Modulehandleiding

iipsen
Software Engineering
Informatica

Studiejaar: 2017-2018
Periode: 4
Auteurs: Alex van Manen, Roland Westveer
Versie: 4.0 (definitief)

Versiebeheer

Versie-nummer	Datum	Auteur	Status	Beschrijving
1.1	13 april 2016	Alex van Manen	Concept	Nieuwe versie voor studiejaar 2015-2016.
1.2	22 april 2016	Alex van Manen	Concept	Feedback verwerkt van docenten (Koen Warner, Henk van Dijk, Thomas Boose, Martijn Jansen en Michiel Boere).
1.3	27 mei 2016	Alex van Manen	Concept	Versienummer verhoogt zodat het conform is met de andere toetsdocumenten.
1.4	24 juni 2016	Alex van Manen	Concept	Versienummer verhoogt zodat het conform is met de andere toetsdocumenten.
2.0	27 juni 2016	Alex van Manen	Definitief	Definitief gemaakt voor de toetscommissie.
2.8	2 mei 2017	Roland Westveer	Concept	Nieuwe versie voor studiejaar 2016-17.
2.8.1	5 mei 2016	Roland Westveer	Concept	Laatste wijzigingen aangebracht, validatie met Alex van Manen
3.8	16 april 2018	Roland Westveer	Concept	Aangepast aan nieuwe collegejaar Validatie met Alex van Manen

Inhoudsopgave

1. Inleiding

Dit document beschrijft het project "IIPSEN – Project Inleiding Software Engineering" en geeft een korte inleiding over de richting Software Engineering.

2. Software Engineering in het kort

Bij de richting Software Engineering leer je software voor diverse doeleinden te ontwerpen, te realiseren en implementeren. Grootschalig (bijvoorbeeld logistieke of financiële systemen) en kleinschalig (bijvoorbeeld voor mobiele telefoon, tablet of smartwatch). Met behulp van vakken als logica en softwarearchitectuur ontdek je hoe je een applicatie ontwerpt en bouwt. Bovendien leer je om een systeem te ontwikkelen dat flexibel is, goed onderhouden kan worden en voldoet aan de andere standaardeisen die hiervoor gelden (ISO/IEC 25010:2011).

Meer weten? Neem contact op met Roland Westveer (westveer.r@hsleiden.nl) of één van de andere SE-docenten.

3. Projectopzet

Dit project staat geheel in het teken van het gestructureerd ontwikkelen van objectgeoriënteerde software in teamverband. Het project behelst een kleinschalig softwareontwikkelingstraject dat gefaseerd van analyse (wat gaan we maken?) tot en met de oplevering (presentatie) verloopt. In dit project combineer je alle kennis van de vakken iopr1, iopr2, inet en imuml. Vervolgens geven we workshops waarin je kennis maakt met diverse technieken zoals RMI, git en (unit)testen en een lagen architectuur.

3.1 Inleiding

Het bouwen van een goed werkende applicatie is veelomvattend. Wat vaak met een eenvoudige klantvraag begint wordt al snel een omvangrijk project. Denk aan vragen zoals van welke gegevens zijn er nodig? Welke scenario's zijn van toepassing? Hoe is de look en feel? En ga zo maar door. Om teleurstellingen bij de klant te voorkomen starten we met het opstellen van een probleemstelling en de requirements. Hierna maken we een planning en de diagrammen zoals je hebt geleerd bij het vak imuml. Daarna komt het echte design- en programmeerwerk. Door middel van het bouwen van een prototype wordt de applicatie tot leven gebracht. Jullie ontwikkelen door tot er een stabiel product is, die de eindtest doorstaat.

