***Task 2 : Domain Model and Sequence Diagram = Bereichsmodell und Ablaufdiagramm***

Für Ihre Anwendung, entwerfen:

1. ein UML-Domänenmodell, das die Hauptklassen abdeckt

* Inklusive Assoziations- Vererbungs- und Aggregationsbeziehungen
* Beachten Sie die allgemeinen Modellierungsrichtlinien und nutzen Sie Responsibility Driven Design (RDD / Verantwortungsgetriebenes Design) oder verwenden Sie CRC-Cards (*Class-Responsibility-Collaborator // Klassen-Verantwortlicher-Mitarbeiter*)

1. Erstellen Sie basierend auf Ihrem Domänenmodell zwei UML-Sequenzdiagramme, jeweils eines für die folgenden Situationen (jedes sollte nicht banal/belanglos sein)

* Wenn der Benutzer bestimmte Informationen aktualisiert
* Wenn das System den Benutzer über ein bestimmtes Ereignis informiert

1. wenn nötig, verfeinern Sie Ihr UML-Klassenmodell, um die Attribute und Operationen einzuschließen, welche für Ihre Sequenzdiagramme benötigt werden.

Präsentation der Ergebnisse am 16.11.2017

10 Punkte für die Dokumente

Hinweise

▶ Halten Sie Ihr Projekttagebuch auf dem neuesten Stand

▶ platzieren Sie alle Arbeitsergebnisse in Ihrem Repository unter Pfad ... \ doc \ task01 \

*UML Domainmodelle*

Basiert sehr vereinfacht auf einem Klassendiagramm

* Illustriert sinnvolle Konzepte in einer Problemdomäne
* Representation von real-world «Things»
* zB Patient, Verschreibung, Doktor, etc.
* UML Domain Model: Strukturdiagrammdarstellung
* Konzepte (Konzeptionelle Klassen) (Manchmal werden hier auch Attribute aufgeführt)
* Verbindungen zwischen den Konzepten
* Wichtige Attribute der Konzepte
* ABER NICHT Operationen (Methodensignaturen) (Schnittstellen werden nie hier definiert)
* DO NOT
* Softwarekomponente oder Implementationsdetails präsentieren
* Klassen mit allen Details (setter/getter) modellieren

*Verantwortlichkeitsgetriebenes Design / Responsibility-Driven Design (RDD)*

Die Herangehensweise ist Objekt orientiert

* Verantwortlichkeitsmetapher: Softwareobjekte sind gleich wie Menschen mit Verantwortung welche mit anderen Menschen zusammenarbeiten
* RDD kreiert ein Objektorientiertes Design mit einer Gemeinschaft von Zusammenarbeitsverantwortlichen Objekten
* Verantwortung «machen»:
* Etwas selber machen (zB ein Objekt kreieren, eine Kalkulation machen)
* Aktion in anderen Objekten injizieren
* Kontrolle und Koordination von Aktivitäten in anderen Objekten
* Verantwortung «wissen»:
* Wissen über private und encapsulierte (verschachtelte) Daten
* Wissen über verwandte Objekte
* Wissen über Dinge, welche abgeleitet oder kalkuliert werden können

=>Hilft um Klassen und Objekte zu identifizieren, zu entscheiden, welche Methoden wohin gehören und wie diese Objekte interagieren sollten

*CRC Cards: Class-Responsibility-Collaborator // Klassen-Verantwortlicher-Mitarbeiter*

* CRC-Cards: Simples Tool um Klassen zu identifizieren (Schlüssel Abstraktionen)
* Klasse
* Welche Sachen sind Teil vom System?
* Welche Sachen kommunizieren oder interagieren mit dem System?
* Sind da spezielle Konsumenten oder Benutzer dieses Systems?
* Verantwortlichkeiten
* Was macht diese Klasse? (Funktionen)
* Welche Informationen sind in diese Klasse bekannt? (Attribute)
* Mitarbeiter
* Braucht diese Klasse Informationen von einer anderen Klasse?
* Ändert diese Klasse Informationen in einer anderen Klasse?
* Besteht eine Zusammenarbeit mit einer anderen Klasse?

***UML Sequenz Diagramm // Ablaufdiagramm***

Sequenzdiagramme zeigen

* Die Interaktion zwischen einer Anzahl von Objekten
* Heben die Zeitabfolge von den ausgetauschten Nachrichten hervor
* Die Reihenfolge von Interaktionen, welche in einem bestimmten Benutzungsfall oder einem Benutzungsszenario auftreten
* Interaktionen zwischen Akteuren und Objekten innerhalb eines Systems

=>zeigen den Datenfluss wo sie wie bearbeitet werden; Kontrollfluss, zeigt auf wer von was abhängig ist