Lösungen 1798

Achtung: Ein Teil des Kurstextes wurde überarbeitet, daher können einige Aufgaben oder Lösungen nicht mehr zu jeweils aktuellen Textversionen passen.

Aufgabe 1: Definition und Strukturen der Softwarearchitektur (6 + 6 Punkte)

a) In der Kurseinheit 1 des Kurses "Softwarearchitektur" wurden drei Definitionen für Softwarearchitektur angegeben. Schreiben Sie eine Definition für Softwarearchitektur stichpunktartig auf.

b) Auf welchen Strukturen beruht die Architektur eines komplexen Softwaresystems? Nennen Sie vier Strukturen und beschreiben Sie sie mit einem Satz.

Lösung 1: Definition und Strukturen der Softwarearchitektur (6 + 6 Punkte)

a) Im Kurstext wurden drei Definitionen aus der Literatur angegeben:

1. Unter der Softwarearchitektur eines Programms oder Computersystems versteht man die Struktur oder die Strukturen des Systems, die Softwarekomponenten, die von außen sichtbaren Eigenschaften dieser Komponenten und die Beziehungen zwischen ihnen umfassen.
   Zu den von außen sichtbaren Eigenschaften gehören zum Beispiel bereitgestellte Dienste (provided services), Performance-Eigenschaften, gemeinsame Nutzung von Ressourcen und Fehlerbehandlung.
   Aus dieser Definition folgt:
   a) Die Architektur definiert Komponenten.
   b) Das System kann mehr als eine Struktur enthalten.
   c) Jedes Software-System hat eine Architektur.
   d) Das Verhalten jeder Komponente ist Teil der Architektur, wenn es von einer anderen Komponente aus beobachtet oder wahrgenommen werden kann.
2. Die Architektur eines Softwaresystems definiert das System mittels Komponenten (computational components) und Interaktionen zwischen diesen Komponenten. Zu Komponenten gehören Klienten und Server, Datenbanken, Filter und Schichten in einem hierarchischen System. Interaktionen können Prozeduraufrufe und Zugriff auf gemeinsame Variablen sein, aber auch Client-Server-Protokolle, Datenbankzugangsprotokolle, asynchrone Ereignisbehandlung (event multicast) und Streambehandlung (piped streams). Die Architektur zeigt zusätzlich zur Struktur und Topologie des Systems die Zusammenhänge zwischen den Systemanforderungen und den Elementen des konstruierten Systems, wobei sie Gründe für die Designentscheidung liefert. Auf der Architekturebene sind Eigenschaften wie Kapazität, Durchsatz, Konsistenz und die Kompatibilität der Komponenten relevante Eigenschaften.
3. Eine Softwarearchitektur ist eine Beschreibung der Subsysteme und Komponenten eines Softwaresystems und deren Beziehungen. Subsysteme und Komponenten werden typischerweise in verschiedenen Sichten (views) spezifiziert, um die wesentlichen funktionalen und nichtfunktionalen Eigenschaften eines Softwaresystems zu zeigen. Die Softwarearchitektur ist das Ergebnis des Softwaredesigns.

b)
- *Konzeptionelle Struktur (conceptual structure)*: Ihre Komponenten sind Abstraktionen der funktionalen Anforderungen des Systems. Diese Abstraktionen werden mittels der Tausche-Daten-mit-Beziehung verbunden.
- *Datenfluss (data flow)*: drückt die Sende-Daten-zu-Beziehung auf Systemkomponenten verschiedener Abstraktionsebenen aus.
- *Kontrollfluss (control flow)*: drückt die Wird-aktiv-nach-Beziehung auf Systemkomponenten verschiedener Abstraktionsebenen aus.
- *Hierarchische Struktur (hierarchical structure)*: beschreibt eine hierarchische Strukturierung des Systems in Untersysteme mittels einer Ist-Subsystem-von-Beziehung. Am Ende der Hierarchie stehen Programm-Module.
- *Benutzungs/Aufrufstruktur (uses/call structure)*: Die Komponenten sind Programmmodule, Klassen, Prozeduren. Die Struktur drückt die Beziehung zwischen Moduln aus, die Benutzungsrelation bei Klassen, die Aufrufrelation bei Prozeduren.
- *Prozesstruktur (process structure)*: Die Komponenten sind Prozeduren und Threads. Typische Beziehungen sind synchronisiert-mit, läuft-nicht-ohne, beginnt, endet,etc.
- *Physikalische Struktur (physical structure)*: beschreibt die Verteilung von Code und Daten an die zugrunde liegenden Plattformen.