3.2 Opdrachtbeschrijving

Het bedrijf '2020 Games' wil een aantal van zijn bordspellen beschikbaar maken voor mensen met een lichamelijke beperking. Aan jullie is de taak om één van deze bordspellen beschikbaar te maken voor deze doelgroep. De spellen die 2020 games graag gedigitaliseerd wil zien zijn: Karuba, Raptor, Kingsburg, Machi Koro, Splendor, Pandemic, Queendomino, 10 minuten kraak - De tovenaarstoren, Ticket to Ride - Nordic countries, Photosynthese, Magic Maze, Prodigals Club, Leeuwenhart, Specter Ops, Clank!, Mission: Red Planet (2nd edition), Quadropolis, Tiny Epic Galaxies, Suburbia, Above and Below, Near and Far, Deus, Burgle Bros.

Het te maken spel moet aan de hieronder genoemde eisen voldoen.

Functionele eisen:

- Het spel moet speelbaar zijn volgens de bijbehorende regels;
- De spelregels dienen te allen tijde te raadplegen te zijn;
- Elk spel dient tussentijds opgeslagen en hervat te kunnen worden.
- Het spel moet online door meerdere spelers tegelijk te spelen zijn.

Niet-functionele eisen:

- Het spel moet aansluiten bij de oorspronkelijke look en feel;
- Het spel moet geschreven worden in Java zodat deze speelbaar is in Windows, Mac OSX en Linux.
- Het spel moet gebruik maken van RMI (Remote Method Invocation).
- Het spel dient een bepaalde complexiteit te hebben zoal Monopoly. Dus geen boter kaas en eieren.

Ontwerpeisen:

- De applicatie dient volledig beschreven te zijn in UML.
- Er moet een grafisch ontwerp van de GUI worden gemaakt.

3.3 Opzet projectteam en speltoekenning

Elke projectgroep bestaat uit minimaal vier en maximaal zes leden. Elke projectgroep krijgt een uniek spel. Zorg dat je via internet of andere bronnen kennis neemt van de spelregels. Sommige spellen zijn te leen bij Alex van Manen tegen inlevering van je collegekaart.

3.4 Organisatie

Tijdens dit project heb je één dag per week een projectdag waarin je een stand-up meeting hebt, workshops en inhoudelijke begeleiding. Daarnaast heb je een groepscoach die op een andere dag met je afspreekt. Alle projectgroepen worden dus begeleid door een groepscoach en een inhoudelijk begeleider. Beide vormen van begeleiding zijn hieronder uitgebreider beschreven.

3.4.1 Groepscoach

Ter ondersteuning van het projectmatig werken staat er elke week een afspraak met de groepscoach in de agenda. Tijdens deze coachsessies zal je ingaan op het groepsproces en de samenwerking in de groep.

3.4.2 Inhoudelijke begeleiding

Tijdens het project wordt inhoudelijke begeleiding geboden door docenten. Elke groep krijgt een inhoudelijk begeleider toegewezen. De taak van de inhoudelijk begeleider is zuiver een adviesrol. Als groep ben je zelf verantwoordelijk voor de kwaliteit van de eindproducten van de betreffende fase.

3.4.2 Workshops

Naast de inhoudelijke begeleiding worden er ook workshops gegeven. In de onderstaande tabel vind je een conceptoverzicht van de workshops.

Week	**Workshop 1**	**Workshop 2**
Week 1 (23 april)	Bordspelkeuze – Charlotte Ipema	Test scenario's – Alex van Manen
Week 2 (7 mei)	Java RMI - Koen Warner	OO Design Principes – Henk van Dijk
Week 3 (14 mei)	Java RMI - Koen Warner	OO Design Principes – Henk van Dijk
Week 4 (21 mei)	Testen & testrapportage – Roland Westveer (weblecture i.v.m. Pinksteren).	Geen tweede workshop (i.v.m. Pinksteren)
Week 5 (28 mei)	GIT – Jeroen Rijsdijk	OO Design Principes – Henk van Dijk
Week 6 (4 juni)	Unit testen – Jeroen Rijsdijk	Javadoc – Michiel Boere

4. Op te leveren producten

Gedurende het project dient er stapsgewijs (per fase) een aantal producten gemaakt en opgeleverd te worden. Al deze producten dragen bij aan het stap voor stap realiseren van het spel. De planning en deze producten bevinden zich hieronder. De beschrijving en eisen per product zijn te vinden als bijlage (zie bijlage 1).

