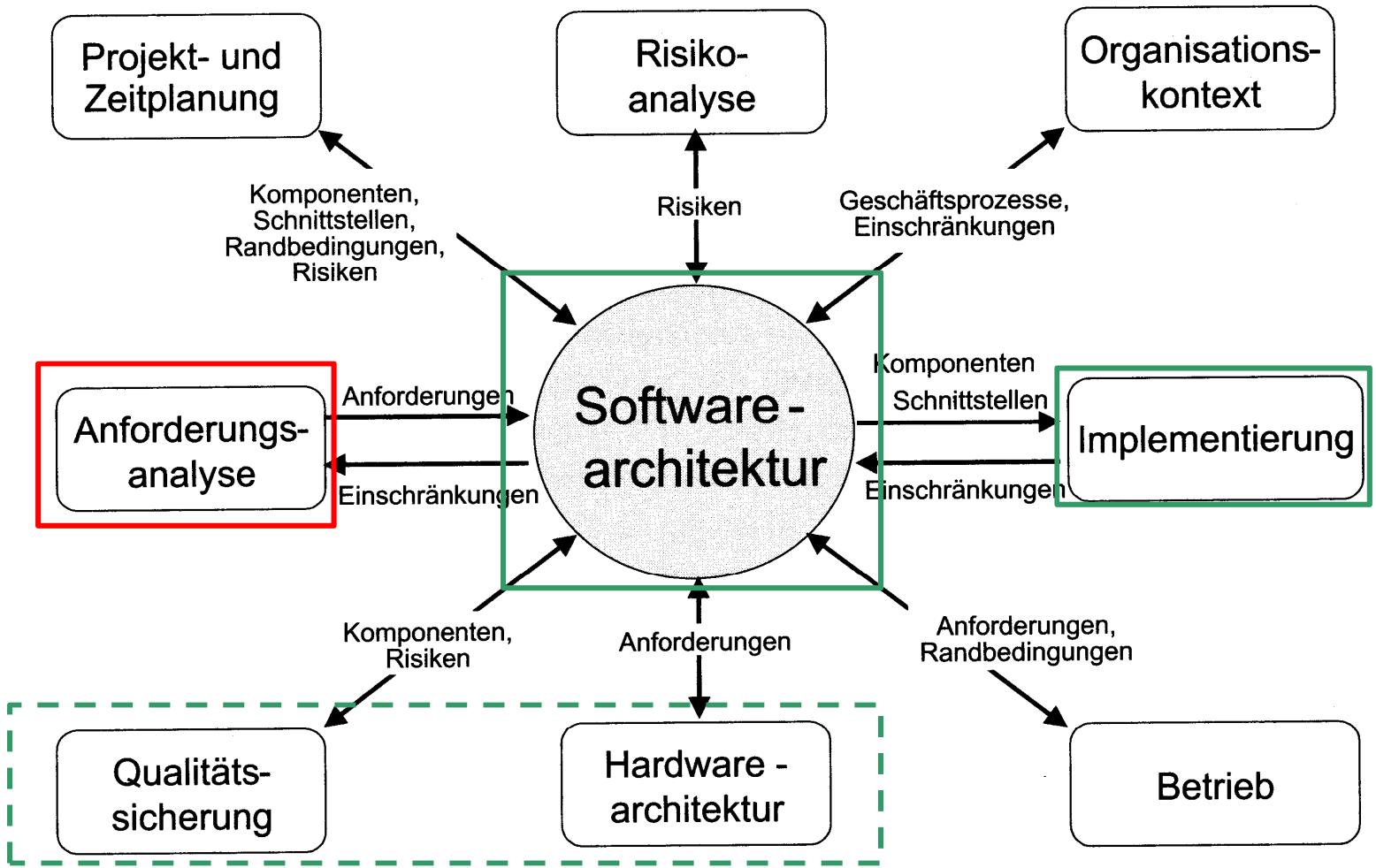
Fallstudie SW-Architektur und Design – Einsatz von Mustern (Patterns)

Beschreibung von MedDev siehe eigener Foliensatz:

