Serious Gaming in Software Engineering

Bart Leusink, Informatiekunde

Begeleiders: Erik Barendsen en Stijn Hoppenbrouwers

4 juli 2011

afstudeernummer 155

Inhoudsopgave

1 Inleiding

Regelmatig zijn er verhalen in het nieuws over gefaalde ICT projecten. Miljoenen euro's per jaar worden er verspild aan software die niet alleen te laat af is, maar ook veel duurder uitvalt dan vooraf beraamd. Natuurlijk gaat het in andere branches ook wel eens mis -men hoeve alleen maar te kijken naar de Betuweroute of de Noord/Zuid lijn- maar in geen vakgebied is ontevredenheid over het uiteindelijk opgeleverde product zo vanzelfsprekend als in de ICT. Uit onderzoek van adviesbureau Ernst & Young [1] blijkt dat bijna de helft van de opgeleverde ICT projecten niet aan de verwachtingen voldoen. Gelukkig is er ook een lichtpunt aan de horizon: volgens hetzelfde onderzoek was het aantal succesvolle projecten in 2010 met 10% gestegen. Nu het ICT vakgebied langzamerhand volwassen wordt is het hard nodig dat het aantal gefaalde ICT projecten nog verder afneemt. Een nieuwe generatie IT'ers kan hier een rol in spelen, mits ze de noodzakelijke opleiding krijgen die hen hierop voorbereid.

Door een uitgebreid software engineering curriculum leren studenten informatica en informatiekunde tegenwoordig de belangrijkste theorieën over het succesvol uitvoeren van software ontwikkelingsprojecten maar hierbij mist veelal de praktijk component.

SimSE is een educatief computerspel dat erop gericht is software engineering studenten in weinig tijd in aanraking te laten komen met veel verschillende situaties uit het software ontwikkelingsproces om hen zo een deel van de vereiste project ervaring te laten opdoen. In deze scriptie wordt er onderzocht of dit spel daadwerkelijk een positieve bijdrage kan leveren aan het software engineering curriculum.

2 Probleemstelling

In het onderzoek staat de volgende vraag centraal: *Wat is de onderwijseffectiviteit van SimSE?*

Om deze vraag te kunnen beantwoorden wordt eerst een antwoord gezocht op de volgende deelvragen:

1. Hoe kunnen we SimSE beschouwen als een serious game?
2. In hoeverre gebruiken studenten software engineering kennis bij het spelen van SimSE?
3. In hoeverre zijn de studenten in staat de kennis die wordt opgedaan tijdens het spelen van SimSE te gebruiken buiten het spel?

Aan de hand van het antwoord op deze vragen kunnen we bepalen of SimSE een bijdrage kan leveren aan de software engineering vakken.

Er is al veel eerder onderzoek gedaan naar SimSE, een aantal van deze onderzoeken is te vinden in sectie 3.3 op pagina 7. Wat dit onderzoek echter toevoegt is dat er ook gekeken wordt naar de toepassing van de kennis die SimSE aanleert buiten het spel. Hiermee kan aangetoond worden of de leerdoelen van SimSE behaald worden.

3 Theoretisch kader

3.1 Serious gaming

Er zijn een aantal definities voor de term "serious game". Een veelgebruikte is die van Mike Zyda [2]. Hij omschrijft een serious game als een mentale wedstrijd, gespeeld met een computer volgens specifieke regels, waarbij entertainment wordt gebruikt om doelen met betrekking tot bestuurlijke of commerciële training, onderwijs, gezondheid, publiek belang en strategische communicatie te bewerkstelligen.

Susi et al. [3] omschrijven serious games als spellen die gebruikt worden voor andere doeleinden dan louter entertainment. De toevoeging van pedagogiek (activiteiten die instrueren of aanleren en daarmee kennis of vaardigheden overdragen) maakt een spel een serious game.