Aufgabe 2: Softwaresysteme und Architektur (6 Punkte)

Geben Sie ein Beispiel für eine 3-Lagen-Architektur (three tier architecture) an und beschreiben Sie die drei Lagen.

Lösung 2: Softwaresysteme und Architektur (6 Punkte)

Die 3-Lagen-Architektur wird oft verwendet, wenn ein effizientes verteiltes Client/Server-Design benötigt wird. Beispiele dafür sind Internetanwendungen und Informationssysteme für das Internet.
Im Kurstext wurde das WebAssign-System vorgestellt.

- 1. Lage (1. tier):
  Benutzerschnittstelle (user system interface): Benutzer des Systems sind Studenten und lehrende Assistenten. Sie kommunizieren mit dem System über Webseiten.
- 2. Lage (2. tier)
  Prozessverwaltung (process management): Der WebAssign-Anwendungsserver (application server) ist die zentrale Ausführungseinheit. Er akzeptiert Benutzeranfragen, führt die angeforderten Dienste aus und gibt Ergebnisse der Dienste in Form von Webseiten zurück, wenn das möglich ist. Er kommuniziert mit der zugrunde liegenden Datenbank und externen Servern.
- 3. Lage (3. tier):
  Datenbankverwaltung (database management): Sie kommuniziert mit einem externen Datenbankverwaltungssystem.

Aufgabe 3: Aspekt eines Softwaresystems (12 Punkte)

Was versteht der Kurs Software-Architektur unter einem Aspekt?
Nennen Sie drei funktionale Aspekte und sieben nichtfunktionale Aspekte.
Diskutieren Sie, wo man die Fehlerbehandlung einordnen würde.

Lösung 3: Aspekt eines Softwaresystems (12 Punkte)

Ein Aspekt eines Softwaresystems ist eine Eigenschaft oder Anforderung, die sich auf das ganze System bezieht. Man unterscheidet funktionale und nichtfunktionale Aspekte.
Funktionale Aspekte sind:

- Eigenschaften der partiellen Korrektheit
- Termination
- Lebensdauer (liveness)
- Freiheit von Deadlocks

Nichtfunktionale Aspekte sind:

- Performance
- Sicherheit
- Verfügbarkeit (availability)
- Verlässlichkeit (reliability)
- Testbarkeit
- Modifizierbarkeit
- Wiederverwendbarkeit
- Skalierbarkeit
- Portierbarkeit
- Benutzerfreundlichkeit (usability)

Die Fehlerbehandlung ist eng mit Verlässlichkeit verbunden, aber sie umfasst auch funktionales Verhalten, da sich funktionale Aspekte mit dem korrekten Verhalten des Systems beschäftigen.

Aufgabe 4: Sichten (6 + 2 + 7 + 6 Punkte)

a) Nennen Sie die vier Sichten des 4-Sichten-Modells (4-view-model).

b) Erklären Sie mit einem Satz den Unterschied zwischen dynamischen und statischen Strukturen.

c) Beschreiben Sie vier mögliche Beziehungen zwischen dynamischer und statischer Sicht, indem Sie in einer Tabelle vier Elemente der dynamischen und der statischen Strukturen gegenüberstellen.

d) Im Kurstext wurden zwei Beispiele behandelt, ein einfacher Web-Browser und eine Bankanwendung. Diese wurden mittels verschiedener Sichten dargestellt. Anbei finden Sie die zugehörigen Graphiken. Geben Sie zu jeder Grafik an, was sie für eine Sicht/Struktur darstellt. Welche dieser Strukturen sind dynamische Strukturen, welche statische?