MIT_ASD_Muster(Patterns)_WS2014 - Übungsprojekt

Übungsprojekt - Arch. Umfeld

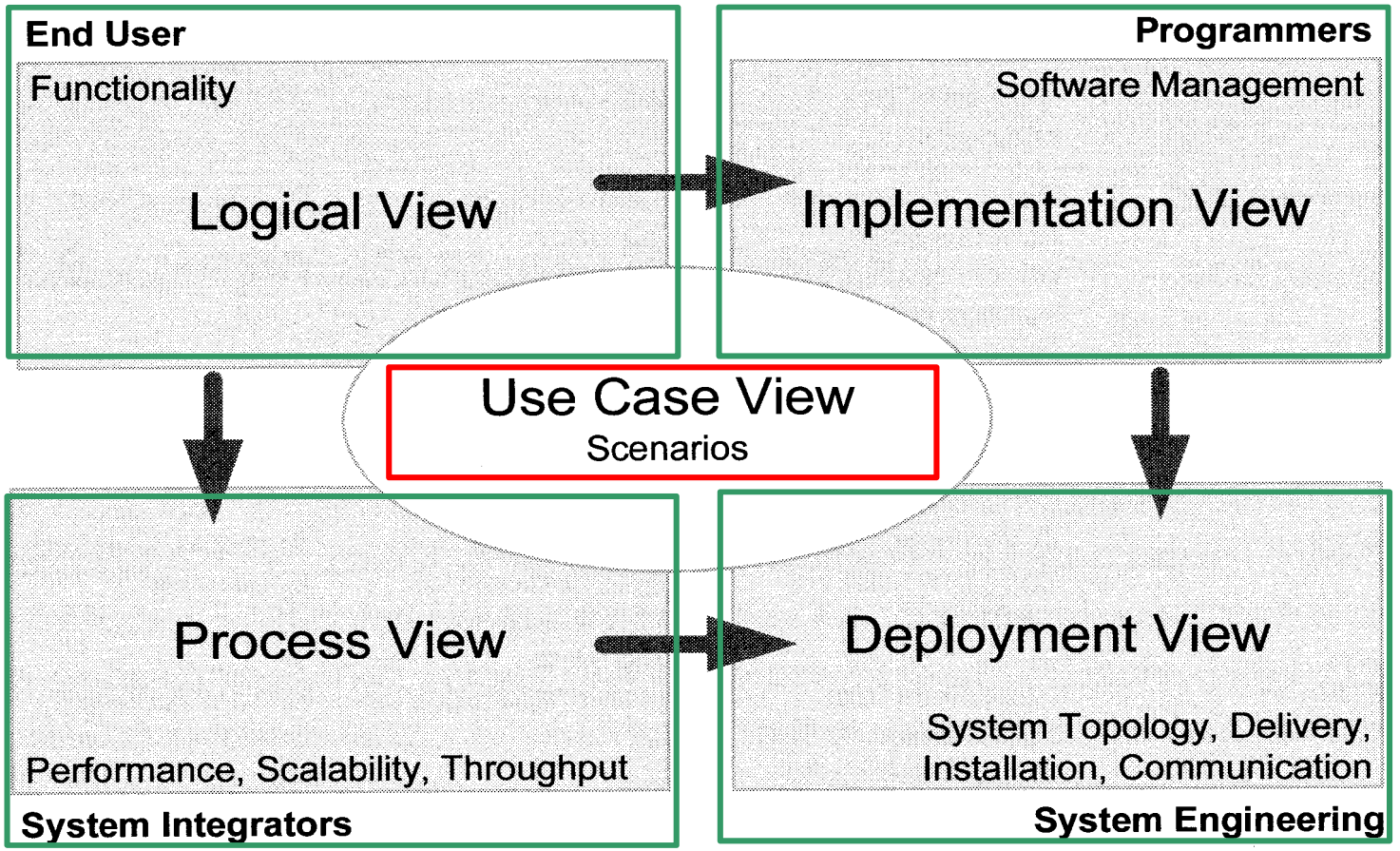
Architektur und Umfeld:



A. Schatten et.al., Best Practice Software-Engineering, Spektrum, 2010

Übungsprojekt - Arch. Views

Architektur und Views:

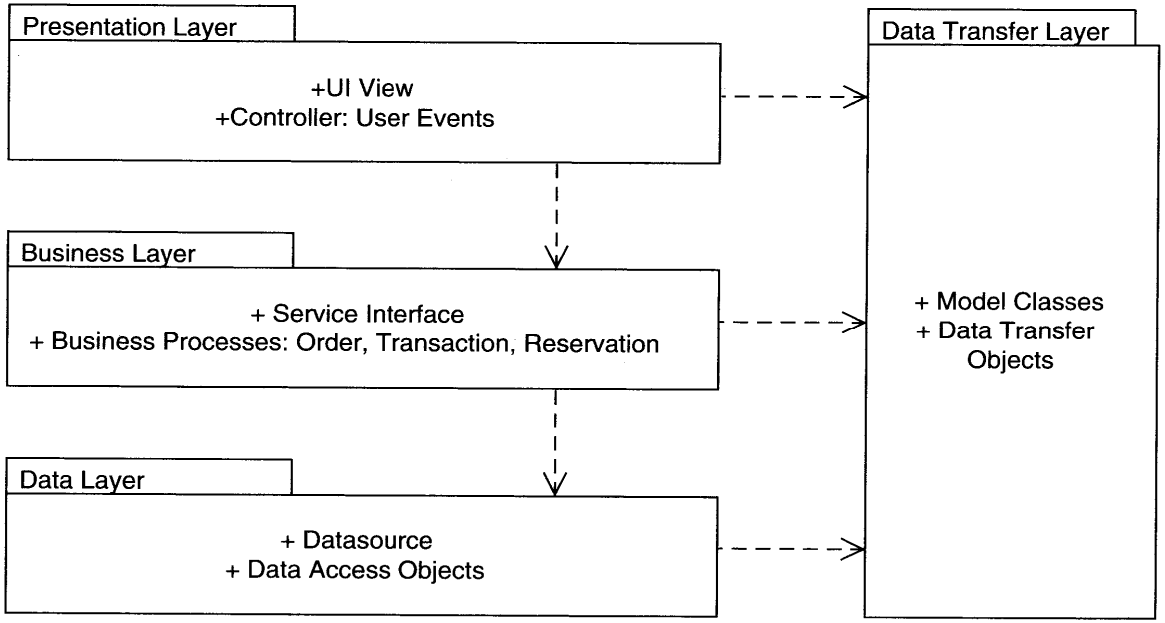


A. Schatten et.al., Best Practice Software-Engineering, Spektrum, 2010

Übungsprojekt - Arch. Schichten

Architektur und Schichten - z.B.:

- **Presentation Layer** (GUIs)- *Präsentations-Schicht*
  Benutzerinteraktionen, Logik der Bedienelemente von GUIs
- **Business Layer** (Kernfunktionalität) - *Logik-Schicht*
  eventuell nochmals unterteilt in Data Access Layer (DAOs) und *Service Layer*
- **Data Layer** (Daten) - *Daten-Schicht*
  Persistierung von Daten



A. Schatten et.al., Best Practice Software-Engineering, Spektrum, 2010

Übungsprojekt - Musterpool

- **Architekturmuster: …**

  welche kämen in Frage ?
  welche ausgesucht ?
  welche wie eingesetzt ?

- **Entwurfsmuster:**
  - **elementare**

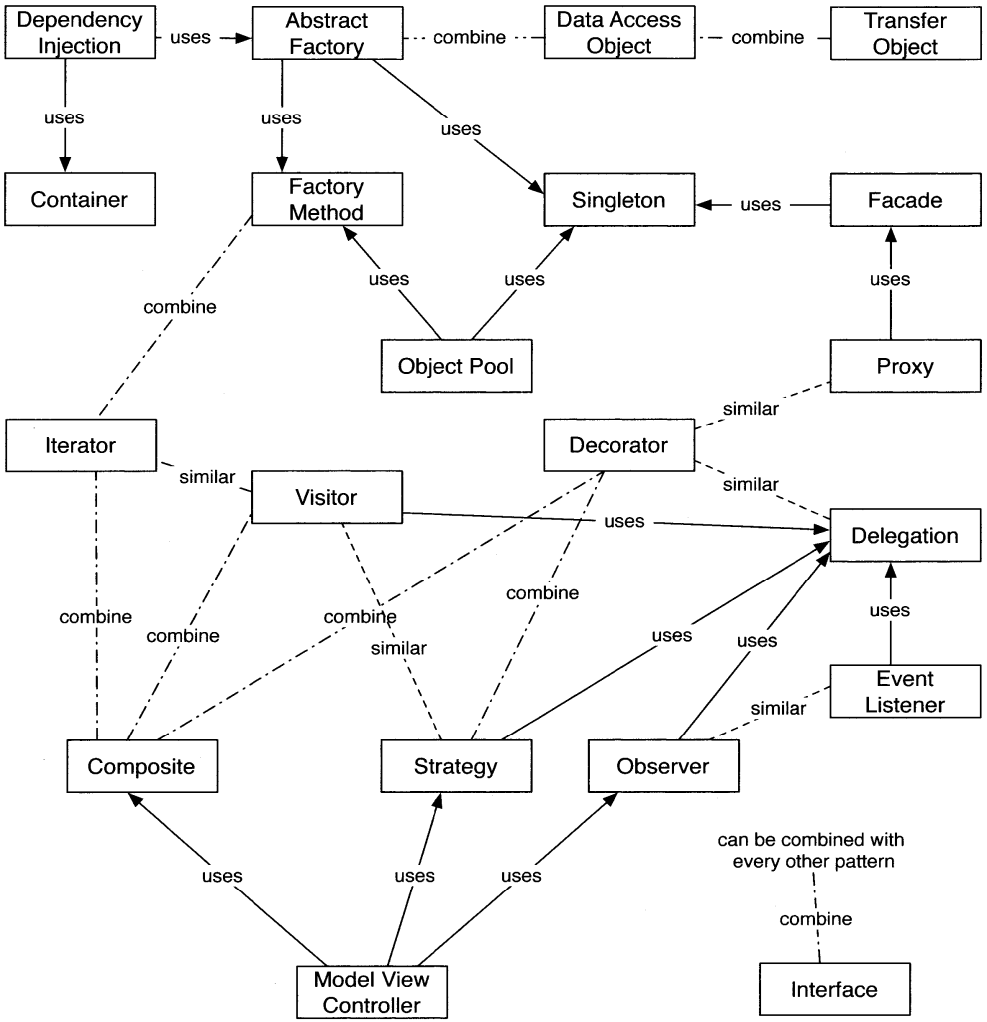
    welche kämen in Frage ?
    welche ausgesucht ?
    welche wie eingesetzt ?