4.1 Producten en planning

Het onderstaande schema geeft de op te leveren producten per fase weer. Daarnaast is beschreven in welke week (voor de begeleidingsbijeenkomst, zie verderop in het document) de conceptversie en de definitieve versie opgeleverd moet worden. Het product dient bij de beoordelaar ingeleverd te worden.

Product	**Soft deadline***	**Harde deadline**
Probleemstelling Samenwerkingscontract	Week 1 - dinsdag 24 april	Week 2 - dinsdag 8 mei
Plan van Aanpak	Week 2 - dinsdag 8 mei	Week 3 - dinsdag 15 mei
Functioneel ontwerp: • User stories - Requirements • Use-Case diagram • Use-Case beschrijvingen • Activiteitendiagram (use case overstijgend) • Grafisch ontwerp (inclusief spelregels) • Domeinmodel Test scenario's	Week 3 - dinsdag 15 mei	Week 4 - dinsdag 22 mei
Technisch ontwerp • Applicatiemodel • Implementatiemodel • Sequentiediagrammen • Toestandsdiagrammen Javadoc Java applicatie Testrapport	Week 5 - dinsdag 29 mei	Week 6 - dinsdag 5 juni
Java applicatie (fouten uit testrapport hersteld) Procesverslag Agenda's en Notulen	Week 7 - dinsdag 12 juni	Week 7 - vrijdag 15 juni
Presenteren (dit is niet een product, maar een activiteit. Zie voor meer informatie het rooster)		Week 8 – maandag 18 juni

* Een soft deadline is een deadline waarop je niet op wordt beoordeeld en dus ook geen Go of No Go krijgt. Het is een deadline om aan te geven dat jullie op schema lopen of juist niet.

4.2 Oplevercriteria

Alle definitieve producten dienen in verslagvorm** uiterlijk dinsdag om 23:59 van de week van oplevering via ELO aangeleverd te worden (zie bovenstaand schema). Voorbeeld: de definitieve versie van het functioneel ontwerp moet in week 4 opgeleverd worden. Dat houdt in dat deze uiterlijk dinsdag 30 mei om 23:59 ingeleverd kan en mag worden. Gebeurt dit niet op tijd, dan is het een NO GO voor het betreffende product(en). Je moet die week erop alsnog de betreffende product(en) opleveren.

** De applicatie (jar-files) en de code moeten in een zip worden aangeleverd en niet in verslagvorm.

Op woensdag beoordelen de iipsen-docenten jullie producten en geven een GO of een NO GO. Een GO betekent dat je samen op de groep op de goede weg bent en ga je door naar de volgende fase. Heb je een NO GO, dan betekent dat je product(en) onvoldoende zijn en dat je volgende week de product(en) nogmaals moet opleveren. Dit betekent dat je een week uitloopt.

Als je dit project nominaal doorloopt, dan ben tot en met week 7 bezig met je project. Je hebt daarnaast uitloop (lees herkansing) tot en met week 9. Dit houdt in dat je maximaal twee keer een NO GO kan veroorloven. Bij de

tweede NO GO krijg je je project niet voor de deadline van week 9 af en heb je automatisch een onvoldoende voor dit project.

Voor alle ontwerpproducten geldt dat alle modellen (onder een beschrijvend kopje) in een pdf-document, met een beschrijvende voorkant (groep, productnaam, versie, status, etc.) worden aangeleverd. Het opsturen van losse diagrammen wordt niet geaccepteerd!

Naamgeving documenten

Opgeleverde documenten moeten zich aan de volgende naamgevingconventie houden: *{Productnaam}_{Groepsnaam}_{versienummer}.* Ter verduidelijking staan hieronder de naamgeving van een aantal documenten voor groepje 1:

- Plan_van_Aanpak_Groep1_1.0.pdf
- Procesverslag_Groep1_2.0.pdf
- Functioneel_Ontwerp_Groep1_definitief.pdf

5. Samenwerking en werkverdeling

Ieder groepslid is verantwoordelijk voor minimaal 2 use-cases. Afhankelijk van de complexiteit van de use-cases kunnen dit er meer dan twee use-cases worden. De inhoudelijk begeleider verdeelt in samenspraak met de groep de use-cases nadat de use-cases zijn vastgesteld.

