
**Probeklausur zur Vorlesung „Einführung in Software Engineering“
im Wintersemester 2015/2016**

Universität Heidelberg, Institut für Informatik
Prof. Dr. Barbara Paech, Marcus Seiler

Hinweise

- Die Bearbeitungszeit beträgt **90 Minuten**.
- Tragen Sie auf diesem Blatt Ihren Namen, Ihren Vornamen, Ihre Matrikelnummer und Ihren Studiengang ein.
- Bitte schreiben Sie Ihren Namen und Matrikelnummer auf **ALLE Aufgaben- und Lösungsblätter**.
- Bitte schreiben Sie Ihre **Lösungen auf die ausgeteilten Lösungsblätter**. Verwenden Sie bitte auch die **Rückseiten**. Falls Sie weitere Blätter für Ihre Lösungen brauchen, melden Sie sich bitte und Sie erhalten weitere Blätter von uns (bitte keine eigenen verwenden). Schreiben Sie auch auf die neuen Blätter Ihren Namen und Ihre Matrikelnummer und kennzeichnen Sie, welche Aufgabe Sie bearbeiten.
- Bitte schreiben Sie leserlich, kennzeichnen Sie eindeutig Ihre Lösungsvorschläge und streichen Sie Irrwege durch.
- Die **Klausur umfasst 4 Aufgaben**. Die erreichbare Höchstpunktzahl beträgt 40 Punkte. Mit 17 Punkten ist die Klausur gerade noch bestanden.

Ihre Daten

Name: _____

Vorname: _____

Matrikelnummer: _____

Studiengang: _____

Ergebnis (bitte nicht beschriften)

	1 (13)	2 (9)	3 (10)	4 (8)	Summe (40)
erreichte Punkte					
korrigiert von					

Aufgabe 1 (13) Definitionen

Aufgabe 1.1 (2 Punkte)

Qualitätsmanagement ist ein wichtiger Aufgabenbereich im Software Engineering. Insgesamt haben wir 5 Aufgabenbereiche betrachtet. Zählen Sie die 4 weiteren Aufgabenbereiche des Software Engineering (außer Qualitätsmanagement) auf.

Aufgabe 1.2 (3 Punkte)

Geben Sie 3 typische Bestandteile einer Persona-Beschreibung an und jeweils ein kurzes Beispiel dafür.

Aufgabe 1.3 (3 Punkte)

Welche 3 Arten von Entwurfsmustern gibt es? Erklären Sie jede Art kurz und nennen Sie ein Beispiel für jede Art.

Aufgabe 1.4 (2 Punkte)

Geben Sie für die Qualitätsattribute Benutzerbarkeit (Usability) und Sicherheit jeweils einen wichtigen Teillaspekt an (z.B. bei Wartbarkeit wäre das Testbarkeit oder Modifizierbarkeit) und eine zu diesem Teillaspekt passende, konkret nachprüfbare nicht-funktionale Anforderung an (bei Modifizierbarkeit wäre das z.B. „Beim Austausch der Datenbank sollen die Klassen der UI-Schicht nicht verändert werden müssen“).

Benutzerbarkeit, Teillaspekte :

- Appropriateness recognizability (Verständnis der Features, z.B. Dokumentation)
- Learnability (Bsp : Das System Movie Manager verfügt über eine Hilfsfunktion)
- Operability (steuerbar)
- User error protection
- User interface aesthetics
- Accessibility(auch spezielle NutzerInnen)

Sicherheit, Teillaspekte:

- Confidentiality (nur autorisierte Zugang, Bsp. : Die Verbindung zwischen Movie Manager und EMF Store Datenbank ist gegen Man in the Middle Angriffe geschützt.)
- Integrity(nur autorisierte Änderungen)
- Non-repudiation (Ereignisse nachweisbar)
- Accountability (Ereignisse zuweisbar)
- Authenticity (Identität zuweisbar)

Aufgabe 1.5 (3 Punkte)

Geben Sie drei Maßnahmen aus dem Bereich Qualitätsmanagement an, mit denen eine Organisation mittelfristig die Qualität ihrer Software erhöhen kann. Beschreiben Sie jede Maßnahme in mindestens einem Satz.

