Modul Application Architecture
Musterlösung zur Wiederholungsprüfung

Name: _____ Note: _____

Nr.	Aufgabe	Max. Punkte	Erreichte Punkte
1	Methodenelemente (NFRs, CRC, SCD/AOD)	20	
2	Verständnisfragen zu Layers, Tiers, Integration	18	
3	Architektur- und Integrationspatterns (DDD, EIP, PoEAA)	20	
4	JEE-Konzepte und JEE-APIs	14	
5	API Design & Management, REST, Web Services	14	
6	Container und Aspect-Oriented Programming	14	
7	Analyse und Design mit Patterns und Technologien	20	
		Total 120	

Hinweise allgemein:
- Zeit: 120 min, d.h. ein Punkt entspricht ca. einer Minute Lösungsaufwand.
- Mit Ausnahme von Prüfungen und Musterlösungen aus früheren Semestern sind alle gedruckten Unterlagen erlaubt (Open Book).
- Es sind keine elektronischen Hilfsmittel zugelassen; auch kein Handy oder Smart Watch.
- Alle Antworten und Lösungen in diese Aufgabenblätter schreiben.
- Keinen Bleistift und keine rote Farbe verwenden.
- Die Antworten und Lösungen müssen nachvollziehbar sein. Bitte alles in leserlicher Schrift!
- Keine Fragen in der Prüfung. Bei Unklarheiten sind sinnvolle Annahmen zu treffen und diese klar zu dokumentieren.
- Die Bewertungspunkte der Aufgaben können nach der Korrektur noch etwas variieren.

Hinweise zum Prüfungsbeginn:
- Versehen Sie das Titelblatt mit Ihrem Namen.
- HSR-Badge oder pers. Ausweis (ID/Führerausweis mit Foto) sichtbar auf den Tisch legen.
- Allfällig bereitliegende Prüfung nicht umdrehen oder umblättern, bis die Aufsichtsperson explizit das Ok dazu gegeben hat.

Hinweise zur Prüfungsabgabe:
- Kontrollieren Sie nochmals, ob die Prüfung mit Ihrem Namen versehen ist. Geben Sie alle Blätter – auch von nicht gelösten Aufgaben – geordnet und geheftet ab.
- Bis 20 min. vor Prüfungsende: Geben Sie ein Zeichen und lassen Sie sich die Prüfung von der Aufsichtsperson abholen. Verlassen Sie dann leise den Raum.
- Ab weniger als 20 min. vor Prüfungsende: Drehen Sie Ihre Prüfung um und warten Sie, bis die Prüfung eingesammelt wird. Bleiben Sie dann sitzen, bis die/eine Aufsichtsperson ein Zeichen zum Verlassen des Raums gegeben hat.

Aufgabe 1: Methodelemente (SMART NFRs, CRC, SCD/AOD) (20 Punkte)