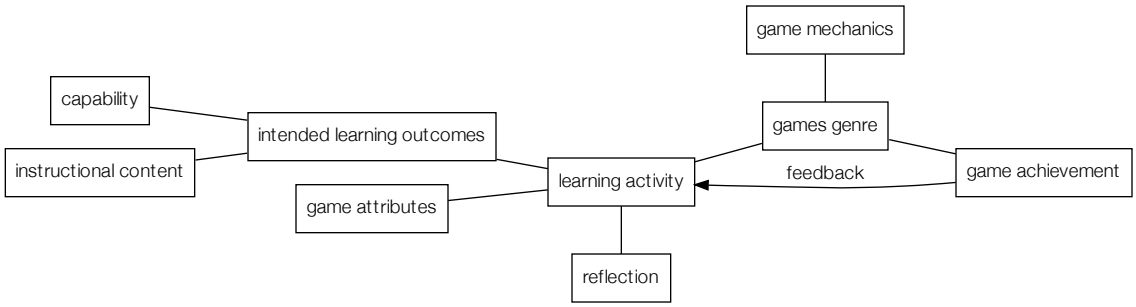
Serious games hebben vele voordelen vergeleken met meer traditionele lesmethoden [6], zoals:

- **Motivatie**: iemand die anders niet of nauwelijks zou leren wordt door een leuk spel verleid om toch een significante tijd met de leerstof door te brengen.
- **Herhaalbaarheid**: als iets fout gaat kan de speler opnieuw beginnen en ditmaal een andere keuze maken. Op deze manier kan de speler leren van zijn fouten en het spel net zo lang herhalen tot hij de stof beheerst.
- **Simulatie**: met behulp van serious games kan een situatie worden geoefend die in een on-the-job training in de echte wereld te duur, tijdrovend, riskant of onmogelijk is. Bedrijven zullen niet snel hun projectteams laten leiden door studenten om ze zo een kans te geven ervaring op te doen met het software ontwikkel proces. Met behulp van SimSE kunnen studenten de benodigde ervaring opdoen zonder een bedrijf ter gronde te richten.
- **Evaluatie**: moderne serious games registreren tijdens het spel een grote schat aan gegevens over de prestaties van de speler. Hiermee is het mogelijk te bepalen hoe goed iemand de lessen uit het spel heeft begrepen of welke specifieke vaardigheden nog verbeterd kunnen worden. In SimSE kan de speler grafieken opvragen over zijn voortgang in het spel en zijn prestaties vergelijken met die van een ideaal spelverloop.

Ondanks deze voordelen worden educatieve spellen nog niet warm onthaald in het onderwijs [7]. Een belangrijke reden hiervoor is dat van weinig serious games wetenschappelijk bewezen is dat ze de spelers daadwerkelijk iets leren. SimSE is een van de weinige software engineering serious games waar een uitgebreid onderzoek is gedaan naar de onderwijseffectiviteit van het spel [16]. Een korte samenvatting van deze onderzoeken en hun uitkomsten is te vinden in sectie 3.3 op pagina 7.

3.1.1 Framework voor serious games

Het serious games framework van Yusoff et al. [4] toont de belangrijkste componenten van een succesvol educatief computerspel. Elke component speelt een rol in het verzorgen van een spelervaring die zowel leuk als educatief is.



Figuur 1: Conceptueel framework voor serious games van Yusoff et al. [4]