Aus Timeas Vortrag:

- Fehlerursachenanalyse, spezifische Rolle eines quality Engineer in jedem Project, UI-check, Metriken (erheben und über längere Zeit analysieren).

Techniken aus der Vorlesung :

- Fehlermanagement, Systematisches erfassen und beheben von Fehlern in der Software
- Versionsmanagement, Gemeinsame Erstellung von Code in verschiedenen Versionen, gemeinsame Abgabekonflikte, Änderungshistorie
- Statische Codeprüfung, Ermitteln von Verstößen gegen Spezifikationen oder einzuhaltende Standards

Aufgabe 2 (9 Punkte) Anforderungen

Im Anhang (Seite 13) finden Sie den Use Case „Manage Loaned Movies with Lenders“. Die Funktionalität des Movie Managers soll um die Behandlung verlorener und überfälliger Filme erweitert werden, d.h.

- Der Aktor kann einen Film als „lost“ kennzeichnen bzw. die Kennzeichnung wieder zurücknehmen. Dies ist unabhängig davon, ob der Film ausgeliehen ist oder nicht.
- Ein als „lost“ gekennzeichneter Film kann nicht ausgeliehen werden.
- Der Aktor kann einen ausgeliehenen Film auf „overdue“ setzen bzw. die Kennzeichnung wieder zurücknehmen.
- Ein Film kann nur entweder als „lost“ oder „overdue“ gekennzeichnet sein. Er muss keine solche Kennzeichnung haben.
- Wird ein als „lost“ oder „overdue“ gekennzeichneter Film zurückgegeben, wird die Kennzeichnung gelöscht.

Im Use Case ist für diese Funktionalität schon die Systemfunktion „Handle Lost and Overdue Movie“ vorgesehen, aber noch nicht vollständig beschrieben.

Ihre Lösungen für Aufgabe 2.1 und 2.2 können auf Deutsch oder Englisch sein.

Aufgabe 2.1 (3 Punkte)

Überarbeiten Sie die untenstehende Aufgabenbeschreibung im Task&Support Schema entsprechend der oben genannten Funktionalität, indem Sie die Vorlage um neue Sub-tasks ergänzen bzw. die bestehenden Sub-tasks überarbeiten. Überlegen Sie, wie das System die neuen Sub-tasks unterstützen kann und nennen Sie das unter „example solution“.

User Task	Movie Management	
Purpose (Goal)	Manage Movies and corresponding Performer data of a Movie Collection	
Frequency	Often and at any time (depending on the user's needs)	
Actors	User who wants to manage Movies	
Sub-tasks:		
1	Describe a Movie Add and describe a Movie with typical data like its title, release date, production country, performers etc. Or change an existing description. Or view Movies.	Provide default values wherever possible, implemented in function <i>Add Movie</i> and <i>Edit Detail Data</i>
1ap	(Variant) Remove existing Movie Problems: Removing all Movies a certain Performer participates in might result in Performers not associated with any Movies	Ensure the consistency of Performers and Movies, implemented in function <i>Delete Movie</i>
2	Lend Movie Record that Movie is given to another person or received back. Or view loans.	Support notification wrt loan period, implemented in function <i>Receive Returned Movie</i>
3	Manage watched Movies Record time when the Movie was last watched. Or sort Movies accordingly	Provide reminder for Movies without watched date, implemented in function <i>Watch Movie</i>
	Manage overdue and lost movies Record if movie is overdue or lost	Provide reminder for overdue movies. Implemented in function <i>handle lost or overdue movie</i>

Aufgabe 2.2 (6 Punkte)

Ergänzen Sie den Use Case „Manage Loaned Movies with Lenders“ entsprechend der im einleitenden Text von Aufgabe 2 beschriebenen Funktionalität (Nicht die „Example solution“ aus Aufgabe 2.1):

- Setzen Sie diese Funktionalität in geeignete Regeln um und ergänzen Sie die Abschnitte 'Flow of Events', 'Exceptions' und 'Rules' entsprechend.
- Beschreiben Sie kurz die Systemfunktion „Handle Lost and Overdue Movie“. Wenn eine existierende Systemfunktionsbeschreibung gegenüber dem bisherigen Use Case verändert wird, beschreiben Sie diese Änderung ebenfalls kurz.

