Regels voor Serious Gaming in Requirements Engineering

Auteur: Jan Vogels
Studentnummer: 0321389
Studierichting: Informatiekunde
Vakcode: I00082
Begeleider: Stijn Hoppenbrouwers
Versie: Definitieve versie
Datum: 2 mei 2012

Inhoudsopgave

Hoofdstuk 1: Inleiding

Aanleiding

Technologie en elektronica zijn in onze maatschappij niet meer weg te denken. Veel projecten in de IT-sector verlopen echter niet naar genoegen. Budgetten en tijden worden regelmatig overschreden. Een van de onderliggende redenen hiervoor is dat de producten niet voldoen aan wat de klant voor ogen had. Aanpassingen laat in het implementatietraject kosten veel geld en veel tijd. Dit is vaak te herleiden tot gebrekkig ontwerp. Dit komt doordat vaak niet duidelijk is wat de klant precies wenst.

Door een degelijk ontwerp te maken kunnen dit soort problemen voorkomen worden. Het is belangrijk om in de ontwerpfase helder te krijgen wat de klant precies wil. Aanpassingen kunnen in deze fase van het traject nog relatief eenvoudig en goedkoop worden doorgevoerd omdat er nog niets gebouwd is.

Het is te vergelijken met het bouwen van een huis. Als tijdens het metselen blijkt dat de afmetingen niet kloppen, omdat deze niet helder gecommuniceerd zijn, dan wordt het een dure klus om dit te corrigeren omdat het huis voor een deel er al staat. Als in de ontwerpfase wordt opgemerkt dat de afmetingen niet overeenstemmen met wat de klant wil, dan kan dit relatief eenvoudig opgelost worden, in het ergste geval moet de gehele bouwtekening opnieuw gedaan worden. Maar omdat er nog niets gebouwd zal dit relatief eenvoudig en goedkoop zijn.

Als het ontwerp in softwareontwikkeling voldoende aandacht krijgt, dan is de kans groot dat het uiteindelijke product ook (beter) aan de wensen van de klant voldoet. In de praktijk lijkt hier echter weinig van terecht te komen.

De ontwerpfase wordt door veel softwarebouwers als een noodzakelijk kwaad gezien. Men wil hier het liefst niet al te veel tijd aan besteden hoewel men zich bewust is van het feit dat deze fase belangrijk is. Een probleem van deze fase is dat er nog geen deliverables worden opgeleverd waar de klant iets mee kan. Men is meer gedreven om code op te leveren en daarom werkt men hier snel naar toe. Requirements engineering wordt hierbij vaak als last gezien, als een extra probleem in plaats van een stap richting de oplossing.[Kna2008] Maar als tijdens of na de bouw blijkt dat de klant het eigenlijk toch anders wilde hebben, dan zijn aanpassingen nodig.

Een goed ontwerp of model begint met helder krijgen wat de klant wil. Wat zijn de eisen en wensen van de klant? Dit is waar de discipline Requirements Engineering om de hoek komt kijken. Goede requirements engineers zijn echter schaars. Ook is de tijd die beschikbaar is om met stakeholders en klanten te praten vaak beperkt vanwege de drukke agenda van de laatstgenoemden.[Hop2010a]

Er bestaan geen goede tools om requirements engineers te ondersteunen in het daadwerkelijk achterhalen van wensen het eisen, het eliciteren van requirements. De meeste tools ondersteunen modelleren zelf en het maken van een syntactisch correct model, maar niet de totstandkoming van de inhoud.[Hop2010a]

Probleemstelling

Er is een behoefte om het elicitatieproces beter te ondersteunen. Deze ondersteuning moet zich richten op hulp bij de totstandkoming van de inhoud. Ondersteuning voor het correct opstellen van modellen zijn er reeds.

Binnen de vakgroep Model Based System Development (MBSD) binnen het Institute for Computing and Information Sciences (ICIS) aan de Radboud Universiteit Nijmegen heeft Stijn Hoppenbrouwers een nieuwe weg ingeslagen om dit te benaderen. Hij maakt daarbij gebruik van *"Serious Gaming"*, een methode die voorzichtig in andere vakgebieden al onderzocht wordt. Het onderzoek van Hoppenbrouwers kijkt onder andere naar het toepassen van speltechnieken in collaboratieve processen.

Het eliciteren van requirements in de vorm van stakeholder interviews is een collaboratief proces. Spellen zijn interactieve systemen waarin de spelers binnen een kader van regels en richtlijnen proberen tot een bepaald einddoel te komen. Dit kan worden toegepast op elicitatieprocessen binnen requirements engineering.

Door zulke gesprekken te benaderen als een spel kan men met spelregels zo een gesprek in goede banen leiden. Zulke spellen bestaan nog niet. Onderzoek in deze richting is heel nieuw en er zijn slechts enkele experimentele pogingen gedaan om Serious Gaming toe te passen in modelleren en requirements engineering.

Onderzoek

Ontwerpsvraag

Zelf willen we aan dit onderzoek bijdragen door in het kader van een bachelorscriptie een klein onderzoek in deze richting uit te voeren. Gezien de aard van het onderzoek zouden we de probleemstelling geen onderzoeksvraag willen noemen maar een ontwerpvraag. Deze ontwerpvraag luidt:

Ontwerp een spelprocedure met een set van regels, procedures en richtlijnen die een requirements engineer leiden in een spel, gericht op het verkrijgen van een use case Basic course of Events, use case triggers, use case preconditions, use case postconditions en use case scenario's.

Inperking & doel

Dit onderwerp wordt op een aantal punten bewust ingekaderd.

Het doel van het onderzoek is niet om een compleet werkend spel op te leveren. Het beperkte kader van een bachelorscriptie Informatiekunde leent zich hier niet voor aangezien het daar om een klein onderzoek gaat. Regels, procedures en richtlijnen vormen het hart van een spelen zijn daarom ook de focus van het onderzoek.

Om een set regels, procedures en richtlijnen op te stellen moet er wel een vorm van spel omheen zitten. Maar dit laatste wordt beperkt en simpel gehouden. Daarom geven we het liever de naam spelprocedure, om niet de indruk te wekken dat het maken van een volledig spel of volledige tool het doel is. Door een set van regels op te stellen zou het in de toekomst wel mogelijk kunnen zijn om een echt werkend spel te maken. Idealiter kan iemand het eindresultaat van dit onderzoek oppakken om een prototype te maken.

Het doel van het spel is een requirements engineer te ondersteunen in het verkrijgen van delen van een use case en use case scenario's. Eerder onderzoek in deze richting liet zien dat mensen zonder modelleerachtergrond vaker moeite hebben met abstractere zaken zoals use cases of modellen op typeniveau. Zulke mensen zijn meer vertrouwd met zaken op instantieniveau, concrete voorbeelden. Dit is precies wat use case scenario's zijn.

Een andere motivatie voor deze keuze is het feit dat er in deze richting nog geen onderzoek is gedaan en dit onderzoek een eerste aanzet gaat zijn. Aanvankelijk wilden we daarom het bewust op instantieniveau houden en ons alleen focussen op use case scenarios. Maar al snel werd duidelijk dat je er niet om heen komt om een abstractiestap hoger te gaan zitten, op niveau van de use case zelf. Ook blijken bij de overgang van instantieniveau naar typeniveau soms interessante dingen te gebeuren.

De set van regels zal alleen betrekking hebben op de requirements engineer, niet de stakeholder. Dit is wederom een bewuste keuze gezien de beperkte omvang van een bachelorscriptie. Het sturen van een stakeholder zal naar verwachting lastiger zijn omdat deze geen of weinig ervaring zal hebben in dit soort zaken, terwijl een requirements engineer zelf al een beter beeld heeft van hoe hij te werk moet gaan. De stakeholder hoeft in principe ook alleen te reageren op de requirements engineer.

Door de requirements engineer aan te sturen wordt de stakeholder indirect ook aangestuurd. De regels en richtlijnen zullen bedoeld zijn voor een requirements engineer met zekere ervaring. De heilige graal van deze onderzoeksrichting is een spel wat door een onervaren requirements engineer met een stakeholder gedaan kan worden of zelfs een volledig geautomatiseerd proces waar alleen de stakeholder nodig is en de taak van de requirements engineer hooguit bestaat uit het opstarten van het programma en het waken voor technische systeemfouten. Maar dit is verre toekomstmuziek, deze onderzoeksrichting bevindt zich nog niet eens in de kinderschoenen.

We richten ons op het verkrijgen van use cases en use case scenario's omdat use cases een aanpak is die in de bacheloropleiding naar voren is gekomen. De cursus Requirements Engineering van Stijn Hoppenbrouwers richt zich op deze methodiek. Het voordeel van een use case en use case scenario is dat deze in natuurlijke taal zijn opgesteld en daardoor makkelijker door een stakeholder gelezen en begrepen kunnen worden.

Use Cases beschrijven verder functionaliteit en niet een specifieke implementatie terwijl ze door het gebruik van natuurlijke taal makkelijker begrepen kunnen worden. Dit is een groot voordeel omdat vaak te snel in termen van implementatie gedacht wordt en daardoor de eigenlijke functionaliteit (wat moet er precies gebeuren) vaak vergeten wordt.

Het opstellen van de complete use case valt buiten de scope. Het doel is om uit de use case scenario's een aantal onderdelen van een use case te halen: triggers, pre-conditions, post-conditions en een Basic Course of Events. Deze vormen in mijn ogen de kern van een use case omdat ze gebruikelijke gang van zaken (Basic Course of Events), eisen vooraf (triggers en pre-conditions) en een eindsituatie (post-conditions) beschrijven.

Use Cases, en dus ook deze onderdelen, worden net als de use case scenario's opgesteld in natuurlijke taal, maar liggen een abstractieniveau hoger, namelijk op typeniveau. Ze zijn voor een door het gebruik van natuurlijke taal nog steeds te begrijpen, zeker met het bijbehorende scenario erbij. De rest van de use case laat ik buiten beschouwing voor deze scriptie omdat het gaat om een eerste poging. Verder onderzoek kan een bijdrage leveren om dit verder te verfijnen om een gehele use case op te stellen.

Hoewel uiteindelijk een deel van Use Case eruit moet rollen voegen we ook een stuk denken in termen van implementatie toe. Implementatie vormt in onze ogen namelijk wel een belangrijke rol in het proces van boven water krijgen van de functionaliteit (eliciteren). Implementatie ligt namelijk een abstractiestap lager dan functionaliteit en daardoor in het algemeen door stakeholders zonder modelleerachtergrond makkelijker te begrijpen. We willen als requirements engineers eigenlijk graag alleen in termen van functionaliteit denken, maar dat blijkt in de praktijk vaak toch lastig.

Methode & opbouw

Het werk wordt als volgt opgebouwd. Hoofdstuk 2 begint met een beknopt theoretisch kader waarin kernbegrippen als *Requirements Engineering, Use Cases, Game Design Theory, Serious gaming en Dialogue games* aangestipt worden. Dit deel zal geen uitputtende theoretische behandeling worden, over deze onderwerpen is genoeg geschreven en in de literatuurlijst zijn verwijzingen te vinden voor de lezer die dieper op deze zaken wil in gaan. Ik beperk me tot zaken die rechtstreeks relevant zijn voor het onderzoek. Hier komt ook het concept *Focused Conceptualisation (FoCon)* naar boven.

In hoofdstuk 3 volgt het eerste ontwerp gebaseerd op de theorie. Eerst wordt het basisidee achter de spelprocedure omschreven gevolgd door het eerste ontwerp. Dit ontwerp is de eerste versie van de spelprocedure zoals het uitgedacht is aan hand van de FoCon theorie.

Hoofdstuk 4 omvat de uitvoering van het spel. In totaal worden drie sessies gedaan waarin het ontwerp getest wordt. Aan hand van de ervaringen en feedback wordt het ontwerp verfijnt en opnieuw getest. Hoofdstuk 4 beschrijft de sessies en de evolutie van het ontwerp die daaruit volgt. De omschrijvingen zijn vrij gedetailleerd. In onze ogen is dit echter noodzakelijk omdat iets soortgelijks nog niet eerder gedaan is. Het hoofdstuk sluit af met een algemene evaluatie die nog eens kort samenvat wat we uit de sessies geleerd hebben.

In hoofdstuk 5 wordt het definitieve ontwerp van de spelprocedure uitgewerkt, aan hand van het verfijnde basisontwerp en de bevindingen uit de eerste sessie. Dit hoofdstuk vormt de kern en antwoord op de ontwerpvraag die we ons gesteld hebben.

Hoofdstuk 6 vormt het afsluitende stuk van het onderzoek. We bekijken kort wat we gedaan hebben in het onderzoek en koppelen dit terug naar de doelen die we ons gesteld hadden. Er volgt nog een korte evaluatie van het definitieve ontwerp aan hand van twee laatste testsessies. Ten slotte staan we stil bij verbetermogelijkheden en de mogelijkheden voor verder onderzoek in de toekomst in deze richting.

Hoofdstuk 2: Achtergrond/Theoretisch kader

Requirements Engineering

Kulak & Guiney omschrijven requirements als de behoeften van de gebruikers, ruwweg wat moet het systeem of programma allemaal gaan doen voor de gebruikers. Requirements Engineering is het boven water krijgen (eliciteren) van deze eisen en het formeel vastleggen hiervan. Kulak & Guiney noemen dit ook *requirements gathering*.

Requirements worden gedocumenteerd in een *requirements definition*. Op basis hiervan kan een systeem ontworpen worden wat voldoet aan de eisen van de klant. De requirements definition draait om functionaliteit.

Volgens Kulak & Guiney wordt de stap vaak gecombineerd met ontwerp van een mogelijke oplossing. Vervolgens wordt al snel meer aandacht besteed aan een mogelijke en minder abstracte oplossing waardoor het verkrijgen van requirements engineering snel een ondergeschoven kindje wordt. [Kul2004]

Pressman verfijnt het proces Requirements engineering in enkele stappen[Pre2010]:

- Inception
- Elicitation
- Elaboration
- Negotiation
- Specification
- Validation

Een dialoogspel voor requirements engineering, zoals we het voor dit onderzoek voor ogen hebben, richt zich op de elicitation fase, de fase waarin de requirements door middel van stakeholdergesprekken naar boven komen. Dit wordt in de volgende twee fases verfijnd, waarin in de fase *negotiations* ook prioriteiten aan wensen van verschillende groepen gegeven worden. In de fase specification wordt het uiteindelijke eindproduct opgeleverd, wat Kulak & Guiney een requirements definition noemen.

Kulak & Guiney stellen dat requirements gathering voornamelijk te weinig aandacht krijgt omdat er geen “flitsende” tools voor bestaan die dit proces goed kunnen faciliteren. Dit proces vereist creativiteit wat moeilijk te ondersteunen is, in tegenstelling tot bijvoorbeeld tools die helpen syntactisch correcte modellen op te stellen.

Er komt echter steeds meer aandacht voor requirements gathering. Kosten en ontwikkeltijd in veel projecten lopen hoog op omdat in latere fases van het project fouten geconstateerd worden. Aanpassingen kosten dan veel in tijd en geld. Aanpassingen in de requirements gathering fase zijn relatief goedkoper. Dit vereist een heldere manier van documenteren die teruggekoppeld kan worden naar de gebruikers en indien nodig bijgesteld kan worden.

Use Cases

Traditioneel requirements engineering schiet volgens Kulak & Guiney vaak te kort. Requirements blijken moeilijk om te zetten in een stuk code. Requirements gathering duurt vaak te lang, documenteert verkeerde dingen en maakt verkeerde assumpties. Als het proces te lang duurt, kunnen requirements al verouderd zijn voordat ze geheel zijn opgesteld.

De uitkomst laat vaak te wensen over. Vaak komt er een lange lijst met opsommingen uit voort. Het valideren van een dergelijke lijst door stakeholders is dan een lastige taak omdat deze stakeholders er tegen opzien een lange tekst met droge opsommingen te lezen. Soms worden de requirements ook als onbegrijpelijk ervaren wanneer men modellen of diagrammen presenteert. Kulak & Guiney dragen daarom de Use Case aanpak voor.

Een Use Case omschrijft één handeling die door het te ontwerpen programma of systeem uitgevoerd moet worden als reactie op een bepaalde handeling. De Use Case zelf blijft abstract op typeniveau en bevat geen implementatieoplossingen. De Use Case beschrijft wat er moet gebeuren, niet hoe het uiteindelijk gaat gebeuren. Dat is voor de latere design fase.

Voorbeeld: in plaats van "de gebruiker drukt op de OK knop" staat er in de Use Case "de gebruiker bevestigt de handeling".

Use Cases hebben een bepaalde opbouw die vastgelegd is in een Use Case template. Een Use Case template bevat de volgende zaken:

- Use Case Name
- Iteration
- Summary
- Basic Course of Events
- Alternative Paths
- Exception Paths
- Extensions Points
- Triggers
- Assumptions
- Preconditions
- Postconditions
- Related Business Rules
- Author
- Date

Als men Use Cases naast het model van Pressman legt dan beginnen Use Cases in de fase Elicitation, wanneer de wensen voor het eerst worden vastgelegd. Gedurende de volgende fases worden de Use Cases verder uitgewerkt. Als alles klaar is dan heeft men een onderdeel van de specificatie.

Voor dit onderzoek richten we ons alleen op de Basic Course of Events, triggers, preconditions, postconditions en use case scenario's. Deze vormen samen het hart van een Use Case.

Use Case Scenarios

Elke Use Case bevat tevens Use Case scenarios. Dit zijn concrete voorbeelden op instantieniveau. Aan hand van scenario's is het makkelijker voor stakeholders om een Use Case te valideren. Use Case Scenarios vormen een instantie voor de verschillende paden die genomen kunnen worden, de Basic Course of Events, Alternative Paths en Exception Paths. Een use case scenario vertelt stapsgewijs een verhaal.

Een groot voordeel van use cases en bijbehorende scenario's is het feit dat deze in natuurlijke taal geschreven zijn. Er komt geen verwarrende terminologie in voor en er staan geen modellen of diagrammen in waarvoor vakkennis vereist is. Valideren door de stakeholders is daarom makkelijker.

De stijl van de use cases leent zich tevens beter om te lezen, omdat het in essentie gaat om verhaaltjes. Lange droge lijsten nodigen niet uit tot lezen. Bij een dik document waar alleen opgesomd wordt wat er moet gebeuren of bij een document met veel technische termen en diagrammen zullen de stakeholders neigen om het maar gewoon door te wuiven onder het motto "hier is goed over nagedacht en zal wel goed zijn" waardoor er feitelijk geen validatie plaatsvindt.

De grootste uitdaging ligt in de fases elicitation, elaboration en negotiation. In deze fases wordt de inhoud van de Use Cases gegenereerd. Juist voor dit kritieke proces is er weinig ondersteuning omdat dit creativiteit vergt. Bestaande tools ondersteunen meestal alleen het modelleerproces, maar niet het tot stand komen van de inhoud. Deze processen zijn collaboratief van aard. De ontwerper zal moeten praten met verschillende groepen stakeholders. Zoals Pressman omschrijft verloopt die communicatie niet vlot. Problemen zijn ondermeer scope en het verzinken in niet relevante technische details (focus).

Een nieuwe kijk suggereert dat spellen een uitkomst kunnen zijn om een scope en focus te bieden. Hierbij wordt gesteund op onderzoek dat al gedaan is naar Serious Gaming.

Serious gaming

Spellen worden traditioneel gespeeld ter ontspanning en vermaak, met elkaar of alleen. Onder *games* wordt tegenwoordig vaak aan computerspellen (videogames) gedacht. Bord- en kaartspellen zijn echter de wereld nog niet uit.

Michael Zyda[Zyd2005] geeft een definitie voor videogames:
"A mental contest, played with a computer according to certain rules for amusement, recreation, or winning a stake."

Serious gaming is een relatief nieuwe discipline. Serious games voegen een extra dimensie aan het spel toe, namelijk een pedagogische factor. Het doel is om kennis of vaardigheden over te dragen. Zyda geeft de volgende definitie voor serious game:

"A mental contest, played with a computer in accordance with specific rules, that use entertainment to further government or corporate training, education, health, public policy, and strategic communication objectives."

Zelf vinden we deze definitie iets te beperkend. We laten de koppeling met videogames los en zoeken de koppeling met spellen in het algemeen. Bordspellen of spellen met pen en papier zijn immers ook spellen die gespeeld worden volgens bepaalde regels voor amusement. Deze spellen bestonden lang voordat de eerste videospellen uitkwamen. Daarom definiëren we een serious game in context van deze scriptie als afgeleide van Zyda's definitie:

"A mental contest, played in accordance with specific rules, that use entertainment to further government or corporate training, education, health, public policy, and strategic communication objectives."

Volgens deze definitie is de betrokkenheid van een computer nog steeds mogelijk maar worden andere speltypen niet uitgesloten.

Een bekend voorbeeld van een serious game is *America's Army*, een zogenaamde *First Person Shooter*, waarin de soldaat een rekruut van het Amerikaanse leger speelt. Dit spel is niet ontworpen als entertainment, maar als rekruteringsmiddel voor het leger. Entertainment is hierbij slechts een middel. Zyda vermeldt dat ten minste één trainingskamp van het leger het spel ook gebruikt bij het opleiden van de rekruten.

Een ander voorbeeld is het strategische oorlogsspel *Operation Flashpoint*. Met de onderliggende technologie en ervaring met het spel werd een militaire simulator gemaakt welke onder meer door de Koninklijke Landmacht gebruikt is voor training en opleiding.[DEF 208]

Serious games zijn niet alleen voor militaire contexten denkbaar. Simulatiespellen kunnen in allerlei richtingen worden toegepast, bijvoorbeeld marktsimulatie. De cursus *Kennismanagement* aan de Faculteit der Managementwetenschappen van de Radboud Universiteit maakt al jaren lang gebruik van een marktsimulatiespel als onderdeel van de eindbeoordeling.

Ook ten behoeve van modelleren is serious gaming in opkomst. Voor de discipline requirements engineering wordt ook onderzoek gedaan naar serious games die kunnen helpen. Een probleem voor ondersteuning van deze processen is dat er bij deze processen een zekere mate van creativiteit benodigd is. Daarvoor is het lastig gebleken ondersteunende tools en hulpmiddelen te maken. De hoop is dat serious games hier een oplossing kunnen bieden.

Serious gaming in modelling

In het dagelijkse leven wordt bij taken of activiteiten vaker metaforisch verwezen naar spellen, zoals "the game of politics" of "zich niet aan de regels houden" binnen een bedrijf. Deze activiteiten kunnen ook vaak aspecten van spellen vertonen, zoals competitie, scores, het bepalen van winners en verliezers, regels etc. Dit wordt de spelmetafoor (*game metaphor)* genoemd.