- **Capability** beschrijft de cognitieve, affectieve of psychomotorische vaardigheden die de speler aanleert door het spelen van de serious game. In Bloom's taxonomie van leerdoelen [17] worden deze vaardigheden verder onderverdeeld in verschillende niveaus van leren.
- **Instructional content** is de educatieve inhoud van het spel, de lesstof die de student bijgebracht wordt door het spelen van het spel. De inhoud kan opgedeeld worden in vier typen: feiten, procedures, concepten en principes.
- **Intended learning outcomes** zijn de leerdoelen van de serious game. Deze leerdoelen zijn te classificeren met behulp van de taxonomie van Bloom [17].
- **Game attributes** beslaan de aspecten van het spel die het leren en de betrokkenheid ondersteunen. Hierbij gaat het om de eigenschappen van de serious game die het spel zowel educatief als leuk maken. Er is vaak een spanningsveld tussen deze twee aspecten aangezien een te leuk spel te veel afleid van de educatieve inhoud maar een saai spel de speler niet weet te betrekken, waardoor deze eerder geneigd zal zijn te stoppen [5].
- **Learning activity** is de handeling die de speler uitvoert in het spel. Zoals hierboven al aangegeven is het belangrijk dat deze activiteit leuk is, maar niet té leuk, om de kennisoverdracht te optimaliseren.
- **Reflection** is het moment waarop de speler nadenkt over het doel van de educatieve activiteiten die hij tot dan toe heeft uitgevoerd. Het is belangrijk dat *reflectie* kan optreden in de spel wereld zelf. Een manier om dit te doen is het toevoegen van adviseurs aan het spel. [5]. Een adviseur is een karakter in het spel die van tijd tot tijd aangeeft wat de speler fout doet. Ook kan de adviseur hints geven over hoe problemen op te lossen zijn.
- **Games genre** is het type of categorie spel. Van elk bijna elk type spel heeft men wel -met meer of minder succes- gepoogd een educatieve variant te maken.
- **Game mechanics** definiëren samen met de spelregels de details van het spel.
- **Game achievement** is de prestatie van de speler in de serious game. Dit kan uitgedrukt worden in bijvoorbeeld een score of de tijd die het de speler kostte om een bepaalde taak te volbrengen.

3.2 Software engineering

De eerste definitie van de term software engineering is: het vaststellen en het gebruik van deugdelijke engineering principes om op economische wijze software te verkrijgen die betrouwbaar is en efficiënt werkt op echte systemen [8].

Hoewel het gewenste eindresultaat hiermee vast staat, zijn er verschillende software engineering proces modellen om tot dit resultaat te komen.

3.2.1 Software engineering processen

In veel organisaties wordt het waterval ontwikkelingsproces model (of een variant hiervan) gebruikt. Dit model werd voor het eerst beschreven door Winston Royce [9]. Royce stelde voor verschillende niveaus van terugkoppeling in te bouwen om het proces zo robuuster te maken en de kans op een succesvol eindresultaat te vergroten. Toch wordt het waterval model tegenwoordig vaak gebruikt als een model dat alle stappen sequentieel afloopt en waarin het niet mogelijk is terug te gaan naar een vorige stap [10]. In de praktijk blijkt een puur sequentiële volgorde van de stappen van het waterval model vaak niet mogelijk. Het is vaak moeilijk voor de klant om al aan het begin van het project alle eisen op een rijtje te hebben en aangezien hij pas aan het eind van het project weer betrokken wordt is de kans op teleurstelling groot [10].

Bij andere modellen kan de volgorde of uitvoering van de stappen anders zijn, maar toch zullen over het algemeen de volgende stappen terugkomen: [10]

- **Requirements**: aan het begin van het project gaan ontwikkelaars met de klant om de tafel zitten om zijn eisen voor het product door te spreken en erachter te komen wat het doel van de software is.
- **Design**: in deze fase worden de eisen van de klant geanalyseerd en een ontwerp voor het software product gemaakt.
- **Coding**: tijdens deze stap wordt de software daadwerkelijk geprogrammeerd en vervolgens geïntegreerd.
- **Testing**: de complete software wordt onderzocht op fouten. Ook wordt er gecontroleerd of de software overeenkomt met het ontwerp en de eisen van de klant.
- **Operations**: uiteindelijk wordt het product bij de klant in gebruik genomen en nog enkele tijd ondersteund door de ontwikkelaar.

Een goed software engineering curriculum onderwijst de student in al deze stappen, aangezien ze de basis vormen van elk software ontwikkelingsproces.

3.2.2 Andere software engineering computerspellen

Voordat we een blik werpen op SimSE kijken we eerst naar haar voorgangers: de inspiratiebronnen voor de ontwerpers van SimSE. Een groot verschil tussen onderstaande spellen en SimSE is dat in vergelijking met SimSE deze spellen minder zijn onderworpen aan evaluatie met software engineering studenten om de educatieve effectiviteit te onderzoeken [16].