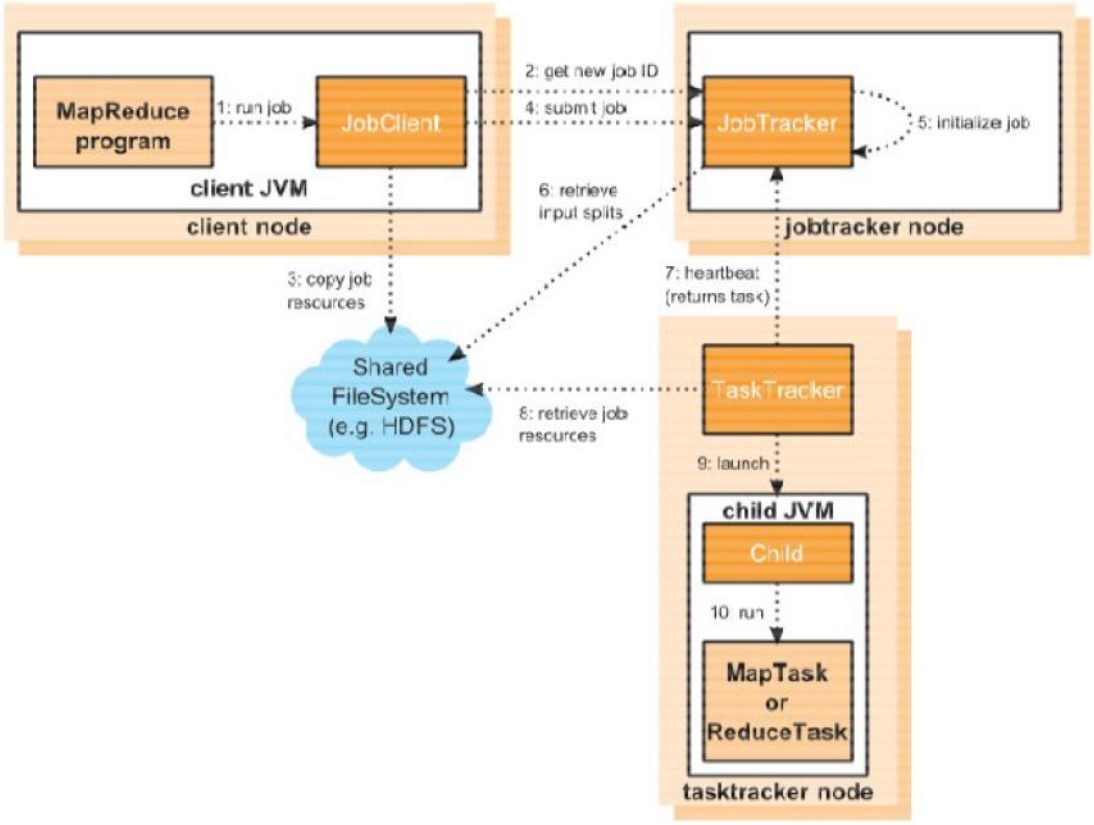
a) Sind die folgenden NFRs SMART (hier: nur Specific und Measurable) definiert? Falls Sie im Zweifel sind, können sie eine Erklärung ergänzen bzw. Annahmen treffen. Antwortformat: Ja/Nein.

NFRs	S(pecific)	M(easurable)
Die Komponente „Account Management Server“ in der Anwendung „Supply Chain Management for Factory 1“ muss hochverfügbar ausgelegt sein; die Verfügbarkeit soll 99.5% pro Jahr betragen. Geplante Wartungsarbeiten, die die Verfügbarkeit beeinträchtigen, sollen nur an Wochenenden durchgeführt werden.	*Ja*	*Ja*
Die Benutzerfreundlichkeit, gemessen in Anzahl Fenster und Benutzerinteraktionen pro Use Case-Realisierung, soll so sein, dass alle aktuellen und zukünftigen Use Cases in einer zumutbaren Zahl von Eingabeschritten durchlaufen werden können.	*Nein („alle aktuellen und zukünftigen Use Cases“ ist zu weich formuliert)*	*Nein (was ist „zumutbar“?)*
Die Antwortzeit im Rich Client Channel der Web-Anwendung darf 3 Sekunden nicht überschreiten in 75% aller Use Case Walkthroughs; in den verbleibenden 25% der Fälle erfolgt die Antwort innerhalb von 8 Sekunden.	*Nein, „Rich Client Channel der Web-Anwendung“ ist nicht präzise genug (welche Anwendung?)*	*Ja*
Auf hohe Wartbarkeit der Implementierung des Use Cases „Create Account“ in der neuen Core Banking Anwendung legen wir als Entwicklungsteam grossen Wert.	*Ja*	*Nein*
Das Order Entry System soll zahlreichen gleichzeitigen Benutzern das Browsen im Produktkatalog erlauben, ohne dass sich das Antwortzeitverhalten signifikant verschlechtert.	*Ja (Anwendung und Use Case bzw. Feature angegeben)*	*Nein (was ist „signifikant“?)*

b) Ausgangssituation: Gegeben sei ein Architekturdiagramm zu der Implementierung des Map-Reduce-Patterns im Open Source Projekt Apache Hadoop (siehe folgende Seite)

Nicht im Architekturdiagramm eingezeichnet, aber ebenfalls spezifiziert ist die Aufgabe und Fähigkeit der Komponente JobTracker, sich Jobs mitsamt deren Ausführungszuständen zu merken.