De verantwoordelijkheid bestaat uit het:

- Schrijven van de use-case beschrijving;
- Maken van de bijbehorende modellen;
- Maken van het grafisch ontwerp;
- Realiseren (programmeren) van betrokken klassen;

Binnen de projectgroep ligt de verantwoordelijkheid bij de aansluiting op elkaars werk en het gezamenlijk opleveren van de producten. Bij elk hoofdstuk moet aan het begin worden aangegeven wie dit heeft gemaakt. Ook al zijn meerdere hoofdstukken achter elkaar door één persoon gemaakt, dan moet dit nog steeds per hoofdstuk genoteerd worden.

6. Projectondersteuning

Het project kent zowel theoretische als praktische ondersteuning:

De theoretische kant wordt is door de onderstaande vakken ondersteund. Daarnaast zijn er ook workshops. Zie hoofdstuk 3.4.2 Workshops.

- Ten eerste door de module “IMUML – Moduleren in UML”. Deze module focust zich op het analyseren van softwarebehoeften, modellen en UML-diagrammen.
- Ten tweede door de module ”IOPR2 – Objectgeoriënteerd programmeren 2”. Deze module leert hoe je de applicatie in Java moet realiseren.
- Ten derde door de module "INET - IT Netwerkstructuren". Deze module leert hoe je een netwerk kan opzetten en verschillende computer met behulp van het TCP/IP protocol met elkaar kunnen communiceren.

Daarnaast is er ook de volgende ondersteuning door software:

- StarUML of Visual Paradigm (UML-tool te vinden op ELO onder de module Software Informatica);
- Tekenapplicatie (vrij te kiezen, bijv.: Paint, Paintbrush, Photoshop, The Gimp, etc.);
- Java Development Kit;
- Eclipse, IntelliJ of iets dergelijks(IDE).

7. Beoordeling

Indien een definitieve versie niet op tijd is ingeleverd, wordt deze met een onvoldoende of het cijfer 1 beoordeeld en wordt verder niet nagekeken. Je krijgt bij het inzagemoment dus ook geen feedback op dit product. De beoordeling per product gaat als volgt:

- Het plan van aanpak wordt beoordeeld op basis van de eisen gesteld in de cursus iprov.

- De samenwerkingsovereenkomst en het procesverslag moeten als voldoende worden beoordeeld om tot een eindcijfer te komen.
- Alle inhoudelijke producten worden beoordeeld met een cijfer. Er mogen geen onvoldoendes (cijfer < 5.5) worden behaald om tot een eindcijfer te komen.
- Het eindcijfer wordt op individuele basis bepaald aan de hand van het opgeleverde gezamenlijke en individuele werk.
- De onderstaande producten worden (gedeeltelijk) individueel beoordeeld. Zorg dat er duidelijk beschreven staat wie welke beschrijving, diagram, stuk code, etcetera heeft gemaakt. Anders kan de betreffende docent je geen cijfer geven en heb je een onvoldoende. Het gaat dus om de volgende producten:
    - Use-Case beschrijvingen
    - Activiteitendiagrammen (per use-case)
    - Sequentiediagrammen (per use-case)
    - Applicatie
    - Testscenario's en testrapport

Aan de onderstaande criteria moeten jij en je groep voldoende om een voldoende te krijgen voor dit project.

- Je moet actief gebruik maken van een versiebeheersysteem (bv. GIT).
- Je moet tenminste aan 10 workshops hebben deelgenomen.
- Je moet tenminste éénmaal een standup en éénmaal een standdown hebben gedaan.
- Je moet tenminste éénmaal door je groepsgenoten uitgeroepen zijn tot "meest krachtige in een bepaalde categorie".
- De groep moet deelgenomen hebben aan tenminste vijf van de inhoudelijke begeleidingssessies.
- De groep moet tenminste vijfmaal deelgenomen hebben aan de intervisiesessies.