- **SESAM** [18] is een tekst gebaseerd computerspel waarin de speler in de schoenen van een projectmanager stapt en een groep virtuele medewerkers een virtueel project kan laten uitvoeren. Het spel is gemaakt als alternatief voor on-the-job training, zodat studenten ervaring kunnen opdoen met het managen van een software ontwikkelingsproces. Een docent kan met behulp van een speciale modelleer taal zelf simulaties ontwikkelen waarna een student deze door middel van tekst commando's kan spelen.

  SESAM is een belangrijke invloed geweest op SimSE. De makers van SimSE probeerden met SimSE een grafische, gebruiksvriendelijker versie van SESAM te maken [12].

- **OSS** [19] is een interactieve multimedia simulatie van een softwarebedrijf. Als lid van een van de projectteams kan de speler rond moet de speler verschillende softwareontwikkeling taken uitvoeren. Na het uitvoeren van elke taak wordt het resultaat van de speler vergeleken met dat van de computer maar in tegenstelling tot SESAM hebben deze taken geen effect op het verloop van het spel. Hierdoor krijgt de speler meer de rol van observant dan van daadwerkelijke medewerker wiens beslissingen ertoe doen.

  Ook het doel van het dit spel is de student praktijkervaring laten opdoen met het uitvoeren van software ontwikkelingsprojecten.

- **SimVBSE** [20] is een grafische simulatie van het software ontwikkelingsproces dat specifiek gericht is op value-based software engineering. Het spel is gebaseerd op een case study van een bestaand softwarebedrijf en vraagt de speler de rol van projectmanager in dit bedrijf op zich te nemen.

  In tegenstelling tot SimVBSE richt SimSE zich niet alleen op value-based software ontwikkeling maar op de simulatie van verschillende software ontwikkelingsprocessen.

- **The Incredible Manager** [21] simuleert de taak van projectmanager in een software ontwikkelingsproces. In een grafisch spel kan de speler alle project management beslissingen nemen en ziet hij zijn medewerkers deze uitvoeren. Net als bij SESAM [18] is de simulatie van The Incredible Manager aan te passen met behulp van een tekst interface. In tegenstelling tot SimSE en de bovenstaande spellen ligt de focus van dit spel specifiek op projectmanagement ervaring.

3.3 Eerder onderzoek naar de onderwijseffectiviteit van SimSE

3.3.1 Pilot studie

In de initiële evaluatie [14] werden 29 studenten die een introductie software engineering vak gehaald hadden gevraagd ongeveer 2 uur lang SimSE te spelen, waarna ze een enquête in moesten vullen. Een deel van deze enquête is gereproduceerd in appendix A en hergebruikt in mijn onderzoek.

De meeste studenten vonden SimSE leuk om te spelen en relatief makkelijk. Ook hadden ze het gevoel dat het spel leerzaam was en een nuttige bijdrage kan zijn aan het software engineering onderwijs.

Een belangrijk resultaat van dit onderzoek was het toevoegen van de uitleg/analyse tool aan de simulatie omdat de proefpersonen aangaven graag meer informatie te hebben gehad tijdens het spelen.

Nadat uit de pilot studie [14] gebleken was dat SimSE de potentie had om een nuttige bijdrage te leveren aan het software engineering onderwijs werd besloten een uitgebreidere evaluatie uit te voeren om erachter te komen *waarom* en *hoe* SimSE studenten helpt het software ontwikkelingsproces beter te begrijpen [15].