  - **kombinierte**

    welche kämen in Frage ?
    welche ausgesucht ?
    welche wie eingesetzt ?

Übungsprojekt - Musterkombin.

ASD-Übungen
Teil-2



uses: Muster wird verwendet von
combine: Muster kann kombiniert werden mit
similar: Muster vom Design her ähnlich

A. Schatten et.al., Best Practice Software-Engineering, Spektrum, 2010

Übungsprojekt - Kräfte+Balance

ASD-Übungen Teil-2

Kräfte und Balance von Mustern und Entwürfen:

Ein wichtiger Bestandteil jedes Musters ist ein System von Kräften („forces"). Diese **Kräfte** sind Teil des Problems, das durch das Muster gelöst wird. Die Lösung muss also eine Balance zwischen Kräften herstellen.

Natürlich kann das Muster diese Balance nur allgemein beschreiben. Der Architekt muss die Ausbalancierung der Kräfte für eine konkrete Lösung in einer Entwurfssituation dem Muster folgend erarbeiten.

Finden der Balance betreffend Muster: eine Hilfestellung dazu sind
angeführte Konsequenzen aus Beschreibungen des Musters.

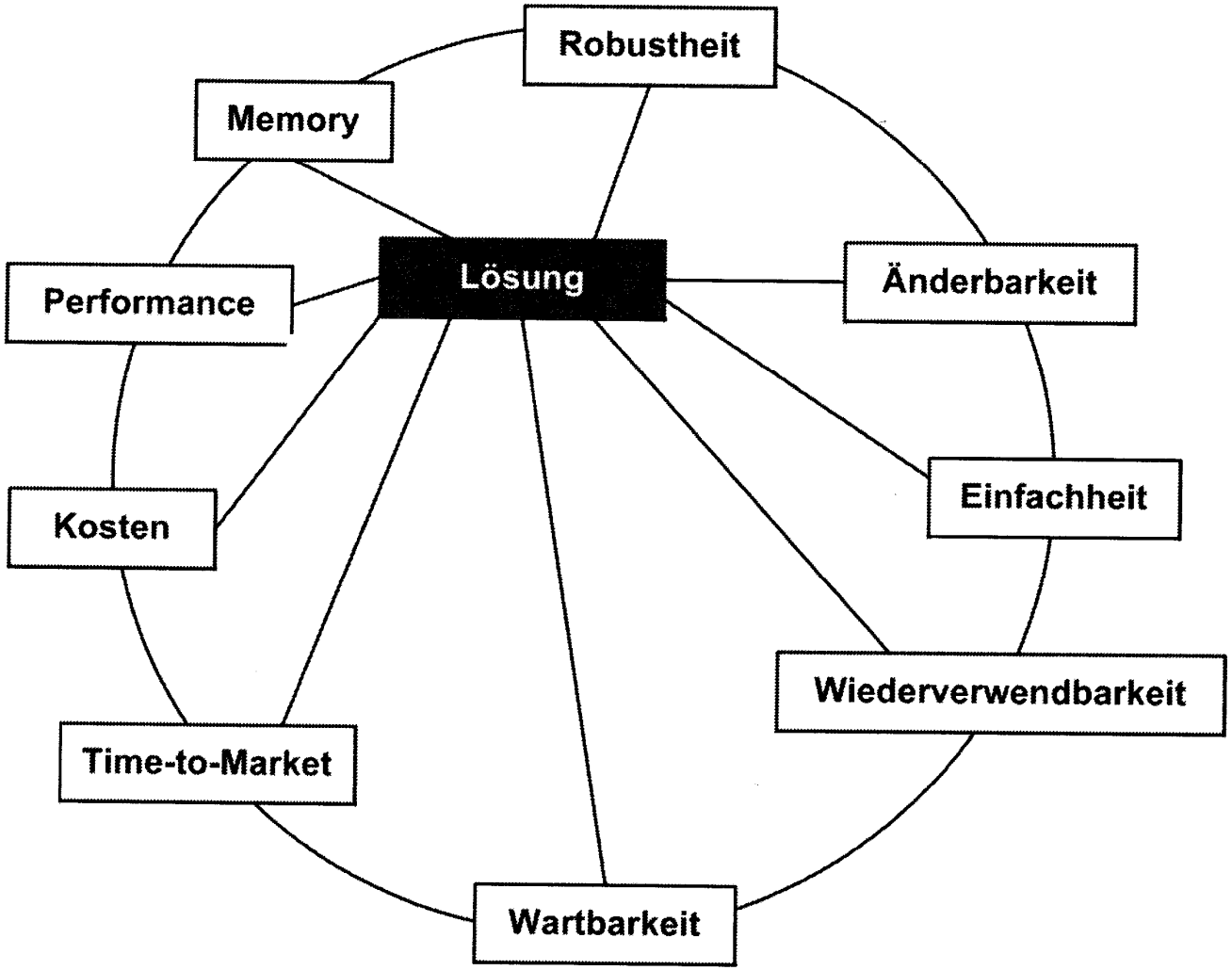
Balance von verwendeten Mustern ->
übergreifend: Balance der Architektur = „Kompromiss"

In der folgenden Abbildung ist dargestellt, wie verschiedene Kräfte, die allesamt typische architektonische Qualitätsattribute sind, eine Lösung beeinflussen und bestimmen.

O. Vogel et.al., Software-Architektur. Grundlagen-Konzepte-Praxis, Spektrum, 2009