Deze spelmetafoor is over te dragen op modelleren. Hoppenbrouwers geeft hiervoor enkele argumenten. [Hop2009]

- Modelleren is nu typisch een taak van specialisten. Niet veel mensen in een organisatie zullen hiertoe in staat zijn. Maar om tot een goed model te komen is juist de input van veel mensen nodig. Om te zorgen dat alle relevante aspecten worden meegenomen moet het mogelijk gemaakt worden dat niet-specialisten in staat zijn een model op te stellen of bij te dragen aan het ontstaan van de modellen zonder dat dit tijdrovend en pijnvol gaat zijn. Spellen bieden een kader en regels waarin zich een activiteit moet bewegen en kunnen zo niet ervaren modelleurs sturen in hun taken.
- Door modelleren aan te pakken als een spel kan de motivatie van modelleurs verhoogd worden. Modelleren wordt niet als leuk ervaren. Door dit te doen in een spelvariant kan modelleren als minder saai ervaren worden. Het is waarschijnlijk een utopie dat modelleren ooit als leuk ervaren wordt, maar alleen al door het minder saai te maken kunnen meer mensen betrokken worden.
- Spellen kunnen helpen de kwaliteit van modellen te verbeteren. Door het beperkte kader en de regels worden mensen gedwongen in een bepaalde richting te werken en zich niet te laten afleiden door niet ter zake doende zaken.
- Spellen kunnen ook helpen in onderzoek naar modelgeoriënteerde interactiesystemen of systemen voor “collaborative modelling”. Een spel kan gespeeld worden door een groot aantal groepen met wisselende samenstelling waardoor makkelijker op grote schaal empirische data ingewonnen kan worden.

De meeste tools tot nu toe ondersteunen alleen het maken van correcte modellen, echter niet de totstandkoming van de modellen, de inhoud. Ze zijn daardoor vaak ook alleen door experts te gebruiken.[Hop2010a]

Serious gaming voor requirements engineering

Deze argumenten kunnen overgedragen worden op requirements engineering, of specifiek het maken van use cases met scenario’s. Use cases en use case scenario’s zijn ook modellen die moeten voldoen aan een bepaalde opbouw. Een spel kan helpen de inhoud van een use case met scenario boven water te halen.

- Net als modelleren wordt het eliciteren van requirements gedaan door specialisten. Deze zijn echter schaars.[Hop2010a] Door aan de slag te gaan met een spel wordt dit toegankelijker voor mensen met minder ervaring. Een spel zal dan gericht zijn op het ondersteuning van de facilitator, meestal de requirements engineering, die met een domeinexpert probeert de eisen boven water te halen. De heilige graal voor het veld Requirements engineering is een spel dat geen requirements engineer als facilitator benodigd en door de domeinexpert alleen gespeeld kan worden.
- Requirements engineering wordt als een tijdrovende en saaie klus gezien. Een spel kan het leuker maken, op zijn minst minder saai. Net als met modelleren is het niet realistisch om te verwachten dat een spel requirements engineering omvormt tot een leuke activiteit, maar het kan zeker helpen met de motivatie als het minder omslachtig en tijdrovend wordt.
- De kwaliteit van het proces kan verbeterd worden doordat kaders en regels van een spel de spelers dwingen zich te richten op het wezenlijke. Het is verleidelijk om buiten de scope te gaan of al in termen van implementatie te gaan denken. Regels kunnen hier helpen een kader te scheppen. Maar zeker bij requirements engineering moet er wel op gelet worden dat het spel niet te rigide wordt. Er moet ruimte zijn voor creativiteit en spontaniteit om alle requirements boven water te krijgen.
- Spellen kunnen ook gebruikt worden bij onderzoek naar het boven water halen van requirements. Men kan een spel gebruiken om verschillende groepen aan de slag te laten gaan (bijvoorbeeld groepen met mensen met een achtergrond in modelleren en ontwerpen en leken op dit gebied) en data vergelijken.

Game design theory

Voor het maken van dergelijke spellen wordt geleund op de Game Design Theory. Game Design Theory is een relatief nieuwe richting en moet niet verward worden met Game Theory. Deze laatste helpt bij het analyseren van strategieën voor het spelen, en winnen, van spellen. De Game Design Theory helpt vooral bij het analyseren en ontwerpen van regels voor spellen zonder het gedrag van de menselijke spelers hierbij te betrekken.

Järvinen geeft een raamwerk spelanalyse en spelontwerp en definieert negen categorieën voor belangrijke spelelementen[Hop2008]:

- Components
- Rule set
- Environment
- Game mechanics
- Theme
- Information
- Interface
- Players
- Context

Volgens Järvinen zijn ten minste de volgende elementen nodig om een spel te ontwerpen:

- Components: objecten die spelers kunnen manipuleren.
- Regels: deze geven aan wat de spelers mogen en niet mogen.
- Een informatiestructuur om alle spelrelevante informatie in op te slaan.
- Ten minste één game mechanic: een mogelijkheid om de components te manipuleren, zodat spelers iets te doen hebben.
- Een doel wat met de game mechanics bereikt kan worden.

Regels en doelen

Een belangrijk aspect bij het maken van regels bij serious games voor modelling is het stellen van een helder doel. Voor een modelleerspel noemen Hoppenbrouwers, Weigand en Rouwette[Hop2009] deze *goals for modeling* en maken onderscheid tussen twee soorten doelen:

- Utility goals: waarvoor is het model bruikbaar of waarvoor dient het modelleren?
- Modeling goals: subdoelen met betrekking tot details voor het modelleerproces in het spel.

Verder hebben ze enkele *key modeling goals* geïdentificeerd waarvan de belangrijkste zijn:

- Creation goals: wat voor deliverables (documenten, objecten, visualisaties etc.) moeten worden opgeleverd worden door het spel? Dit valt in de categorie utility goals.
- Grammar goals: welke taalregels moeten de gebruikers zich aan houden? Dit valt onder de modeling goals.
- Validation goals: welke overeenstemming is noodzakelijk over welke dingen tussen wie? Ook deze vallen onder modeling goals.

Met een theoretisch kader over requirements engineering, serious gaming en game design theory in het achterhoofd kan de volgende stap gezet worden naar een spel voor requirements engineering.

Focused Conceptualisation (FoCon)

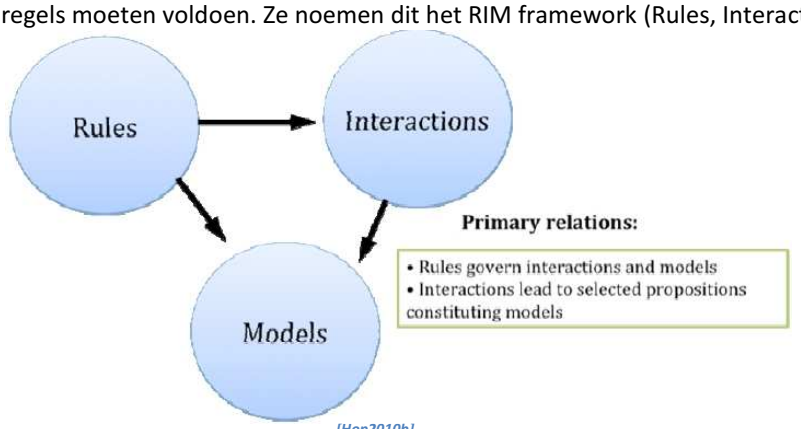
RIM-model

Hoppenbrouwers en Wilmont verrichten onderzoek naar het ondersteunen van het communicatieproces in Group Based Modelling. Zij komen hiervoor met *Focused Conceptualisations* (FoCons).[Hop2010b]

Zij hanteren een perspectief dat modelleren als een *constrained conversation* ziet. De motivatie voor dit onderzoek is het feit dat het modelleerproces zelf nog weinig ondersteund wordt en de meeste bestaande tools alleen door experts te gebruiken zijn, wat in lijn is met eerdere observaties.

Ze nemen een doelgedreven standpunt in bij modelleren: elk model moet toewerken naar een doel. Ze geven de volgende definitie aan modelleren in deze context: *"the purposeful creation of structured and coherent texts or graphical artefacts and subject to strong conceptual (and other) constraints."*

Hierbij zien ze modelleermethodes als interactieve systemen. Modelleren bestaat vooral uit interacties welke gestuurd worden door regels. Deze interacties leiden vervolgens tot modellen die eveneens aan bepaalde regels moeten voldoen. Ze noemen dit het RIM framework (Rules, Interactions, Models).



Afbeelding 1: Het RIM framework[Hop2010b]

Regels nemen een belangrijke positie in bij het RIM framework. Het RIM framework onderscheidt drie type regels:

- *Goal rules*: deze regels leggen het doel van het modelleerproces vast en zijn onder te verdelen in content, syntax, validation en argumentation rules.
- *Interaction rules*: deze leggen restricties op aan de interacties.
- *Procedural rules*: deze regelen de workflow van het proces.

De key modeling goals, zoals geïdentificeerd door Hoppenbrouwes, van Bommel en Järvinen, kunnen vertaald worden naar deze doelen. Creation goals zijn terug te vinden in de content goals en specificeren wat er moet worden opgeleverd. Grammar goals zijn terug te vinden in syntax goals en slaan zowel op de modelleersessie als de op te leveren modellen. Validation goals zijn terug te vinden in de validation goals van RIM.

Naast doelen definiëren ze een standaard set aan interacties:

- Propositions
- Questions
- Agreements
- Disagreements
- Arguments
- Clarifications
- Acceptations
- rejections.

Een set van regels en interacties vormen de basiscomponenten voor een Dialogue Game.

Focus Questions en Abstract Conceptualisation

Modelleren hangt sterk af van abstractievermogen. Er wordt onderscheid gemaakt tussen drie soorten abstract conceptualisations:

- generation of proposals
- classification of proposals
- selection of proposals.

Deze drie stappen kunnen tegelijk, maar afzonderlijk van elkaar gedaan worden. Vooral personen met weinig modelleerervaring zullen de stappen expliciet volgen terwijl ervaren modelleurs dit in 1 keer doen.

Bij het conceptualiseren kunnen twee soorten focussen onderscheiden worden:

Pragmatische focus: focus op informatie die relevant is voor het doel van het model. Mensen schijnen een natuurlijke aanleg hiervoor te hebben en kunnen omschrijvingen toespitsen op het doel.

Semantic-syntactic focus: focus op concepten die een voorgeschreven semantisch of syntactisch raamwerk passen. Deze focus is minder natuurlijk en richt zich niet op inhoud maar vorm en het concept classificatie.

De pragmatische focus zou leidend moeten zijn bij de keuze voor de modelleertaal. Men kan een semantisch-syntactisch volledig correct opstellen dat de foute dingen modelleert en daardoor niet te gebruiken is. De modelleertaal moet kunnen uitdrukken wat men nodig heeft.

De semantisch-syntactische focus is echter wel belangrijk omdat anders de mogelijkheid bestaat dat men zichzelf verliest in een wildgroei aan verschillende conceptualisaties. Dit uit te pluizen en te ordenen is een extreem lastige klus en vaak is het makkelijker om gewoon opnieuw te beginnen.

Inhoud van FoCons

Hoppenbrouwers en Wilmont komen met zogenaamde *Focused Conceptualisations (FoCon)*, mini dialogue games. Het FoCon concept is ontworpen om het denken over vragen en antwoorden te ondersteunen.

De auteurs omschrijven een FoCon als volgt: *een omschrijving van een communicatiesituatie waarin één of meerdere deelnemers een gefocust gesprek aangaan om een specifiek doel te bereiken zoals een gestructureerde abstracte beschrijving, een specificatie of een (deel van een) model.*

FoCons kunnen gebruikt worden voor een analyse van modelgeoriënteerde gesprekken of als raamwerk om zulke conversaties te leiden. Dat laatste maakt ze interessant voor mijn onderzoek.

Voor het gebruik hebben de auteurs een template opgesteld met informatie categorieën die een samen een FoCon vormen:

Korte omschrijving	Uitleg
Wat gaat er in	Soorten informatie en hun bron
Wat zou er uit moeten komen	Constraints, zowel pragmatisch als semantisch-syntactisch, op de resultaten, zowel tekstuele als andere zaken, bij voorkeur als modelleerdoel vastgelegd.
Types abstractie die in de activiteit gebruikt worden	Generation, classification of selection (apart of als één activiteit)
De specifieke focus vragen die gesteld worden (letterlijk)	Zowel de pragmatische als semantische-syntactische focus
De verschillende (soorten) deelnemers en hun competenties en expertise	Onvolkomenheden, andere relevante informatie over de deelnemers
Gegeven instructies en/of procedures, conventies en richtlijnen	Expliciet en impliciet indien relevant; nageleefd door de deelnemers (“rules of the game”)
Overige situationele aspecten of constraints	Bijvoorbeeld: gebruikte media, benodigde voorzieningen, organisatorische issues, sociale aspecten, politieke aspecten en alles wat relevant wordt geacht.

De auteurs gebruikten de FoCon methode om gevoerde interacties te analyseren. Ze bieden echter ook een basis voor het ontwerp van regels voor zulke interacties door vast te leggen hoe een interactie zou moeten verlopen aan hand van de informatiecategorieën. Door na te denken over wat er allemaal moet gebeuren kunnen er regels en richtlijnen naar voren komen die van belang gaan zijn. Voor het ontwerp van een dialoogspel als geheel zijn ze eveneens bruikbaar.

Dit hoofdstuk heeft een theoretische achtergrond geboden over de onderwerpen requirements engineering, de use case aanpak, serious gaming, game design theory, regels en doelen voor modelleren en de FoCon benadering. In het volgende hoofdstuk zullen we deze elementen gebruiken om een dialoogspel voor requirements engineering op te stellen.

Hoofdstuk 3: Basisidee & ontwerp

Doelen

Bij het ontwerpen van het spel ligt de nadruk op de regels. Het doel is geen kant-en-klaar spel op te leveren. Voor de regels zullen we gebruik maken van ondermeer Focused Conceptualisations.

Het spel zal een zogenaamd dialogue game zijn in de vorm van een vraag-en-antwoord spel. Voordat we beginnen met ontwerp van spel en regels moeten we de doelen van het spel duidelijk vaststellen. Hiervoor grijp ik terug naar de eerder genoemde literatuur.

We kunnen de doelen van het spel vertalen naar utility goals en modelling goals zoals deze door Van Bommel et al zijn gedefinieerd[Hop2010a]:

- Utility goal: het doel van het spel is om informatie te vergaren voor use case scenario's en een daarbij horende gedeeltelijke use case op te stellen. De velden basic course of events, preconditions, postconditions en triggers dienen ingevuld te kunnen worden.
- Modelling goals: het opstellen van correcte use cases volgens de use case template. In het kader van dit spel volstaan use cases met een BCoE, triggers en pre- en postconditions. Om het simpel te laten we alternative paths en exception paths buiten beschouwing.

Deze kunnen we verder verfijnen:

- Creation goals: use case scenario's voor het opstellen van use cases, specifiek de triggers, preconditions, postconditions en Basic Course of Events.
- Grammar goals: de spelers communiceren met elkaar in natuurlijke taal. De taal zal afhangen van de talen die de gebruikers machtig zijn, zowel gesproken als geschreven. We beperken ons in dit geval tot de Nederlandse taal.
- Validation goals: de requirements engineering moet use case scenario's opstellen die in lijn zijn met wat de domeinexpert voor ogen heeft. Validatie gebeurt door de opgestelde scenario's terug te koppelen met e domeinexpert en de vraag of deze inderdaad voldoen aan zijn denkbeeld.

In context van deze scriptie gaat deze dialoog tussen een requirements engineer (facilitator) en een domeinexpert (klant). We gaan uit van een facilitator met een degelijke kennis in requirements engineering en modellering met de bijbehorende abstractievaardigheden. Van de domeinexpert wordt geen modelleerkennis verwacht.

Ook in termen van abstractie verwachten we niet al te veel van de domeinexpert. Dit willen we aan de facilitator overlaten zodat de domeinexpert zich volledig kan richting op het creatief genereren van de inhoud. Het spel behandelt één use case. We kiezen voor een simpele aanpak omdat het hier om een eerste poging in deze richting gaat. Ook rolt er geen volledige use case uit het spel na afloop.

De FoConbenadering

We hebben besloten deze dialoog als een FoCon te zien, omdat FoCons proberen structuur aan te brengen in een dialoog.

Het behandelen van een use case in spelvorm is als één grote FoCon te zien. De input zijn de zaken die de domeinexpert aandraagt tijdens de dialoog. De output is een stapsgewijze omschrijving van wat er in de use case gebeurt, waaruit de requirements engineer een gedeeltelijke use case kan opstellen. De requirements engineering stuurt het proces aan hand van de regels en focus questions.

Om structuur aan het spel te geven verdelen we deze FoCon in meerdere subfocons. Om het geheel wat meer het gevoel van een spel te geven, noemen we het behandelen van één use case een *ronde*, naar spelrondes. Als meerdere use cases behandeld worden is elke nieuwe use case op te vatten als een één spelronde

Elke ronde is verdeeld in meerdere stappen, de subfocons. Deze geven we de naam *fase*. In bordspellen kan hier ook vaak de term "beurt" of "stap" voor gebruikt worden. Beurt klinkt te speels. Stap klinkt wat te verwarrend omdat we in elke fase werken aan een stapsgewijze omschrijving.

Het spel zal beginnen met een algemene omschrijving die de klant aan hand van vragen van de facilitator geeft. Aan deze omschrijving worden nog geen eisen gesteld behalve dat deze in chronologische volgorde moet staan.

Deze algemene omschrijving moet uiteindelijk verfijnd gaan worden zodat we inhoudelijk correcte en volledige en semantisch-syntaktisch correcte use cases kunnen maken. Hierbij onderscheiden we twee soorten van omschrijvingen: functionele omschrijvingen (functieomschrijvingen) en implementatieomschrijvingen.

Een functieomschrijving beschrijft de use case in termen van wat er moet gebeuren, los van een eventuele oplossing. Een implementatieomschrijving omschrijft de use case juist in termen van een oplossing. Beide omschrijvingen kunnen weer twee niveaus hebben, ze kunnen omschreven zijn op instantieniveau of een abstracter typeniveau. Instantieniveau is heel concreet terwijl er op typeniveau veralgemeniseerd wordt.

Een voorbeeld van een omschrijving op typeniveau is: *"Een student schrijft zich in voor een cursus",* terwijl instantieniveau specifieker is, bijvoorbeeld *"Jan schrijft zich in voor cursus Requirements Engineering"*. Een instantie is als het ware een voorbeeld. Een use case scenario is hier een perfect voorbeeld van.

Een Use Case Basic Course of Events ligt juist op typeniveau en richt zich op *wat?* en niet op het *hoe?*. Door uiteindelijk tot een functieomschrijving op beide niveaus te komen kan de facilitator een Basic Course of Events met bijbehorend scenario genereren.

Tijdens dit proces wordt beroep gedaan op abstractievermogen van de deelnemers. We onderscheiden twee soorten abstractiestappen: van concrete implementatieomschrijving naar abstractere functieomschrijving en van concreet instantieniveau naar abstracter typeniveau. Onderstaande tabel geeft dit schematisch weer.

Concreet	**Abstract**
Instantie	*Type*
Implementatie	*Functie*

Tabel 1: concrete en abstracte omschrijvingen

Het meest concreet is een implementatieomschrijving op instantieniveau, het meest abstract de functieomschrijving op typeniveau, waaruit we uiteindelijk de Basic Course of Events willen halen.

Hier tussenin liggen twee stappen die half concreet en half abstract zijn, de implementatieomschrijving op typeniveau en de functieomschrijving op instantieniveau. Het is niet duidelijk welke nu meer abstract is of dat de mate van abstractie in beide min of meer gelijk is. Zelf verwachten we dat een stap van implementatie naar functie meer abstractievermogen zal vergen dan de stap van instantie naar type.

Onderstaande tabel geeft de verschillende niveaus van abstractie per omschrijving weer.

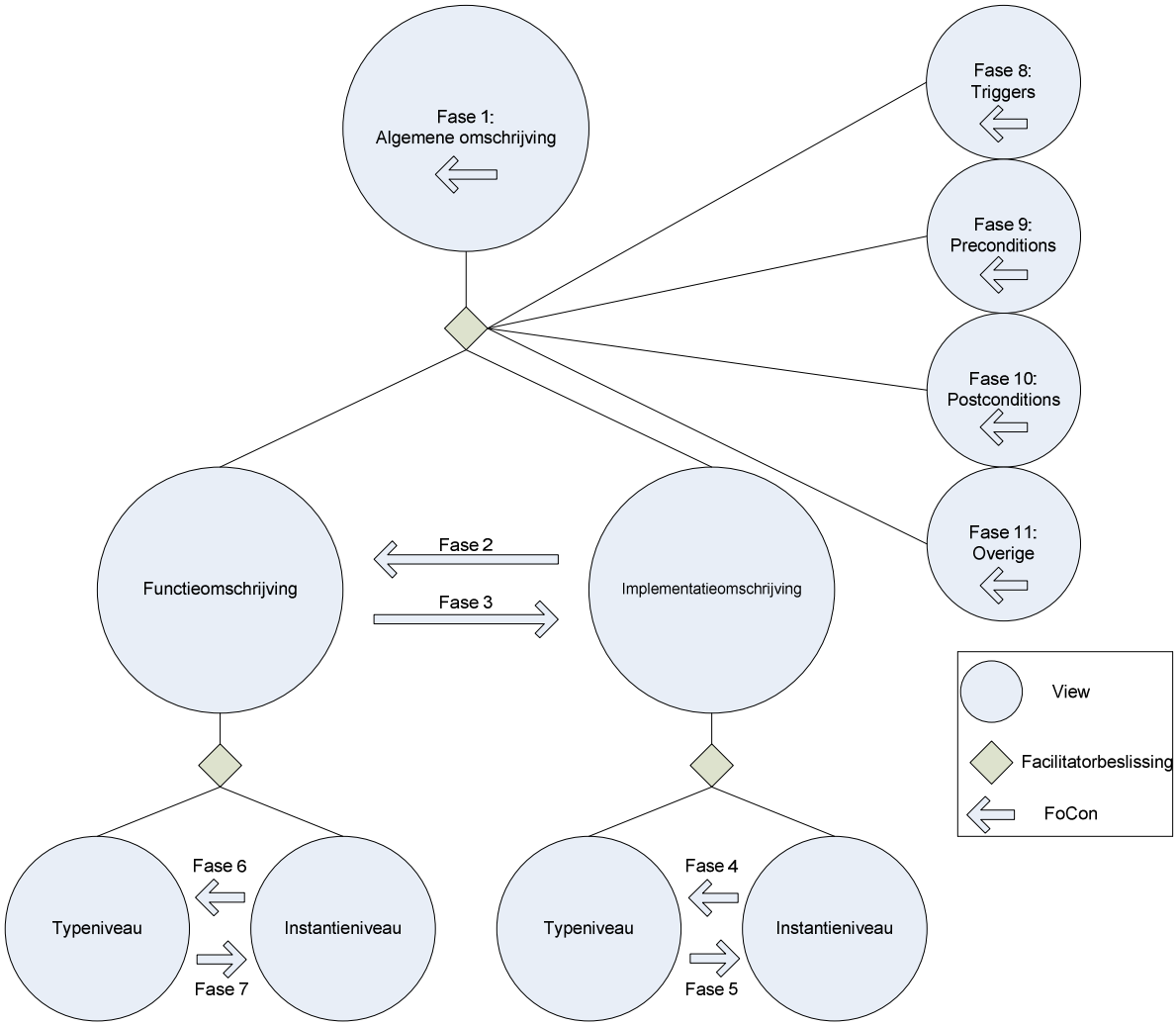
	Implementatie	**Functie**
Instantie	*Concreet*	*Semi-abstract*
Type	*Semi-abstract*	*Abstract*

Tabel 2: verschillende lagen van abstractie

Aan hand van de twee verschillende omschrijvingen en niveaus kunnen we in totaal elf *views* opstellen. De views zijn artefacten, objecten in het spel die door de facilitator gemanipuleerd kunnen worden, in overeenstemming met *components* uit de theorie van Järvinen.