3.3.2 Evaluatie van SimSE in de klas

Om uit te vinden of SimSE past in het traditionele software engineering curriculum gaven de onderzoekers studenten van een inleidend software engineering vak de mogelijkheid bonuspunten (7,5% van het eindcijfer) te verdienen met het spelen van het spel en hier enkele vragen over te beantwoorden [15]. Doordat de vragen zo werden opgesteld dat de studenten SimSE meerdere keren moesten spelen om achter de antwoorden te komen, hoopten de onderzoekers uit te vinden hoe leerzaam SimSE was voor deze studenten. Na het spelen werden de studenten gevraagd de zelfde enquête die ook in de pilot studie gebruikt was in te vullen (zie appendix A), om er zo achter te komen wat de studenten van het spelen van SimSE vinden. Hiernaast hebben de onderzoekers deze evaluatie ook gebruikt om een beter beeld te krijgen van de praktische zaken waar een docent rekening mee moet houden bij het gebruik van SimSE in een software engineering vak.

De studenten vonden over het algemeen SimSE leuk maar na een tijdje spelen wel repetitief.

3.3.3 Vergelijkende studie

Het doel van deze studie was het vergelijken van SimSE met traditionele software engineering lesmethoden voor het aanleren van software ontwikkelingsproces concepten [15].

De proefpersonen -waarvan een deel al een inleidend software engineering vak gevolgd had en een deel niet- werden opgedeeld in drie groepen. Bij alle proefpersonen werd eerst een pre-test afgenomen om hun software engineering kennis te peilen. De vier dagen hierna speelde de eerste groep drie verschillende SimSE modellen, las de tweede groep een aantal teksten die ongeveer de zelfde lessen als de SimSE modellen bevatten en kreeg de derde groep twee hoorcolleges van 50 minuten over de zelfde concepten. Na deze vier dagen kregen alle proefpersonen een post-test om te bepalen hoeveel ze in die vier dagen geleerd hadden over software engineering.

De SimSE groep had de minste vooruitgang tussen pre- en post-test terwijl de groep die teksten over de zelfde software engineering concepten had gelezen de meeste verbetering in hun test scores had. Wel werd aangetoond dat er een sterke correlatie bestaat tussen de speeltijd en de toename van software engineering kennis.

3.3.4 Observatie studie

Als laatste wilden de onderzoekers erachter komen *hoe* SimSE studenten helpt bij het leren van software engineering concepten [15].

Bij dit onderzoek werd een groep studenten dat een inleidend software engineering vak had gehaald geobserveerd terwijl ze SimSE speelden om erachter te komen welke types van leren studenten ervaren tijdens het spelen van het spel. Hierna werden de proefpersonen geïnterviewd door de onderzoeker met vragen die speciaal ontworpen waren om de aanwezigheid van verschillende leer theorieën aan te tonen in het leerproces van de proefpersoon.

Uit de observaties is gebleken dat het cruciaal is dat de instructies voor het spelen van SimSE zo duidelijk mogelijk zijn, aangezien studenten anders belangrijke informatie missen en snel gefrustreerd raken.

Een belangrijke andere conclusie was dat SimSE het effectiefst is als de speler al een basiskennis over de behandelde software engineering concepten heeft, omdat dit de student de mogelijkheid geeft de eerder geleerde kennis in de praktijk te brengen.

Aangezien uit de vergelijkende studie gebleken was dat SimSE zonder andere vormen van informatieoverdracht de studenten te weinig kennis bijbrengt concludeerden de onderzoekers dat SimSE het beste samen met andere lesmethoden kan worden gebruikt.

3.3.5 Evaluatie op andere universiteiten

De bovenstaande onderzoeken zijn allemaal verricht op UC Irvine, de universiteit waar de makers van SimSE werken. Om te onderzoeken of hun ervaringen met SimSE ook te reproduceren waren op andere universiteiten en zonder hun aanwezigheid is SimSE op drie andere universiteiten ingezet bij software engineering vakken [16].

Op alle drie universiteiten kregen studenten de kans bonuspunten te verdienen door drie verschillende SimSE modellen te spelen en hierna een aantal vragen erover te beantwoorden. Alle studenten hadden voldoende software engineering basiskennis en voor het spelen kregen de proefpersonen college over de relevante proces modellen. De onderzoekers gaven de docenten een antwoord model en een sleutel om de cijfers te berekenen. Zowel de cijfers voor de opdracht als de eindcijfers voor het vak werden opgestuurd naar de onderzoekers. Hierna werden de docenten en studenten die mee hadden gedaan aan het onderzoek gevraagd een enquête in te vullen over hun ervaringen met SimSE.