Sie können die **Änderungen/Vervollständigungen** entweder direkt in den Use Case im Anhang eintragen oder Sie beschreiben sie im Abschnitt „Änderungen/Vervollständigungen“ auf der nachfolgenden Seite.

Änderungen/Vervollständigungen

Aktor / System Steps:

Keine Änderungen

Exceptions:

Neue Exception in S4
[Exception : Movie cannot be loaned because it is lost]
The Movie is not accepted as loaned.

Neue Exception in S3
[Exception : Movie cannot be overdue because it is not loaned]
The Movie is not accepted as overdue

Rules:

Neue Rule:
[Overdue Rule]
A Movie can only be overdue if it is loaned

Abgepasste Rule:
[Loan Movie Rule]
Alt: Movie cannot be loaned if it is already loaned
Neu: Movie cannot be loaned if it is already loaned or lost

Systemfunctions:

- Beschreibung für Systemfunktion "Handle Lost and overdue Movie". Depending on the input given by the actor the "lost" or "overdue" attribute of the Movie is set to "true".
- "Receive returned Movie" muss angepasst werden:
 Alt : The attribute "loaned" of the returned Movie is set to false.
 Neu : The attributes "loaned" and "lost" and "overdue" of the returned Movie are set to false.

Aufgabe 3 (10 Punkte) Entwurf

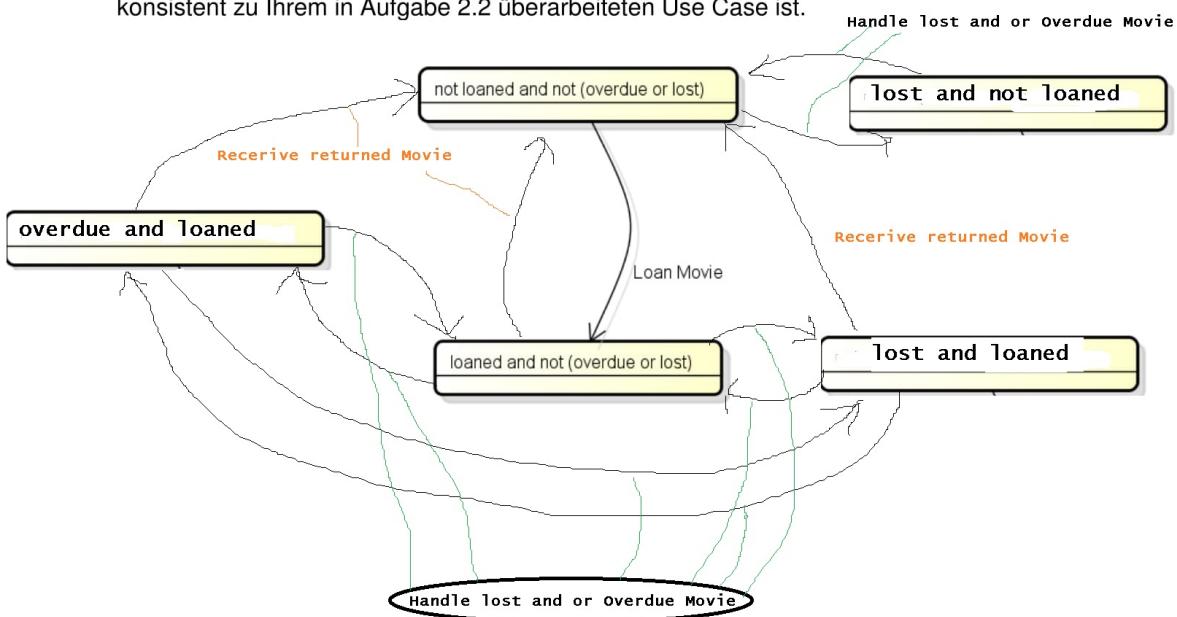
Aufbauend auf dem Use Case „Manage Loaned Movies with Lenders“ im Anhang (Seite 13) und Ihrer Bearbeitung von Aufgabe 2.2 fertigen Sie in dieser Aufgabe Teile eines Entwurfs für diesen Use Case an.