- Algemene omschrijving
- Functieomschrijving
- Implementatieomschrijving
- Functieomschrijving op typeniveau
- Functieomschrijving op instantieniveau
- Implementatieomschrijving op typeniveau
- Implementatieomschrijving op instantieniveau
- Triggers
- Preconditions
- Postconditions
- Overige zaken

Deze worden in afbeelding 2 schematisch weergegeven.



Afbeelding 2: schema spelverloop van het eerste ontwerp

Elke cirkel staat voor een *view*. Tussen de lagen zit steeds een zogenaamde facilitatorbeslissing. De facilitator classificeert de in de vorige view gegeven omschrijving als een van de onderliggende views. Dit dient als input voor de onderliggende stappen. Bij pijltjes komen focusvragen kijken. Hier speelt het daadwerkelijke spel zich af. Elke pijltje kan daarom als een FoCon beschouwd worden en staat voor een spelfase.

Voor de implementatie van dit spel hebben we gekozen om gebruikt te maken van MS Excel. Microsoft Excel biedt ons de mogelijkheid om per use case een werkboek aan te maken met diverse tabbladen. Elke view wordt weergeven als een tabblad waarin de betreffende omschrijving wordt opgeschreven door de facilitator.

Alleen de facilitator mag aanpassingen doen. De klant heeft te allen tijde inzage en kan de facilitator vragen de views te tonen. We overwogen aanvankelijk met pen en papier werken, maar omdat veel kopieerwerk nodig gaat zijn hebben we uiteindelijk gekozen voor een computerprogramma. We verkiezen Excel boven tekstbewerkers als MS Word of MS Notepad omdat Excel de mogelijkheid geeft in één werkboek/bestand meerdere tabbladen te hebben om views mooi te ordenen.

De facilitator en klant zitten daarvoor samen achter een computerscherm zodat beiden de views kunnen zien, de domeinexpert ziet wat er met zijn input gebeurt en de facilitator om validatie kan vragen. Het spel wordt gespeeld door het schema te doorlopen.

Het is te alle tijden mogelijk om terug te gaan naar een voorgaande fase als de klant iets wil toevoegen. Deze optie willen we specifiek bieden omdat we het over een creatief proces hebben en de klant spontaan dingen kan bedenken die hij in een eerder stadium over het hoofd gezien heeft. Omdat we de basisfunctionaliteit naar boven willen halen is het belangrijk dat we dit toestaan zodat geen dingen onbelicht blijven omdat een rigide regelsysteem het niet toestaat.

Spelfasen

Elke view wordt in één spelfase doorlopen.

Fase 1: het maken van de algemene omschrijving

Het spel begint een algemene omschrijving die de klant geeft als reactie op vragen die de facilitator stelt. De klant wordt in deze fase zo vrij mogelijk gelaten en mag opnoemen wat in hem opkomt zolang het ook maar enige relevantie voor de use case heeft.

De facilitator moet de domeinexpert proberen niet te veel te belemmeren en alleen af te kappen als het gaat over zaken die er niet direct toe doen. Het idee hierachter is dat requirements elicitation een creatief proces is. Door de domeinexpert zo weinig mogelijk te reguleren hopen we dat deze zo creatief mogelijk is.

In deze fase van het ontwerptraject willen we zeker gaan dat we geen zaken over het hoofd zien. Schrappen kan altijd nog. Dingen toevoegen in latere fases wordt al lastiger. De facilitator documenteert wat de domeinexpert allemaal bedenkt in view 1 en brengt een chronologische volgorde aan in het geheel.

Vragen die de facilitator stelt zijn vragen als “waar begint u mee in dit proces?” en “waar in de huidige omschrijving komt deze stap?”. Deze stap wordt afgerond wanneer de domeinexpert aangeeft dat er geen verdere zaken zijn. Eventueel moet de facilitator hier navragen.

Indien niet gegeven gaat de facilitator nu op zoek naar triggers, preconditions en postconditions. Voor triggers vraagt de facilitator “waarom doen we dit?” Een andere mogelijke vraag is “Wie of wat start dit?”. Voor preconditions vraagt hij “zijn er bepaalde omstandigheden die moeten gelden voordat we dit gaan doen?” Voor postconditions vraagt hij “wanneer zijn we klaar?”. Andere mogelijke vragen zijn “wat moet de situatie zijn voordat we klaar zijn?” of “Wat moet bereikt worden?”.

Het kiezen van de “juiste” vraag is een lastige kwestie. Experimenteel moet gekeken worden met welke soort vragen het beste antwoord naar voren komt. Maar er zal nooit één perfecte vraag zijn. Elke communicatiesituatie is uniek doordat personen allemaal anders zijn. De uiteindelijke regels zullen dan ook sets van vragen bevatten die gesuggereerd worden aan de facilitator. Het is verder mogelijk dat dit soort zaken al aan bod zijn gekomen, dan kan deze stap worden overgeslagen.

Het type abstractie in deze fase is generation. Er is nog geen input, deze wordt in deze fase volledig bedacht. De uiteindelijke output is een algemene omschrijving van de use case.

Facilitatorbeslissing na fase 1

De uitkomst van deze fase is nu een stapsgewijze omschrijving. De soorten omschrijvingen en het niveau van de omschrijvingen kan in deze stap volledig door elkaar heenlopen. Dat is niet erg. De bedoeling is namelijk om de domeinexpert in deze stap nog zo vrij mogelijk te laten en nog geen beperkingen op te leggen door hem te laten denken in termen van alleen functie of implementatie. Het is onze mening dat anders al meteen in het beginproces waardevolle informatie verloren kan gaan.

De facilitator moet deze informatie categoriseren. Allereerst worden triggers, pre- en postconditions en eventuele overige zaken als dusdanig geclassificeerd en naar de betreffende views gekopieerd. Onder overige zaken komt alles wat de klant genoemd heeft, niet valt onder onze uiteindelijke deliverables maar belangrijk genoeg geacht wordt door de facilitator om ergens te documenteren. Men kan hierbij denken aan business rules. De facilitator voert in deze stap de abstractietypes classification en selection uit.

Wat overblijft, is een stapsgewijze omschrijving van het proces wat in de use case behandeld wordt. Deze categoriseert de facilitator als functieomschrijving of implementatieomschrijving en kopieert ze naar de betreffende views.

Fase 2: van implementatieomschrijving naar functieomschrijving

In deze fase gebruiken we fase 1 als input, of beter gezegd een door de facilitator geselecteerd deel van fase 1. Ook de implementatieomschrijving uit view 3 wordt gebruikt als input. We gebruiken view 2. Het doel van deze fase is het opstellen van een volledige functieomschrijving. Een deel is mogelijk reeds ingevuld.

Het overige deel zal moeten worden aangevuld vanuit de implementatieomschrijving. Deze moet daarvoor worden omgezet naar een functieomschrijving. De facilitator zal daarom de stappen uit de implementatieomschrijving (view 3) stap voor stap nalopen. De facilitator vraagt bij deze stappen “waarom doet u dit?”. Als de klant kan uitleggen waarom hij een bepaalde handeling verricht, dan wordt daaruit de functie duidelijk.

Voorbeeld: de facilitator vraagt “waarom haalt u uw pas door de scanner?”. De klant antwoort “Dat doe ik om mezelf bekend te maken”.

Het doel is dan blijkbaar “identificatie”. De facilitator kan dit terugkoppelen met de vraag “dus u haalt de pas door de scanner om uzelf bekend te maken?”

Zijn de facilitator en klant het eens over een nieuwe omschrijving van een stap, dan wordt deze stap in view 2 ingevuld. Stap voor stap worden alle stappen uit de implementatieomschrijving behandeld en de functieomschrijving bijgewerkt totdat deze compleet is.

Het is mogelijk dat de domeinexpert opeens inziet dat hij dingen vergeten is. In dat geval wordt teruggegaan naar view 1, de algemene omschrijving. Hier worden de aanvullingen gedaan. In de stap facilitatorbeslissing worden deze nieuwe stappen naar de betreffende views gekopieerd. Daarna gaat men weer verder met fase 2.

De uiteindelijke output van deze stap is een beschrijving in de vorm van een functieomschrijving. De abstractie in deze methode is generation en classification. Nieuwe inhoud wordt gegenereerd aan hand van de input. De focusvragen zorgen er in deze fase voor dat de omschrijvingen gegeven worden als een functieomschrijving, hierdoor vindt er ook (gedwongen) classification plaats.

Facilitatorbeslissing na fase 2

Als alle stappen in de algemene omschrijving een corresponderende stap in de functieomschrijving hebben, is deze klaar. De facilitator beoordeelt nu welke stappen op typeniveau en welke op instantieniveau liggen. Hij kopieert de stappen naar de onderliggende views. De facilitator voert in deze stap de abstractietypes classification en selection uit.

Fase 3: van functieomschrijving naar implementatieomschrijving

Deze fase gebruikt view 3 en heeft view 1 en 2 als input. De functieomschrijving in view 2 is compleet. View 3 omvat mogelijk al implementatiestappen uit view 1. De resterende stappen komen uit view 2 en worden stap voor stap omgezet.

Hiervoor stelt de facilitator vragen als “zou u een voorbeeld kunnen geven hoe u dit doet, op dezelfde manier zoals u de overige stappen beschreven hebt?”.

Hierbij dient opgemerkt te worden dat het implementatieniveau niet belangrijk is om compleet uit te werken. We zijn geïnteresseerd in de functies. We doen deze stap echter voor volledigheid. Als tijdens deze fase opeens stappen nieuw worden bedacht, dan springt men eerst weer terug naar Fase 1 om de algemene omschrijving aan te passen. Deze aanpassingen worden doorgegeven naar de onderliggende views.

Indien nodig wordt fase 2 eerst nog eens doorlopen om de functieomschrijving weer compleet te maken in het geval dat een of meerdere nieuwe stappen op implementatieniveau liggen. Als alle stappen gedaan zijn is fase 3 fase afgesloten. Als output is een beschrijving in de vorm van een volledige implementatieomschrijving opgeleverd. Net als in fase 2 vinden hier generation en classification abstracties plaats.

Facilitatorbeslissing na fase 3

Als alle stappen in de algemene omschrijving een corresponderende stap in de implementatieomschrijving hebben, is deze klaar. De facilitator beoordeelt nu welke stappen op typeniveau en welke op instantieniveau liggen. Hij kopieert de stappen naar de onderliggende views. De facilitator voert in deze stap de abstractietypes classification en selection uit.

Fase 4: van instantieniveau naar typeniveau in de functieomschrijving

Deze fase heeft als input view 2 en 5 en gebruikt view 4. De functieomschrijvingen uit view 2 zijn nu gescheiden naar typeniveau en instantieniveau.

In deze fase wordt de beschrijving op typeniveau volledig gemaakt door deze aan te vullen met stappen die op instantieniveau (view 5) zijn omschreven. In deze zaken moeten actoren gegeneraliseerd worden.

De facilitator stelt vragen als “Wat voor een soort persoon is dit?” of “Wie mag dit allemaal zijn?” of “Mag iedereen deze rol vervullen?”. Het doel is helder te krijgen wat voor soort actoren allemaal participeren in de use case, los van een eventuele implementatie.

In geval van een use case “geld opnemen” willen we hier omschrijvingen als “geldautomaat” graag vermijden. In geval van personen willen we weten wat voor personen dat precies zijn. In het geval van geld opnemen kan de facilitator bijvoorbeeld vragen “Mag iedereen geld opnemen?”

Ook in deze fase is het mogelijk dat opeens zaken boven water komen die de klant niet eerder bedacht heeft. In dat geval wordt teruggegaan naar fase 1, deze fase bijgewerkt en de onderliggende views bijgewerkt en de onderliggende fases weer doorlopen.

In deze fase vinden generation en classification abstracties plaats. Fase 4 heeft als output een functieomschrijving op typeniveau. Deze moet zo opgesteld zijn dat deze voldoet als input voor een Basic Course of Events van een use case.

Fase 5: van typeniveau naar instantieniveau in de functieomschrijving

Deze fase heeft als input view 2 en 4 als opgesteld en gebruikt view 5. De omschrijving typeniveau is af. De omschrijving op instantieniveau wordt nu compleet gemaakt door deze aan te vullen met stappen die alleen nog op typeniveau zijn omschreven.

Hiervoor worden de stappen uit view 4 stap voor stap doorlopen. Vragen die hier gesteld worden zijn “Kun je een concreet voorbeeld geven van dit type persoon?”. In plaats van “een klant” kan dan gesproken worden over “Stijn”.

Ook in deze fase is het mogelijk dat opeens zaken boven water komen die de domeinexpert niet eerder bedacht heeft. In dat geval wordt teruggegaan naar fase 1, deze fase bijgewerkt en de onderliggende views bijgewerkt en de onderliggende fases weer doorlopen.

In deze fase vinden generation en classification abstracties plaats. Fase 5 heeft als output een functieomschrijving op instantieniveau. Deze moet zo opgesteld zijn dat deze voldoet als input voor een use case scenario die past bij de Basic Course of Events die uit view 4 opgesteld kan worden.

Fase 6: van instantieniveau naar typeniveau in de implementatieomschrijving

Deze fase heeft als input view 3 en 7 en gebruikt view 6. Deze fase is gelijk aan fase 4, met als enige verschil dat deze in het vlak van een implementatieomschrijving ligt. Deze fase levert geen einddeliverable op maar dient alleen voor de compleetheid. Mogelijkerwijs komen in deze fase nog nieuwe dingen boven water waardoor het wel dient als mogelijke input voor de functieomschrijving.

In deze fase vinden generation en classification abstracties plaats. Fase 6 heeft als output een implementatieomschrijving op typeniveau

Fase 7: van typeniveau naar instantieniveau in de implementatieomschrijving

Deze fase heeft als input view 3 en 6 en gebruikt view 7. Deze fase is gelijk aan fase 5, met als enige verschil dat deze in het vlak van een implementatieomschrijving ligt. Deze fase levert geen einddeliverable op maar dient alleen voor de compleetheid. Mogelijkerwijs komen in deze fase nog nieuwe dingen boven water waardoor het wel dient als mogelijke input voor de functieomschrijving.

In deze fase vinden generation en classification abstracties plaats. Fase 7 heeft als output een implementatieomschrijving op instantieniveau.

Fase 8: triggers

Deze fase heeft als input view 1 en gebruikt view 8. In de stap facilitatorbeslissing na fase 1 zijn de relevante zaken van view 1 naar view 8 gekopieerd. Per genoemde trigger wordt gevraagd of dit klopt. Zo niet, dan worden deze bijgesteld. Als alle triggers doorlopen zijn, is deze fase klaar.

Deze fase heeft als output een omschrijving van triggers die dienen als input voor de triggers van de use case.

Fase 9: preconditions

Deze fase heeft als input view 1 en gebruikt view 9. In de stap facilitatorbeslissing na fase 1 zijn de relevante zaken van view 1 naar view 9 gekopieerd. Per genoemde preconditie wordt gevraagd of dit klopt. Zo niet, dan worden deze bijgesteld. Als alle triggers doorlopen zijn, is deze fase klaar. De zaken in deze view vormen de bron voor de preconditie van de use case Net als in de overige fasen geldt dat als hier nieuwe zaken naar boven komen dat de facilitator teruggaat naar fase 1 en alle views stapsgewijs per fase bijwerkt.

Deze fase heeft als output een omschrijving van preconditions die dienen als input voor de preconditions van de use case.

Fase 10: postconditions

Deze fase heeft als input view 1 en gebruikt view 10. In de stap facilitatorbeslissing na fase 1 zijn de relevante zaken van view 1 naar view 10 gekopieerd. Per genoemde postconditions wordt gevraagd of dit klopt. Zo niet, dan worden deze bijgesteld. Als alle postconditions doorlopen zijn, is deze fase klaar. De zaken in deze view vormen de bron voor de triggers van de use case. Net als in de overige fasen geldt dat als hier nieuwe zaken naar boven komen dat de facilitator teruggaat naar fase 1 en alle views stapsgewijs per fase bijwerkt.

Deze fase heeft als output een omschrijving van postconditions die dienen als input voor de postconditions van de use case.

Fase 11: overige zaken

Deze fase heeft als input view 1 en gebruikt view 11. In de stap facilitatorbeslissing na fase 1 zijn de relevante zaken van view 1 naar view 11 gekopieerd.. Hierin staan eventuele informatie die aan bod is gekomen maar geen plek heeft in de overige views. Elke stap wordt voor validatie doorlopen. Net als in de overige fasen geldt dat als hier nieuwe zaken naar boven komen dat de facilitator teruggaat naar fase 1 en alle views stapsgewijs per fase bijwerkt.

Deze fase heeft als output omschrijvingen die buiten de scope van een use case vallen maar volgens de facilitator belangrijk genoeg zijn om gedocumenteerd te worden. Deze view kan leeg zijn.

De spelronde eindigt als alle 11 fasen zijn doorlopen. Voor dit onderzoek eindigt dan ook meteen het spel omdat we ons richten op één use case. Een spel wat later echt gebruikt wordt springt dan naar de volgende spelronde, de volgende use case die op de agenda staat. Hierbij geldt dat men ook terug kan springen naar vorige spelrondes als nieuwe zaken boven water komen.

Net als terugspringen binnen de spelronde wordt dan teruggesprongen naar Fase 1 en allereerst deze bijgewerkt alvorens de onderliggende views bijgewerkt worden. Dit wordt geheel afgesloten voordat men met de huidige spelronde doorgaat zodat er geen twee “open spelrondes” zijn. Alle vorige spelrondes dienen afgesloten te zijn voordat met de huidige ronde kan worden verder gegaan.

Opzet

Als opzet voor het uitvoeren van het spel hebben we een Excel werkboek aangemaakt. Uit praktische overwegingen hebben we besloten views te gaan combineren omdat we anders te veel moet gaan springen tussen tabbladen. Views 2 & 3, 4 & 5, 6 & 7 en 8 – 11 worden gecombineerd zodat we uiteindelijk vijf views over houden:

- View 1: Algemene omschrijving
- View 2: Functie- & Implementatieomschrijving
- View 3: Functieomschrijving op type- en instantieniveau
- View 4: Implementatieomschrijving op type- en instantieniveau
- View 5: overige.

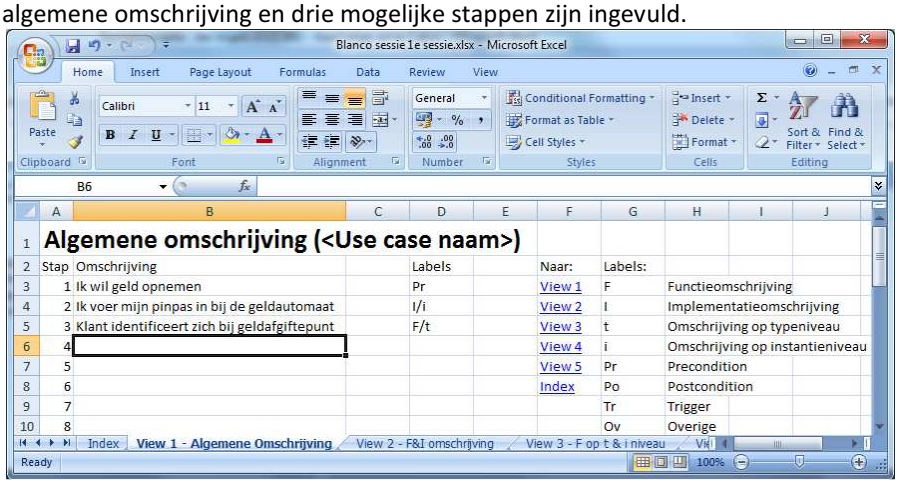
Het voordeel hiervan is dat elke FoCon (fase) binnen één tabblad kan plaatsvinden. Bijvoorbeeld bij het opstellen van de functieomschrijving uit de implementatieomschrijving hoeven we niet steeds terug naar het tabblad implementatieomschrijving te springen om een stap te kopiëren.

We kunnen beide omschrijvingen naast elkaar zetten waardoor het makkelijker te overzien of alles gedaan is. Beide omschrijvingen zouden uit evenveel stappen moeten bestaan en dat zou meteen duidelijk moeten zijn.

Elk tabblad is opgebouwd uit een aantal kolommen. De eerste kolom is een simpele nummering om stappen uit elkaar te ordenen en met elkaar te associëren. In de volgende kolom komt de omschrijving van de stap.

Een derde kolom geeft de facilitator de mogelijkheid labels aan te brengen waar nodig (F voor Functieomschrijving, I voor implementatieomschrijving, t voor typeniveau, i voor instantieniveau, Pr voor preconditie, Po voor postconditie, Tr voor trigger en Ov voor overige zaken. Er is nog een extra kolom om eventueel een notitie te maken.

Afbeelding 3 geeft laat een voorbeeld view van het oorspronkelijke ontwerp zien. Getoond is view 1, de algemene omschrijving en drie mogelijke stappen zijn ingevuld.



Afbeelding 3: screenshot van een view

Regels

Voor het verloop zijn al enkele regels aan bod gekomen, ze worden hier nog eens opgesomd:

- Het spel wordt gespeeld per ronde, één use case tegelijk
- De facilitator leidt het spel door gericht vragen te stellen
- Binnen een ronde wordt gespeeld per fase, één FoCon tegelijk
- Het is te allen tijde mogelijk om naar een vorige ronde terug te springen. Die ronde moet dan helemaal worden afgewerkt voordat men met de huidige ronde verder gaat.
- Het is te allen tijde mogelijk om naar een vorige fase terug te springen. Die en alle onderliggende fases moeten dan helemaal worden afgewerkt voordat men met de huidige ronde verder gaat.
- Een ronde begint altijd met fase 1, de algemene beschrijving.
- De overige fasen worden in volgorde afgewerkt.
- De facilitator heeft de mogelijkheid views te bewerken.
- De domeinexpert mag te allen tijde views bekijken.

Game Design Theory

Het bovenstaande ontwerp voldoet aan de minimale criteria die door Järvinen in de Game Design Theory genoemd worden.

- Components: de spelers, of in dit geval de facilitator, hebben enkele components om te manipuleren, namelijk de verschillende views (tabbladen).
- Regels: deze geven aan wat de spelers mogen en niet mogen. Deze zijn in de vorige paragraaf kort omschreven.
- Een informatiestructuur om alle spelrelevante informatie in op te slaan. Deze bestaat uit het Excel werkboek.
- Ten minste één game mechanic: een mogelijkheid om de components te manipuleren, zodat spelers iets te doen hebben. Ze mogen onderling met elkaar communiceren. De facilitator heeft daarbij ook de mogelijkheid om de views te bewerken door middel van een toetsenbord en een muis.
- Een doel wat met de game mechanics bereikt kan worden. Dit doel is het opstellen van omschrijvingen om een Use Case Basic Course of Events met bijbehorend scenario, triggers, preconditions en postconditions op te stellen.

Hoofdstuk 4: Uitvoering

Het ontwerp uit het vorige hoofdstuk hebben we vervolgens in de praktijk gebracht in drie sessies. In deze sessie traden we zelf op als facilitator. De interviewpartner trad steeds op als klant.