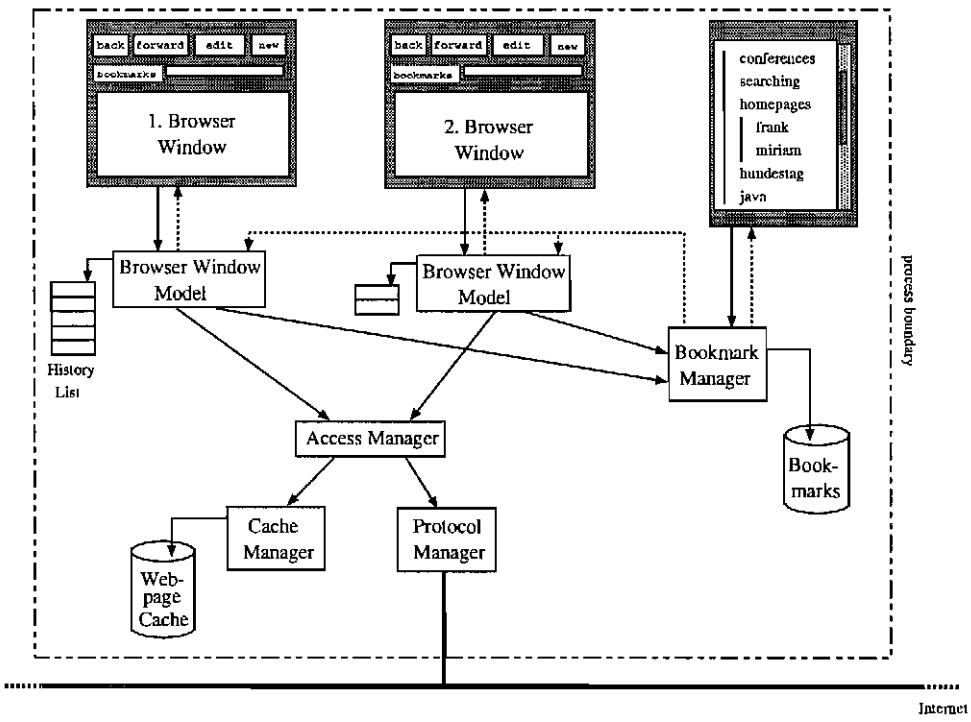


Abbildung 1:

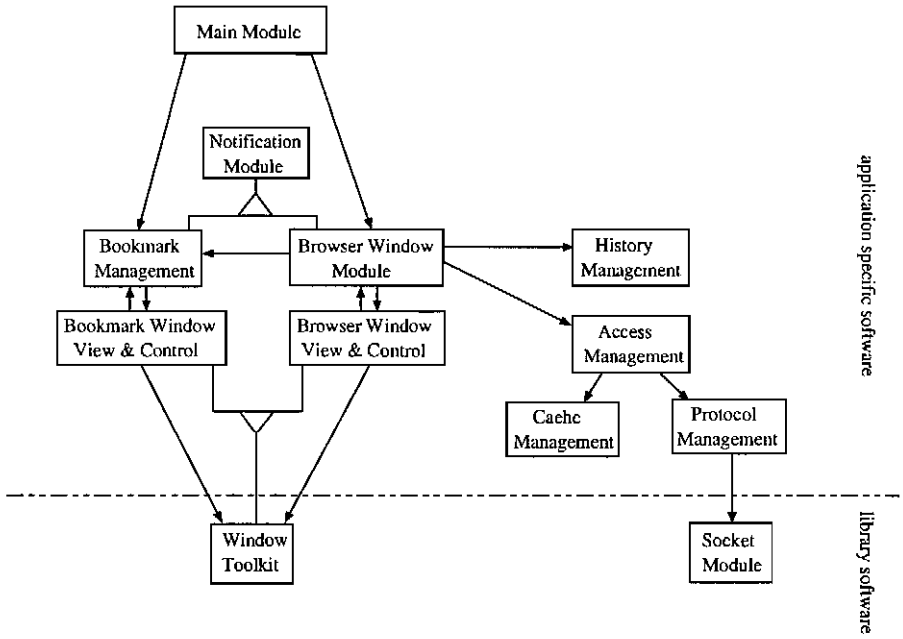


Abbildung 2:

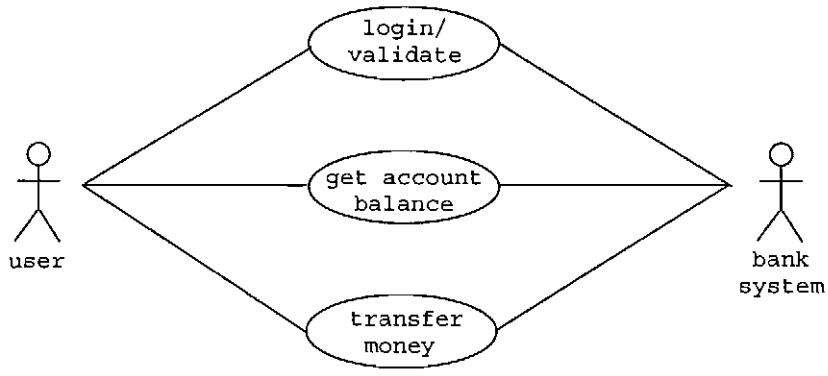


Abbildung 3:

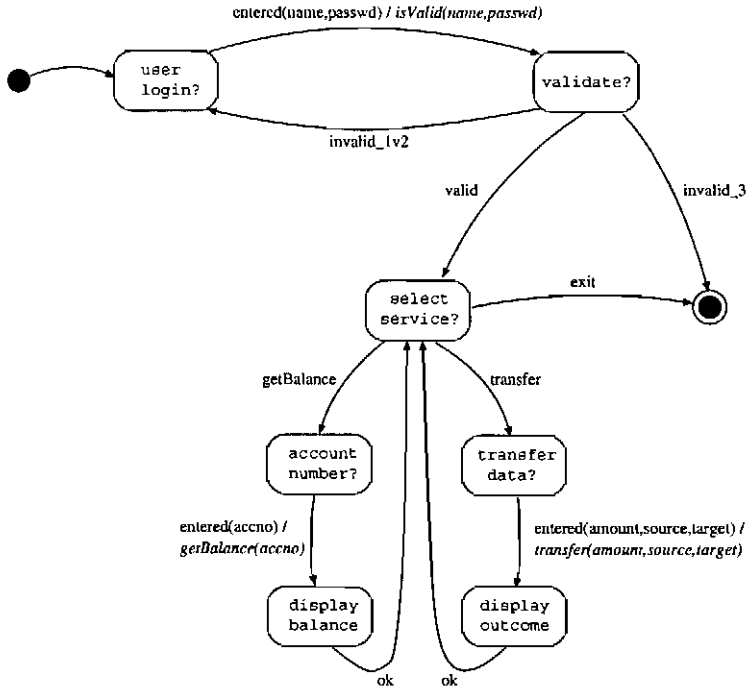


Abbildung 4:

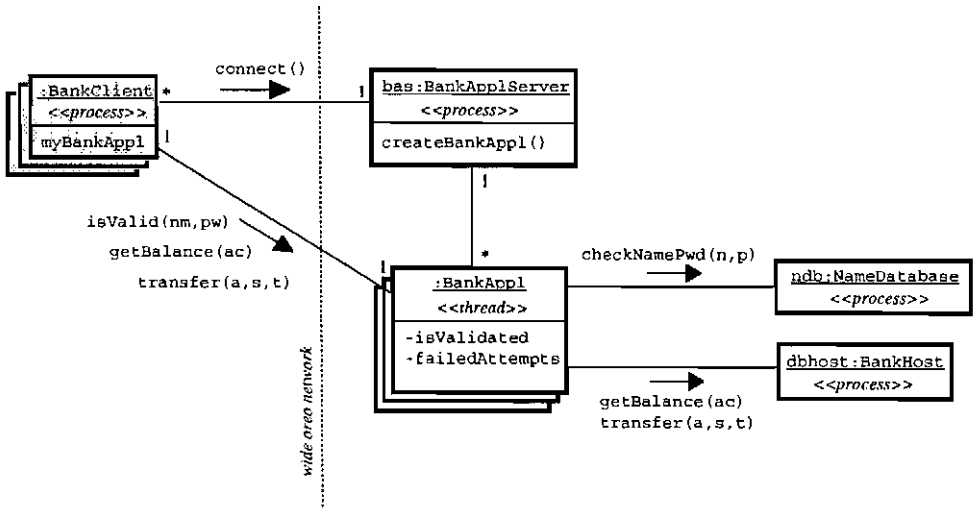


Abbildung 5:

Lösung 4: Sichten (6 + 2 + 7 + 6 Punkte)

a) Bei diesem Modell unterscheidet man die

- konzeptuelle Sicht (conceptual view): sie beschreibt das System mit Hilfe seiner Hauptdesignelemente und deren Beziehungen.
- Modulsicht (module view): sie enthält die funktionale Zerlegung und die Schichten.
- Ausführungssicht (execution view): sie beschreibt die dynamische Struktur des Systems.
- Code-Sicht (code view): sie beschreibt, wie Quellcode, Binärdateien und Bibliotheken in der Entwicklungsumgebung organisiert sind.

b) Die dynamische Sicht beschäftigt sich mit den Laufzeitstrukturen, die statische Sicht mit Software und Code-Strukturen.

c) Beziehung zwischen dynamischer und statischer Sicht:

Elemente der dynamischen Struktur	Elemente der statischen Struktur
Prozess	Programm
Objekt	Klasse
Prozedurinkarnation	Prozedur
Zustand	Variablen/files/execution points/...
Protokoll	??
Ereignis	?? (Prozedur)
??	Module/Pakete

d)
- Abb.1: Konzeptstruktur (Conceptual Structure) eines Browsers
- Abb.2: Modulstruktur eines Browsers
- Abb.3: Use Case Diagram in UML: Die Grundaufgaben der Bankanwendung
- Abb.4: Zustandsübergangssystem (State Transition System) der Bankanwendung
- Abb.5: Prozessstruktur der Bankanwendung

Die Modulstruktur gehört zur statischen Sicht, alle anderen Sichten zur dynamischen Sicht.

Aufgabe 5: Architekturmuster (Architectural patterns) (8 + 5 + 3 Punkte)

a) Diskutieren Sie die erwünschten Eigenschaften der Schichtenarchitektur (Layered Organization).
Welche Eigenschaft dieses Musters kann zu unnatürlichem oder ineffizientem Design führen?

b) Beschreiben Sie das Warenlagermuster (repository pattern).
Welche zwei Spezialisierungen spielen in der Praxis eine wichtige Rolle?

c) Welche Architekturentscheidungen legt das AWT fest, welche nicht?

Lösung 5: Architekturmuster (Architectural patterns) (8 + 5 + 3 Punkte)

a)
- Schichten unterstützen ein auf wachsenden Abstraktionsebenen basierendes Design. Dies erlaubt es den Implementierern, ein komplexes Problem in eine Folge inkrementeller Schritte aufzuteilen.
- Sie unterstützen Erweiterung: Da jede Schicht nur Wechselwirkungen auf die Schichten darüber und darunter hat, können Änderungen sich nur auf zwei andere Schichten auswirken.
- Sie unterstützen Wiederverwendung: Da sie klare Schnittstellen zwischen den Schichten unterstützen, erlauben sie verschiedene Implementierungen derselben Schicht, die austauschbar sind. Sie ermöglichen es, Schnittstellen zu standardisieren, so dass sie von verschiedenen Verkäufern implementiert werden können.

Diese strikte Strukturierung ist gleichzeitig Vorteil und Nachteil. Speziell die Einschränkung, dass Kommunikation nur mit den direkten Nachbarschichten erfolgen soll, kann zu unerwünscht kompliziertem Design führen.

b) Eine Warenlager-Architektur besteht aus einer zentralen Speicherkomponente, auf die von verschiedenen Klienten zugegriffen wird. Die Speicherkomponente wird zur Speicherung von gemeinsamen Daten genutzt und ermöglicht asynchrone Kommunikation zwischen den Kunden: Die von dem einen Kunden gespeicherten Daten kann später ein anderer Kunde lesen.
Das Muster legt nicht fest, wer die Aktivität innerhalb des Systems anstößt und kontrolliert.
Spezialisierungen:
- Muster für Datenbanksysteme
- Muster der Tafel-Architektur (blackboard architecture)

c) Das AWT legt die Grundlagen der Architektur für die Konstruktion von Sichten fest sowie den Ereignisbehandlungsmechanismus.
Es lässt offen, wie man Sichten und Anwendungskern trennt.