Aufgabe: Erstellen Sie für die Komponente Job Tracker eine CRC-Karte gemäss Template aus der Vorlesung. Nennen Sie dabei mindestens drei Responsibilities, zwei Collaborations und einen Known Use (also den Namen eines Produktes oder einer Open Source Software).



Component: Job Tracker	
Responsibilities: • *Assigns job ids* • *Initializes jobs that get submitted* • *Responds to heartbeat requests* • *Keeps track of submitted jobs*	*Collaborations (Interfaces):* • *Job Client* • *Task Tracker* • *HDFS*
Known uses (implementations): • *Apache Hadoop*	

c) Erklären Sie die Aufgabe des System Context Diagrams (SCD) und des Architecture Overview Diagrams (AOD) und mindestens zwei Unterschiede zwischen diesen beiden Methodenelementen und Diagrammtypen.

Aufgaben: Vertragsverhandlungen zum Projekt-Scope, Design, Reviews, Test (z.B. Integration, Interoperability); Unterschiede: Innen- vs. Aussensicht; Detaillierungsgrad; White Box vs. Black Box

Aufgabe 2: Verständnisfragen zu Layers, Tiers, Integration (18 Punkte)

Stimmen die folgenden Aussagen? Tragen Sie wahr oder falsch ein und begründen Sie Ihre Antwort mit 2-3 Stichworten. Hinweise zum Ausfüllen:

- Falls die Antwort einer Vorlesungsfolie direkt entnommen werden kann, reicht der Verweis auf die Folie (in der Syntax „Lektion x, Folie y"). Man kann aber auch anders begründen, der Folienverweis ist *nicht* explizit gefordert.
- Falls eine verallgemeinernde Aussage getroffen wird, reicht ein Gegenbeispiel, um diese zu falsifizieren. Bei wahren Aussagen kann ein Beispiel ebenfalls als Begründung dienen.
- Falls nur ein Teil der Aussage stimmt, ist die Gesamtaussage als falsch einzustufen; der falsche Teil der Aussage kann gekennzeichnet (und begründet) werden.