De omschrijvingen zijn heel gedetailleerd. De essentie wordt in de evaluatie kort samengevat. We hebben besloten toch een uitgebreide beschrijving van de sessies in de hoofdtekst op te nemen omdat het uiteindelijke ontwerp gebaseerd is op de ervaringen die in deze sessies zijn opgedaan. Een lezer die vooral in de evaluatie geïnteresseerd is kan de beschrijvingen overslaan. De beschrijvingen zijn in een kleiner formaat en schuingedrukt.

Sessie 1

De eerste spelsessie werd uitgevoerd zoals de in hoofdstuk 3 beschreven opzet. De sessie werd uitgevoerd met een voormalige studiegenoot. Als use case gebruikten we de handeling “Geld opnemen”. Voor de uitvoering zaten we gezamenlijk achter één PC in HG00.075. De ruimte was redelijk vol, maar dit vormde voor deze test geen belemmering. Voor echte sessies in requirements engineering wordt een afgesloten ruimte wel aanbevolen, maar voor deze test volstond deze setting.

Op de PC werd het Microsoft Excel werkboek geopend wat ik voor de gelegenheid had aangemaakt. Ik informeerde mijn interviewpartner dat dit een eerste poging was en het doel van de sessie was om de opzet te testen. Daarna gingen we aan de slag.

View 1: Algemene omschrijving (FoCon 1, sessie 1)

De facilitator legde het onderwerp van de use case voor aan de klant. Het ging over “geld opnemen”. De facilitator opende view 1 (Algemene omschrijving) en vroeg de klant te omschrijven wat hij doet wanneer hij geld gaat opnemen.

De klant begon met “Je gaat naar een geldautomaat”. Stapsgewijs haalden we stappen naar boven over geld opnemen bij een geldautomaat. Alle omschrijvingen werden gegeven in termen van “ik loop”, welke de facilitator noteerde als “je loopt”.

De stappen werden niet chronologisch gegeven. Na de stap “ik kijk welk opties er zijn” maakte hij de opmerking “Oh wacht, ik moet eerst mijn pas invoeren”. De stap “ik moet mijn pincode intoetsen” kwam pas helemaal op het einde. Bij elke stap vroeg de facilitator hem op welke plek deze moest komen zodat onze omschrijving wel op chronologische volgorde kwam.

Na een aantal stappen gaf de klant aan dat we klaar zijn. De facilitator vroeg de klant of er nog bepaalde zaken moeten gelden. Hieruit kwam naar voren dat je saldo op je bankrekening moet hebben. Ook zei hij dat het van invloed kan zijn of de geldautomaat waar je bij staat van een andere bank is dan de bank waar jij de rekening hebt. Hij kwam met een gedetailleerde uitleg hierover.

Deze gedachtegang heeft de facilitator uiteindelijk afgekapt nadat hij wel een korte notitie hiervan gemaakt heeft. De reden voor het afkappen was dat het uiteindelijk voor onze deliverables niet interessant is en te zeer in (implementatie)details gaat. Om te voorkomen dat we gingen verzanden in deze details, besloot de facilitator verder te gaan.

De facilitator vroeg hem vervolgens naar de reden waarom de klant dit doet. De klant gaf aan dat de reden om geld op te nemen was dat hij iets wilde kopen. De facilitator haakte na of dit altijd zo is. Uiteindelijk kwam naar boven dat de reden een behoefte aan contant geld was.

De facilitator vroeg hem vervolgens wat moet gelden als je klaar bent. Hierop antwoordde de domeinexpert dat het geld wat je opneemt ook van je bankrekening afgeboekt moet worden.

De omschrijving lag in zijn geheel op het gebied van implementatie en op instantieniveau. De facilitator nam hier voorlopig genoegen mee. De facilitator meldde dat hij een handeling moest verrichten om dingen te classificeren. Hij voegde labels toe en markeerde zo omschrijvingen die volgens hem betrekking hadden op implementatieomschrijving, triggers, preconditions, postconditions en overige zaken. Het hele stuk over handelingen aan de automaat werd gekopieerd naar view 2 onder implementatieomschrijving. De opmerking “Je hebt saldo nodig op jouw bankrekening”, “Het kan uitmaken of de geldautomaat van een andere bank is dan waar jij die bankrekening hebt.”, “Je hebt contant geld nodig” en “Geld wat je opneemt moet je van bankrekening afgeboekt worden.” werden gekopieerd naar view 5 (overige).

View 4: implementatieomschrijving op type- en instantieniveau (FoCon 6 & 7, sessie 1)

De facilitator besloot daarna ad-hoc af te wijken van het van te voren bedachte patroon en ging naar view 4, het uitsplitsen van de implementatieomschrijving in type- en instantieniveau. De reden hiervoor is dat de oorspronkelijke algemene omschrijving geheel op implementatieniveau lag. Dit uitsplitsen naar type en instantieniveau lijkt makkelijker, de abstractieslag is in de ogen van de facilitator net wat minder.

De omschrijving op implementatieniveau heeft de facilitator daarvoor gekopieerd naar view 4 en wel onder instantieniveau. De facilitator en de klant liepen stap voor stap de zaken af. De omschrijving ging uit van “ik”. Op de vraag of alleen de klant dit mocht doen antwoorde hij dat iedereen met een bankrekening en pinpas dit kan doen. We hebben besloten dit te generaliseren tot bankklant en verder achterwege te laten wat nu precies een bankklant is, dit valt buiten de scope van het spel in het onderzoek.

De klant realiseerde zich op dit moment dat de omschrijving niet compleet was. Ook de geldautomaat bleek handelingen te verrichten. De facilitator ging terug naar view 1 waar stappen voor de geldautomaat werden toegevoegd.

Toen de klant aangaf dat het overzicht compleet was, heeft de facilitator eerst view 2 bijgewerkt en daarna view 4. Verder is in deze stap weinig gebeurd. De beschrijving geldautomaat bleef staan omdat dit een implementatiespecifieke omschrijving is. Ook bij de beschrijving op instantieniveau is dit verder niet aangepast.

View 3: functieomschrijving op type- en instantieniveau (FoCon 4 & 5, sessie 1)

Vanuit view 4 ging De facilitator naar view 3. Hiervoor heeft hij de implementatieomschrijving op typeniveau gekopieerd naar view 3. De taak was nu de vertaalslag te maken van implementatieniveau naar functieniveau.

Een use case beschrijft uiteindelijk wat gebeurt en niet hoe. Om dit wat te achterhalen vroeg de facilitator naar het waarom. De vraag was dan ook steeds bij elke stap "Waarom doe je dit?". Bij "De bankklant gaat naar een geldautomaat" antwoordde de klant dat dit een plaats is waar geld opgenomen kan worden. Dit klonk heel triviaal en voor de hand liggend.

Zaken als "pas invoeren" werden gemakkelijk tot functie omgezet. Op de vraag "waarom voert de bankklant de pas in" antwoordde de klant "Zodat de automaat weet wie hij is". Zo konden we achterhalen dat de reden voor deze stap "identificatie" is. Stapsgewijs werd de omschrijving verder doorlopen.

Vervolgens stelde de facilitator de vraag of de beschrijving "geldautomaat" wat algemener kon omdat dit een implementatiespecifieke actor is. De domeinexpert gaf daarop te kennen dat dit ook anders zou kunnen, bijvoorbeeld bij de balie van een bank. Daarom werd "geldautomaat" hernoemd naar "geldafhaalpunt".

Enkele stappen werden samengevoegd. Zo werden de stappen "Klant voert pinpas in" en "Klant toetst pincode in" samengevoegd tot de stap "De bankklant identificeert zich bij het geldafhaalpunt" nadat de klant op de vragen "waarom gebeuren deze stappen" antwoordde dat het bij beide stappen om identificatie ging.

De klant merkte meteen op dat dit specifiek is voor de omschrijving die we in view 1 gegeven hadden. Identificatie bij een balie gaat waarschijnlijk anders. De stappen "Pas terug geven" en "Pas terug nemen" zijn daarom in deze omschrijving ook geschrapt omdat deze wederom specifiek voor de implementatie zijn.

De klant merkte op dit punt op dat de geldautomaat waarschijnlijk intern nog meer handelingen verricht. We hebben besloten dit verder niet te behandelen omdat de klant geen expert op dat gebied was. Specifieke handelingen binnen de geldautomaat beschouwen we als een black box. Een interview met een bankmedewerker levert misschien juist detailbeschrijvingen voor de geldautomaat en dus ook voor functionaliteit, bijvoorbeeld het controleren van rekeninggegevens en controleren of voldoende saldo aanwezig is.

De omschrijving uit deze stap is gekopieerd naar het instantieniveau binnen deze view. Hier veranderde bijna niets en deze stap werd door beide participanten ook een beetje als "flauw" ervaren. In een use case scenario wordt een abstracte actor benoemd. In plaats van bijvoorbeeld "Klant identificeert zich" zal in een use case scenario staan "Jan identificeert zich" staan. De domeinexpert besloot de bankklant "Henk" te noemen.

View 2: functie- en implementatieomschrijving (FoCon 2 & 3, sessie 1)

Met view 2 is weinig gedaan. De implementatieomschrijving is gelijk aan de implementatieomschrijving op instantieniveau in view 4. Omdat we ons voorgenomen hadden naar compleetheid te streven hebben we in view 2 nog de functieomschrijving ingevuld door de implementatieomschrijving te combineren met de functieomschrijving op typeniveau. Deze stap was erg geforceerd en beide deelnemers vonden het een beetje overbodig.

Evaluatie sessie 1

We vonden de sessie onbevredigend en bevredigend. De sessie verliep erg stroef en soms geforceerd. Dit leverde ons echter wel nuttige inzichten op. We hebben geconcludeerd dat de top-down benadering misschien niet de juiste aanpak is geweest. De algemene omschrijving lag geheel in het implementatievlak en op instantieniveau. We verwachten dat dit eerder de norm dan een uitzondering zal zijn met de meeste stakeholders.

Daarom lijkt het ons meer voor de hand te liggen om daar te beginnen. Volgens het oorspronkelijke ontwerp zou de algemene omschrijving opgesplitst worden in een functieomschrijving en een implementatieomschrijving. Beide zouden compleet moeten worden gemaakt vanuit de andere. Daarna zou de functieomschrijving uitgewerkt worden in type- en instantieniveau en daarna de implementatieomschrijving.

Zoals we al eerder beschreven, is deze laatste stap geen deliverable, maar zou alleen ter compleetheid gebeuren en in de hoop dat misschien nog zaken boven water komen. Tijd is echter een tegenargument om dit te doen. De sessie duurde ongeveer 75 minuten. Verder we ook niet tevreden met het combineren van FoCon's in één view, dit had te maken met veel scrollwerk. Op een computer met een breder beeldscherm is dit wellicht minder een issue.

Spelontwerp 2

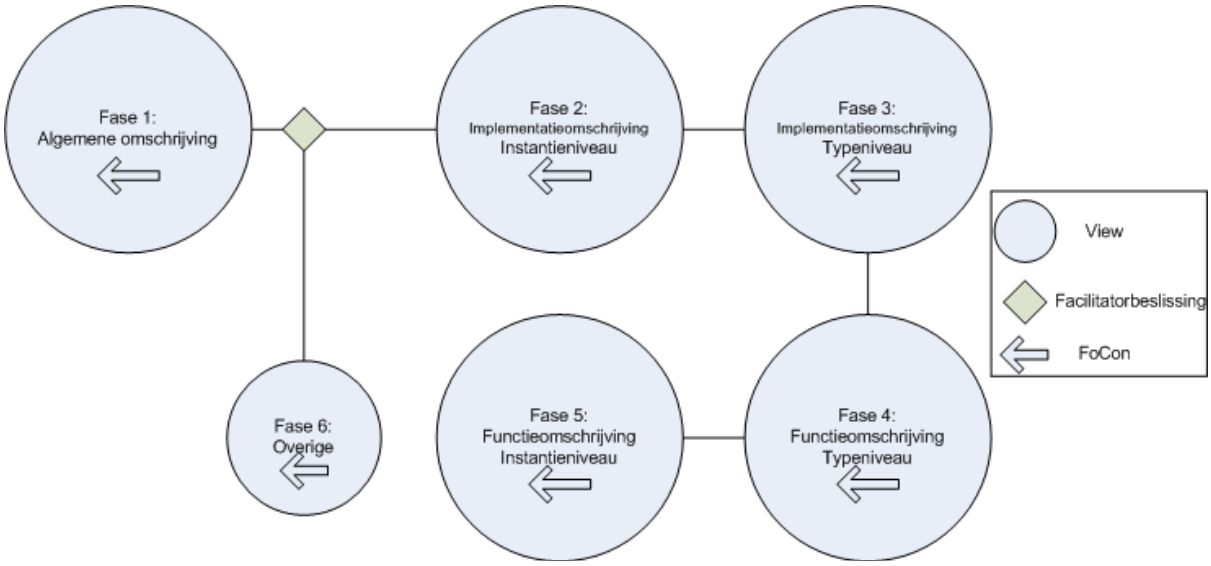
Op basis van de ervaringen van spelsessie 1 hebben we besloten de volgorde van spelfases aan te passen. In de eerste sessie zijn we al afgeweken van de volgorde. We hebben besloten fase 1 bij te behouden. Een ronde begint met een algemene omschrijving (FoCon 1) waarna de facilitator labels toevoegt en zaken categoriseert naar triggers, preconditions, postconditions, overige zaken en een type omschrijving.

De volgende fase wordt de implementatieomschrijving op instantieniveau (FoCon 2), omdat we verwachten dat de algemene omschrijving grotendeels op dit niveau zal liggen. Dan volgen de implementatieomschrijving op typeniveau (FoCon 3), de functieomschrijving op typeniveau (FoCon 4) en de functieomschrijving op instantieniveau (FoCon 5). Afsluitend volgt FoCon 6 waar triggers, pre- en postconditions en overige zaken nog bekeken worden. We hebben besloten deze zaken in een stap te behandelen omdat ze naar verwachting klein zijn en weinig extra werk behoeven.

Voor elke FoCon wordt een aparte view gemaakt. Ook werden de fasen niet langer behandeld in de vorm van "van implementatie naar functie". Hierdoor werd constant springen tussen tabs teruggedrongen. Omschrijvingen van de vorige fasen werden geheel gekopieerd.

In de tweede sessie zal een algemene omschrijving worden opgesteld. Deze wordt meegenomen naar fase 2 en omgezet in een implementatieomschrijving op instantieniveau. De opgeleverde beschrijving wordt steeds meegenomen naar de volgende fase en dient compleet te zijn voordat we verder gaan. Het is mogelijk dingen toe te voegen.

Net zoals in ontwerp 1 wordt dan teruggesprongen naar view 1 en de algemene omschrijving eerst geheel bijgewerkt. Daarna worden de overige views helemaal bijgewerkt totdat men weer terug is op het punt waar men gebleven was.



Afbeelding 4: schema spelverloop ontwerp 3. De lijnen geven de verplichte volgorde en inputstromen weer.

Sessie 2

Sessie twee vond plaats met een studiegenoot. Als use case gebruikten we de handeling “Geld opnemen”. Voor de uitvoering zaten we gezamenlijk achter één PC in de groepswerkruimte van de Library of Science. Op de PC werd het Microsoft Excel werkboek geopend wat ik voor de gelegenheid had aangemaakt.

We informeerden de interviewpartner dat dit een eerste poging was en het doel van de sessie was om de opzet te testen. Daarna gingen we aan de slag.

View 1: Algemene omschrijving (FoCon 1, sessie 2)

Net als sessie 1 werd begonnen met een algemene omschrijving in view 1. De klant kreeg de opdracht een scenario te omschrijven voor geld opnemen. Als eerste stelde de facilitator de vraag “waar begin je mee”. De domeinexpert gaf een omschrijving over een geldautomaat.

Toen de omschrijving af was stelde de facilitator de vraag “waarom doe je dit” om triggers te achterhalen. Hierop omschreef de klant een heel concreet scenario verhaal waarin hij contant geld nodig had voor een gedeelde pot met huishoudgeld.

Dit hebben we algemener kunnen maken nadat de facilitator nahaakte of dit de enige reden is of dat er nog ook andere redenen kunnen zijn. Uiteindelijk kwamen we op de omschrijving “ik heb contant geld nodig”. De omschrijving omvatte grotendeels stappen van de klant zelf. Ook zaten er enkele business rules tussen. De facilitator had zelf enkele verwachtingen maar heeft zorg gedragen niet te pushen in die richting en de klant zo vrij mogelijk gelaten.

Nadat de domeinexpert aangaf dat de handeling klaar is, kopieerde de facilitator bijdragen die niet aan een Basic Course of Events bijdragen naar view 6. De resterende beschrijving kopieerde hij naar view 2. Deze omschrijving lag geheel op het vlak van een implementatieomschrijving op instantieniveau.

View 2: Implementatieomschrijving op instantieniveau (FoCon 2, sessie 2)

In deze fase is bijna niets gebeurd. De omschrijving lag al geheel op dit vlak. De facilitator vond het derhalve niet nodig om deze nog eens stap voor stap te doorlopen. Dat werd in de vorige sessie als storend en overbodig ervaren. D de omschrijving werd gekopieerd naar view 3.

View 3: Implementatieomschrijving op typeniveau (FoCon 3, sessie 2)

De taak in deze stap was om de concrete actoren een abstractielaag hoger te krijgen. De facilitator opende view 3. Hij stelde de vraag of de geldautomaat perse een geldautomaat moest zijn of dat dit ook iets anders kon zijn.

Hierop antwoordde de klant dat niet perse hoeft. Hij gaf aan dat het ook mogelijk is contant geld “op te nemen” bij de balie van een bank onder bepaalde voorwaarden of door “bij te pinnen” bij een kassa in de supermarkt. De handelingen zijn dan echter anders. De facilitator heeft deze opmerkingen in view 6 opgeschreven. We hebben er verder geen aandacht aan besteed omdat ze er verder nu even niet toe doen.

Door de klant te vragen of we dan een generieke omschrijving voor geldautomaat konden vinden kwamen we op “geldafgiftepunt”. Omdat de handeling echter in dit geval specifiek op een geldautomaat is toegesneden (implementatieomschrijving) heeft de facilitator echter in deze view de tekst niet aangepast. Vervolgens vroeg de facilitator wie de ik-persoon in het verhaal kan zijn. De vraag aan de klant was of dit alleen de klant zelf zijn of dat ook andere mensen deze handelingen mogen verrichten.

De klant gaf aan dat deze persoon een klant van een bank moet zijn. Dit hebben we bankklant genoemd. Aan welke eisen deze klant moet voldoen hebben we achterwege gelaten zodat we ons konden richten op de omschrijving van de handelingen. De facilitator heeft de omschrijving gekopieerd naar view 4 en hierin geldautomaat vervangen door “geldafgiftepunt”.

View 4: functieomschrijving op typeniveau (FoCon 4, sessie 2)

De facilitator ging verder met view 4. Het omzetten van implementatieomschrijving naar functieomschrijving bleek een hele interessante stap te zijn. De vragen bij elke stap waren “waarom doe je dit?”. Zo wilde de facilitator de functie achterhalen.

De facilitator kwam er meteen achter dat dit niet voor alle stappen geschikt is. Een vraag “waarom loopt de bankklant naar een geldafgiftepunt” klinkt heel triviaal. Als een klant niet bij een geldafgiftepunt bent, dan kun je de handelingen niet verrichten. Bij de vraag “waarom voert de bankklant zijn pas in bij het geldafgiftepunt” kwam de klant met extra informatie. We concludeerden dat de omschrijving nog niet compleet was. Ook het geldafgiftepunt voert stappen uit.

Daarom sprong de facilitator terug naar view 1. De klant vulde omschrijving aan met enkele stappen die de geldautomaat onderneemt, zoals “Geldautomaat vraagt om pinpas”. De omschrijvingen waren wederom heel generiek. De facilitator heeft in view 1 de toegevoegde stappen gemarkeerd met een kleur om later terug te zien welke stappen in een later stadium zijn toegevoegd. Na de vraag of de omschrijving nu volledig is ging de facilitator naar view 2.

Deze werd aangevuld met de nieuwe stappen. Deze lagen al op het juiste vlak, daardoor ging de facilitator meteen naar view 3. Hier vroeg de facilitator of we voor de nieuwe stappen hetzelfde konden zeggen als voor de andere stappen. De klant antwoordde dat het nog steeds om dezelfde ik-persoon en dezelfde geldautomaat ging.

De extra stappen werden daarom herverwoord in dezelfde stijl. De facilitator ging vervolgens weer naar view 4 en kopieerde de nieuwe stappen op de juiste plekken en ging verder met het proces. Op de vragen “waarom wordt deze stap gedaan” en “wat is de functie hiervan” konden de overige stappen omgezet worden naar een functieomschrijving. Dit verliep verder soepel.
De afgeronde omschrijving werd naar view 5 gekopieerd.

View 5: functieomschrijving op instantieniveau (FoCon 5, sessie 2)
Deze stap werd geforceerd ervaren. Er was immers al een omschrijving op instantieniveau en een functieomschrijving. De facilitator ging de omschrijving stap voor stap door waarbij hij steeds vroeg of de klant een concreet voorbeeld kon geven, bijvoorbeeld een voorbeeld van een klant. Hij gaf de klant een naam, "Henk" waardoor "Bankklant" overal vervangen werd door Henk. Dit had in feite ook de ik-persoon uit view 2 kunnen zijn.

Evaluatie sessie 2

Deze sessie verliep een stuk soepeler. Door het aantal fases terug te brengen en opnieuw te rangschikken ontstond een natuurlijker proces in onze ogen.

De algemene omschrijving was weer een implementatieomschrijving op instantieniveau, dus vanuit daar werken was in onze ogen een juiste keuze. We hebben geforceerd fase 2 doorlopen, maar hier viel eigenlijk niets te doen. De testpersoon gaf aan deze fase een beetje overbodig gevonden te hebben en deze mening delen wij.

Fase 3 was wel nuttig, hier kwamen de types naar boven. Fase 4 was ook heel nuttig en hier kwamen ook extra stappen naar boven. De testpersoon gaf achteraf aan dat hij het wel een beetje als dubbelop ervaren heeft om de vorige fasen nog eens te doorlopen en compleet te maken. Ook deze mening delen we. Dit kostte veel tijd en voegde weinig toe in onze ogen.

Ook Fase 3 op zichzelf vonden we achteraf weinig toevoegen. In deze sessie had dit gemakkelijk met Fase 4 gecombineerd kunnen worden. Dit zou wat tijd gescheeld hebben en de testpersoon zei dat hij dan minder het gevoel zou hebben gehad dingen dubbelop te doen.

Fase 5 vond hij helemaal geforceerd en ook daar waren wij het me eens. Omdat we ons richten op een use case scenario, wilden we instantievoorbeelden hebben. Maar eigenlijk hebben we die al uit voorgaande fases.

We denken dat het beter is om voor deze fase een combinatie te maken van de functieomschrijving op typeniveau met de implementatieomschrijving op instantieniveau. Bij de omschrijving "De bankklant identificeert zich tegenover het geldafgiftepunt" hoort de implementatieomschrijving "Ik voer mijn pas in". Door in fase 5 nog eens om een concreet voorbeeld te vragen kan bij de klant een gevoel opkomen dubbelop bezig te zijn.

