SEP 2020

– Aufgabenblatt 2 –

spätester Abgabetermin: 25.05 2020 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 bzw. den Entwurf durchgeführt.

Dieser Schritt hilft unter anderem besser die fachliche Domäne zu verstehen. Aufbauend darauf sollen Sie einen Entwurf des Systems gestalten, das als Grundlage für die Implementierung dienen wird.

Sie müssen zunächst ein fachliches Klassendiagramm für die Systemanalyse erstellen, wobei Sie die in der fachlichen Domäne verwendeten Entitäten als Klassen darstellen. Dieses Diagramm erweitern Sie danach mit technischen Details, damit Sie die statische Struktur des Systems detailliert entwerfen.

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

Der nächste Schritt ist der wichtigste Teil der dynamischen Abläufe des Systems (das Spiel) in einem Sequenzdiagramm darzustellen. Dadurch legen Sie die Grundlage für die Implementierung dieser Dynamiken fest.

Abschließend stellen Sie die Ausnahmefälle dar und bereiten einen Gantt-Plan für die Implementierungsphase sowie eine kleine Präsentation des Zwischenstandes für die Abnahme.

Aufgaben

1. Erstellen Sie ein statisches Analysemodell mittels eines oder mehrerer Klassendiagramme. Erläutern Sie das, was nicht offensichtlich aus dem Diagramm hervorgeht. Stellen Sie sicher, dass das Diagramm alle wichtigen fachlichen Einzelheiten darstellt, die für die Umsetzung der Use-Cases wichtig sind. Hier sind Datentypen der Attribute aber auch Methoden nicht relevant.

2. Ergänzen Sie das Analysemodell um technische Aspekte (Typen, Methodensignaturen, Stereotypen, RMI, Client/Server usw.), um die statische Struktur der Software zu veranschaulichen. Das Klassendiagramm soll die wichtigen Zusammenhänge zwischen Entitäten darstellen, die für die korrekte Funktion während der Laufzeit

relevant sind. Teilen Sie dabei sinnvoll alle Klassen in Pakete ein, sodass Sie ein Paketdiagramm erstellen. Die GUI-Klassen, die unmittelbar für die Darstellung der Benutzeroberfläche verwendet werden brauchen Sie nicht darzustellen.

3. Erstellen Sie ein Sequenzdiagramm, das den Ablauf einer kompletten Spielrunde veranschaulicht. Verwenden Sie dabei die bereits vorhandenen Klassen. Es ist hier wichtig, das Zusammenspiel zwischen Client und Server zu verstehen und darzustellen, insbesondere die Synchronisation des Spielvorgangs unter mehreren Clients.

4. Überlegen Sie sich, welche Exceptions Ihre Klassen werfen werden. Definieren Sie eine Hierarchie fachlicher Exceptions und veranschaulichen Sie diese in einem Klassendiagramm.

5. Beschreiben Sie für einen Computerspieler (Bot) für alle Entscheidungen Handlungsvorgaben. Es kann hierbei gegebenenfalls hilfreich sein einen Zustandsautomaten zu modellieren.

6. Erstellen Sie ein Gantt-Diagramm für die Implementierungsphase. Achten Sie hierbei darauf, dass aus Ihrem Diagramm hervorgeht wer für welchen Teil verantwortlich ist. Diese Informationen sollen auch in Ihrem Diagramm ersichtlich sein.
   Implementierung und Optimierung erstrecken sich über zwei Phasen:
   a) Grundgerüst, Dokumentation, Tests und Implementierung von Chat-, Login- und GUI-bezogene Funktionen
      (Späteste Abgabe: 22.06.)
      i. In dieser Phase sollen die Interfaces/Klassen jeweils mit ihren Methodensignaturen und die Beziehungen der Klassen umgesetzt werden.
      ii. Ferner sind Unit-Tests und Java-Doc für alle öffentlichen Methoden zu schreiben.
      iii. Schließlich müssen Chat-, Login- und GUI-bezogene Funktionen implementiert werden.
   b) Erweiterung, Validierung und Optimierung
      (Späteste Abgabe: 14.07.)
      i. In dieser Phase soll die Implementierung abgeschlossen werden, indem alle verbleibenden Klassen/Methoden implementiert werden, insbesondere die Spiellogik und Synchronisation des Spielvorgang unter mehreren Clients.
      ii. Hier sollen dann auch die auf diesem Blatt entworfenen Bots umgesetzt werden.