Aussage	**(W/F)**	**Begründung (Ausführung, Erklärung)**
Die PCs, von denen aus mit Internet-Browsern auf Servlets in einer Web-Anwendung zugegriffen wird, bilden keinen Tier im Sinne von Two-Tier- und Three-Tier-Architekturen.	*F*	*Siehe VL-Folien aus Semesterwoche 2 und 3: Remote-Kommunikation findet statt zwischen Client Tier PCs und Mid Tier (hier über HTTP, Thin Browser Client).*
Wenn die Software innerhalb eines logischen Layer aktualisiert wird, hat dies immer dann Auswirkungen auf die umliegenden Layer, wenn die Schnittstelle betroffen ist; dies ist aber nicht immer der Fall.	*W*	*Bsp. ein einfaches layerinternes Refactoring ohne Schnittstellenänderung und ohne Einfluss auf QoS/SLA hat keine Auswirkungen, im Unterschied zu Schnittstellenänderungen*
Ein logischer Layer kann auf mehrere physische Tiers verteilt werden.	*W*	*Siehe VL-Folien aus Semesterwoche 2 und 3: Der Presentation Layer ist bei Three-Tier-Architekturen oft aufgeteilt, auch der Business Logic Layer kann auf mehrere Tiers aufgeteilt werden.*
Die Kommunikation zwischen logischen Schichten in einer Architektur kann nicht aus einfachen lokalen Java-Methodenaufrufen bestehen, sondern erfordert Remoting,	*F*	*Das ist eines der definierenden Merkmale der Definition bzw. Abgrenzung Layer vs. Tier. Siehe Übungsbeispiele und DDD Sample Anwendung.*
Wenn man sich für Domain-Driven Design (DDD) im Business Logic Layer entscheidet, sollte man die PoEAA-Patterns für Presentation Layer und Data Access (Persistence) Layer nicht verwenden, um die konzeptionelle Integrität der Anwendungsarchitektur nicht zu gefährden; eine Patternsprache pro Projekt reicht normalerweise aus.	*F*	*Die Entkopplung der Schichten ist ein Ziel des Layers Patterns; für die Kombination von PoEAA und DDD siehe z.B. DDD Sample Application (verwendet Hibernate als Data Mapper).*
Ein Grund für die Umstellung von einer Two-Tier-Architektur auf eine Three-Tier-Architektur kann die bessere Skalierbarkeit sein: man hat mehr Deployment Design-Optionen (ist aber auch mit zusätzlichem Bedarf für Remote-Kommunikation konfrontiert).	*W*	*Ein Mid-Tier kann mit Unterstützung von Middleware wie JEE-Applikationsservern hochperformant und skalierbar ausgelegt werden (Grund für Einführung!).*
Man kann Spring Container nicht mitsamt ihrer Anwendungen (also den Spring-Beans) in JEE-	*F*	*Siehe DDD-Sample in der Übung.*

Applikationsserver wie IBM WebSphere und JBoss deployen.		
Den Java Enterprise Edition (JEE) Support für JEE APIs und Container Management für alle drei logischen Schichten aktiviert man, in dem man die Java Standard Edition (JSE) JDKs und JREs mit dem Command Line Argument (Parameter) `-DenableJEE` startet.	*F*	*JEE Application Server sind eigenständige Produkte oder Open Source Assets*
Ein API Gateway in einer Microservices-Architektur ähnelt dem allgemeineren Remote Facade Pattern und überlappt auch mit dem SOA Pattern Enterprise Service Bus (ESB), z.B. in den Bereichen Message Routing, Message Transformation (Mediation) und Protocol Adaptation.	*W*	*Siehe ESB-Gastlektion 7 sowie Analyse und Diskussion in Lektion 12 und 13 sowie zugehörige Übung (External ESB vs. Internal ESB).*

Aufgabe 3: Architektur- und Integrationspatterns (DDD, EIP, PoEAA) (20 Punkte)

a) Entscheiden Sie, ob die beschriebenen Eigenschaften auf Instanzen der beiden Patterns zutreffen (Antwortformat: „Ja“ oder „Nein; ergänzen Sie ggfs. Begründungen bzw. Erklärungen, falls Ihrer Meinung nach die Aussage nicht einfach mit „Ja“ oder „Nein“ beantwortet werden kann):

Eigenschaft	**~~Pattern 1: Transaction Script~~**	**Pattern 2: Domain Model**
Objektorientierung	*Nein*	*Ja*
Einfach zu designen und zu implementieren (nur eine Klasse und eine Methode pro Patterninstanz)	*Ja*	*Nein*
Objekte von Klassen, die dieses Pattern realisieren, werden direkt in einer relationalen Datenbank abgelegt	*Nein, kein Zustand*	*Nein, O/R Mapping erforderlich (diverse Patterns, Technologien und Frameworks zur Unterstützung dieser Abbildung)*
Geeignet zur Behandlung von anwendungsfachlichen Fehlern (Bsp. Logging invalider Eingangsdaten an der Schnittstelle zum Presentation Layer, Rethrow von technischen Exceptions aus dem Data Access Layer)	*Ja*	*Ja, manche Klassen*
Geeignet als Transaktionsgrenze	*Ja*	*Ja, allerdings nicht alle Klasen (vgl. Domain-Driven Design Patterns und PoEAA-Patterns wie Service Layer und Remote Facade vs. Data Transfer Object)*

b) Ordnen Sie die folgenden Beschreibungen einem Domain-Driven Design (DDD)-Pattern zu:

Beschreibung	**Pattern**
Erlaubt es an seiner Schnittstelle, Aggregates anzulegen, zu finden und zu löschen.	*Repository*
Stellt die Beziehungen zwischen Bounded Contexts dar (Beziehungstypen sind z.B. Shared Kernel, Customer-Supplier).	*Context Map*
Ist im Business Logic Layer angesiedelt, hat eine Identität und einen Lifecycle (also State und Behavior) und kann über DDD-Repositories gefunden werden.	*Entity*
Ist im Business Logic Layer angesiedelt, hat keine eigene Identität (z.B. beim Kopieren) und kein Behavior und kann in einer Remote-Schnittstelle daher gut als Parameter fungieren.	*Value Object (Data Transfer Object auch ok als Antwort)*
Hat einen globalen Identifier und dient als Einstieg in ein Aggegate, das weitere Entities und Value Objects enthalten kann.	*Root Entity*

c) Entscheiden Sie, ob die beschriebenen Eigenschaften auf die beiden Enterprise Integration Patterns (EIP) zutreffen (Ja/Nein).

Eigenschaft	**Recipient List**	**~~Transactional Client~~**
Unterstützt, wie alle Patterns, den plattformunabhängigen Wissensaustausch und das plattformunabhängige Design.	*Ja*	*Ja*
Ist eines der Message Transformation (Mediation) Patterns.	*Nein*	*Nein*
Besitzt eine eigene Klasse im JMS API.	*Nein*	*Nein*
Dient dazu, eine Nachricht einer dynamisch spezifizierten, also nicht statitsch festgelegten, Zahl von Empfängern zuzustellen (Routing).	*Ja*	*Nein*
Erlaubt einem Message Endpoint (also einem Client des Massaging Systems), die Transaktionsgrenze zu kontrollieren und ggfs. an verteilte Transaktionen teilzunehmen (Bsp. Lesen von Queue und Schreiben in Datenbank).	*Nein*	*Ja*

Nennen Sie drei besonders wichtige Vorteile (Stärken) von Message-Oriented Middleware (MOM) und zwei besonders gravierende Kritikpunkte an MOM (Schwächen). Nehmen Sie entweder Bezug auf die Literatur oder begründen Sie ihre Nennungen kurz.

- Vorteil:
  Loose Kopplung in der zeitlichen Dimension

- Vorteil:
  Einfache Skalierung z.B. durch Competing Consumers (Throttling)

- Vorteil:
  MOM-Provider bieten APIs in vielen Sprachen, JMX API unterstützt Portabilität von Java-Clients

- Nachteil:
  Erhöhter Designaufwand, z.B. Message Expiration und Dead Letter Queues, Message Content, Empfängertyp, Anzahl Empfänger

- Nachteil:
  Erhöhter Aufwand im Systemmanagement, noch schwache Broker-Interoperabilität

Aufgabe 4: JEE-Konzepte und JEE-APIs (14 Punkte)

Finden Sie für jede Anforderung mindestens ein JEE-Konzept und/oder mindestens ein API, die zur Realisierung der Anforderung bzw. zur Unterstützung des Architekturprinzips geeignet sind. Falls Ihrer Meinung nach kein derartiges Konzept bzw. API existiert, tragen Sie „n/a“ für „not applicable“ ein.