We denken dat het beter is om deze stap als facilitator zelf in te vullen en alleen ter verificatie voor te leggen aan de klant. Het enige wat immers moet gebeuren is een instantie geven voor de types in de omschrijving. Als dat gebeurd is kan een use case scenario bij de Basic Course of Events opgesteld worden.

Een algemene bevinding die we hadden, was dat het te rigide vasthouden aan het schema niet altijd bevorderlijk werkt. Deze sessie was weliswaar al een stuk korter dan de vorige, 45 minuten, maar gezien de geringe omvang vonden beide deelnemers dit wel aan de lange kant.

Spelontwerp 3

Met de bevindingen uit sessie 2 hebben we het spelontwerp wederom ter hand genomen. De volgorde van algemeen naar implementatie naar functie hebben we besloten te blijven hanteren. Deze werkte in de vorige sessies goed omdat de algemene omschrijvingen geheel in het implementatievlak lagen. Omdat onderzoek langzaam aantoont dat dit ook eerder de norm dan uitzondering is, lijkt het me relevant dit bij te behouden. Echter geven we de facilitator meer vrijheid.

De facilitator kan fasen overslaan als hij van mening is dat ze weinig zullen bijdragen. Ligt de omschrijving bijvoorbeeld inderdaad geheel op het vlak van implementatieomschrijving op instantieniveau, dan heeft het weinig zin om fase 2 stap voor stap te doorlopen.

Ligt de omschrijving echter al geheel op typeniveau, dan we ook niet het nut om fase 2 te doen. Het lijkt ons beter meteen richting het functionele te gaan. En in plaats van stap voor stap dan eerst de functieomschrijving op typeniveau af te werken en dan het geheel nog eens op instantieniveau kan besloten worden dit in één keer te doen. Hierdoor wordt minder dubbelop gedaan, dit werkt vooral voor de motivatie van de klant beter in onze ogen. Technisch gezien is het dan misschien handig om dit toch weer binnen één view te combineren.

Een ander punt dat we bij sessie 2 opgemerkt hebben dat het nogal tijdrovend was om terug te gaan naar vorige fasen als in een latere fase de klant met nieuwe informatie aan komt zetten. Door de vorige fasen nog eens helemaal te doorlopen zijn we extra tijd kwijt en ondernemen we stappen die eigenlijk niet meer nodig zijn. Als nieuwe stappen boven water komen in de functieomschrijving op typeniveau, dan is het onderliggende instantieniveau vaak al duidelijk.

Voor de einddeliverables is verder de implementatieomschrijving niet nodig. Als de facilitator van mening is dat de klant inmiddels vertrouwd genoeg is met de omschrijving, dan is het beter om gewoon in de huidige fase verder te werken en de vorige views niet opnieuw aan te vullen.

Als blijkt dat de klant toch moeite heeft met het omschrijven van de nieuwe stappen op dit niveau, dan kan men altijd nog terug springen en de klant vragen om het op een concreter vlak te omschrijven. We laten de strikte regels van het hanteren van de volgorde vallen, evenals de vereiste compleetheid van alle views. Deze zetten we om in *richtlijnen*.

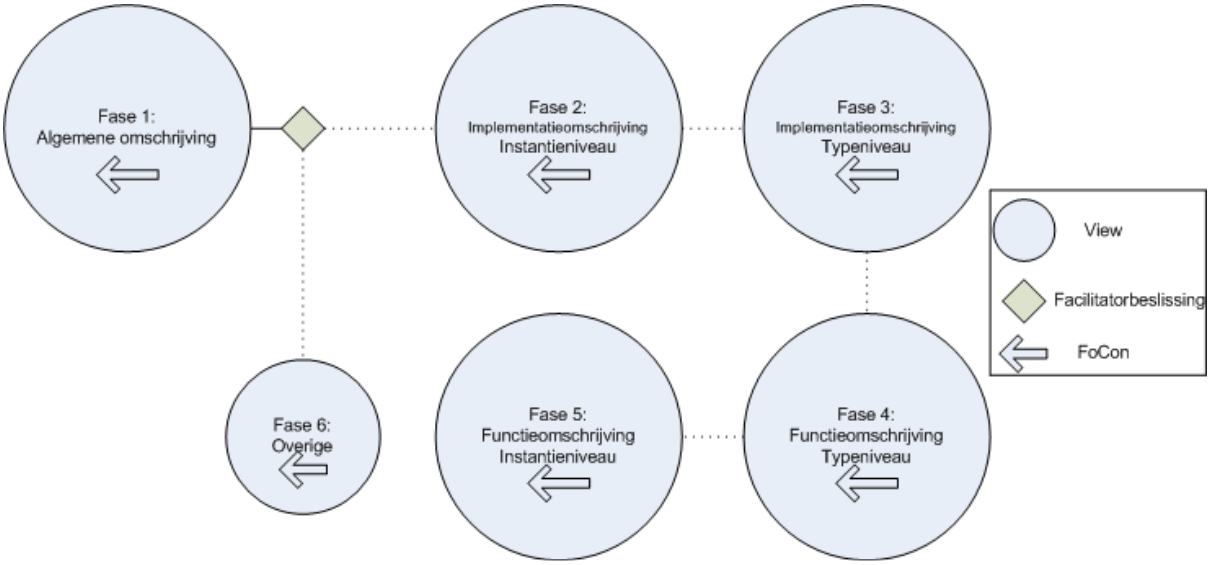
Als de facilitator denkt dat de domeinexpert een abstractieslag niet kan maken of dat de facilitator door onervarenheid zelf niet meteen weet hoe hij het beste verder kan gaan dan zullen de regels als richtlijn geven even een fase terug te springen naar een lager abstractieniveau. Acht de facilitator dit niet nodig, dan kan hij in de huidige fase blijven zitten.

Hierdoor krijgt de facilitator veel meer flexibiliteit. De regel om te beginnen met Fase 1: algemene omschrijving blijft wel staan als harde regel. De klant moet niet vanaf het begin in een bepaalde richting geduwd worden, we willen zijn creativiteit juist vrije loop laten.

Sessie 3

Deze sessie vond plaats met dezelfde persoon die deelgenomen heeft aan sessie 1. Daarom hebben we een andere use case uitgekozen. Als use case gebruikten we “boek inleveren bij een bibliotheek”. De reden waarom we deze persoon nog eens uitgekozen heb was de reden dat we van hem feedback wilden hebben over de nieuwe opzet. Dit keer werd de sessie gehouden op een groepswerkplek in de Library of Science, net zoals in sessie 2. Aangezien deze persoon bekend was met de procedure behoefde het geen extra uitleg vooraf.

Deze sessie verliep anders, zoals reeds in het spelontwerp naar boven is gekomen. We hebben de strakke volgorde losgelaten en gingen ons als facilitator laten leiden door de dingen die in het gesprek naar boven kwamen.



Afbeelding 5: schema spelverloop ontwerp 3. De stippellijnen geven een gesuggereerde volgorde en inputstroom aan.

View 1: Algemene omschrijving (FoCon 1, sessie 3)

Net als in voorgaande sessies begonnen we eerst met een algemene omschrijving in view 1. De vraag waar we mee begonnen was: "Wat doe je als je een boek gaat inleveren?".

Hierop kwam de klant met een beschrijving van een procedure met een inleverautomaat. In tegenstelling tot de voorgaande twee sessies was dit een omschrijving die we zelf niet verwacht had.

Nadat bleek dat de omschrijving klaar was vroeg we de klant "wat is de reden om dit te doen?". Het antwoord hierop was: "Ik heb een boek dat ik wil inleveren". Daarna vroegen we "wanneer is dit proces afgesloten?". Hierop antwoordde de domeinexpert "wanneer ik al mijn boeken heb ingeleverd". We vroegen als laatste "zijn er nog bepaalde dingen of regels die gelden?". Hieruit kwamen de omschrijvingen "De klant moet lid van de bibliotheek zijn" en "een eventuele boete wordt op een ander punt betaald".

De laatste vier dingen hebben we gekopieerd naar view 6 (overige). De rest van de omschrijving zat in het vlak van een implementatieomschrijving op instantieniveau. Deze hebben we daarom gekopieerd naar view 2. We hebben besloten deze niet nog eens stap voor stap na te lopen en eveneens fase 3, de implementatieomschrijving op typeniveau over te slaan om meteen naar fase 4 gaan, de functieomschrijving op instantieniveau.

View 2 veranderde daarom niet, view 3 bleef leeg. De reden om deze sprong te maken was omdat we de noodzaak niet voelden om fase 3 uit te voeren. De beschrijving was tot dat punt klein en helder. We verwachtten dat de omschakeling van implementatieomschrijving naar functieomschrijving lastiger is dan de omschakeling van instantieniveau naar typeniveau. Het leek ons daarom beter om deze stap eerst te maken.

Uit de voorgaande sessies leek dit ook de meest interessante stap te zijn. Het komen van instantieniveau naar typeniveau was meestal niet zo een probleem. Gezien de kleine omvang van de omschrijving dachten we beide zaken te kunnen combineren in één stap en gelijk naar fase 4 te gaan. Op deze manier kon in onze ogen tijd bespaard worden doordat dingen minder vaak herhaald worden.

Onze verwachting was dat als er nog extra dingen boven water zouden kunnen komen, dat dit in deze stap wel zou gebeuren, baserende op ervaringen van de vorige sessies.

View 4: Functieomschrijving op typeniveau (FoCon 4, sessie 3)

In deze fase gingen we stap voor stap de algemene omschrijving af. Hiervoor hebben we de omschrijving zoals deze naar view 2 gekopieerd was nogmaals gekopieerd naar view 4. De vragen waren "Waarom doe je deze stap?".

Specifiek, voor de eerste omschrijving "een klant loopt naar de inleverautomaat" klonk dit een beetje raar. De reden was "ik wil een boek inleveren". Dit is mogelijk een preconditie of zelfs een postconditie. Door na te vragen "waarom loop je specifiek naar een inleverautomaat?" kreeg we als antwoord "daar kan ik een boek inleveren".

We konden hierdoor de slag maken naar een type. De klant gaf aan dat een inleverautomaat een mogelijke variant is maar dat het ook anders zou kunnen, bijvoorbeeld bij een balie. We besloten het woord "inleverpunt" te kiezen.

Ook nu kwamen in de stap van implementatie naar functie extra stappen naar boven. Namelijk de inleverautomaat doet ook iets. Op de vraag "waarom leg je het boek in de inleverautomaat neer?" kwam het antwoord "omdat de automaat dit vraagt." Dit was een extra stap. Deze stap werd ingevoegd. Dit werd eerst in de algemene omschrijving gedaan, daarna werd ook view 2 bijgewerkt alvorens we verder ging met view 4.

Stap voor stap werd de omschrijving afgewerkt.

View 5: Functieomschrijving op instantieniveau (FoCon 5, sessie 3)
We hebben de stap naar een scenario (fase 5) weggelaten. Door de implementatieomschrijving op instantieniveau naast de functieomschrijving op typeniveau te leggen hadden we het idee dat we het scenario zelf konden invullen conform de eisen voor een use case scenario.

De klant hebben we daarom niet "lastig gevallen" door nog eens alle stappen te doorlopen en om te vormen naar een omschrijving die direct in een use case scenario past.

Evaluatie sessie 3

De testpersoon gaf in een korte discussie achteraf aan dat hij de keuze, om niet nog de fase functieomschrijving op instantieniveau te doorlopen, een goede keuze van onze kant vond. Hij vond het in de eerste sessie al lichtelijk overbodig als klant zijnde. In sessie 2 kregen we dezelfde opmerking. Dit vonden we een belangrijk leermoment.

De facilitator moet naast de keuzevrijheid in het doorlopen van het schema ook de mogelijkheid krijgen om de ronde af te sluiten als hij denkt dat hij voldoende heeft om een correcte Basic Course of Events met bijbehorend scenario te kunnen opstellen zonder dit scenario ook nog daadwerkelijk met de klant op te stellen. Dit scheelt tijd maar ook in motivatie voor de klant, wat zeker belangrijk gaat zijn wanneer meerdere use cases behandeld gaan worden.

In onze ogen vormde het ook geen probleem om fase 3 over te slaan. Door deze fase over te slaan verliep het proces vlotter. Desalniettemin kwamen er in fase 4 nog extra stappen boven water. We hebben daardoor ook het gevoel weinig tot geen relevante informatie gemist te hebben.

Door flexibel om te springen in het doorlopen van de fasen ging deze sessie redelijk vlot, het duurde slechts 20 minuten, in tegenstelling tot 75 en 45 minuten in de eerste twee sessies. Gezien het geringe aantal sessies kan echter niet stellen dat dit alleen komt door de vernieuwde aanpak. Deze use case in zijn geheel was kleiner en we kunnen ook niet uitsluiten dat het vlotter ging omdat het spel al eens met deze persoon gespeeld is, ook al waren de opzet en use case anders.

Concluderend waren we met deze sessie heel tevreden. De flexibiliteit gaf ons de mogelijkheid enkele dingen over te slaan die in onze ogen te repetitief waren en daardoor onnodig tijd zouden kosten. Dit zou ook de motivatie van de klant niet ten goede komen.

Algemene Evaluatie

Het voorgaande deel van dit hoofdstuk heeft de spelsessies beschreven. Voor elke spelsessie hebben we een korte evaluatie gegeven. De spelsessies hebben ons enkele inzichten gegeven voor een definitief ontwerp met regels. Deze inzichten worden hieronder samengevat en worden meegenomen voor de uiteindelijke set regels.

Omschrijvingen

De eerste sessie bracht meteen een gebrek in het eerste ontwerp naar boven. Bij het eerste ontwerp zijn we uitgegaan van een algemene omschrijving waar alle niveaus door elkaar lopen, zowel functie- en implementatieomschrijvingen op zowel type- als instantieniveau. Niets bleek echter minder waar.

De algemene omschrijving van de eerste sessie lag geheel op het niveau van een implementatieomschrijving op instantieniveau. Dit bleek ook in de volgende sessies zo te zijn. Dit hoeft natuurlijk niet altijd waar te zijn. Het aantal sessies is te klein om uitspraken te doen, maar het is te verwachten dat veel mensen op dit niveau denken niet heel gemakkelijk de slag naar een ander abstractieniveau maken. Meer onderzoek in deze richting is noodzakelijk om te kijken of dit klopt.

Volgorde

De gehanteerde volgorde in de eerste sessie voldeed in onze ogen niet. Deze gaat uit van een top-down benadering. Doordat de omschrijving echter geheel rechts onder in de pyramide lag was de aanpak enorm geforceerd door deze omschrijving eerst in een gehele functieomschrijving om te zetten en daarna aan hand van de functieomschrijving weer een implementatieomschrijving op te stellen.

Nadat de functieomschrijving was opgesteld was ook de slag naar een omschrijving geheel op typeniveau eerder frustrerend omdat alles herkauwd werd en alleen kleine wijzigingen werden doorgenomen. Dit bevordert motivatie van beide deelnemers niet.

De opbouw van het spel is daarom aangepast. Er werd een bottom-up benadering gehanteerd en gewerkt vanuit de implementatieomschrijving. Deze werd omgezet naar typeniveau. Vervolgens werd deze omschrijving omgezet naar een functieomschrijving op typeniveau en deze vervolgen omgezet naar instantieniveau.

Ook dit werkte niet geheel bevredigend. Vooral de laatste stap leek nutteloos omdat we immers al een functieomschrijving op typeniveau en een implementatieomschrijving op instantieniveau hadden. Deze stap was daardoor erg geforceerd, wat zo ook door de deelnemers werd ervaren.

In de derde sessie hebben we ons daardoor laten leiden door de soort omschrijving die gegeven werd, welke wederom een implementatieomschrijving op instantieniveau was. De stappen werden grotendeels gecombineerd en we gingen rechtstreeks naar een functieomschrijving op typeniveau. Deze werd met de implementatieomschrijving op instantieniveau gecombineerd tot de benodigde functieomschrijving op implementatieniveau.

De sessie verliep vlotter en minder geforceerd. Overigens dienen we op te merken dat we niet kunnen uitsluiten dat het feit dat de testpersoon al eens eerder deel had genomen geen invloed heeft gehad. Wel heeft deze persoon niet vooringenomen gereageerd en zijn kennis van requirements engineering naar achteren gedrongen om te reageren zoals een gewone klant dit ook zou doen.

Deze aanpak werkte in onze ogen bevredigend. Dit geeft een facilitator flexibiliteit. De regels moeten niet te rigide na worden geleefd als dit geforceerd en demotiverend gaat werken. De facilitator kan op basis van de gegeven algemene omschrijving zelf beslissen welke stappen hij onderneemt.

Door geen vaste volgorde voor te schrijven kan de facilitator afwijken van het patroon als de algemene omschrijving juist in de vorm van een functieomschrijving gegeven wordt en vanuit hier verder gaan. Daarbij is het niet eens noodzakelijk om elke fase uitgebreid door te voeren om tot een eindresultaat te komen. Dit komt de beleving van de deelnemers en de tijd ten goede.

Toch willen we de bottom-up benadering als suggestie meegeven. Als een facilitator niet goed weet hoe hij na de algemene omschrijving verder moet, bijvoorbeeld door onervarenheid of een omschrijving die daadwerkelijk op alle niveaus ligt, geeft dit patroon hem houvast en aan hoe hij verder moet. Is de facilitator ervaren genoeg, kan hij van het patroon afwijken. De keuze is hierbij aan hem. Een deel van de regels zouden we derhalve ook graag niet regels maar *richtlijnen* willen noemen.

Regels blijven wel bestaan, dit zijn dingen die moeten gebeuren, onder andere door afhankelijkheden. Zo moet altijd een complete algemene omschrijving als eerste gegeven worden. Ook blijven we bij het principe van één use case tegelijk. Springt men terug naar een vorige use case, dan moet deze eerst volledig afgewerkt worden. Maar binnen de use cases krijgt de facilitator dus meer vrijheden.

Verwerken
De manier van dataverwerking was soms wat omslachtig. Maar dit komt omdat we ons niet zo zeer op het spel zelf richten maar op het proces. Ons gaat het om het wat, niet om het hoe. Dit spel had ook met pen en papier of een geavanceerde applicatie gespeeld kunnen worden.

Er kwam veel kopieerwerk van toepassing. Er kwamen soms ook momenten van stilte, vooral tijdens de stap facilitatorbeslissing. Dit is in onze ervaring onze ervaring geen goed iets. Een stilte kan dodelijk zijn als men werkt met een klant, die dan even niets te doen heeft, zich af laat leiden en misschien nutteloos voelt. Een tool die het verwerken van gegevens beter ondersteund kan hier een oplossing bieden. Dit is in het kader van deze scriptie niet gedaan omdat het buiten de scope viel.

Er vielen ook pauzes waarin we op papier aantekeningen maakten over de verloop zelf. Deze laatste soort van onderbrekingen zal in sessies voor requirements engineering niet optreden omdat het doel dan puur en alleen requirements engineering is en niet testen van de spelprocedure. We hadden de geïnterviewde vooraf hierover moeten informeren. Dit hebben we in sessie 2 en 3 gedaan en dit werd positief ontvangen opdat men wist wat er te wachten stond.

Hoofdstuk 5: Definitief ontwerp

Nu is het tijd geworden voor het definitieve ontwerp, waarbij de nadruk op de regels en richtlijnen zullen liggen. Het ontwerp en de regels zijn afgeleid van het 3e en laatste ontwerp dat in hoofdstuk 4 gegeven is aangevuld met de bevindingen van de 3e sessie.

De nadruk ligt niet zozeer op regels (strikte volgorde, volledige compleetheid) maar op richtlijnen om de facilitator flexibiliteit te geven. We beschrijven de spelregels los van een implementatievorm. Hierdoor zijn de regels makkelijker toe te passen op een andere vorm van het spel, bijvoorbeeld een speciaal ontwikkelde tool of pen en papier.

Net als bij requirements engineering is het "wat" belangrijker dan het "hoe". De regels hebben betrekking op een spel dat gespeeld wordt door één facilitator en één domeinexpert. Hoewel we het over een "spel" hebben, proberen we de zaken net iets formeler te verwoorden. Zo zullen we het niet hebben over spelers van deelnemers.

De regels zijn toegespitst op de facilitator. Voor de klaten zijn geen expliciete regels opgenomen. De regels richten zich op een spel dat één use case behandelt. Schuingedrukt staan aanbevelingen voor de facilitator. Deze zijn op te vatten als richtlijnen.

Bij de regels van het spel wordt de koppeling met het Excel werkboek losgelaten. De spelregels zullen alleen omschrijven wat er moet gebeuren, niet hoe het uiteindelijk uitgevoerd wordt. Een ontwikkelaar die een applicatie wil bouwen kan hierdoor onbevangen aan het werk gaan.

De opbouw van de spelregels is geïnspireerd door de spelregels van het bordspel "De Kolonisten van Catan". Dit bordspel is zeer populair en de regels van dit spel zijn kort maar goed opgezet. De spelprocedure bestaat uit een spelronde, kortweg ronde, welke opgebouwd is uit meerdere fasen. Als een spel meerdere use cases bevat, dan zouden meer rondes gespeeld worden.

De regels voor onze spelprocedure bestaan uit:

- Opzet: het begin van het spel. Dit is vergelijkbaar met een uitleg over hoe een speelbord is op te bouwen, hoe beginopstellingen gedaan moeten worden etc.
- Rondeverloop in het kort: dit geeft een kort overzicht van alle fasen in een ronde ter overzicht.
- Rondeverloop in detail: het rondeverloop wordt nu in detail beschreven, per fase. Dit vormt de kern van de regels en bevat veel richtlijnen om een facilitator te sturen.
- Einde van het spel: deze sectie omschrijft waar aan voldaan moet zijn om een spel te beëindigen. Dit is vergelijkbaar met de overwinningscriteria in een bordspel.
- Kort voorbeeld: er wordt een kort voorbeeld gegeven ter illustratie van de regels.

Opzet

Aantal deelnemers: 2, een facilitator en een domeinexpert.
Benodigdheden: een mechanisme om de verschillende views bij te houden.

De facilitator treedt op als *spelleider*. De facilitator stelt de vragen waarop de domeinexpert antwoord geeft. De facilitator mag de verschillende views aanpassen. De facilitator mag de domeinexpert afkappen als hij dat nodig acht.

De facilitator stelt zichzelf kort voor aan de domeinexpert. De facilitator zorgt dat de views klaar staan om gebruikt te worden en geeft een korte uitleg aan de domeinexpert over het doel en voortgang. De formaliteit van het gesprek bepaalt of facilitator en domeinexpert elkaar tutoyeren of vousvoyeren. De facilitator moet zelf bepalen welke omgangsnorm hij gaat hanteren.

Indien de facilitator twijfelt welke vorm gepast is, wordt het aangeraden om te vousvoyeren en de gesprekspartner met u aan te spreken. Een informele setting met tutoyeren kan echter positief bijdragen aan de sfeer van het gesprek waardoor de domeinexpert zich vrijer voelt. Het wordt de facilitator aanbevolen om de domeinexpert vooraf te vragen naar zijn of haar voorkeur.