Aufgabe 3.1 (6 Punkte)

Ergänzen Sie das unten angegebene Zustandsdiagramm für den von Ihnen in Aufgabe 2.2. überarbeiteten Use Case „Manage Loaned Movies with Lenders“. Gehen Sie dazu wie folgt vor:

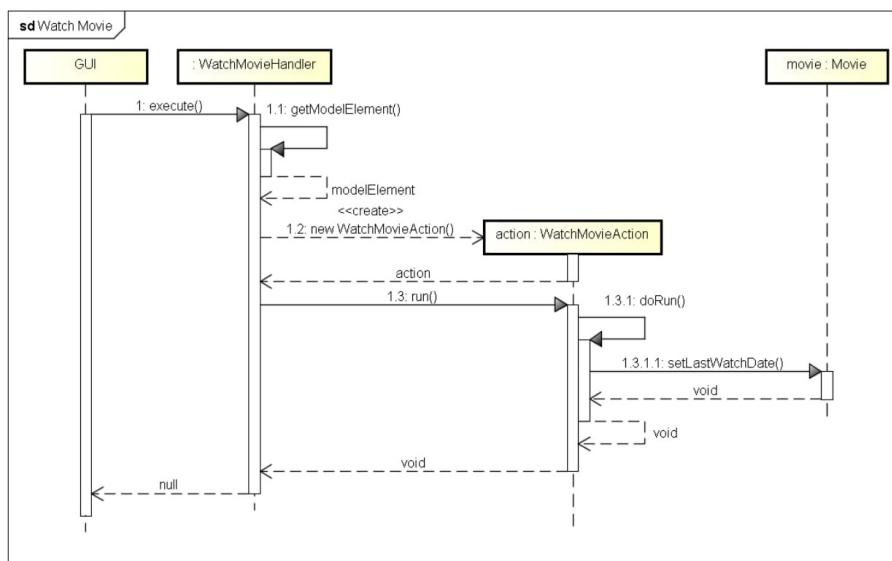
- Ergänzen Sie weitere Zustände, welche ein Film in Bezug auf die Attribute lost, loaned und overdue annehmen kann. Beschreiben Sie jeden Zustand durch eine Bedingung über diesen Attributen. Beachten Sie, dass die Bedingungen verschiedener Zustände sich gegenseitig ausschließen müssen.
- Zeichnen Sie in Ihr Zustandsdiagramm die möglichen Transitionen zwischen den Zuständen ein und benennen Sie diese mit der entsprechenden Systemfunktion.
- Stellen Sie sicher, dass alle durch „Loan Movie“, „Receive Returned Movie“ und „Handle Lost and Overdue Movie“ möglichen Zustandsübergänge im Diagramm beschrieben sind.

Beachten Sie bei Ihrer Überarbeitung insbesondere, dass Ihr Zustandsdiagramm konsistent zu Ihrem in Aufgabe 2.2 überarbeiteten Use Case ist.



Aufgabe 3.2 (4 Punkte)

Das nachfolgende Sequenzdiagramm zeigt den Ablauf der Systemfunktion „Watch Movie“ zum Ansehen eines Films.



- a) Verwenden Sie das Sequenzdiagramm um die nachfolgend aufgeführten Java Klassen an den gekennzeichneten Stellen zu vervollständigen.

1	<code>public class WatchMovieHandler extends AbstractHandler {</code>
2	<code> public Object execute(ExecutionEvent event) throws ExecutionException {</code>
3	<code> EObject modelElement = this.getModelElement(event);</code>
4	<code> WatchMovieAction action = new WatchMovieAction(modelElement);</code>
5	<code> action.run();</code>
6	<code> return null;</code>
7	<code> }</code>
8	<code> public static EObject getModelElement(ExecutionEvent event) {...}</code>
9	<code>}</code>
1	<code>public abstract class EPCCommand {</code>
2	<code> abstract protected void doRun() {}</code>
3	<code> protected final void run() {this.doRun();}</code>
4	<code>}</code>
1	<code>public class WatchMovieAction extends EPCCommand {</code>
2	<code> private Movie movie;</code>
3	<code> public WatchMovieAction(EObject movie) {</code>
4	<code> this.movie = (Movie) movie;</code>
5	<code> }</code>
6	<code> protected void doRun() {</code>
7	<code> movie.setLastWatchDate(new Date());</code>
8	<code> }</code>
9	<code>}</code>

- b) Beschreiben Sie, was im Sequenzdiagramm und damit bei den ergänzten Code Teilen noch fehlt, damit der Code vollständig ist, d.h. ausgeführt werden kann?

