Furkan Hidayet Rafet Aydin 1630039

Zaid Alakad 1630327

Aleksander Aiple 1629988

Teil 1: Architekturkonzepte

1. Architekturkonzepte

Erklären Sie in einem kurzen Aufsatz oder mit einigen Folien die vier in der Vorlesung behandelten Architektursichten. Für zwei der Sichten entwickeln Sie ein kleines UML-Beispieldiagramm mit kurzer Erläuterung hierfür, z.B. anhand eines Ausschnitts ihres aktuellen Bachelor-Projekts oder anderer Beispiele.

1. Kontextabgrenzung / Kontextsicht (Context View)

-betrachtung der angeschlossenen Nachbarsysteme.

1. Bausteinsicht / Struktursicht (Component View)

-beschreibt Architekturbausteine mit deren Schnittstellen und Kommunikationsbeziehungen

1. Laufzeitsicht / Verhaltenssicht (Dynamic View)

-Interaktionen und Zustandsbesc2hreibqung von Architekturbausteinen

1. Verteilungssicht / Abbildungssicht (Deployment View)

-Verteilung bzw. Abbildung der Architekturbausteine auf

- Hardware

- Softwaremodule

Component View

Ein Bild, das Diagramm enthält.

Automatisch generierte Beschreibung

Context View

Ein Bild, das Diagramm enthält.

Automatisch generierte Beschreibung

1. Physische Verteilung

Diskutieren Sie in einer Tabelle die Vor- und Nachteile für die verschiedenen Architekturen zur physischen Verteilung der Softwareschichten für Zweischichten-, Dreischichten und Mehrschichtenarchitektur.

| **Architektur** | **Vorteile** | **Nachteile** |
| --- | --- | --- |
| Zweischichten | Einfache Implementierung und Wartung | Keine klare Trennung von Präsentation und Logik |
|  | Geringe Komplexität | Sicherheitsrisiken |
|  | Geringere Netzwerkbelastung | Schlecht flexibel bei Änderungen |
|  | Geringere Anforderungen an Hardware |  |
| Dreischichten | Klare Trennung von Präsentation, Logik und Daten | Komplexere Implementierung und Wartung |
|  | Wartbarkeit durch Aufteilung der Schichten | Höhere Netzwerkbelastung |
|  | Einfache Integration von neuen Technologien | Höhere Anforderungen an Hardware |
|  | Bessere Wiederverwendbarkeit von Komponenten | Höherer Implementierungsaufwand für kleine Projekte |
| Mehrschichten | Noch bessere Trennung von Funktionalitäten | Noch höhere Komplexität |
|  | Höhere Flexibilität bei der Wartung | Noch höhere Netzwerkbelastung |
|  | Bessere Anpassung an unterschiedliche Systeme | Noch höhere Anforderungen an Hardware |
|  | Bessere Wiederverwendbarkeit von Komponenten | Höherer Implementierungsaufwand für kleine Projekte |

Teil 2: Nebenläufigkeit / Asynchrone Aufrufe