Aufgabe 6: Tiny Architectural Framework (TAF) (15 Punkte)

Die mit einem Editor erstellte Klausuranmeldung zum Kurs 1798 muss ein bestimmtes Format haben, um weiterverarbeitet werden zu können. Vereinfacht gehen wir davon aus, dass die Anmeldung aus

```
Klausurort Matrikelnummer Nachname Vorname
```

besteht, also z.B.

```
Hamburg 12345678 Boss Hugo
```

oder

`Rom 87654321 Borgia Lucrezia,`
wobei in den Namen keine Sonderzeichen oder Umlaute vorkommen dürfen und die Matrikelnummer achtstellig ist.
Ein Programm soll diese Anmeldung entgegennehmen und sie auf Korrektheit überprüfen.

a) Welche Teilaufgaben können Sie dafür identifizieren? Für jede der Teilaufgaben soll eine eigene Komponente zuständig sein.

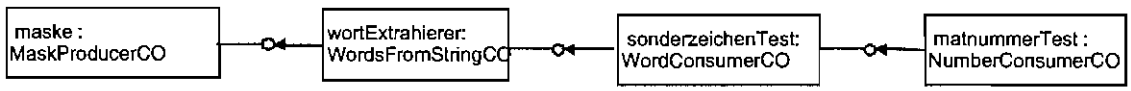
b) Stellen Sie die Topologie der Architektur grafisch mit Hilfe des im Kurs eingeführten TAF-Frameworks dar.
Stellen Sie die benötigten Komponentenschnittstellen jeweils auch einzeln grafisch dar zusammen mit den für eine Verbindung benötigten Methoden (link method).
Deklarieren Sie für die Ports der verschiedenen Komponenten geeignete Java Interfaces.

Lösung 6: Tiny Architectural Framework (TAF) (15 Punkte)

a) Man kann die folgenden Teilaufgaben identifizieren:

- Maske, mit der die Anmeldung erstellt wird (*Maske*)
- Zerlegen der Zeile in vier Wörter (*Wort-Extrahierer*)
- Test der Wörter auf Sonderzeichen (*Sonderzeichentest*)
- Test der Matrikelnummer auf Korrektheit (*Matrikelnummertest*)

b) Die Topologie der Architektur kann mit Hilfe von TAF wie folgt dargestellt werden:

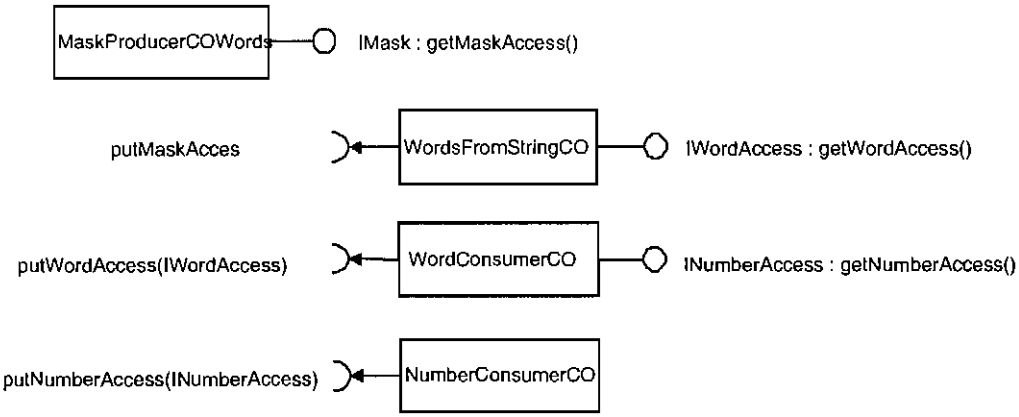


Die Komponentenschnittstellen und die zu deren Ports gehörenden Java Interfaces sehen wie folgt aus:

```
interface IMask {
  String getMask();
}

interface IWordAccess {
  String[] getWords ();
}

interface INumberAccess {
  String getNumber ();
}
```



Aufgabe 7: Programm- und Architekturrahmenwerk (5 + 6 Punkte)

a) Was ist der Unterschied zwischen einem Programmrahmenwerk und einer Menge von Bibliotheksklassen?

b) Was ist ein Architekturrahmenwerk?