Die Übergabeparameter von operationen sind im Sequenzdiagramm nicht vorhanden (z.B.: new WatchMovieAction() und setLastWatchDate()).

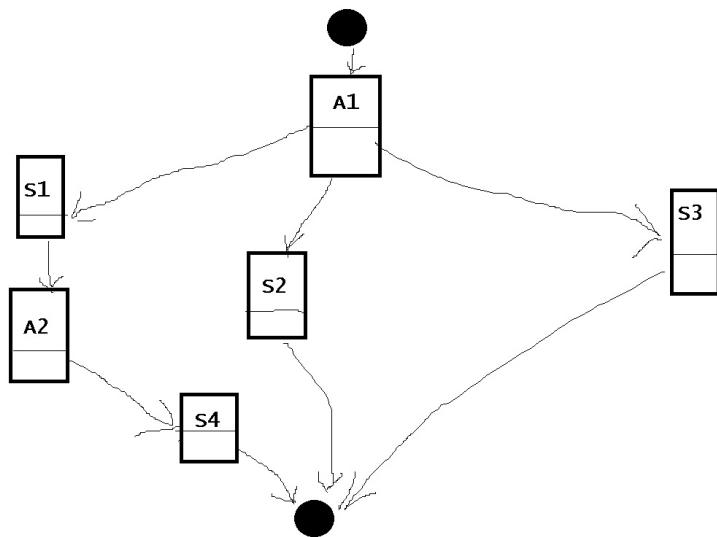
Aufgabe 4 (8 Punkte)

Im Anhang (Seite 13) finden Sie den Use Case „Manage Loaned Movies with Lenders“. Im Folgenden erstellen Sie Systemtestfälle aus diesem Use Case.

Aufgabe 4.1 (3 Punkte)

Zeichnen Sie (auf dem nächsten Blatt) einen Kontrollflussgraphen, der den „Flow of Events“ des Use Cases „Manage Loaned Movies with Lenders“ darstellt. Benennen Sie die Knoten des Kontrollflussgraphen nach den Schritten des Use Cases (z.B. „A1“ für Aktorschritt 1 oder „S2“ für Systemschritt 2). Achten Sie darauf, dass für jeden Schritt genau ein Knoten im Kontrollflussgraphen vorkommt. Verwenden Sie einen einzigen Start- und Endknoten zusätzlich. Vernachlässigen Sie die Ausnahmen.

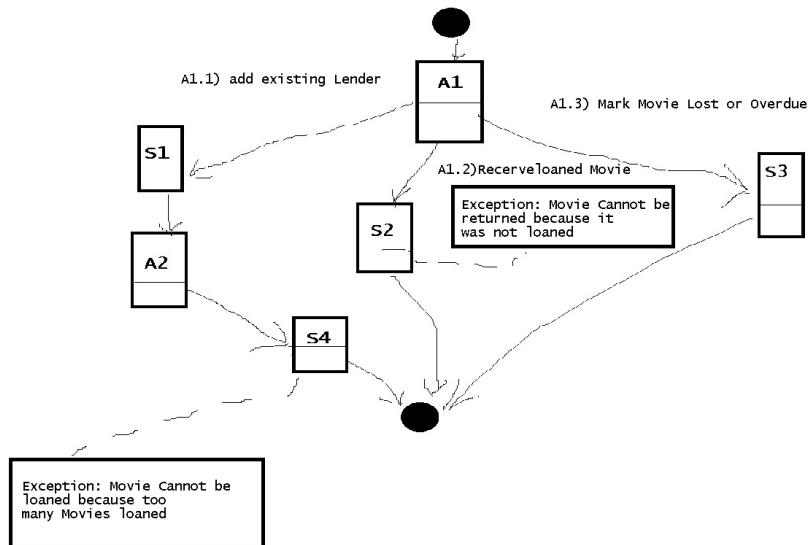
Kontrollflussgraphen für ‚Flow of Events‘ für Use Case ‖Manage Loaned Movies with Lenders‖



Aufgabe 4.2 (5 Punkte)