Übungsprojekt - Kräfte+Balance

ASD-Übungen Teil-2



Übungsprojekt - Q-Merkmale

Liste - Qualitätsmerkmale – primärer Pool:

- Performance
- Memory/Speicherbedarf(belastung)
- Persistenz
- Load Balancing
- Skalierbarkeit
- Verfügbarkeit
- Einfachheit
- Wiederverwendbarkeit
- Erweiterbarkeit
- Security
- Schnittstellen und Robustheit
- Wartbarkeit

Übungsprojekt - Q-Merkmale

Liste - Qualitätsmerkmale – sekundärer Pool:

- Backup und Recovery
- Konfiguration
- Abhängigkeiten (HW, OS, Entw.-Sprache, Plattformen, FW, ...)
- Logging
- Monitoring
- Authentifizierung und Authorisierung
- GUIs
- Workflows
- ...

Techniken der SW-Architektur-1

ASD-Übungen Teil-2

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

ASD-Übungen Teil-2

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

ASD-Übungen Teil-2

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

Klassifikation von Mustern - 1

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

F. Buschmann et.al.: Pattern-orientierte Softwarearchitektur, Addison-Wesley, 1998

Klassifikation von Mustern - 2

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

F. Buschmann et.al.: Pattern-orientierte Softwarearchitektur, Addison-Wesley, 1998

Auswahl von Mustern - 1

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 !

