SEP 2019

– Aufgabenblatt 2 –

spätester Abgabetermin: 20.05 2019 12 Uhr

Dieses Blatt richtet sich ausschließlich an die Teilnehmer des ***Softwareentwicklungsprojekts****.*

Ziel des Blattes

Auf dem letzten Aufgabenblatt wurde ein Pflichtenheft erstellt. Aufbauend auf diesen Ergebnissen wird nun die Systemanalyse durchgeführt.

Dieser Schritt hilft unter anderem besser die fachliche Domäne zu verstehen sowie ein Grundgerüst für weiteren Entwurf zu erzeugen.

Die Schnittstellen (Methoden) werden in diesem Fall durch exploratives Prototyping mit Hilfe von Aktivitätsdiagrammen bestimmt. Diese Diagramme sollen dazu dienen, die nötigen Methoden ausfindig zu machen und zu zeigen, dass diese prinzipiell in der Lage sind, die gestellten Aufgaben erfüllen zu können.

Dann wird eine Grobstruktur mittels eines Paketdiagramms festgelegt. Danach wird dieses durch ein Klassendiagramm verfeinert. Eine Aufteilung in Client und Server findet hier noch nicht statt. Dies wird erst Aufgabe für den Entwurf sein.

Beim Modellieren soll auf eine starke Bindung und lose Kopplung geachtet werden und jeder Klasse eine klar definierte Aufgabe gegeben werden (Single Responsibility Principle).

Aufgaben

1. Stellen Sie die Use Cases mithilfe der Aktivitätsdiagrammen dar. Falls Ihnen weitere Vorgänge auffallen, die es sinnvoll ist zu berücksichtigen, erstellen Sie die zusätzlichen entsprechenden Use Cases und Aktivitätsdiagramme. Achten Sie darauf, dass jeder Anwendungsfall Ihres Pflichtenheftes in mindestens einem Aktivitätsdiagramm abgebildet ist.

2. Erstellen Sie, auf Basis Ihrer Aktivitätsdiagramme, ein Paketdiagramm, das die Grobstruktur des Systems veranschaulicht. Ignorieren Sie dabei jegliche Verteilungsaspekte (also die Unterscheidung zwischen Client und Server). Geben Sie zu jedem Paket jeweils dessen Zweck an.

3. Erstellen Sie ein statisches Analysemodell mittels eines oder mehrerer Klassendiagramme. Erläutern Sie das, was nicht offensichtlich aus dem Diagramm hervorgeht. Stellen Sie sicher, dass sich jeder der Workflows aus Aufgabe 1 darin widerspiegelt.

4. Wenn Sie sich zwischen mehreren alternativen Lösungswegen entscheiden, dokumentieren Sie jeweils Ihre Gründe.

Hinweise

1. Wenn Ihr Modellierungstool Beschränkungen hinsichtlich der Gestaltung der Diagramme aufweist, sodass Sie von dem Abweichen müssen, was in SE2 gelehrt wurde, schreiben Sie das bitte dazu.
2. Wenn Sie absichtlich von dem abweichen, was in SE2 gelehrt wurde, ist das auch kein Problem, solange Sie das sinnvoll begründen.
3. Für die Grobstruktur kann es sinnvoll sein, Schichten zu definieren.
4. Benutzen Sie `«use»`-Beziehungen um zu zeigen, welche Pakete auf welche anderen Pakete zugreifen dürfen.
5. Machen Sie im Klassendiagramm deutlich, in welchen Paketen sich die einzelnen Klassen befinden. Das kann dadurch passieren, dass Sie die Klassen in das Paket „schachteln“.
6. Klassen die auf `-Manager` enden oder auf ähnliche Weise mehr einen technischen als einen fachdomänenspezifischen Aspekt darstellen, gehören i. d. R. nicht in das Analysemodell. Technische Aspekte sind Sache des Entwurfs. Analyseklassen sollten vorrangig „etwas sein“ und nicht „etwas tun“.
7. Versuchen Sie, mit den Aktivitätsdiagrammen neben dem groben Spielablauf auch die Spielregeln zu verdeutlichen.
8. Bei Aktivitätsdiagrammen sind die Knoten Aktivitäten und die Kanten Kontrollflüsse.
9. Verlieren Sie sich beim Modellieren der dynamischen Aspekte nicht in Details. Fragen Sie sich immer, was sinnvoll ist. Diagramme, die Sie sich später nicht mehr ansehen, sind wertlos.
10. Für die Grobstruktur kann es sinnvoll sein, Schichten zu definieren.