Lösung 7: Programm- und Architekturrahmenwerk (5 + 6 Punkte)

a)
- Die Klassen eines Programmrahmenswerks sollen zusammenarbeiten, um die Funktionalität von komplexen Teilen des Softwaresystems zu erzeugen, die nicht durch eine einzelne Klasse ausgedrückt werden kann.
- Die Klassen sollen eine Kernfunktionalität bereitstellen, um verschiedene konkrete Systeme implementieren zu können.
- Ein Programmrahmenwerk beruht oft auf Architekturmustern, so etwa der GUI-Baukasten auf dem Model-View-Controller-Muster.

b)
- Eine Sammlung von Sprachen zur Architekturbeschreibung mit wohldefinierten Semantiken. Die Semantiken sollen auch die Beziehungen zwischen den Sprachen erklären. Eine Sprache darf verschiedene Notationen unterstützen, z.B. textorientierte und graphische Notationen.
- Eine Sammlung von Stilen, Regeln, Invarianten und Eigenschaften, die die Verwendung der Sprache und ihrer Sätze regelt.
- Eine Definition der wesentlichen Elemente einer Architekturbeschreibung, d.h. das Rahmenwerk soll uns mitteilen, was wir liefern müssen, um eine vollständige Architekturbeschreibung zu geben.
- Eine Definition, was eine Implementierung einer Architekturbeschreibung ausmacht.
- Optional: Eine Sammlung von vordefinierten Elementen, Diensten, Standards und Werkzeugen (oder ihren Spezifikationen)

Aufgabe 8: COM und Enterprise JavaBeans (10 + 4 + 10 Punkte)

a) Erläutern Sie die in Abbildung 6 gegebene Grafik: Was ist eine COM-Klasse, was ein COM-Objekt? Wie könnte die Implementierung der Schnittstellen aussehen? Wie könnte die Implementierung der Klasse C1A aussehen?

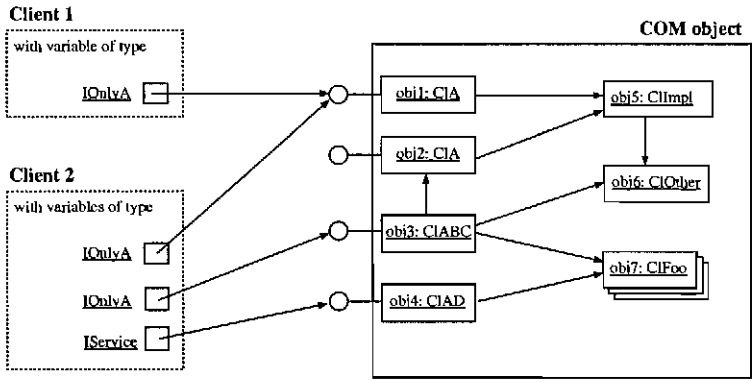


Abbildung 6: COM object as a collection of pure objects

b) Wofür verwendet man Enterprise JavaBeans, wofür COM?

c) Erläutern Sie Abbildung 7: Was ist der Unterschied zwischen Session-Beans und Entity Beans? Wofür stehen Home und Remote? Was ist der Behälter?

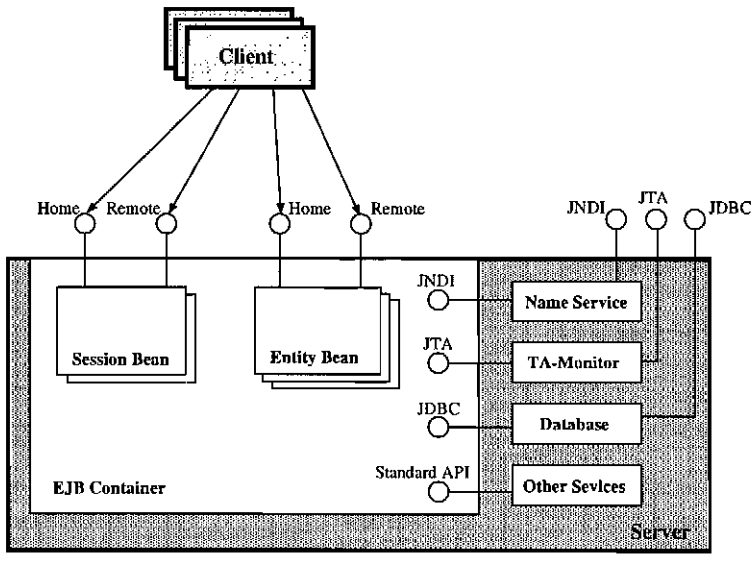


Abbildung 7: The principal architecture of an EJB based system

Lösung 8: COM und Enterprise JavaBeans (10 + 4 + 10 Punkte)