7. Bereiten sie sich auf eine Zwischenabnahme vor. Hierbei sollen sie als Team präsentieren was sie bisher im Projekt geleistet haben. Jeder Teilnehmer in ihrer Gruppe soll seine eigene Leistung selbst vorstellen. Hierzu ist nicht zwingend eine Beamer

Präsentation notwendig, aber falls diese von Ihnen als hilfreich angesehen wird können Sie eine solche vorbereiten.

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 Modellierung von Software-Systemen (bzw. SE2) gelehrt wurde, ist das auch kein Problem, solange Sie das sinnvoll begründen.
3. Vermeiden Sie es, sofort das Entwurfsklassendiagramm anzufangen. Sie sollen zunächst die wichtigen Zusammenhänge zwischen den fachlichen Entitäten verstehen.
4. Benutzen Sie `«use»`-Beziehungen um zu zeigen, welche Pakete auf welche anderen Pakete zugreifen dürfen.
5. Mittels Stereotypen lassen sich bestimmte Informationen leicht an Klassen „anheften“. So ist es beispielsweise möglich, Remote-Interfaces durch einen Stereotyp `«remote»` zu kennzeichnen, anstatt jedes Mal einen Generalisierungspfeil zu `java.rmi.Remote` zu zeichnen. Auf diese Weise können Sie Ihre Diagramme übersichtlicher gestalten.
6. Bedenken Sie, dass bei RMI alle Daten, die zwischen Client und Server ausgetauscht werden, entweder `Remote` oder `Serializable` sein müssen.
7. Sehen Sie sich die *Java Collection API*[1] an. Sie werden einige dieser Klassen bzw. Interfaces brauchen.
8. Es ist nicht nötig alle GUI-Klassen von Client zu modellieren. Es reicht die Klassen zu modellieren, die unmittelbar in der Kommunikation mit dem Server stehen und so die Schnittstellen zwischen eurem System und GUI-Framework darstellen.
9. 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“.
10. Es ist kaum möglich das Sequenzdiagramm ohne Verwendung von Combined Fragments zu erledigen[2].
11. 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 gehören in den den Entwurf. Analyseklassen sollten vorrangig „etwas sein“ und nicht „etwas tun“.

[1] http://docs.oracle.com/javase/8/docs/api/java/util/Collections.html
[2] https://www.uml-diagrams.org/sequence-diagrams-combined-fragment.html

12. java.util.concurrent kann eine Hilfe darstellen, sowohl zum Verständnis von Nebenläufigkeit als auch um Deadlocks zu vermeiden[3] (insbesondere lohnt es sich die Klassen CountDownLatch, CyclicBarrier, and Semaphore anzuschauen).

13. Wählen Sie sinnvolle Abstraktionsebenen für jedes Diagramm: Diagramme sollen wichtige Informationen veranschaulichen.

14. Die Verwendung von Design Patterns können wesentlich hilfreich sein. Insbesondere können die folgenden Patterns relevant sein: Observer (Publish/Subscribe), Facade, Iterator und Strategy.

Checkliste zur Vermeidung typischer Fehler

- ☐ Die Lösungen beziehen sich unmittelbar auf das Pflichtenheft: Es werden die Begriffe, die Datentypen sowie weitere Entitäten verwendet, die früher bereits ihre Namen bekommen haben, damit man nicht das gleiche unter unterschiedlichen Namen hat.
- ☐ Die dynamischen Diagramme haben eine leicht verständliche Verbindung zur statischen Struktur.
- ☐ Diagramme werden nach UML-Syntax und Semantik erstellt und zwar nicht mit "Wissen" aus Wikipedia. Wikipedia stellt einige Dinge von UML falsch dar.
- ☐ Benutzung von RMI wird bei Modellierung berücksichtigt
- ☐ Modellierung wird unter Beachtung von Java-spezifischen Einzelheiten erledigt.
- ☐ UML-Syntax wird beachtet (insbesondere von Sequenzdiagrammen[4])
- ☐ Der Teil des Systems welches auf dem Client läuft verwendet das GUI-Framework aber ist nicht dem Framework gleich.

Hilfreiche Links

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

[3] https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/package-summary.html
[4] https://www.uml-diagrams.org/sequence-diagrams.html

Kontakt

OLAT	https://olat.vcrp.de/url/RepositoryEntry/2565867421
Leitung	apl. Prof. Dr. Achim Ebert ebert@cs.uni-kl.de
Organisation	M.Sc. Pavel Weber
Hiwis	Ann Kathrin Peters
	Samir Bouchama
	Roman Reimche
	Mail an alle sep-support@cs.uni-kl.de