F. Buschmann et.al.: Pattern-orientierte Softwarearchitektur, Addison-Wesley, 1998

Auswahl von Mustern - 2

Basierend auf unserem Klassifikationsschema, unserem Schema zur Beschreibung eines Musters (s. Kapitel 1, *Muster*) und den Beziehungen zwischen den Mustern können wir jetzt das folgende einfache Verfahren zur Auswahl eines Musters definieren. Es besteht aus sieben Schritten:

1 *Spezifizieren Sie das Problem.* Um ein Muster finden zu können, das Ihnen bei Ihrem konkreten Problem hilft, müssen Sie dieses Problem erst genau beschreiben: Was ist, ganz grundsätzlich gesprochen, das zu lösende Problem aus Sicht der Anwendung, und welche Kräfte kommen zur Wirkung? Wenn das grundsätzliche Problem mehrere Aspekte besitzt, zum Beispiel wenn die Architektur eines verteilten und zugleich interaktiven Systems spezifiziert werden soll, dann zerlegen Sie das Problem in mehrere Teilprobleme. Beschreiben Sie jedes Teilproblem und die darauf einwirkenden Kräfte getrennt. Versuchen Sie, für jedes Teilproblem ein Muster zu finden, das bei der Lösung des Teilproblems hilft.

Nehmen wir zum Beispiel einmal an, daß Sie die grundsätzliche Architektur eines interaktiven Texteditors definieren wollen. Das System soll verschiedene Benutzerschnittstellen-Bibliotheken benutzen und an verschiedene anwendungsspezifische Gestaltungsrichtlinien (engl. style guide) angepaßt werden können. Wir werden dieses Beispiel benutzen, um die folgenden Schritte der Musterauswahl zu erläutern.

F. Buschmann et.al.: Pattern-orientierte Softwarearchitektur, Addison-Wesley, 1998

Auswahl von Mustern - 3

2 *Wählen Sie die Musterkategorie aus*, die der Entwurfsaktivität entspricht, die Sie gerade durchführen. In unserem Beispiel wollen wir die grundsätzliche Architektur des Texteditors festlegen. Daher wählen wir die Kategorie der Architekturmuster aus.

Auch wenn dieser Schritt kein Detailwissen über das Entwurfsproblem verlangt, schränkt er die Anzahl der Muster, die potentiell auf das Problem anwendbar sind, deutlich ein.

3 *Wählen Sie die Problemkategorie aus*, die der grundsätzlichen Problemstellung entspricht. Jede Problemkategorie beschreibt durch die Muster, die in diese Kategorie fallen, die Art der Probleme, die zu dieser Kategorie gehören. Für unser Beispiel wählen wir die Problemkategorie der interaktiven Systeme aus, in der das Model-View-Controller-Muster (124) und das Presentation-Abstraction-Control-Muster (145) liegen. Wenn das aktuelle Problem in keine Problemkategorie paßt, wählen Sie, sofern möglich, eine andere Problemkategorie aus (Schritt 7).

F. Buschmann et.al.: Pattern-orientierte Softwarearchitektur, Addison-Wesley, 1998

Auswahl von Mustern - 4

4 *Vergleichen Sie die Problembeschreibungen.* Jedes Muster in der von Ihnen ausgewählten Problemkategorie kann einen bestimmten Aspekt Ihres konkreten Problems adressieren. Daher kann entweder ein einzelnes Muster oder auch eine Kombination mehrerer Muster Ihr Problem lösen. Wählen Sie diejenigen Muster aus, deren Problembeschreibungen und Kräfte am besten auf Ihr aktuelles Problem zutreffen. Dieser Schritt ist damit der erste, der konkretes Wissen über das zu lösende Entwurfsproblem voraussetzt. Für den Texteditor wählen wir das Model-View-Controller-Muster. Sowohl dieses als auch das Presentation-Abstraction-Control-Muster unterstützen die Veränderbarkeit von Benutzeroberflächen. Da aber die Funktionalität für das Bearbeiten von Text eher in einer Menge miteinander verwandter Funktionen als in einer Reihe voneinander unabhängiger Funktionen liegt, besteht kein Bedarf daran, eine agentenbasierte Architektur zu nehmen, wie vom Presentation-Abstration-Control-Muster vorgeschlagen wird.

Wenn die Muster der ausgewählten Problemkategorie die Aspekte Ihres Entwurfsproblems nicht ansprechen, wählen Sie eine andere Problemkategorie aus, sofern möglich (Schritt 7).

5 *Vergleichen Sie die Vorteile und die Nachteile.* Dieser Schritt beleuchtet die Konsequenzen der Anwendung der von Ihnen bisher vorausgewählten Muster. Wählen Sie das Muster aus, dessen Vorteile für Sie wichtig sind und dessen Nachteile für Sie keine große Rolle spielen. Da wir bereits ein bestimmtes Muster für die Architektur unseres Texteditors ausgesucht haben, überspringen wir diesen Schritt.