a) Eine COM-Klasse ist die Software, die eine Komponente implementiert, ein COM-Objekt ist eine Instanz einer solchen Komponentenklasse. Ein COM-Objekt muss kein reines Objekt im Sinne einer objektorientierten Sprache sein, es kann sich aus mehreren solcher reiner Objekte mit verschiedenem Typ oder aus anderen Datenstrukturen zusammensetzen. Ein COM-Objekt wird wie ein Objekt erzeugt, kann einen Zustand haben, liefert einen Mechanismus, um es auf Identität zu prüfen, ist nur über wohldefinierte Schnittstellen zugreifbar.
Ein COM-Objekt unterstützt seine Dienste, indem es Schnittstellen implementiert. Eine Schnittstelle hat einen Namen und beschreibt die Signatur der Prozeduren/Methoden, die von Objekten, die die Schnittstelle implementieren, bereitgestellt werden.
In der Grafik sieht man ein COM-Objekt, das aus den reinen Objekten obj1 bis obj7 zusammengesetzt ist und die Schnittstellen IOnlyA und IService hat. Auf die sog. Schnittstellenobjekte obj1 bis obj4 kann von aussen zugegriffen werden. Sie erlauben es Kunden des COM-Objektes, auf Dienste zuzugreifen oder seinen Zustand zu ändern.

```
interface IOnlyA extends IUnknown {
   int doOpA(...);
}


interface IService extends IUnknown {
   int doOpB(...);
}
```

Die Klasse C1A könnte so aussehen:

```
class C1A extends C1Unknown implements IOnlyA {
   int doOpA(...) {...}
}
```

b) COM ist ein Rahmenwerk, um Softwaresysteme auf Binärebene zusammenzusetzen. Insbesondere können Komponenten, die in unterschiedlichen Programmiersprachen geschrieben sind, miteinander kooperieren.
COM kann in allen Bereichen genutzt werden, in denen es Sinn macht, Softwaresysteme aus Komponenten zusammenzusetzen.
Das Enterprise JavaBeans-Rahmenwerk unterstützt die Realisierung von Applikationsservern gemäß dem 3-Lagen-Architekturmuster (three tier architectural pattern).

c) Die Serverseite besteht aus einem EJB-Behälter und Diensten. Der EJB-Behälter enthält und verwaltet die Komponenten des EJB-Rahmenwerks, die sog. Enterprise JavaBeans. Es gibt Session Beans und Entity Beans.

Entity Beans stellen im Wesentlichen die Datenobjekte dar, mit denen das System umgeht.
Session Beans werden verwendet, um Modellierungsaufgaben, Prozesse oder das Verhalten des Applikationsservers zu modellieren. Ein Session Bean entspricht einem Dienst für einen Klienten.
Die Beans stellen das Home- und das Remote-Interface bereit. Klienten kommunizieren mit den Beans über die Home und Remote- Schnittstellen.
Das Remote Interface definiert die Methoden, die das Entity Bean dem Klient bereitstellt. Es spiegelt die Funktionalität des Beans wider.
Das Home Interface definiert die Methoden, die den Lebenszyklus des Beans betreffen. Der Klient benutzt diese Schnittstelle, um Bean-Instanzen zu erzeugen, sie zu finden und zu entfernen.
Die Bean-Klasse stellt Implementierungen für die Methoden von Home - und Remote-Interface bereit.
Die Beans können auf einen Namensdienst über das Java Naming and Directory Interface (JNDI), auf Transaktionsmonitore über das Java Transaction API (JTA), auf Datenbanken über das Java Database Connectivity API (JDBC) und auf andere Dienste über das Java Standard API zugreifen. Die meisten dieser Dienste sind auch direkt vom Server erhältlich.