Anforderung	**JEE-Konzept**	**JEE-API**
Serverseitige Steuerung des HTML Page Flows	*Servlets im Web Container*	*Servlet API*
All-or-Nothing Semantik im Business Logic Layer	*Programmatische oder deklarative Festlegung der Transaktionsgrenzen*	*JTA, Transaktions-Annotations*
Wechsel der relationalen Datenbank ohne Änderung des eigengeschriebenen Zugriffscodes	*Produktneutraler Zugriff auf SQL-Tabellen per selbstgeschriebenem SELECT Statement;*	*JDBC , (JPA)*
Nichtblockierendes Aufruf-Verhalten (Komponente)	*Asynchrone Remote-Kommunikation (Messaging)*	*Message-Driven Beans, (JMS)*
Persistente Speicherung eines komplexen Domain Models	*Objektrelationale Mappings*	*JPA*

Nennen Sie zwei besonders praxisrelevante Vorteile von JEE (Stärken) und zwei Kritikpunkte an JEE (Schwächen). Nehmen Sie entweder Bezug auf die Literatur oder begründen Sie ihre Nennungen kurz.

- Vorteil:
  Managed Container, viele Cross Cuting Concerns z.B. in den Bereichen Management und Security müssen nicht im Projekt programmiert, sondern können konfiguriert werden

- Vorteil:
  Standardisierte APIs, Produkte im Markt vorhanden

- Nachteil:
  Technische Komplexität, Laufzeit-Footprint des Containers

- Nachteil:
  Spezifikation wIrd von Oracle (früher Sun) kontrolliert, also z.B. Gefahr von Vendor Lockin

Aufgabe 5: API Design & Management, REST, Web Services (14 Punkte)

Markieren Sie die richtige(n) Antwort(en) zu jeder Frage. Mindestens eine Antwort ist korrekt.

1. POINT steht für…
a. Protocol-Oriented (Web Services) Interface.
b. Public Organization and Integration of Nearby Tiers.
c. Published Ontology to Inject Native Transactions.
d. Keine der Alternativen a bis c. X

2. Eine Message-Driven Bean in einem JEE Application Server implementiert die folgenden EIPs…
a. Transactional Client. X
b. Event-Driven Consumer. X
c. Aggregator.
d. Content Filter.

3. Spezielle Message Router sind laut EIP…
a. Content-Based Router und Recipient List. X
b. Recipient List und Content Enricher.
c. Point-to-Point Channel und Selective Consumer.
d. Enterprise Service Bus und Command Message.

4. Ein Service Contract in einer SOA…
a. muss keinen fachlich-geschäftlichen, sollte aber einen technisch-syntaktischen Teil haben.
b. ist auch bei Nutzung des WWW als Middleware (RESTful HTTP) erforderlich. X
c. beschreibt im technischen Teil die Ausrufsyntax mit Parametern von Remote Service Interfaces (z.B. WSDL, Swagger). X
d. muss nicht maschinenlesbar sein, da nur Entwickler ihn lesen in Programmierung und Test.

5. Im Kontext von API Design & Management …
a. sind Evolvability und Compatibility wichtige Anfoderungen. X
b. kümmert sich die Messaging Middleware um die Überwindung von Formatunterschieden.
c. kann ein ESB benutzt werden, um Service Requests zu einer geeigneten Version des Service Providers zu routen. X
d. Keine der Alternativen a bis c.

6. Das Service Design Pattern „Dotted Line“ beschreibt die Verwendung …
a. mehrerer skalarer Parameter in der Schnittstellensignatur. X
b. genau eines skalarer Parameters in der Schnittstellensignatur.
c. mehrerer komplexer Parameter in der Schnittstellensignatur.
d. genau eines komplexen Parameters in der Schnittstellensignatur.

7. Web Services (WSDL, SOAP) und RESTful HTTP…
a. sind zwei von vielen Technologieoptionen, um verteilte Anwendungen zu integrieren. X
b. lösen völlig verschiedene Designprobleme (Backendintegration vs. Web Engineering).
c. sind zwei Architekturstile, die man nicht direkt miteinander vergleichen kann.
d. unterscheiden sich hinsichtlich Verbreitung, Reifegrad und Toolunterstützung. X

Aufgabe 6: Container und Aspect-Oriented Programming (14 Punkte)

Stimmen die folgenden Aussagen? Tragen Sie wahr oder falsch ein und begründen Sie Ihre Antwort mit 2-3 Stichworten. Hinweise zum Ausfüllen:

- Falls die Antwort einer Vorlesungsfolie direkt entnommen werden kann, reicht der Verweis auf die Folie (in der Syntax „Lektion x, Folie y“). Man kann aber auch anders begründen, der Folienverweis ist nicht explizit gefordert.
- Falls eine verallgemeinernde Aussage getroffen wird, reicht ein Gegenbeispiel, um diese zu falsifizieren. Bei wahren Aussagen kann ein Beispiel ebenfalls als Begründung dienen.
- Falls nur ein Teil der Aussage stimmt, ist die Gesamtaussage als falsch einzustufen; der falsche Teil der Aussage kann gekennzeichnet (und begründet) werden.

Aussage	**(W/F)**	**Begründung (Ausführung, Erklärung)**
Spring Boot Applikationen müssen im Unterschied zu JEE-Anwendungen nicht über WAR-Files auf einen Applikationsserver deployed werden.	*W*	*Spring Boot hat einen integrierten Webserver.*
Mit Spring-Boot Profilen lassen sich Properties setzen, um die Konfiguration der Applikation zu beeinflussen.	*W*	*Woche 8, Folie 16*
Dependency Injection (DI) in Java ist erst seit der Einführung von Annotationen möglich.	*F*	*Falsch, man kann auch manuell Dependencies injizieren (z.B. Dependencies im Konstruktor mitgeben)*
Inversion of Control bedeutet, dass der Code, den wir schreiben, die Kontrolle hat und bei Bedarf das Framework aufruft (z.B. wenn ein Request eintrifft).	*F*	*Genau umgekehrt, Woche 8, Folie 8*
AOP steht für Annotation Oriented Programming, ein Beispiel dafür ist Dependency Injection.	*F*	*Aspect Oriented Programming*
Die Laufzeiteigenschaften einer Datenbanktechnologie und einer Datenbank sind unabhängig von den umzusetzenden Use Cases.	*F*	*Woche 10, Folie 7*
Concurrency ist eine Aufgabe des Spring-Containers; dieser verwaltet einen Threadpool und weist jedem Request einen Thread zu.	*W*	*Woche 8, Folie 9*

Aufgabe 7: Analyse und Design mit Patterns und Technologien (20 Punkte)

Ausgangslage: Gegeben sei eine Web-Anwendung zur Verwaltung der Versicherten und Gelder von Pensionskassen. Die Anwendung unterstützt Pensionskassenmitarbeiter bei den folgenden Aufgaben (Use Cases):

1. Versichertenkonto erstellen
2. Einzahlungen einpflegen (Monatsabschluss)
3. Pensionskassenausweis drucken (Jahresabschluss)
4. Auszahlung prüfen und vornehmen
5. Arbeitgeber- und Pensionkassenwechsel vornehmen
6. Versichertenkonto auflösen

Die Aufgaben werden jeweils einmal im Monat bzw. Jahr durchgeführt; dabei kommt es aufgrund von Terminen (Deadlines) zu vorhersehbaren Lastspitzen. Die Performance insbesondere bei Monatsabschlüssen soll auch bei hohen Benutzerzahlen und vielen Zugriffen so sein, dass ein flüssiges Arbeiten möglich ist („Sub-Second Response Time“). Domain- und Datenmodell der Anwendung sind recht einfach strukturiert (5-10 Klassen bzw. Entities; kaum komplexe Objektreferenzen).

Die Anwendung soll als Tier 3 Java Web Anwendung realisiert werden, welche die folgenden Patterns und Technologien verwendet:

- Model-View-Controller (MVC) zur Organisation des Presentation Layers,
- Plain Old Java Objects (POJOs) für die Businesslogik-Schicht (Mitarbeiter, Firma, Versicherungsform und einige nicht näher spezifizierte Value Objects zur weiteren Umsetzung des Domain Models),
- direkter JDBC-Zugriff auf die relationale Datenbank Oracle 11g im Persistence Layer.