F. Buschmann et.al.: Pattern-orientierte Softwarearchitektur, Addison-Wesley, 1998

Auswahl von Mustern - 5

6 *Wählen Sie die Variante aus*, die für Ihr Entwurfsproblem die beste Lösung darstellt. In unserem Beispiel sind die Funktionalität der Ansicht (engl. view) und der Steuerung eng miteinander verwoben. Daher wählen wir die Document-View-Variante des Model-View-Controller-Musters aus, um damit die Architektur unseres Editors festzulegen.

Wenn Sie in Schritt 3 oder 4 auf kein Problem gestoßen sind, dann haben Sie jetzt Ihre Musterauswahl abgeschlossen.

7 *Wählen Sie eine andere Problemkategorie aus*. Wenn es keine geeignete Problemkategorie gibt oder es in der von Ihnen ausgewählten Problemkategorie kein passendes Muster gibt, versuchen Sie es damit, eine andere Problemkategorie auszuwählen, die Ihr Entwurfsproblem weiter verallgemeinert. Vielleicht enthält diese Kategorie Muster, die – eventuell verfeinert – Ihnen helfen, Ihr Problem zu lösen. Dann kehren Sie zu Schritt 4 zurück.

F. Buschmann et.al.: Pattern-orientierte Softwarearchitektur, Addison-Wesley, 1998

Auswahl von Mustern - 6

Wenn die Schritte 2, 3 und 4 Ihnen auch dann kein Ergebnis liefern, wenn Sie es mit anderen Problemkategorien versucht haben, sollten Sie die Suche beenden – das Mustersystem enthält kein Muster, das Ihnen bei Ihrem Entwurfsproblem helfen kann. Sie sollten dann andere Mustersprachen, -systeme oder -kataloge untersuchen, um festzustellen, ob Sie dort vielleicht ein geeignetes Muster finden. Oder Sie versuchen, Ihr Entwurfsproblem ganz ohne Anwendung von Mustern zu lösen.

Sie brauchen dieses Auswahlverfahren nicht anzuwenden, um ein Muster zu implementieren oder zu verfeinern, das Sie bereits ausgewählt haben. Im Abschnitt über die Implementierung des von Ihnen ausgewählten Musters finden Sie Verweise auf diejenigen Muster, die Ihr Muster auf natürliche Weise ergänzen.

F. Buschmann et.al.: Pattern-orientierte Softwarearchitektur, Addison-Wesley, 1998

Referenzarchitekturen - 1

Definitionen:

Eine **Referenzarchitektur** (engl. Reference Architecture) ist in der Informatik eine Schablone für eine Klasse von Softwarearchitekturen für einen bestimmten Anwendungsbereich. Charakteristisch für eine Referenzarchitektur sind unter Anderem ein eigenes einheitliches Vokabular um Implementationen zu besprechen und eine Liste von Funktionen und Schnittstellen der Funktionen mit anderen in- und externen Funktionen.

Eine **Referenzarchitektur** ist eine abstrakte Softwarearchitektur, sie definiert Strukturen und Typen von Software-Elementen sowie deren erlaubte Interaktionen und ihre Verantwortlichkeiten speziell für einen Anwendungsbereich. Die Strukturen sind jeweils für alle Systeme innerhalb einer Domäne anwendbar. Referenzarchitekturen bilden somit die Grundlage für neue Softwaresysteme. Ein System ist zu einer Referenzarchitektur konform, wenn sich die Struktur oder die Strukturen der Referenzarchitektur darin wieder finden. Es muss Schnittstellen, Schichten oder Komponenten enthalten, die wie in der Referenzarchitektur in Beziehung stehen. Ein System kann zu mehreren Referenzarchitekturen konform sein, da es verschiedene Strukturen aufweisen kann.

Referenzarchitekturen - 2

Eine **Referenzarchitektur** ist in der Informatik ein Referenzmodell für eine Klasse von Architekturen. Die Referenzarchitektur kann als Modellmuster – also ein idealtypisches Modell – für die Klasse der zu modellierenden Architekturen betrachtet werden. Neben fachlichen Referenzarchitekturen gibt es (vor allem) auch technische Referenzarchitekturen.