- a) Kennzeichnen Sie die Knoten (Schritte), an denen die Ausnahmen „Movie cannot be returned, because it was not loaned“ und „Movie cannot be loaned, because of too many movies“ auftreten können.
- b) Stellen Sie sich vor, Sie testen ein System, bei dem die Ausnahme „Movie cannot be loaned because too many Movies loaned“ **nicht umgesetzt** wurde, d.h. das System **erkennt** diesen **Ausnahmefall nicht**. Geben Sie ein Testprotokoll für dieses **fehlerhafte** Verhalten des Movie Managers an. Achten Sie darauf, die konkreten Eingaben (Aktorschritt und Datenzustände) und konkreten **falsche** Ausgaben (inkl. Systemschritt) anzugeben, die man auf der Oberfläche erkennen kann. Das Testprotokoll soll im Startzustand beginnen. Beachten Sie die Hinweise in der Vorlage für das Testprotokoll:

Name	
Beschreibung	<i>Beschreibung, was überprüft wird</i>
Vorbedingung	<i>Datenzustand vor Ausführung des Testfalls</i>
Nachbedingung	<i>Datenzustand nach Ausführung des Testfalls</i>
Testschritte	
Schritt 1	<p>Eingabe:</p> <ul style="list-style-type: none"> • Aktorschritt: • Daten auf Bildschirm: • Eingabedaten: <p>Ausgabe:</p> <ul style="list-style-type: none"> • Systemschritt: • Ausgabedaten: (<i>Änderung im Systemzustand und auf dem Bildschirm</i>)
Schritt 2	<ul style="list-style-type: none"> •



Testprotokoll

Name	
Beschreibung	
Vorbedingung	
Nachbedingung	
Testschritte	
Schritt 1	
Schritt 2	

Anhang - Use Case „Manage Loaned Movies with Lenders“

Name	Manage Loaned Movies with Lenders	
Actor	Person	
Goal	The actor wants to loan or receive returned Movies or handle lost and overdue Movies	
Pre-condition	The system "Movie Manager" is started. W2.2 Detail Movie	
Flow of Events	Actor A1) 1.1) Actor chooses to add an existing Lender to the selected Movie Next: S1 1.2) Actor chooses to receive the previously loaned Movie. Next: S2 1.3) Actor chooses to mark or unmark the selected Movie as "lost" or "overdue" Next: S3 A2) Actor selects the Lender to loan the Movie to and specifies the end date of the loan period. Next: S4	System S1) System shows a list of Lenders W2.2 Detail Movie [System function: List Lenders] Next A2 S2) System executes the "Receive Returned Movie" function. W2.2 Detail Movie [System function: Receive Returned Movie] [Exception: Movie cannot be returned because it was not loaned]
		S3) System executes the "Handle Lost or Overdue Movie" function W2.2 Detail Movie [System function: Handle Lost and Overdue Movie] [Exception : Movie cannot be overdue because it is not loaned]
		S4) System executes the "Loan Movie" function. W2.2 Detail Movie [System function: Loan Movie] [Exception: Movie cannot be loaned because it is already loaned] [Exception: Movie cannot be loaned because too many Movies loaned] [Exception : Movie cannot be loaned because it is lost]
Exceptions	[Exception: Movie cannot be returned because it was not loaned] - The Movie is not accepted as returned. [Exception: Movie cannot be loaned because it is already loaned] - The Lender of the Movie is not changed. [Exception: Movie cannot be loaned because too many Movies loaned] - The Movie is not accepted as loaned. [Exception : Movie cannot be overdue because it is not loaned] - The Movie is not accepted as overdue [Exception : Movie cannot be loaned because it is lost] - The Movie is not accepted as loaned.	
Rules	[Return Movie Rule] : Movie cannot be returned if it is not loaned. [Loan Movie Rule] : Movie cannot be loaned if it is already loaned. [Lender Rule] : A Lender can only lend 5 Movies at the same time [Overdue Rule] : A movie can only be overdue if it is loaned	
Data, System Functions	Data: Movie, Lender System Functions: Loan Movie, Receive Returned Movie, List Lenders, Handle Lost and Overdue Movie	
Postcondition	A Movie is loaned to a Lender or a Movie is received from a Lender or a Movie is marked as lost or as overdue W2.2 Detail Movie	