Aufgaben: a) Erstellen Sie ein vollständiges Quality Attribute Scenario (QAS) für das Quality Attribute Performance anhand der Angaben in der Ausgangslage.

Die Angaben aus dem SMART NFR in der Aufgabenstellung können analog zum QAS aus Übung 2 in die QAS-Struktur überführt und ggfs. ergänzt werden.

Scenario (Portion)	*Performance-QAS 1 (QAS-P1):* *„Die Antwortzeit in Rich Client/Web Channels für Use Cases 1 bis 4 in der Ausgangslage darf 2 Sekunden nicht überschreiten in 80% aller Use Case Walkthroughs; in den verbleibenden 20% der Fälle erfolgt die Antwort innerhalb von 7 Sekunden.“*
Source	*Ein externer Systembenutzer über die Kanäle aus der Ausgangslage, also Pensionskassenmitarbeiter.*
Stimulus	*Ausführung einer der Use Cases 1 bis 6 aus der Ausgangslage.*
Artifact	*Pensionskassen-Webanwendung, alle Tiers*
Environment	*Zur Laufzeit (normaler Betrieb), mindestens 80% der Use Case Walkthroughs*
Response	*Die Antwortzeit aus Sicht der externen Source (s.o.) soll gemessen werden, vom Absenden des Stimulus (s.o.) bis zum Empfang der ersten Antwort der Pensionskassen -Webanwendung (also dem Artifact). Bsp. Bestätigung des Eingangs eines Gebots an den Pensionskassenmitarbeiter im Web bzw. Mobile Channel.*
Response Measure	*Die Response (wie oben definiert) soll innerhalb von weniger als einer Sekunde eingehen – im Kontext von (Source, Stimulus, Artifact, Environment).*

b) Wählen Sie mindestens ein weiteres PoEAA- oder SOA-Pattern und eine Middleware für den Tier 2 der Anwendung (also insbesondere Business Logic Layer und/oder Data Access Layer). Begründen Sie eine Ihrer Architekturentscheidungen in einer der Notationen zur Erfassung von Architekturentscheidungen aus der Vorlesung (Umfang: ein ausgefülltes Architekturentscheidungs-Template und ggfs. 4-5 Sätze mit zusätzlichen Erklärungen).

Im Kontext des Backend Tiers der Pensionskassen-Webanwendung,
konfrontiert mit der Notwendigkeit der zuverlässigen, aber auch performanten Speicherung der Versichertendaten,
haben wir uns für das Data Mapper Pattern wie von Hibernate realisiert entschieden und gegen andere Data Access Layer Pattern wie Active Record,
um eine flexible, portable und wenig Programmierwaufwand verursachende zu erreichen;
wir nehmen die daraus resultierende Abhängigkeit von O/R Mapper Hibernate in Kauf, sowie die fehlende Kontrolle über die einzelnen SQL-Statements, die vom O/R-Mapper abgesetzt werden.

c) Diskutieren Sie, ob und wie Sie die Anwendungsarchitektur verändern müssen, wenn:

- die Anwendung neu Versicherte als zusätzliche Benutzergruppe mit den Use Cases „Selbstauskunft“ und „Adressänderung“ unterstützen soll.
- das interne und externe Benutzerverhalten sowie sämtliche Mutationen von Instanzen des Domain Models aufgrund einer gesetzlichen Auflage für 10 Jahre gespeichert werden sollen und
- ein Missbrauch der Anwendung über das Internet verhindert werden soll.

Gehen Sie auf dabei auf Vorlesungskonzepte wie Quality Attributes, Patterns und Middleware-Technologien ein (3-4 Sätze).

Quality Attribute bzw. Quality Attribute Group: Security

Einführung von Application Security Konzepten wie in der Vorlesungslektion 9 eingeführt: Authentication, RBAC, Audit Trail.

Unterstützende Funktionen im Systemmanagement wie z.B. Logs