Aangezien er geen correlatie werd gevonden tussen SimSE scores en eindcijfers en de studenten op de verschillende universiteiten vergelijkbare antwoorden op de enquête gaven, concludeerden de onderzoekers dat SimSE toepasbaar is bij studenten met een grote verscheidenheid aan vaardigheden en achtergronden.

Een andere conclusie die de onderzoekers trokken op basis van de enquête resultaten was dat studenten het over het algemeen redelijk leuk vinden om SimSE te spelen.

In voorgaande onderzoeken [15] had het gebrek aan adequate instructie de educatieve effectiviteit van SimSE in de weg gestaan. Doordat deze keer de studenten uitgebreide instructie van hun docenten gehad hebben over SimSE en bovendien een aantal leidende vragen kregen waardoor ze gedwongen werden meer na te denken en te onderzoeken tijdens het spelen waren ditmaal de SimSE scores aanzienlijk hoger. Dit heeft waarschijnlijk ook meegespeeld bij de betere evaluaties die SimSE deze keer van studenten kreeg.

4 Methode

Om de onderwijseffectiviteit van SimSE te onderzoeken is er gekeken naar het gebruik van software engineering kennis tijdens en buiten het spel. Dit laatste was nodig om te kijken of SimSE spelers in een praktijksituatie gebruik maken van de kennis die SimSE ze bijgebracht heeft en dus of de leerdoelen van SimSE zijn behaald. Hierbij is het gebruik van software engineering kennis buiten het spel vergeleken tussen SimSE spelers en niet-SimSE spelers om te kijken of SimSE spelers beter zijn in het analyseren van een software ontwikkelingsproces in een praktijksituatie en het geven van een oplossing dan studenten die het spel niet hebben gespeeld.

Aangezien onderzoek heeft aangetoond dat SimSE vooral effectief is in combinatie met meer traditionele lesmethoden [15, 16] is ervoor gekozen de proefpersonen te werven onder studenten die een inleidend software engineering vak volgden. Dit had als bijkomend voordeel dat alle proefpersonen ongeveer even veel voorkennis over het software engineering proces hadden. Omdat in het inleidend software engineering vak uitgebreid wordt stilgestaan bij het waterval proces model en dit model ook bij eerder onderzoek naar de educatieve effectiviteit van SimSE gebruikt is [14], is ervoor gekozen gebruik te maken van de waterval versie van SimSE. Deze is te downloaden op de SimSE website [11].

4.1 Observatie

Om het gebruik van software engineering kennis bij het spelen van SimSE aan te kunnen tonen werd gekeken naar de acties van de spelers en hun eventuele argumenten hiervoor.

Zes studenten zijn geobserveerd bij het spelen van één spel met het waterval model in SimSE. De duur van het spel was gemiddeld ongeveer drie kwartier. In tegenstelling tot het eerder uitgevoerde observatie onderzoek [15] is deze keer de studenten gevraagd hardop na te denken bij het spelen. Op deze manier is het denkproces van de speler voordat hij een bepaalde keuze maakt in het spel, samen met een screen capture van SimSE opgenomen op video voor latere analyse. Zogeheten think aloud protocols [22] worden veel gebruikt in de cognitieve psychologie om het oplossen van problemen te onderzoeken. Tegenwoordig wordt deze methode ook gebruikt bij het onderzoeken van het gebruik van educatieve software [23].

Aangezien eerder onderzoek had aangetoond dat proefpersonen met te weinig instructie makkelijk gefrustreerd raakten [15] werden de studenten voor het spelen uitgebreid ingelicht over het gebruik van SimSE. Tijdens het spelen van het spel werden ze echter niet geholpen, tenzij dat absoluut noodzakelijk was voor de voortgang van het onderzoek. Alle studenten speelden het spel afzonderlijk van elkaar.