Het doel van de spelprocedure is het boven water halen van functionaliteit. De facilitator heeft beschrijvingen nodig waaruit de volgende use case onderdelen gegenereerd kunnen worden:

- Basic Course of Events
- Preconditions
- Postconditions
- Triggers
- Use case scenario

Hiervoor gaat de facilitator een gestuurde vraag-en-antwoord sessie aan met de domeinexpert.

De facilitator benadrukt dat niets wat de domeinexpert zal doen fout is. De facilitator legt uit dat hij mogelijkerwijs wel gedachtegangen moet afkappen, maar nodigt de domeinexpert uit zijn creativiteit volop te gebruiken.

Rondeverloop in het kort

In een ronde wordt één enkele use case behandeld. Deze ronde wordt in verschillende fasen onderverdeeld:

- Fase 1: algemene omschrijving
- Fase 2: implementatieomschrijving op instantieniveau
- Fase 3: implementatieomschrijving op typeniveau
- Fase 4: functieomschrijving op typeniveau
- Fase 5: functieomschrijving op instantieniveau
- Fase 6: overige omschrijvingen (triggers, preconditions, postconditions en overige zaken)

Een ronde begint altijd met Fase 1. De overige fasen hoeven niet de aangegeven volgorde te worden doorlopen. Fase 1 moet compleet doorgevoerd worden. De overige fasen hoeven niet compleet doorgevoerd te worden. De facilitator mag fasen helemaal overslaan. Bij elke fase hoort een overeenkomstige “view”, een spelelement (artefact) waar de beschrijvingen in opgeschreven, bewerkt en bewaard worden.

Aanbeveling: het wordt een onervaren facilitator aangeraden om deze volgorde wel te hanteren en alle stappen in volledigheid door te voeren. Deze aanbeveling geldt ook wanneer de inhoud complex is. Het wordt aan de facilitator zelf overgelaten om hierover een beslissing te nemen.

Rondeverloop in detail

De regels voor de fasen bevatten gedeeltelijk overlap. Elke fase is echter volledig beschreven zodat de facilitator per fase één blad met regels voor die fase heeft en niet hoeft te bladeren naar een vorige fase waarnaar verwezen wordt.

Fase 1: algemene omschrijving

Deze fase is verplicht en moet altijd aan het begin van een ronde gestart worden.

De facilitator begint met introductie van de use case en neemt view 1 ter hand. Daarna stelt de facilitator een algemene omschrijving op door een vraag en antwoordsessie. De facilitator probeert omschrijvingen voor een basic course of events & use case scenario’s, triggers, preconditions en postconditions te vergaren. De facilitator noteert alle antwoorden als een stapsgewijze omschrijving in de view. De facilitator classificeert elke stap op de volgende criteria: implementatieomschrijving, functieomschrijving of overige en instantieniveau of typeniveau.

De facilitator stuurt de domeinexpert niet in een bepaalde richting qua inhoud of vorm van de omschrijving maar laat de domeinexpert vrije loop. Als de facilitator van mening is dat een bepaalde gedachtengang niet relevant is, mag hij de domeinexpert afkappen. Het is aan de facilitator te bepalen wanneer dit moet gebeuren.

Aanbeveling: kap niet te gretig af. Vaak kunnen dingen boven water komen die niet direct voor de use case relevant zijn maar wel moeten worden meegenomen.

De facilitator probeert samen met de domeinexpert te bepalen wanneer de omschrijving klaar is. Als de domeinexpert hier geen uitsluitsel over kan geven, dan moet de facilitator op een gegeven moment zelf een beslissing nemen om de fase af te ronden. Als besloten is dat de omschrijving in deze fase compleet is, dan controleert de facilitator of alle stappen geclassificeerd zijn. Waar nodig voert hij nog classificatiestappen uit. Daarna besluit de facilitator met welke fase wordt verder gegaan.

De bestaande beschrijving wordt meegenomen en gekopieerd in de view van de nieuwe fase en wordt daar verder bewerkt. Zaken voor fase 6 worden gekopieerd naar view 6 en worden niet meegenomen naar de volgende fase. De facilitator kan zelf kiezen met welke fase hij verder gaat. Hiervoor kopieert hij de omschrijving uit view 1 met uitzondering van de zaken die naar view 6 gekopieerd zijn.

Aanbeveling: een onervaren facilitator wordt aangeraden de fases in de volgorde van de regels uit te voeren. Dit geldt ook als de omschrijving dusdanig divers is dat het moeilijk te beslissen is op welk niveau men het makkelijkst door kan gaan.

Algemeen geldt de richtlijn dat de facilitator zelf de volgende fase kiest op basis van de algemene omschrijving.

Voorbeeld: ligt de omschrijving grotendeels op functieniveau, dan is het misschien niet zinvol om het implementatieniveau uit te werken. De deliverables liggen in het functievlak. Ligt de algemene omschrijving grotendeels op implementatievlak op typeniveau, dan is het misschien minder zinvol om de implementatieomschrijving eerst op instantieniveau uit te werken. Als de facilitator van mening is van een implementatieomschrijving op instantieniveau in één stap naar een functieomschrijving op typeniveau te gaan, dan is dit mogelijk.

Focusvragen fase 1
Om de benodigde omschrijvingen boven water te halen heeft de facilitator enkele focus vragen tot zijn beschikking. De facilitator mag de lijst zelf aanvullen met aanvullen met vragen. De gegeven vragen zijn slechts een suggestie en niet alle vragen hoeven gebruikt te worden.

- Wat doet u als u <…> (Basic course of events/use case scenario)
- Wat is de volgende stap? (Basic course of events/use case scenario)
- Waar komt deze stap te staan? (volgorde)
- Wie of wat start dit? (Triggers)
- Zijn er bepaalde omstandigheden waaraan voldaan moet zijn voordat u kunt beginnen met <…>? (Preconditions)
- Wat moet bereikt worden? (Postconditions)
- Wat moet de situatie zijn opdat u klaar bent? (Postconditions)

Fase 2: implementatieomschrijving op instantieniveau

Deze fase is optioneel. Deze fase hoeft niet te volgen op fase 1.Het is aan te raden om deze stap over te slaan als de voorgaande omschrijving geheel ligt op het vlak een implementatieomschrijving op instantieniveau.

Als de facilitator besluit deze fase uit te voeren dan neemt hij view 2 ter hand. De omschrijving van de voorgaande fase wordt gebruikt in deze view.

Voorbeeld: als fase 2 direct na fase 1 volgt, dan wordt de omschrijving uit view 1 gekopieerd naar view 2 voor fase 2 en hier verder bewerkt. Als dit het geval is worden de stappen die betrekking hebben op fase 6 niet meegenomen naar view 2.

Het doel van deze fase is de omschrijving geheel te geven als een implementatieomschrijving op instantieniveau. Dit is het meest concrete niveau.

De facilitator loopt alle stappen op chronologische volgorde na. De stappen die reeds op het vereiste vlak liggen kunnen worden overgeslagen.

Voorbeeld: om te komen van een implementatieomschrijving op typeniveau is het alleen nodig om het typeniveau om te zetten naar instantieniveau, de rest bevindt zich immers al op het vereiste implementatieniveau.

De overige stappen worden omgezet naar een omschrijving op het juiste vlak.

Optioneel kan de facilitator de domeinexpert vragen de stap nog eens te bevestigen. Het wordt aangeraden dit niet te doen.

Het is mogelijk dat tijdens deze fase nieuwe stappen in de omschrijving naar boven komen. Deze worden chronologisch door de facilitator ingevoegd. De facilitator moet kiezen hoe hij hier mee omgaat. Hij kan eerst de vorige fasen opnieuw bezoeken en aanvullen alvorens hij verder gaat met deze fase of hij voert de veranderingen direct door in deze fase en laat de vorige fasen ongemoeid.

Aanbeveling: een onervaren facilitator kan ervoor kiezen om op het moment dat nieuwe stappen boven water te komen terug te gaan naar fase 1 en eerst de algemene omschrijving aan te vullen met deze nieuwe stappen. Dan wordt eerst fase 1 nog eens doorlopen, alle nieuwe stappen toegevoegd alvorens men verder gaat met fase 2.

Zijn er vooraf aan fase 2 al andere fasen gedaan, dan worden deze eerst bijgewerkt voordat men verder gaat met fase 2. Alternatief kan de facilitator ervoor kiezen om de omschrijving in fase 2 compleet te maken en dan fase 1 en eventuele overige fasen bij te werken.

Hij kan er ook voor kiezen om voorgaande fasen alleen bij te werken als deze dienen als input voor de uiteindelijke deliverables, namelijk fasen 4 t/m 6. De keuze ligt bij de facilitator. Een onervaren facilitator wordt aangeraden alle voorgaande fases geheel bij te werken voordat hij verder gaat. Een ervaren facilitator wordt aangeraden fase 1 en eventueel fase 3 niet bij te werken.

Als geen nieuwe stappen worden toegevoegd, dan is deze fase afgesloten zodra alle stappen uit de omschrijving zijn doorgenomen. Als wel nieuwe stappen zijn toegevoegd, maar deze zijn toegevoegd door naar een vorige fase terug te gaan, dan is fase 2 eveneens afgelopen wanneer alle stappen doorlopen zijn.

Als de stappen ook daadwerkelijk in deze fase zijn toegevoegd, dan moet de facilitator de domeinexpert vragen of er nog meer stappen bij moeten nadat alle overige stappen zijn doorlopen. Als zich deze niet zeker is, dan moet de facilitator een beslissing maken om de fase te beëindigen of te proberen om toch nog nieuwe stappen te vinden.

Aanbeveling: een minder ervaren facilitator kan er beter voor kiezen om fase 2 te beëindigen.

De facilitator kan zelf kiezen met welke fase hij verder gaat.

Focusvragen fase 2
De vragen in deze stappen moeten de omschrijvingen omvormen tot een implementatieomschrijving op instantieniveau voor zover dit nodig is. Van functie naar implementatie vooral vragen die vragen naar "hoe". Vragen van type naar instantie zijn vooral wie vragen of vragen naar voorbeelden.

- Kunt u een concreet voorbeeld geven hoe u dit zou doen? (van functie naar implementatie)
- Hoe zou u dit doen? (van functie naar implementatie)
- Kunt u hiervan een concreet voorbeeld geven? (van type naar instantie)
- Wie kan deze persoon zijn? (van type naar instantie)
- Waar komt deze stap te staan? (volgorde)

Fase 3: implementatieomschrijving op typeniveau

Deze fase is optioneel. Deze fase hoeft niet te volgen op fase 2. Het is aan te raden om deze stap over te slaan als de voorgaande omschrijving geheel ligt op het vlak een implementatieomschrijving op typeniveau.

Als de facilitator besluit deze fase uit te voeren dan neemt hij view 3 ter hand. De omschrijving van de voorgaande fase wordt gebruikt in deze view.

Voorbeeld: als fase 3 direct na fase 1 volgt, dan wordt de omschrijving uit view 1 gekopieerd naar view 3 voor fase 3 en hier verder bewerkt. Als dit het geval is worden de stappen die betrekking hebben op fase 6 niet meegenomen naar fase 3.

Het doel van deze fase is de omschrijving geheel te geven als een implementatieomschrijving op typeniveau. Dit is een semi-abstract niveau. De implementatieomschrijving is vrij concreet, het typeniveau is echter abstracter dan het voorgaande instantieniveau.

De facilitator loopt alle stappen op chronologische volgorde na. De stappen die reeds op het vereiste vlak liggen kunnen worden overgeslagen. Stappen die gedeeltelijk in het juiste vlak zitten kunnen een deel overslaag kan dat deel worden overgeslagen.

Voorbeeld: om te komen van een implementatieomschrijving op instantieniveau is het alleen nodig om het instantieniveau om te zetten naar typeniveau, de rest bevindt zich immers al op het vereiste implementatieniveau.

De overige stappen worden omgezet naar een omschrijving op het juiste vlak.

Optioneel kan de facilitator de domeinexpert vragen de stap nog eens te bevestigen. Het wordt aangeraden dit niet te doen.

Het is mogelijk dat tijdens deze fase nieuwe stappen in de omschrijving naar boven komen. Deze worden chronologisch door de facilitator ingevoegd. De facilitator moet kiezen hoe hij hier mee omgaat. Hij kan eerst de vorige fasen opnieuw bezoeken en aanvullen alvorens hij verder gaat met deze fase of hij voert de veranderingen direct door in deze fase en laat de vorige fasen ongemoeid.

Aanbeveling: een onervaren facilitator kan ervoor kiezen om op het moment dat nieuwe stappen boven water te komen terug te gaan naar fase 1 en eerst de algemene omschrijving aan te vullen met deze nieuwe stappen. Dan wordt eerst fase 1 nog eens doorlopen, alle nieuwe stappen toegevoegd alvorens men verder gaat met fase 3.

Zijn er vooraf aan fase 3 al andere fasen gedaan, dan worden deze eerst bijgewerkt voordat men verder gaat met fase 3. Alternatief kan de facilitator ervoor kiezen om de omschrijving in fase 3 compleet te maken en dan fase 1 en eventuele overige fasen bij te werken.

Hij kan er ook voor kiezen om voorgaande fasen alleen bij te werken als deze dienen als input voor de uiteindelijke deliverables, namelijk fasen 4 t/m 6. De keuze ligt bij de facilitator. Een onervaren facilitator wordt

aangeraden alle voorgaande fases geheel bij te werken voordat hij verder gaat. Een ervaren facilitator wordt aangeraden fase 1 en eventueel fase 2 niet bij te werken.

Als geen nieuwe stappen worden toegevoegd, dan is deze fase afgesloten zodra alle stappen uit de omschrijving zijn doorgenomen. Als wel nieuwe stappen zijn toegevoegd, maar deze zijn toegevoegd door naar een vorige stap terug te gaan, dan is fase 3 eveneens afgelopen wanneer alle stappen doorlopen zijn.

Als de stappen ook daadwerkelijk in deze fase zijn toegevoegd, dan moet de facilitator de domeinexpert vragen of er nog stappen bij moeten nadat alle overige stappen zijn doorlopen. Als zich deze niet zeker is, dan moet de facilitator een beslissing maken om de fase te beëindigen of te proberen om toch nog nieuwe stappen te vinden.

Aanbeveling: een minder ervaren facilitator kan er beter voor kiezen om fase 3 te beëindigen.

De facilitator kan zelf kiezen met welke fase hij verder gaat.

Focusvragen fase 3
De vragen in deze stappen moeten de omschrijvingen omvormen tot een implementatieomschrijving op typeniveau voor zover dit nodig is. Van functie naar implementatie vooral vragen die vragen naar "hoe". Vragen van instantie naar type zijn vooral vragen om te veralgemeniseren. Hier kan werken met voorbeelden een goede uitkomst bieden als de domeinexpert moeite heeft om de abstractieslag te maken.

- Kunt u een concreet voorbeeld geven hoe u dit zou doen? (van functie naar implementatie)
- Hoe zou u dit doen? (van functie naar implementatie)
- Kunt u dit algemener omschrijven? (instantie naar type)
- Wat voor soort personen mogen dit zijn? (instantie naar type)
- Een hond is een dier. Een klant is een …? (instantie naar type vanuit een voorbeeld)
- Waar komt deze stap te staan? (volgorde)

Fase 4: Functieomschrijving op typeniveau

Deze fase is verplicht. Het is echter niet verplicht om deze fase voor fase 5 uit te voeren of na fase 3. Als de facilitator van mening is dat de domeinexpert de abstractieslag naar functie moeilijker vind dan de abstractieslag naar van instantie naar type, dan kan de facilitator besluiten eerst fase 5 uit te voeren.

Uitzondering: als de vorige omschrijving al geheel op het vlak van een functieomschrijving op typeniveau ligt, dan wordt de facilitator niet verplicht deze ter validatie door te lopen, maar omdat deze fase het belangrijkste deel van de use case gaat opleveren is dit wel aan te raden.

Als de facilitator fase 4 uitvoert, neemt hij view 4 ter hand. De omschrijving van de voorgaande fase wordt gekopieerd naar view 4.

Voorbeeld: als fase 4 direct na fase 1 volgt, dan wordt de omschrijving uit view 1 gekopieerd naar view 4 voor fase 4 en hier verder bewerkt. Als dit het geval is worden de stappen die betrekking hebben op fase 6 niet meegenomen naar fase 4.

Het doel van deze fase is de omschrijving geheel te geven als een functieomschrijving op typeniveau. Dit is het meest abstracte niveau in het spel. De facilitator loopt alle stappen op chronologische volgorde na. De stappen die reeds op het vereiste vlak liggen kunnen worden overgeslagen.

Optioneel kan de facilitator de domeinexpert vragen de stap nog eens te bevestigen. Het wordt aangeraden dit niet te doen.

De overige stappen worden omgezet naar een omschrijving op het juiste vlak. Als een vraag al gedeeltelijk in het juiste vlak ligt, kan dat deel worden overgeslagen.

Voorbeeld: om te komen van een implementatieomschrijving op typeniveau is het alleen nodig om de implementatieomschrijving om te zetten naar een functieomschrijving, de rest bevindt zich immers al op het vereiste typeniveau.

Het is mogelijk dat tijdens deze fase nieuwe stappen in de omschrijving naar boven komen. Deze worden chronologisch door de facilitator ingevoegd. De facilitator moet kiezen hoe hij hier mee omgaat. Hij kan eerst de vorige fasen opnieuw bezoeken en aanvullen alvorens hij verder gaat met deze fase of hij voert de veranderingen direct door in deze fase en laat de vorige fasen ongemoeid.

Aanbeveling: een onervaren facilitator kan ervoor kiezen om op het moment dat nieuwe stappen boven water te komen terug te gaan naar fase 1 en eerst de algemene omschrijving aan te vullen met deze nieuwe stappen. Dan wordt eerst fase 1 nog eens doorlopen, alle nieuwe stappen toegevoegd alvorens men verder gaat met fase 4. Zijn er vooraf aan fase 4 al andere fasen gedaan, dan worden deze eerst bijgewerkt voordat men verder gaat met fase 4.

Alternatief kan de facilitator ervoor kiezen om de omschrijving in fase 4 compleet te maken en dan fase 1 en eventuele overige fasen bij te werken. Hij kan er ook voor kiezen om voorgaande fasen alleen bij te werken als deze dienen als input voor de uiteindelijke deliverables, namelijk fasen 5 en 6. De keuze ligt bij de facilitator. Een onervaren facilitator wordt aangeraden alle voorgaande fases geheel bij te werken voordat hij verder gaat. Een ervaren facilitator wordt aangeraden fase 1 t/m 3 niet bij te werken.

Als geen nieuwe stappen worden toegevoegd, dan is deze fase afgesloten zodra alle stappen uit de omschrijving zijn doorgenomen. Als wel nieuwe stappen zijn toegevoegd, maar deze zijn toegevoegd door naar een vorige stap terug te gaan, dan is fase 4 eveneens afgelopen wanneer alle stappen doorlopen zijn.

Als de stappen ook daadwerkelijk in deze fase zijn toegevoegd, dan moet de facilitator de domeinexpert vragen of er nog stappen bij moeten nadat alle overige stappen zijn doorlopen. Als zich deze niet zeker is, dan moet de facilitator een beslissing maken om de fase te beëindigen of te proberen om toch nog nieuwe stappen te vinden.

Aanbeveling: een minder ervaren facilitator kan er beter voor kiezen om fase 4 te beëindigen.

De facilitator kan zelf kiezen met welke fase hij verder gaat.

Aanbeveling: het wordt aanbevolen na fase 4 niet door te gaan met fase 2 of 3 indien deze zijn overslagen maar naar fase 5 te gaan indien deze nog niet gedaan is. Als fase 5 al gedaan is wordt aangeraden om naar fase 6 te gaan.

Een ervaren facilitator kan besluiten fase 5 over te slaan als hij in combinatie met voorgaande fasen genoeg informatie heeft voor een use case scenario. Dit kan alleen als deze voorgaande fasen compleet zijn. Als dit niet het geval is moet fase 5 gedaan worden of de voorgaande fase eerst bijgewerkt worden met nieuwe stappen. Een onervaren facilitator wordt aangeraden fase 5 niet over te slaan.

Focusvragen fase 4
De vragen in deze stappen moeten de omschrijvingen omvormen tot een functieomschrijving op typeniveau voor zover dit nodig is. Vragen om van implementatie naar functie te komen zijn vooral vragen die vragen naar “waarom” of een doel. Het waarom kan dan dienen als een middel om naar dit doel (de functionaliteit) te komen. Vragen van instantie naar type zijn vooral vragen om te veralgemeniseren. Hier kan werken met voorbeelden een goede uitkomst bieden als de domeinexpert moeite heeft om de abstractieslag te maken.

- Wat is het doel van deze stap? (van implementatie naar functie)
- Waarom wordt deze stap uitgevoerd? (van implementatie naar functie)
- Kunt u dit algemener omschrijven? (instantie naar type)
- Wat voor soort personen mogen dit zijn? (instantie naar type)
- Een hond is een dier. Een klant is een …? (instantie naar type vanuit een voorbeeld)
- Waar komt deze stap te staan? (volgorde)

Fase 5: Functieomschrijving op instantieniveau

Deze fase is optioneel. Een ervaren facilitator wordt aangeraden deze fase over te slaan als hij fase 4 gedaan heeft en van mening is dat door een combinatie van fase 4 met voorgaande fasen tot een use case scenario kan komen, bijvoorbeeld door de omschrijving uit fase 4 naast de omschrijving van fase 3 te leggen. Dit kan alleen als beide omschrijving helemaal volledig zijn.

Als fase 3 onvolledig is, omdat bijvoorbeeld in fase 4 stappen toegevoegd zijn die niet in vorige fasen zijn bijgewerkt, dan is het nodig om fase 5 alsnog te doen.

Een onervaren facilitator wordt aangeraden deze fase in ieder geval te doen. Een facilitator kan kiezen deze fase voor fase 4 uit te voeren wanneer de domeinexpert meer moeite heeft met de abstractieslag van instantie naar typeniveau.

De omschrijving van de voorgaande fase wordt gebruikt in deze view.

Voorbeeld: als fase 5 direct na fase 1 volgt, dan wordt de omschrijving uit view 1 gekopieerd naar view 5 voor fase 5 en hier verder bewerkt. Als dit het geval is worden de stappen die betrekking hebben op fase 6 niet meegenomen naar fase 5.

Het doel van deze fase is de omschrijving geheel te geven als een functieomschrijving op instantieniveau. Dit is een semi-abstract niveau in het spel, maar vormt wel een deliverable, namelijk het use case scenario. De facilitator loopt alle stappen op chronologische volgorde na. De stappen die reeds op het vereiste vlak liggen kunnen worden overgeslagen.

Optioneel kan de facilitator de domeinexpert vragen de stap nog eens te bevestigen. Het wordt aangeraden dit niet te doen.

De overige stappen worden omgezet naar een omschrijving op het juiste vlak. Als een vraag al gedeeltelijk in het juiste vlak ligt, kan dat deel worden overgeslagen.

Voorbeeld: om te komen van een implementatieomschrijving op instantieniveau is het alleen nodig om de implementatieomschrijving om te zetten naar een functieomschrijving, de rest bevindt zich immers al op het vereiste instantieniveau.

Het is mogelijk dat tijdens deze fase nieuwe stappen in de omschrijving naar boven komen. Deze worden chronologisch door de facilitator ingevoegd.