Checkliste zur Vermeidung typischer Fehler

- ☐ Die Lösungen beziehen sich unmittelbar auf das Pflichtenheft.
- ☐ Die dynamischen Diagramme haben leicht verständliche Verbindung mit der statischen Struktur.
- ☐ Diagramme werden nach UML-Syntax und Semantik erstellt und zwar nicht mit “Wissen” aus Wikipedia (insbesondere die Aktivitätsdiagramme). Wikipedia stellt einige Dinge von UML falsch dar.

☐ Analysemodell ist nicht technisch.

Anforderungen an Aktivitätsdiagramme

Da es oft zu Problemen beim Thema Aktivitätsdiagramme kommt, geben wir Ihnen diese kleine Auszüge aus dem OMG UML 2.5 Standard, damit ihr nicht das Dokument bearbeiten müsst[1][2]

1. Alle Startknoten bekommen zu Beginn der Aktivität einen Token. Dieser Token geht dann durch die Kontrollflows/Aktionen und solange keine Joins kommen, läuft alles unsynchronisiert, d.h. sobald ein Token sich weiter bewegen kann, macht er das auch[3].

2. Executable sind die eigentlichen Aktionen, dargestellt als Rechteck mit abgerundeten Ecken: 'An ExecutableNode shall not execute until **all** incoming ControlFlows (if any) are offering tokens.'

3. Fork und Join werden beide als einfache Balken dargestellt, die auch zu einem Element kombiniert werden können, wenn sie hintereinander stehen: "A ForkNode is a ControlNode that splits a flow into multiple **concurrent** flows[4]. A ForkNode shall have **exactly one** incoming ActivityEdge, though it may have multiple outgoing ActivityEdges."

4. "A JoinNode is a ControlNode that synchronizes multiple flows. A JoinNode shall have **exactly one** outgoing ActivityEdge but may have multiple incoming ActivityEdges."[5]

5. "If a JoinNode does not have a joinSpec[6], then this is equivalent to a joinSpec Expression with the Boolean operator 'and'."

6. Merge und Decision werden beide als Rauten dargestellt, die auch zu einem Element kombiniert werden können, wenn sie hintereinander stehen (unübersichtlich).

7. "A MergeNode is a control node that brings together multiple flows **without synchronization**[7]. A MergeNode shall have **exactly one** outgoing ActivityEdge but may have multiple incoming ActivityEdges."

[1] Das solltet Ihr aber in Zweifelsfällen tun (oder euren HiWi fragen).
[2] Nichtbeachtung dieser Hinweise ist nicht empfehlenswert.
[3] Wie das funktioniert, könnt Ihr hier anschauen (1,5 Minuten): https://coder-coacher.github.io/Udacity/Example-Activity-Diagram-qeWrvNFX-cY.html (Achtung! Fehler im Video bei durchgang vom ersten DecisionNode, es gibt keine Multiplizierung der Tokens beim DecisionNode.) und hier: (c.a. 4 Minuten): https://www.youtube.com/watch?v=dkd1Bw_B6jA
[4] D.h. jede ausgehende Kante bekommt einen Token.
[5] D.h. mehrere Tokens gehen rein, nur Einer geht raus.
[6] Die Bedingung dafür, dass die ausgehende Kante einen Token ausgibt == Kontrollflow weiter geht.
[7] D.h. keine Tokens werden vermehrt oder gelöscht. Das ist auch so beim DecisionNode.

8. "All tokens offered on the incoming edges of a MergeNode are offered to the outgoing edge. There is **no synchronization of flows or joining of tokens**."

9. "A DecisionNode is a ControlNode that chooses between outgoing flows. A DecisionNode shall have at least one and at most two incoming ActivityEdges, and at least one outgoing ActivityEdge."

10. Bei einer Entscheidung müssen die ausgehenden Kanten eindeutig unterscheidbar sein, da es keine Auswertungsreihenfolge gibt und somit das Verhalten sonst nichtdeterministisch wird: "A DecisionNode accepts tokens on its primary incoming edge and offers them to all its outgoing edges. However, each token offered on the primary incoming edge shall traverse **at most one** outgoing edge. **Tokens are not duplicated**."

Hilfreiche Links

- OMG UML 2.5: http://www.omg.org/spec/UML/2.5/PDF
- UML-Referenz (kompakter, weniger Infos): http://uml-diagrams.org/

Kontakt

Website	http://hci.uni-kl.de/~ebert/SEP
Leitung	apl. Prof. Dr. Achim Ebert ebert@cs.uni-kl.de
Organisation	Dr. Taimur Khan
Hiwis	Samir Bouchama
	Roman Reimche
	Matthias Müller
	Jonas Noglik
	Mail an alle sep-support@cs.uni-kl.de