Vorteile bei der Verwendung von Referenzarchitekturen ist die Zeitersparnis durch die erzielte Wiederverwendung – hierbei reicht das Spektrum der Wiederverwendung von den architektonischen Blaupausen bis hin zu konkreten Software Komponenten, da diese ja alle der Referenzarchitektur inhärenten Anforderungen erfüllen. Ein System, welches einer Referenzarchitektur folgt, muss also nicht „von Null weg" auf dem Reißbrett geplant werden, sondern fußt auf – im besten Fall – vielfach bewährten Konzepten, Entscheidungen und Lösungsmustern. Damit ist die Verwendung von Referenzarchitekturen auch angewandtes Wissensmanagement, da sich viele evolutorisch gewonnene Erfahrungen in einer Referenzarchitektur manifestieren.
Nachteile der Verwendung einer Referenzarchitektur sind der Aufwand der Erstellung und Pflege (technologische und konzeptuelle Neuerungen im Technologie- und Anwendungsbereich der Referenzarchitektur müssen ja nachgezogen werden), die Vererbung von Fehlern der Referenzarchitektur in konkrete, von dieser Referenz abgeleiteten, Architekturen und die Einschränkung des kreativen Freiraumes der Beteiligten, da viele Rahmenbedingungen und Entscheidungen bereits vorweg genommen sind

Referenzarchitekturen - 3

Links:

http://www.sigs.de/download/oop_2011/downloads/files/Do1-2_Gharbi_Koschel-IT-Projekte-Refernzarchitekturen.pdf

http://riemke.net/unternehmensportale-intranet/referenzarchitektur-portal/

http://www.ibm.com/developerworks/rational/library/2774.html

Bücher:

http://link.springer.com/chapter/10.1007/978-3-642-02439-9_8#page-1

Literaturverzeichnis - 1

1. Heide Balzert, Helmut Balzert:
   Lehrbuch der Objektmodellierung: Analyse und Entwurf mit der UML 2,
   Spektrum, 2.Auflage, 2004,
   ISBN 978-3827411624
2. Erich Gamma et.al.:
   Design Patterns, Elements of Reusable Object-Oriented Software,
   Addison-Wesley, 1995,
   ISBN 978-0201633610
3. Frank Buschmann et.al.:
   Pattern-orientierte Softwarearchitektur: Ein Pattern System,
   Addison-Wesley, 1998,
   ISBN 978-3827312822
4. Douglas Schmidt, Michael Stal, Hans Rohnert, Frank Buschmann:
   Pattern-orientierte Softwarearchitektur: Muster für nebenläufige und vernetzte Objekte,
   Addison-Wesley, 2002,
   ISBN 978-3898641425
5. Martin Fowler:
   Analysis Patterns: Reusable Object Models,
   Addison-Wesley, 1996,
   ISBN 978-0201895421

Literaturverzeichnis - 2

6. Karl Eilebrecht, Gernot Starke:
Patterns kompakt. Entwurfsmuster für effektive Software-Entwicklung,
Spektrum, 2010,
ISBN 978-3827425256

7. Alexander Schatten et.al.:
Best Practice Software-Engineering,
Spektrum Verlag, 2010,
ISBN 978-3827424860

8. Oliver Vogel et.al.:
Software-Architektur. Grundlagen-Konzepte-Praxis,
Spektrum, 2009,
ISBN 978-3827419330

9. Thomas Grechenig et.al.:
Softwaretechnik. Mit Fallbeispielen aus realen Entwicklungsprojekten,
Pearson, 2010,
ISBN 978-3868940077

10. Joachim Goll:
Methoden und Architekturen der Softwaretechnik,
Vieweg-Teubner, 2011,
ISBN 978-3-834815781

Literaturverzeichnis - 3

11. Helmut Balzert: Lehrbuch der Software-Technik: Software-Entwicklung, Spektrum, 2001, **ISBN 978-3827404800**
12. Gernot Starke: Effektive Softwarearchitekturen. Ein praktischer Leitfaden, Hanser, 2011, **ISBN 978-3446427280**
13. Bernd Oestereich: Analyse und Design mit der UML 2.5: Objektorientierte Softwareentwicklung, Oldenbourg Wissenschaftsverlag, 2012, **ISBN 978-348671667**
14. Stephan Schmidt: PHP Design Patterns, O'Reilly, 2009, **ISBN 978-3897218642**
15. Stephen John Metsker: Design Patterns in C# (Software Patterns Series), Addison-Wesley, 2004, **ISBN 978-0321718938**
16. Scott Millett: Professional ASP.NET Design Patterns, Wiley (Wrox), 2010 **ISBN 978-0470292785**