Aanbeveling: een onervaren facilitator kan ervoor kiezen om op het moment dat nieuwe stappen boven water te komen terug te gaan naar fase 1 en eerst de algemene omschrijving aan te vullen met deze nieuwe stappen. Dan wordt eerst fase 1 nog eens doorlopen, alle nieuwe stappen toegevoegd alvorens men verder gaat met fase 5. Zijn er vooraf aan fase 5 al andere fasen gedaan, dan worden deze eerst bijgewerkt voordat men verder gaat met fase 5.

Alternatief kan de facilitator ervoor kiezen om de omschrijving in fase 5 te compleet te maken en dan fase 1 en eventuele overige fasen bij te werken. Hij kan er ook voor kiezen om voorgaande fasen alleen bij te werken als deze dienen als input voor de uiteindelijke deliverables, namelijk fasen 4 en 6. De keuze ligt bij de facilitator. Een onervaren facilitator wordt aangeraden alle voorgaande fases geheel bij te werken voordat hij verder gaat.

Een ervaren facilitator wordt aangeraden fase 1 t/m 3 niet bij te werken.

Als geen nieuwe stappen worden toegevoegd, dan is deze fase afgesloten zodra alle stappen uit de omschrijving zijn doorgenomen. Als wel nieuwe stappen zijn toegevoegd, maar deze zijn toegevoegd door naar een vorige stap terug te gaan, dan is fase 5 eveneens afgelopen wanneer alle stappen doorlopen zijn.

Als de stappen ook daadwerkelijk in deze fase zijn toegevoegd, dan moet de facilitator de domeinexpert vragen of er nog stappen bij moeten nadat alle overige stappen zijn doorlopen. Als zich deze niet zeker is, dan

moet de facilitator een beslissing maken om de fase te beëindigen of te proberen om toch nog nieuwe stappen te vinden.

Aanbeveling: een minder ervaren facilitator kan er beter voor kiezen om fase 5 te beëindigen.

De facilitator kan zelf kiezen met welke fase hij verder gaat.

Aanbeveling: het wordt aanbevolen na fase 5 niet door te gaan met fase 2 of 3 indien deze zijn overslagen.

Focusvragen fase 5
De vragen in deze stappen moeten de omschrijvingen omvormen tot een functieomschrijving op instantieniveau voor zover dit nodig is. Vragen om van implementatie naar functie te komen zijn vooral vragen die vragen naar "waarom" of een doel. Het waarom kan dan dienen als een middel om naar dit doel (de functionaliteit) te komen. Vragen van type naar instantie zijn vooral wie vragen of vragen naar voorbeelden.

- Wat is het doel van deze stap? (van implementatie naar functie)
- Waarom wordt deze stap uitgevoerd? (van implementatie naar functie)
- Kunt u hiervan een concreet voorbeeld geven? (van type naar instantie)
- Wie kan deze persoon zijn? (van type naar instantie)
- Waar komt deze stap te staan? (volgorde)

Fase 6: Overige
Deze fase is optioneel. Als de facilitator van mening is dat de omschrijvingen, die als triggers, preconditions en postconditions geclassificeerd zijn, voldoende zijn dan kan deze stap worden overgeslagen.

In deze fase treedt validatie op. De zaken die in view 6 zijn weggezet worden stap voor stap nog eens doorgelopen. Als de domeinexpert aangeeft op grond van de ervaringen in het vorige proces nog wijzigingen te willen aanbrengen, dan is dat mogelijk.

Als de domeinexpert op grond van zaken in fase 6 met nieuwe stappen voor de algemene omschrijving komt, dan gaat de facilitator terug naar een voorgaande fase. Het is de keuze aan de facilitator naar welke fase hij terugspringt.

Een onervaren facilitator wordt aangeraden terug te springen naar fase 1, de omschrijving daar bij te werken en stapsgewijs de overige fasen weer te doorlopen.

Een ervaren facilitator kan het beste naar fase 4 terug springen. Is deze fase eenmaal compleet dan kan fase 5 door de facilitator automatisch worden aangevuld door de nieuwe omschrijvingen te combineren met de reeds bestaande instanties in fase 5. Als er compleet nieuwe types naar boven zijn gekomen dan wordt de facilitator aangeraden de nieuwe stappen kort in fase 5 opnieuw te behandelen.

Als de domeinexpert geen wijzigingen wil aanbrengen en ook niet met nieuwe stappen komt, dan sluit de facilitator deze fase af.

Einde van de ronde
Een ronde is voorbij als de facilitator van mening is dat hij voldoende materiaal heeft om de volgende zaken van een use case template in te vullen: een Basic Course of Events, triggers, preconditions en postconditions. *Als de facilitator twijfelt, dan is het aanbevolen om ten minste fasen 4, 5 en 6 volledig afgewerkt te hebben.*

Einde van het spel
Als de facilitator besluit dat de laatste ronde voorbij is, dan is het spel voorbij.

Tips voor de volgorde

De volgorde waarin het beste gespeeld kan worden zal sterk afhangen van de ervarenheid van de facilitator, het abstractievermogen van de domeinexpert en de algemene omschrijving die deze domeinexpert in fase 1 geeft.

Het is verplicht met fase 1 te beginnen. Zaken voor fase 6 worden in view 6 weggezet. Van de overige omschrijvingen kijkt de facilitator naar de labels die hij gegeven heeft. Als alle stappen in één vlak liggen, bijvoorbeeld implementatieomschrijving in instantieniveau, dan is het niet nodig om deze fase te doorlopen en wordt de facilitator aangeraden door te gaan naar fase 3, 4 of 5. Het wordt aangeraden fase 6 altijd op het einde te doen omdat deze verder los staat van de resterende omschrijvingen.

Met welke fase de facilitator doorgaat is afhankelijk van de omstandigheden. Als de facilitator voldoende vertrouwen in zijn kunnen heeft en de kunde van de domeinexpert kan hij besluiten meteen met fase 4 verder te gaan en fase 3 over te slaan.

Als hij echter denkt dat de domeinexpert moeite heeft om de abstractieslag van implementatie naar functie te maken, dan kan hij besluiten eerst de slag van instantie naar type te maken en met fase 3 verder te gaan. Denkt hij juist dat de domeinexpert daar moeite mee heeft dan kan hij kiezen om eerst fase 5 te doen en vanuit daar naar fase 4 te gaan en fase 3 over te slaan. Als de facilitator twijfelt dan kan hij de fasen het beste in de gesuggereerde genummerde volgorde afwerken.

Als de algemene omschrijving op een gemixt vlak ligt gelden dezelfde overwegingen met betrekking tot eigen vaardigheden en inschatting van de domeinexpert. Twijfelt de facilitator dan kan hij zich laten leiden door de labels. Hij kan de labels tellen door te kijken in welk vlak de meeste omschrijvingen liggen en in dit vlak door te gaan.

Liggen bijvoorbeeld 2 omschrijvingen op het vlak van implementatieomschrijving op instantieniveau, 7 omschrijvingen op implementatieomschrijving op typeniveau, 1 omschrijving op functieomschrijving op typeniveau en 0 omschrijving op functieomeschrijving op instantieniveau dan kan de facilitator zich laten leiden door de meerderheid en besluiten fase 2 (tijdelijk) over te slaan en te beginnen met fase 3 en alles om te zetten naar een implementatieomschrijving op typeniveau.

Kort voorbeeld

Facilitator John, een ervaren requirements engineer, gaat aan de slag met domeinexpert Stan. John en Stan hebben elkaar al eerder ontmoet en hoeven zich niet voor te stellen. Ze tutoyeren elkaar al sinds de eerste keer dat ze elkaar ontmoet hebben. John heeft de views al klaar staan. John informeert Stan dat ze de use case “geld opnemen” gaan behandelen.

John begint met fase 1. Bij elke stap die John opschrijft, zet hij meteen een label om te markeren of het om functie of implementatie en of het om type of instantie gaat of dat het triggers, preconditions of postconditions gaat. Stan beschrijft enkele zaken die niet rechtstreeks tot de deliverables horen maar meer in het vlak van business rules thuis horen. John vindt deze dingen interessant en is van mening dat ze uiteindelijk wel belangrijk zijn. Hij laat Stan zijn gang gaan en markeert de zaken als “overige”. Een keer kapt hij Stan af wanneer hij begint een heel omslachtig en uiterst zeldzame gebeurtenis omschrijft. Hij neemt alleen het begin op zodat het niet vergeten wordt en in een later stadium hierop alsnog kan worden teruggekomen. De overige omschrijving die John verkrijgt is een volledige implementatieomschrijving op instantieniveau.

John besluit daarom fase 2 over te slaan. John kopieert te zaken voor fase 6 naar view 6. John weet dat voor Stan de slag van implementatie naar functie wat lastiger en is en besluit daarom de slag van instantie naar type te maken. John kiest er daarom voor om met fase 3 verder te gaan. Hij kopieert de relevantie stappen naar view 3 en begint fase 3. Vervolgens gaat John naar fase 4. Hij kopieert weer de stappen uit view 3 naar view 4.

Tijdens deze fase realiseert Stan zich dat er nog stappen zijn die hij vergeten is. John besluit niet terug te gaan naar fase 1 maar deze in fase 4 in te voegen om tijd te sparen. Nadat alle stappen, oud en nieuw aan bod geweest zijn, vraagt hij Stan of hij nog wat dingen kan bedenken. Stan zegt nee. John concludeert dat hij alles heeft om een Basic Course of Events op te stellen en sluit fase 4 af.

Omdat nieuwe stappen zijn toegevoegd ziet John zich genoodzaakt om fase 5 te doen. Als er geen nieuwe stappen waren bij gekomen dan had John deze fase over geslagen omdat hij door fase 3 met fase 4 te combineren alle benodigdheden zou hebben gehad om een use case scenario op te stellen.

Hij kopieert de beschrijving van view 4 naar view 5 en past de oude stappen meteen aan met de instantieomschrijvingen uit view 3. De in fase 4 nieuw toegevoegde stappen behandelt hij nu nog. Als alle stappen geweest zijn besluit John fase 5 te beëindigen omdat hij voldoende heeft om een use case scenario op te stellen voor de Basic Course of Events.

John gaat naar fase 6. Hier vraagt hij Stan slechts om validatie en de vraag of hij misschien nog wat dingen kan bedenken. Stan schudt zijn hoofd. John beëindigt daarom fase 6 en de ronde. Aangezien er verder geen use cases op het programma staan is de procedure daarmee ook afgelopen. John bedankt Stan voor de moeite waarna Stan weer richting zijn werkplek loopt en John een gedeeltelijk use case opstelt.

Hoofdstuk 6: Conclusie

In de inleiding hebben we ons de volgende ontwerpvraag opgelegd:
Ontwerp een spelprocedure met een set van regels, procedures en richtlijnen die een requirements engineer leiden in een spel, gericht op het verkrijgen van een use case Basic course of Events, use case triggers, use case preconditions, use case postconditions en use case scenario's.

Hoofdstuk 2 schiep een theoretisch kader om dit te verwerkelijken. In hoofdstuk 3 hebben we een basisidee en ontwerp geschetst. Dit hebben we in de praktijk getest en geëvalueerd in hoofdstuk 4. Aan hand van de inzichten die we vergaard hebben, hebben we een definitieve spelprocedure gemaakt welke in hoofdstuk 5 op papier is gezet.

Deze is afgeleid van het 3^{e} en laatste ontwerp dat in hoofdstuk 4 gegeven is. In tegenstelling tot het allereerste ontwerp legt het definitieve ontwerp de nadruk op flexibiliteit en geeft deze meer vrijheid aan de facilitator. Veel regels zijn versoepelt en laten juist keuzemogelijkheden aan de facilitator.

Het doel van het spel is het ondersteunen van een requirements engineering door een vraag-en-antwoord sessie in te kaderen in een spelachtige procedure. Door regels op te weken werken we dit weer tegen. Daarom is er in het definitief ontwerp veel aandacht voor richtlijnen. Deze geven de facilitator suggesties wanneer deze vastloopt. Voor een erg onervaren requirements engineering wordt default de aanbeveling gegeven de volgorde aan te houden die gegeven wordt en te streven naar volledigheid in alle fasen. Hierdoor geeft het spel ondersteuning voor minder ervaren requirements engineers terwijl flexibiliteit voor meer ervaren personen aanwezig is.

Evaluatie

Het definitieve ontwerp is gebruikt voor twee laatste sessies, dit keer niet met mensen die uit een informatiekundehoek maar juist van weerszijden hiervan: een uit de informaticahoek en iemand zonder informatica- of informatiekundeachtergrond. Deze sessies worden niet in het detail beschreven zoals de voorgaande sessies omdat ze niet meer gediend hebben voor het ontwerp. Een korte omschrijving van deze sessies is te vinden in Appendix A.

In deze sessies hebben we voor iets ander publiek gekozen. Nadat in de eerste twee sessies twee personen met Informatiekundeachtergrond gekozen waren werd nu andere personen gekozen: een student Informatica en een recentelijk afgestudeerde studente kunstgeschiedenis zonder formele achtergrond. Beide sessies verliepen ongeveer gelijk.

De facilitatoraanpak was gelijk aan de aanpak van sessie 3. In beide gevallen lag de algemene omschrijving weer op het vlak van een implementatieomschrijving op instantieniveau. Daar zijn we doorgegaan naar fase 3 en later naar fase 4. In beide gevallen kwam er heel soepel een functieomschrijving op typeniveau uit. We merkten op dat in sessie 5 iets meer doorgevraagd moest worden in fase 4, maar ook de persoon zonder modelleerachtergrond wist vrij soepel de abstractieslagen te maken aan hand van de vragen. In beide gevallen werd fase 5 overgeslagen. Fase 3 en 4 hadden samen voldoende informatie voor een use case scenario.

Reflectie

Terugkoppelend naar de onderzoeksvraag kunnen we concluderen dat op deze beperkte schaal het ontwerp gewerkt heeft. Het aantal deelnemers was gering en niet gevarieerd. De gebruikte use cases waren klein en relatief eenvoudig van aard. Maar we zijn erin geslaagd een hele specifieke en reproduceerbare spelprocedure op te stellen. Dit biedt een opstap naar verder onderzoek.

Toekomstig onderzoek

Dit onderzoek vormt een eerste stap in een nieuwe richting die veel mogelijkheden belooft. De schaal was echter zeer beperkt. De enige juiste manier om dit verder te onderzoeken is in onze ogen dit verder in de praktijk te brengen. Dit onderzoek kan globaal in twee richtingen verlopen: technische ondersteuning voor de facilitator tijdens het uitvoeren en verder onderzoek naar abstractievermogen van mensen om de set van focusvragen uit te breiden en te verbeteren.

Facilitatorondersteuning

Een mogelijk direct vervolg op dit onderzoek is het bouwen van een tool om het dialoogspel te ondersteunen en faciliteren. De druk op de facilitator was in deze tests erg zwaar. Hij moet veel keuzes maken, veel zaken bijhouden en vooral bij veranderingen opletten dat niets vergeten wordt. De aanpak met Excel sheets was omslachtig.

Bijvoorbeeld bij het omzetten van instantie naar types moeten vaak ook alle persoonsvormen worden aangepast, wanneer de omschrijving van “Ik” of “Je” of “Men” naar “Een klant” springt. Dit hebben we in tussenliggende fasen daarom vaak overgeslagen om het tempo erin te houden. Het voldeed voor dit onderzoek gezien de geringe omvang van het onderzoek en de beperkte testcases. We hebben slechts vijf sessies gedaan met één use case per sessie en één domeinexpert.

Een vervolgonderzoek kan proberen zich te richten op meer ondersteuning bij de processen met behulp van een editor. Hierbij kan men denken aan automatisch aanvullen van andere views bij veranderingen, documenteren van veranderingen, automatisch aanpassen van alle instanties van een instantie naar het bijbehorende type, markeren van stappen die nog afgewerkt moeten worden en adviezen voor de facilitator hoe verder te gaan. Door te komen met een prototype zou het mogelijk kunnen zijn om deze spelaanpak op een groter publiek met meer en uitgebreidere use cases los te laten. Dit leidt ons tot het volgende onderwerp.

Abstractievermogen en focusvragen

Het onderzoek was heel beperkt met vier personen. Alle personen hadden een academische achtergrond, drie ervan een formele achtergrond door een opleiding Informatiekunde of Informatica. Met een prototype tool kan de aanpak niet alleen op een groter maar ook op een breder publiek worden losgelaten.

In de cursus Requirements Engineering op de Radboud Universiteit werd tot dusver geprobeerd om het denken in implementaties zo vroeg mogelijk uit te bannen. De sessies in dit onderzoek suggereren echter dat domeinexperts toch voornamelijk op dit gebied denken en minder snel zelf al een abstractieslag maken van implementatie naar functionaliteit en zelfs van instantie naar type.

De vraag is of dit de regel of een uitzondering is. Dat is gezien de beperkte omvang uit dit onderzoek niet te concluderen. Hiervoor moet deze aanpak op meer mensen en vooral op een breder publiek met verschillende achtergronden worden losgelaten. Dit kan ook helpen in meer focusvragen boven water te krijgen.

Dialogue games

De spelprocedure van dit onderzoek was vrij specifiek. Door meer specifieke spelprocedures te ontwerpen en toe te passen in de praktijk is het in de toekomst misschien mogelijk algemene ondersteuntools voor dialogue games in het algemeen te bouwen die ondersteuning voor content generation op verschillende vlakken kunnen bieden. Dit zou kunnen door een algemene tool te bouwen die voor verschillende situaties gebruik kan maken van verschillende sets focusvragen, toegespitst op verschillende situatie.

Literatuurlijst

[DEF 2008] 13 Gemechaniseerde Brigade gaat 'serious gamen', Ministerie van Defensie *http://www.defensie.nl/landmacht/actueel/nieuws/2008/12/01/46123250/13_Gemechaniseerde_Brigade_gaat_serious_gamen*

[Hop2008] Hoppenbrouwers, S.J.B.A.; Bommel, P. van; Järvinen, A. (2008). "Method Engineering as Game Design: an Emerging HCI Perspective on Methods and CASE Tools", *Proceedings of EMMSAD'08 (Exploring Modelling Methods for System Analysis and Design), held in conjunction with CAiSE'08*. Montpellier, France, June 2008.

[Hop2009] Hoppenbrouwers, S.J.B.A., Weigand, H., & Rouwette, E.A.J.A. (2009). Setting Rules of Play for Collaborative Modeling. *International Journal of e-Collaboration (IJeC),* **5(4)**, 37–52.

[Hop2010a] Hoppenbrouwers, S.J.B.A., Schotten, B., & Lucas, P. J. F. (2010). "Towards Games for Knowledge Acquisition and Modeling", *International Journal of Gaming and Computer-Mediated Simulations, Special issue on AI and Games,* **2(4)**: 48–66.

[Hop2010b] Hoppenbrouwers, S.J.B.A., & Wilmont, I. (2010). Focused Conceptualisation: Framing Questioning and Answering in Model-Oriented Dialogue Games. In P. Bommel, S. Hoppenbrouwers, S. Overbeek, E. Proper & J. Barjis (Eds.) (2010), "The Practice of Enterprise Modeling.", *Proceedings of the Third IFIP WG 8.1 Working Conference on the Practice of Enterprise Modelling (PoEM 2010). Lecture Notes in Business Information Processing (LNBIP),* **68**: 190–204. Berlin, Heidelberg: Springer

[Hop2012] Hoppenbrouwers, S.J.B.A., & Rouwette, E.A.J.A. (2012). "A Dialogue Game for Analysing Group Model Building: Framing Collaborative Modelling and its Facilitation", *International Journal of Organisational Design and Engineering (IJODE). Nog niet gepubliceerd.*

[Kna2008] Knauss, E., Schneider, K. & Stapel, K. (2008). "A Game for Taking Requirements Engineering More Seriously," *Third International Workshop on Multimedia and Enjoyable Requirements Engineering - Beyond Mere Descriptions and with More Fun and Games, 2008. MERE '08:* 22-26.

[Kul2004] Kulak, D. & Guiney, E. (2004). *Use Cases: Requirements in Context,* Second Edition. Boston: Addison-Wesley.

[Pre2010] Pressman, R. (2010). *Software Engineering: A practitioner's Approach,* Seventh Edition. Boston: McGraw-Hill.

[Zyd2005] Zyda, M. (2005). "From Visual Simulation to Virtual Reality to Games", *IEEE Computer,* **38(9)**: 25–32

Appendix A: Evaluatiesessies

Evaluatie sessie 4

Deze sessie werd gehouden met een student Informatica. Als use case gebruikten we de handeling “Geld opnemen”.

We hebben eerst samen een algemene omschrijving opgesteld. Vanuit daar heb ik als facilitator besloten naar fase 3 te gaan omdat de beschrijving geheel op het vlak van implementatie op instantieniveau lag. Dit was wederom opvallend. Hierbij dient aangegeven te worden dat de testpersoon is gevraagd om tijdens het spel wel uit een “klant” oogpunt te denken, omdat hij aangaf ook te kunnen denken vanuit de andere kant. Hij had duidelijk wel kennis van use cases en functioneel denken, maar als klant zijnde redeneerde ook hij vooral op het meest concrete vlak.

Na de abstractieslag naar typeniveau gemaakt te hebben gingen we verder naar stap 4. Nadat deze afgerond was heb ik besloten fase 5 over te slaan omdat met de informatie in view 3 en 4 voldoende informatie vergaard was. Fase 6 werd ter validatie nog kort nagelopen.

Deze sessie verliep zoals verwacht en verschilde niet wezenlijk van sessie 3. Het nam ongeveer 40 minuten in beslag. Het valt op dat het dit keer langer duurde dan de voorgaande sessie, maar dat kan met de omvang van de use case te maken hebben en met het feit dat de deelnemer in sessie 3 het spel al een keer eerder gespeeld heeft.

Een verschil met voorgaande sessies is dat er weinig toegevoegd werd in latere fasen. Op 1 omschrijvingstap na verschilde de omschrijving in fase 1 niet met de omschrijving in fase 4 in termen van ondernomen stappen. Alleen enkele heel implementatiespecieke stappen (het teruggeven van een bankpas) werden in fase 4 verwijderd.

De aanpak zelf goed te werken. Dit ervoer de testpersoon ook als zodanig. Deze is zelf werkzaam in het gebied van softwareontwikkeling en hanteert zelf de stap van implementatie naar functie. Het verwerken verliep soms wat omslachtig, maar dat ligt aan de simpelheid van het ontwerp. De testpersoon gaf achteraan dat het wel nuttig kan zijn om transities expliciet vast te leggen om de gedachtenpatronen van de domeinexpert beter te kunnen volgen. Dit nemen we mee als aandachtspunt. Dit zou kunnen gebeuren door alle sessies geheel te loggen.