Zie bijlage 2 voor meer informatie over de beoordeling.

8. Herkansing

Herkansen van het project is mogelijk door week 7 en week 9 van periode 4 te benutten om je producten op te leveren. Er zijn twee mogelijkheden op je applicatie te presenteren. Dit is in eind week 7 en eind week 9. Mocht je de deadline van week 7 dus niet halen, dan moet je sowieso in week 9 de applicatie presenteren. Indien er na de herkansing geen voldoende is behaald, zal het project het volgend studiejaar in zijn geheel moeten worden overgedaan.

Bijlage 1: Producteisen en beschrijvingen

Hieronder staan per product (opgedeeld per fase) de aanpak, eisen en aansluiting beschreven.

Fase 1: Probleemstelling

Gebruik voor het maken van je probleemstelling en samenwerkingscontract de literatuur die je hebt gebruikt bij iprov.

Fase 2: Projectopzet

Plan van Aanpak

De onderdelen die in het Plan van Aanpak terug moeten komen zijn te vinden bij de ELO course iprov. Maak dus gebruik van de literatuur die je ook hebt gebruikt bij iprov.

Voor alle ontwerpproducten gelden dezelfde eisen als gesteld binnen de module ondersteunende IMUML. Deze worden hieronder dan ook niet genoemd.

Use-case diagram

Een van de belangrijkste onderdelen van een softwareontwikkeltraject is het definiëren van gebruikers (actoren) en gebruikershandelingen (use-cases). Gebruikelijk is om deze schematisch weer te geven met de tussenliggende relatie. UML heeft hiervoor het use-case diagram gedefinieerd.
Om tot een use-case diagram te komen dienen van te voren de volgende vragen beantwoord te worden:

1. Welke actoren kent het spel?
2. Welke use-cases zijn er nodig om het spel te spelen?
3. Welke actor kan/mag welke use-case uitvoeren?

De antwoorden op de vragen dienen als directe input voor het maken van het diagram. Het diagram dient als basis voor de verdere uitwerking van het systeem. Op dit onderdeel wordt je als groep beoordeeld.

Use-case beschrijvingen

Om tot de realisatie van een softwaresysteem te komen dienen alle gebruikershandeling (use-cases) van start tot eind beschreven te worden. Immers kan een onvolledige beschrijving niet geïmplementeerd worden (of erger: leiden tot interpretatieverschillen). Het is dan ook de kunst deze stap in het ontwikkelproces zo zorgvuldig mogelijk uit te voeren.
Het onderstaande formulier geeft een goed format weer om tot een volledige use-case beschrijving te komen. Voor elke use-case in het use-case diagram dient een apart formulier ingevuld te worden.

Use case naam	Titel van de Use Case. Altijd werkwoord <meervouds vorm> + zelfstandig naamwoord (of andersom). Je mag ter **verduidelijking** iets toevoegen. Van elke use case moet een beschrijving worden gemaakt.
Auteur	Naam van het groepslid die de use case heeft gemaakt
Related requirement	Nummer of codering van een requirement
Goal in the context	Korte kernachtige uitleg van de use case (zonder scenario's) waarbij het doel wordt gehaald. (Wat ?)
Pre conditions	Voorwaarden om de use case te laten beginnen. Let er op dat er dit in verhouding staat met de primairy actor en de main flow.
Succesfull end condition	Beschrijving van eindresultaat.
Failed End condition	Beschrijving van het resultaat als het doel niet wordt gehaald.
Primary actor	De actor die de use case in beweging zet (Wie ?)
Secondary actor	De actor die meewerkt aan de use case. Vaan zij dat use case ten behoeve van de output van de use case.
Trigger	Actor geeft het hijsen zeil signaal.
Base use cases	In geval van een include of extends relatie wordt deze hier vermeld.
Main Flow	Alle zinvolle stappen op volgorde aangegeven voor het bereiken van de **'succesfull end condition'**. Dit wordt ook de happy flow genoemd.
Extensions	Scenario's die kunnen ontstaan bij het uitvoeren van de happy flow

Het resultaat is een volledig functioneel beschreven systeem. Deze beschrijvingen dienen als input voor het verdere ontwerp. Op dit onderdeel word je individueel beoordeeld. Geef dus bij elke use-case beschrijving duidelijk aan wie deze beschrijving heeft gemaakt.