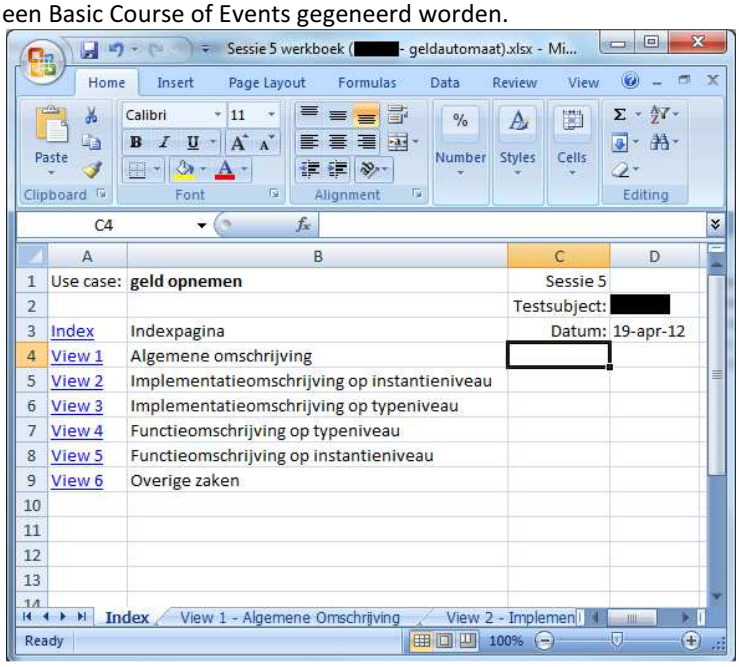
Evaluatie sessie 5

De laatste sessie met een afstudeerde kunstgeschiedenisstudente. Als use case gebruikten we de handeling “Geld opnemen”.

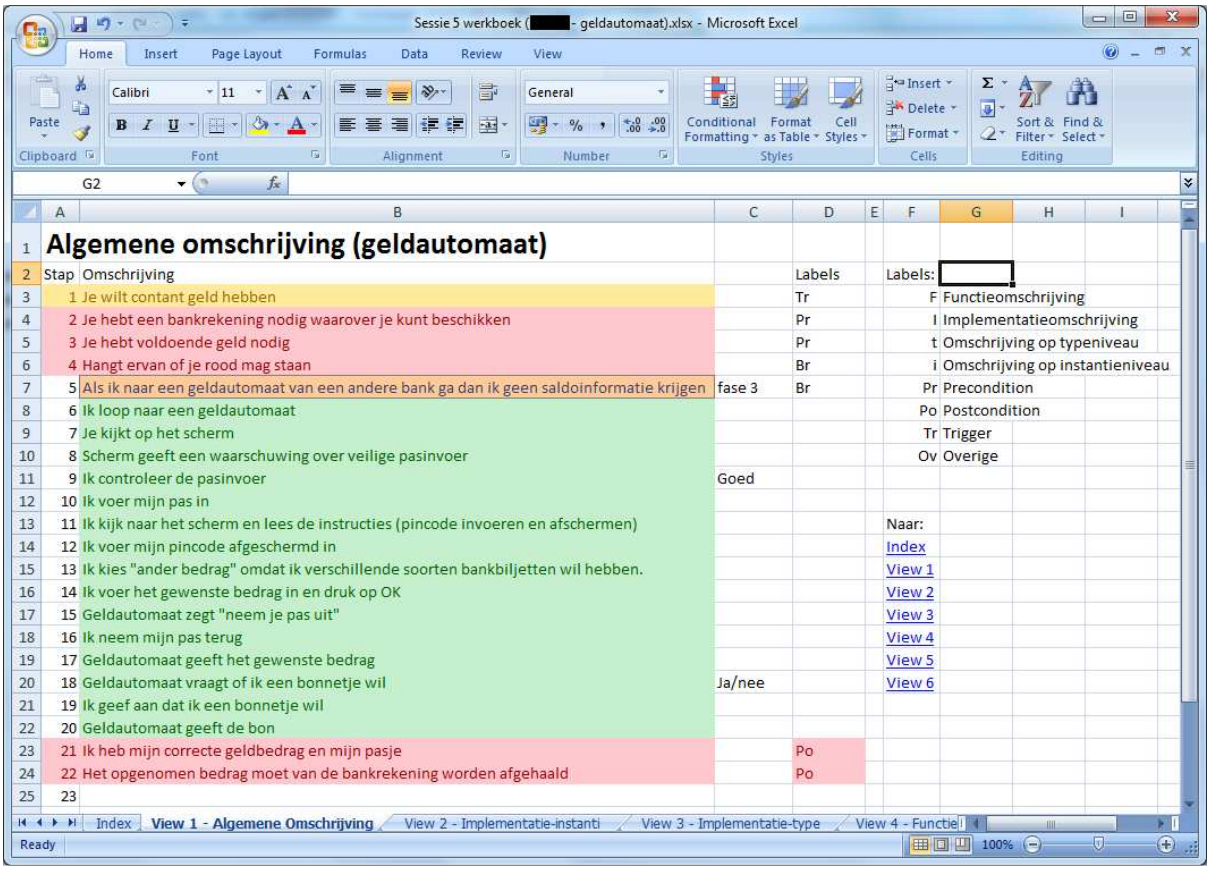
Als eerste werd een algemene omschrijving opgesteld. Deze lag geheel in het vlak van een implementatieomschrijving op instantieniveau. Als facilitator besloot ik daarom fase 2 over te slaan. Via fase 3 gingen we naar fase 4. Deze aanpak werkte ook in dit geval naar tevredenheid, in fase 4 verkregen we uiteindelijk een functionele omschrijving op typeniveau. Fase 5 hebben we overgeslagen, als facilitator was ik van mening met de omschrijvingen in fase 3 en 4 voldoende te hebben om een use case scenario te maken. We hebben in fase 6 gekeken naar view 6 ter validatie. Hier werd nog een opmerking gemaakt. De sessie duurde ongeveer 35 minuten.

Appendix B: views van sessie 5

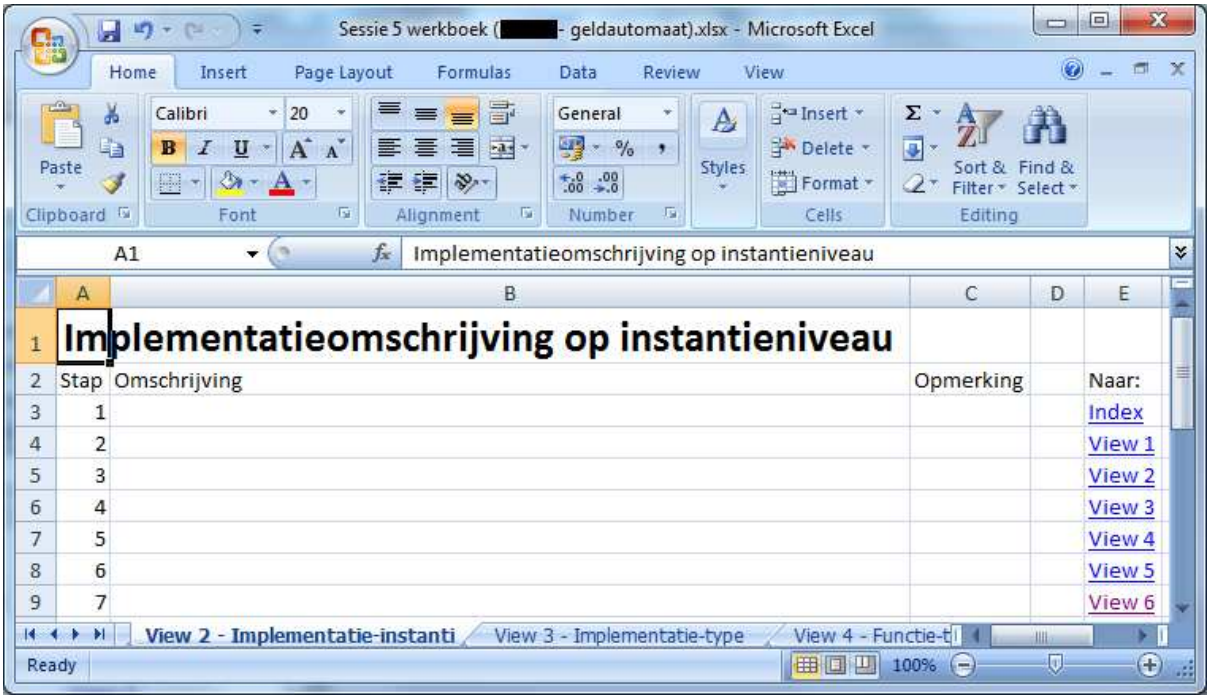
Ter illustratie geven we in dit hoofdstuk de views van sessie 5. De views zijn opgenomen in één Excel werkboek welke bestaat uit zeven tabbladen: een overzichtsview en één tabblad per view. De naam van de participant is zwart gemaakt uit privacyoverwegingen. Fase 2 is overgeslagen. Enkele zinnen lopen raar omdat bijvoorbeeld persoonsvormen niet zijn aangepast bij het omzetten van instantie naar type. We hadden besloten bij de sessie geen aandacht aan te besteden om tijd te besparen. Uit de gegeven omschrijving kan desalniettemin een Basic Course of Events gegeneerd worden.



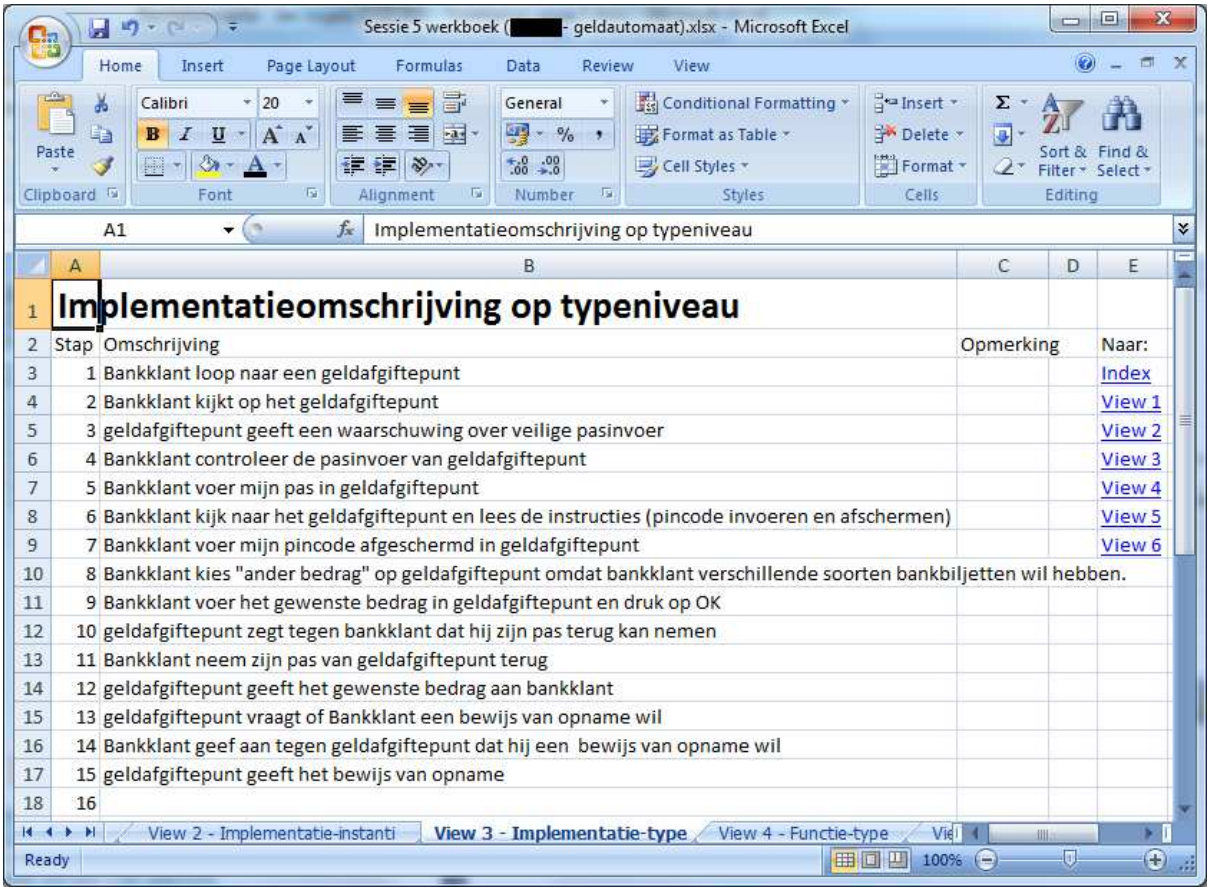
Afbeelding 6: overzichtsview sessie 5



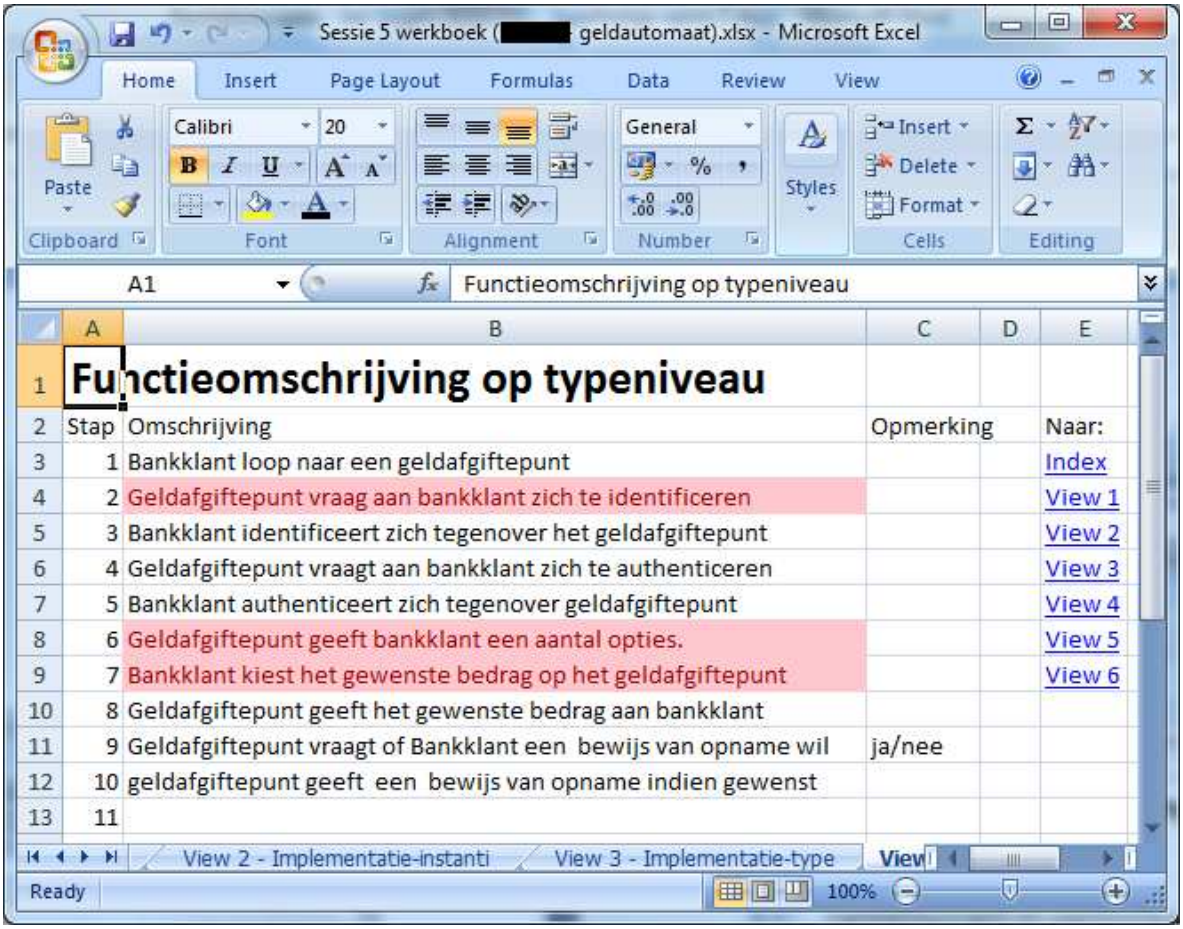
Afbeelding 7: view 1, algemene omschrijving, van sessie 5. Groene markeringen geven de oorspronkelijke stapsgewijze omschrijving aan, de andere kleuren geven zaken voor view 6 aan die aan het einde van fase 1 zijn opgesteld.



Afbeelding 8: view 2, sessie 5, implementatieomschrijving op instantieniveau. Deze view is leeg omdat deze fase is overgeslagen.

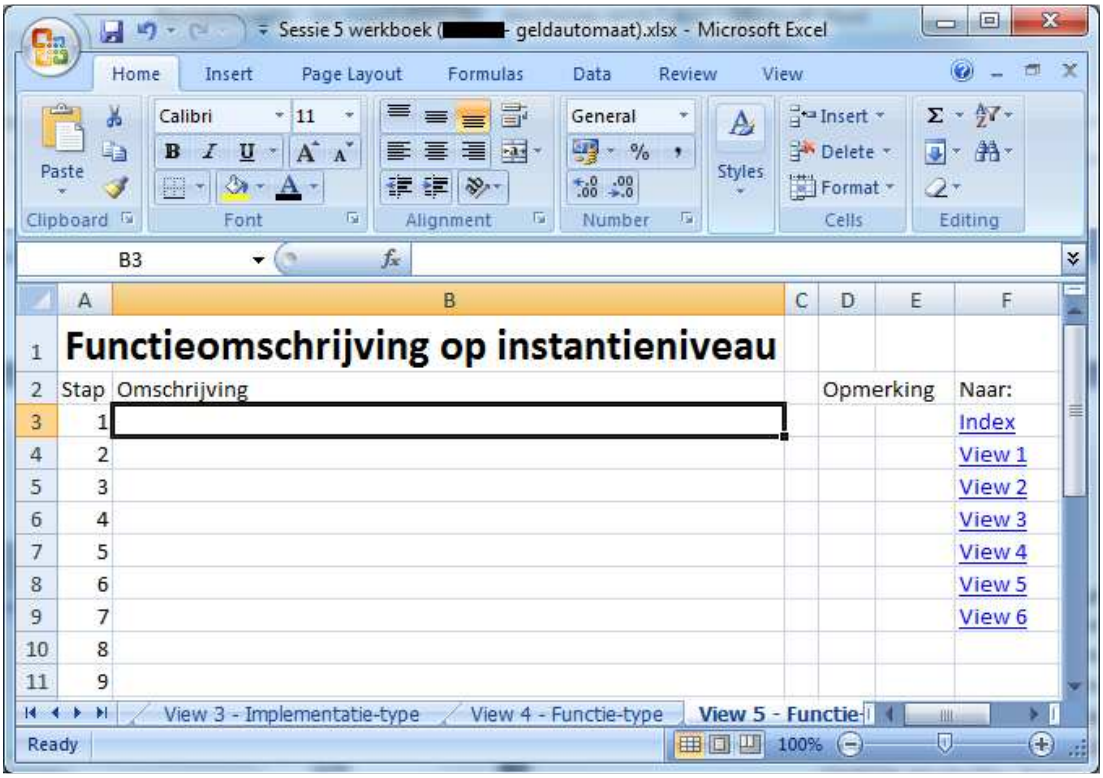


Afbeelding 9: view 3, sessie 5, implementatieomschrijving op typeniveau



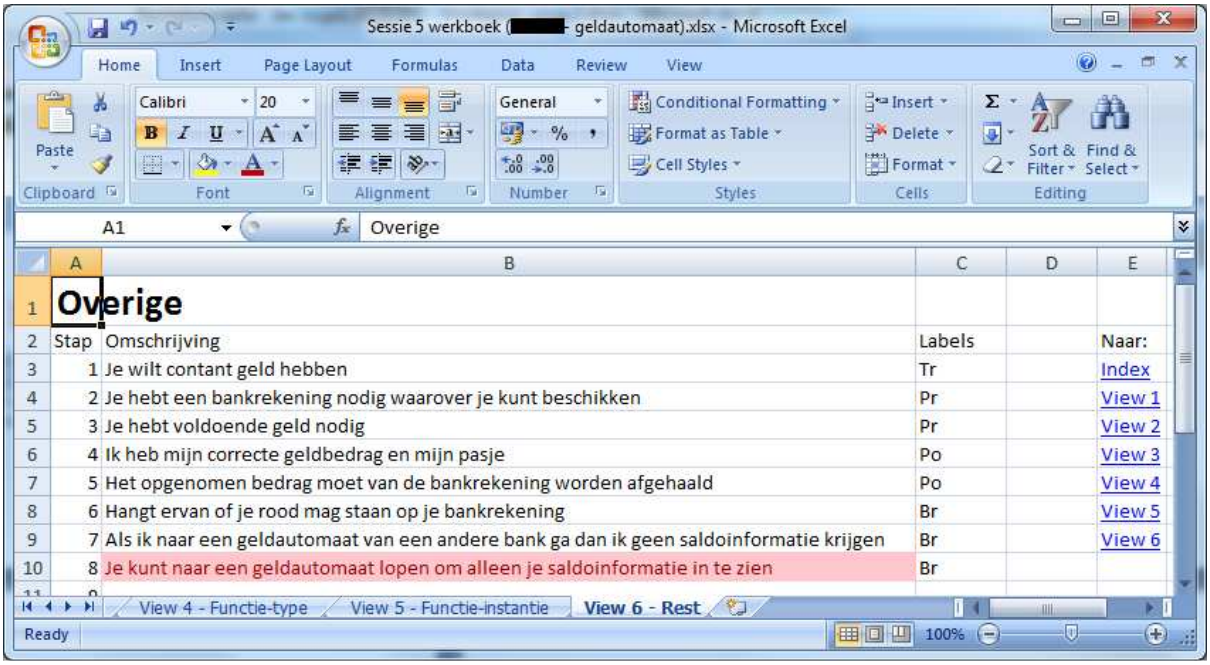
	A	B	C	D	E
1	**Functieomschrijving op typeniveau**				
2	Stap	Omschrijving	Opmerking		Naar:
3	1	Bankklant loop naar een geldafgiftepunt			Index
4	2	Geldafgiftepunt vraag aan bankklant zich te identificeren			View 1
5	3	Bankklant identificeert zich tegenover het geldafgiftepunt			View 2
6	4	Geldafgiftepunt vraagt aan bankklant zich te authenticeren			View 3
7	5	Bankklant authenticeert zich tegenover geldafgiftepunt			View 4
8	6	Geldafgiftepunt geeft bankklant een aantal opties.			View 5
9	7	Bankklant kiest het gewenste bedrag op het geldafgiftepunt			View 6
10	8	Geldafgiftepunt geeft het gewenste bedrag aan bankklant			
11	9	Geldafgiftepunt vraagt of Bankklant een bewijs van opname wil	ja/nee		
12	10	geldafgiftepunt geeft een bewijs van opname indien gewenst			
13	11				

Afbeelding 10: view 4, sessie 5, functieomschrijving op typeniveau. Rood gemarkeerde velden geven stappen aan die in deze fase nieuw bedacht zijn.



	A	B	C	D	E	F
1	**Functieomschrijving op instantieniveau**					
2	Stap	Omschrijving		Opmerking		Naar:
3	1					Index
4	2					View 1
5	3					View 2
6	4					View 3
7	5					View 4
8	6					View 5
9	7					View 6
10	8					
11	9					

Afbeelding 11: view 5, sessie 5, de functieomschrijving op instantieniveau. Deze is leeg omdat fase 5 is overgeslagen. Aan hand van views 3 en 4 kan een scenario op instantieniveau opgesteld worden.



Afbeelding 12: view 6, sessie 5, overige zaken (rest). Het rood gemarkeerde veld is in fase 6 nog boven water gekomen, de resterende zaken zijn rechtstreeks uit view 1 overgenomen.