Activiteitendiagrammen

Er dient één algemene activiteitendiagram gemaakt worden. Zie imuml presentatie 3a - Activity & Scenarios sheet 9.

Grafisch ontwerp

Een grafisch ontwerp kan worden gezien als een set van screenshots van het spel, voordat het daadwerkelijk gerealiseerd is. Het ontwerp dient een goed beeld te geven van de werking (samenhang use-cases en look-and-feel) van het spel en de informatievoorziening. Hierbij is het belangrijk dat de volledige applicatie is weergegeven (bijv. het menu, speelveld, spelregels, etc.).
Volg de onderstaande stappen om tot het grafisch ontwerp te komen:

1. Definieer alle schermen van de applicatie;
2. Definieer de stijl (look-and-feel) van de applicatie;
3. Werk alle schermen uit met een tekenapplicatie.

Aan de hand van dit ontwerp kan er later in het ontwerpproces bepaald worden welke grafische componenten opgenomen moeten worden.
Het ontwerp mag met een willekeurig tekenprogramma gemaakt worden. Zie modulewijzer voor mogelijke programma's.

Testscenario's

Een testscenario is een op een use-case beschrijving gebaseerde testbeschrijving. Één use-case beschrijving heeft meerdere scenario's. Al deze scenario's moeten worden beschreven in tests. Zodat nadat de applicatie gerealiseerd is deze testscenario's uitgevoerd kunnen worden en er een rapportage gemaakt kan worden. Op basis

van dit testrapport kan worden gezegd of alle functionaliteit die van te voren is opgesteld ook daadwerkelijk is gerealiseerd. Een template van een testscenario staat op ELO in de cursus iipsen. Lever elk scenario als een aparte spreadsheet op.

Fase 3: Realisatie

Twee domeinmodellen (eerste versie klassendiagram)

Het objectgeoriënteerde paradigma stelt ons in staat om een concept van de werkelijkheid om te zetten naar software. Hetzelfde geldt ook voor het spel. Dit model geeft alle domeinklassen met daarbij de tussenliggende relaties. Met domeinklassen worden alle losse concepten omgezet naar klassen die puur met het spel te maken hebben (dus niet met de applicatie!). De tussenliggende relaties geven de samenwerking tussen deze klassen weer.
Het domeinmodel wordt als volgt opgesteld:

1. Definieer alle domeinconcepten (incl. stereotype);
2. Definieer alle tussenliggende relaties (incl. naam en multipliciteit).

Dit model dient als absolute bases van de applicatie (het fundament).
Er dienen twee domeinmodellen gemaakt te worden. Één domeinmodel die gemaakt is met behulp van de brainstormmethode en één die is gemaakt met de zelfstandig naamwoordenmethode. Van de twee diagrammen die gemaakt zijn kies je één waarop je verder je gaat bouwen. Deze komt in het hoofdstuk domeinmodel. Het andere domeinmodel voeg je toe als bijlage aan het Technisch Ontwerp.

Toestandsdiagrammen

Gedurende het verloop van een applicatie kan deze in verschillende toestanden verkeren. Toestanden kunnen gelden voor de hele applicatie, een enkele use-case of slechts een klasse. Toestandsdiagrammen haken hier op in door deze toestanden te definiëren samen met de mogelijke overgangsvormen.
Een toestandsdiagram wordt als volgt opgebouwd:
1. Definieer toestanden (en eventuele subtoestanden);
2. Plaats initial node(s);
3. Plaats en definieer transities. Denk hierbij dat een toestand een entry, do en exit actie kan hebben.
Deze diagrammen dienen als input voor zowel de sequentiediagrammen als het implementatiemodel.

Sequentiediagrammen

Sequentiediagrammen geven heel concreet inzicht in hoe een use-case (Controller) wordt uitgevoerd door het systeem. Het diagram koppelt de betrokken actoren en klassen aan elkaar doormiddel van berichten (messages). Voor één use-case per persoon dient een sequentiediagram gemaakt te worden.
Regels voor het opstellen van een sequentiediagram:

- Begint met een message van de actor naar de uit te voeren Controller (use-case);
- Vanuit de Controller worden de businessklassen, factories en forms centraal aangestuurd (en,indien deze nog niet bestaan, gecreëerd).

Deze diagrammen dienen als directe input voor de realisatiefase.

Applicatiemodel (tweede versie klassendiagram)

Het applicatiemodel is een uitbreiding van het domeinmodel. In dit model worden, naast alle domeinklassen, ook de applicatiespecifieke klassen weergegeven. Met applicatiespecifieke klassen worden Forms (grafische elementen) en Controller (uitgewerkte use-cases) bedoeld. In dit geval moet ervoor de gekozen sequentiediagrammen (elke groepslid één) het deel van het applicatiemodel maken dat hoort bij het betreffende sequentiediagram/use case.
Volg de onderstaande stappen voor het maken van het model:

1. Maak een kopie van het domeinmodel;
2. Voeg aparte klassen toe voor de betreffende use-case als Controller;
3. Voeg aparte klassen toe voor elk grafisch component als Form;
4. Leg de bijbehorende relaties tussen de Controllers, Forms en Business klassen.
5. Verwijder de business en factory klassen die niet behoren bij het betreffende sequentiediagram

Het resultaat is een nieuw klassendiagram dat de relatie tussen domeinconcepten, use-cases en grafische elementen weergeeft.

Implementatiemodel (derde versie klassendiagram)

Het implementatiemodel is een volledig ingevuld applicatiemodel (met alle eigenschappen en methoden per klasse).
Om tot dit (zo goed als) volledige model te komen dienen de sequentiediagrammen als input. Het volgende stappenplan beschrijft het proces:

1. Maak een kopie van het applicatiemodel;
2. Vertaal elke message vanuit de sequentiediagrammen naar een methode van de bijbehorende klasse.

Het resultaat is een zo goed als implementeerbaar (realiseerbaar) klassendiagram.

Applicatie

De applicatie is de vertaling van de UML-diagrammen naar code (het realiseren, programmeren, van het spel). Tijdens de realisatie kan het mogelijk zijn dat niet alles uit te werken valt zoals in de modellen beschreven. In deze gevallen mag er afgeweken worden. Het is uiteraard niet de bedoeling om de modellen geheel los te laten en het spel op een compleet andere wijze te implementeren. Voor de beoordeling is de vertaling van de diagrammen belangrijker dan een werkend spel!

Fase 4: Oplevering

Procesverslag

Aan het begin van iedere week is er een reflectiepitch op maandagochtend. Voorafgaand aan deze pitch vult ieder groepslid het digitale weekverslag in waarin je kort maar krachtig antwoord geeft op de volgende vragen:
- Wat heb je afgelopen week gedaan?
- Hoe heb je dit aangepakt en waarom op die manier?
- Hoe verliep de samenwerking? Wat ging er goed en/of niet goed?
- Klopt de planning nog? Zo nee, wat zijn de aanpassingen?
- Wat is je planning voor komende week?

Alle weekverslagen voeg je uiteindelijk samen tot een procesverslag, dus bewaar deze goed.

Applicatie

De oplevering van het spel bestaat uit de volgende delen:

1. Het spel als executable JAR bestand;
2. Broncode in een apart ZIP-bestand.
3. Installatiehandleiding

Presentatie

Elke projectgroep dient aan het eind van het project (zie rooster) zijn geprogrammeerde spel en de werking hiervan te demonstreren.

Bijlage 2: Beoordeling

De matrices in deze bijlage geven een indicatie over de beoordeling van het iipsen project. Als ingangseis voor de beoordeling geldt dat alle deliverables tijdig en volledig aangeleverd zijn. Om een excellent (cijfer 10) te scoren voor een bepaald onderdeel moet je iets uitzonderlijks opleveren. Wat dat is laten we aan jullie als studenten over, maar uiteindelijk beoordeeld de docent. Hij wordt graag verrast met mooie extra's waar hij zelf niet aan heeft gedacht.

Onderdelen	Onvoldoende (0-5)*	Voldoende (6 - 7)	Goed (8 - 10)
Plan van Aanpak (5%)	Voldoet niet aan het iprov-template en/of is onvoldoende qua inhoud (bv. Onjuiste doelstelling) (NO GO)	Plan van Aanpak voldoet aan de eisen die gesteld zijn bij iprov-colleges	De planning is realistisch en gedetailleerd.
Functioneel Ontwerp (15%)	Voldoet niet aan drempel zoals gesteld zijn bij imuml (NO GO)	Voldoet aan drempelwaarde** Alle diagrammen zijn aanwezig sluiten op elkaar aan en op het spel. Vrijwel alle scenario's zijn beschreven in de desbetreffende diagrammen.	Grafische design is in overeenstemming met de diagrammen. De juiste symbolen zijn gebruikt in relatie tot Java RMI.
Technisch ontwerp (20%)	Voldoet niet aan drempel zoals gesteld zijn bij imuml (NO GO)	Controllers in lijn met het grafisch lay-out. (aan elke knop of overgang (Form) is een controller gemoduleerd). Sequence en activity diagram komen overeen Pre-condities in de UC zijn herkenbaar gemoduleerd. Methoden in het state-, sequence-, en classdiagram komen overeen.	Triggers in de UC beschrijving is herkenbaar gemoduleerd. Implementatiemodel heeft interfaces of abstracte klassen. SOLID principes zijn op meerdere plekken goed toegepast en bieden meerwaarde.
Realisatie (40%)	Applicatie werkt niet of niet goed en/of de uitslag van het assessment is onvoldoende. (NO GO)	De applicatie werkt volledig. Vrijwel elk scenario voldoet aan de opgestelde eisen. De code komt overeen met de gemaakte modellen. Er is sprake van een overzichtelijke indeling van geprogrammeerde klassen (views, controllers, klassen) binnen de applicatie (packages en\of directories). Uitslag van het assessment is voldoende.	Van alle superklassen en interfaces is een complete en uitgebreide Javadoc aanwezig State en sequentiediagrammen komen vrijwel geheel overeen met de code.
Testen (10%)	Er is geen rapport of de indeling is onjuist. De student heeft onevenredig weinig testscenario's uitgevoerd.	Er is een testrapport met juiste indeling Alle testscenario's zijn correct uitgevoerd en de resultaten hiervan zijn vermeldt in een testrapport. De student heeft een evenredig deel van de testscenario's uitgevoerd.	Het testrapport bevat een valide conclusie en bevat aanbevelingen over de kwaliteit van de applicatie. De uitgevoerde testscenario's zijn uitvoerig beschreven en het is zeer duidelijk welke stap welke uitvoer heeft geproduceerd.
Oplevering (5%)	Geen oplevering of niet conform gestelde eisen.	Spel is opgeleverd en er is een demonstratie van het werkende spel.	Er is een installatie handleiding gemaakt en deze is volledig en correct.
Proces & Reflectie (5%)	Het procesverslag is incompleet en/of geeft onvoldoende inzicht in de individuele bijdrage van de student en/of er is niet iedere week een weekverslag ingeleverd.	Het procesverslag is compleet en geeft voldoende beeld van de individuele bijdrage van de student. Daarnaast is er iedere week een weekverslag ingeleverd.	Het procesverslag is compleet, geeft een goed beeld van de individuele bijdrage van de student en de reflectie op het proces is van hoog niveau.