De belangrijkste acties van de spelers, eventuele argumenten voor keuzes en gebeurtenissen in het spel zijn geannoteerd in appendix C. Door de argumenten naast de acties te leggen kon worden bekeken of een actie gebaseerd was op correcte software engineering kennis.

Met de lijst van acties kon verder gezien worden hoe vaak een speler terug ging naar een eerdere fase, aangezien dit tegen de principes van het waterval model (en dus een van de lessen van het spel) in gaat.

Tussen haakjes staat achter elke actie van de speler hoeveel virtuele medewerkers er bij deze actie betrokken waren. Hieruit kon afgeleid worden of de speler rekening houdt met de specifieke vaardigheden van individuele virtuele medewerkers en de communicatie overhead van een groot team aangezien dit twee van de lessen van de waterval versie van SimSE zijn.

Door de bovenstaande indicatoren te combineren kon er een beeld gevormd worden van gebruik van software engineering kennis van de spelers.

4.2 Enquête

Een effectief educatief spel moet niet alleen leerzaam zijn. Een student is beter gemotiveerd als hij het spel ook leuk vind en hij overtuigd is dat het spel niet te moeilijk is [24]. Het is dus belangrijk dat studenten vinden dat SimSE leuk en niet te moeilijk is.

Om dit te onderzoeken is de proefpersonen na het spelen van SimSE gevraagd de enquête van appendix A in te vullen. Deze enquête is grotendeels gelijk aan de enquête die in eerder onderzoek naar SimSE [14] gebruikt is. Hierdoor is het mogelijk de resultaten van de enquête te vergelijken met de resultaten van eerder onderzoek naar SimSE [14, 15, 16].

4.3 Scenario discussie

Om te kijken of studenten die SimSE gespeeld hebben ook in staat waren de kennis die ze hebben opgedaan bij het spelen van SimSE te gebruiken buiten het spel zijn drie groepen van 2 studenten 2 weken na het spelen van SimSE gevraagd samen te discussiëren over het scenario in appendix B.

Drie controlegroepen van 2 studenten die SimSE niet hadden gespeeld werd gevraagd het zelfde te doen. Hierdoor was het mogelijk de argumenten van studenten die SimSE gespeeld hadden te vergelijken met studenten die dat niet gedaan hadden.

Om de discussies van de groepen met elkaar te vergelijken werden het aantal correcte argumenten uit elke scenario discussie opgeteld. Dit getal geeft een indicatie van de hoeveelheid software ontwikkelingsproces kennis van de groepen. Hiermee was het dan ook mogelijk deze kennis te vergelijken tussen studenten die SimSE hadden gespeeld en studenten die dat niet hadden gedaan.

Het scenario was ontworpen om de proefpersonen enkele van de leerdoelen uit sectie 5.1 te laten noemen en bestond uit een van de meest voorkomende problemen bij software ontwikkelingsprojecten: nieuwe eisen van de klant. De proefpersonen werd gevraagd de risico's te noemen die zich in de casus manifesteerden en een oplossing te bedenken om deze risico's te mitigeren. Deze taak bevind zich op het *evaluation* niveau van Bloom's taxonomie [17] aangezien de proefpersonen de casus moeten analyseren om de risico's te vinden en vervolgens een beargumenteerde mening moeten vormen over een oplossing. De discussies van beide groepen werden opgenomen en naderhand uitgeschreven (zie appendix D). De argumenten van de proefpersonen werden opgedeeld in een aantal categorieën en de transcripties werden geannoteerd door achter elk argument de betreffende categorie te vermelden. Zie appendix D op pagina 40 voor een legenda. Achter elke annotatie staat tussen haakjes een plus- of een minteken om aan te geven of het argument gebaseerd is op een correct begrip van de betreffende software ontwikkeling concepten. Dit werd bepaald door te kijken naar correct gebruik van terminologie en de kwaliteit van de argumenten.

Ik heb iemand anders ook een discussie laten annoteren om de intersubjectiviteit van de annotaties te controleren.

5 Resultaten

5.1 SimSE

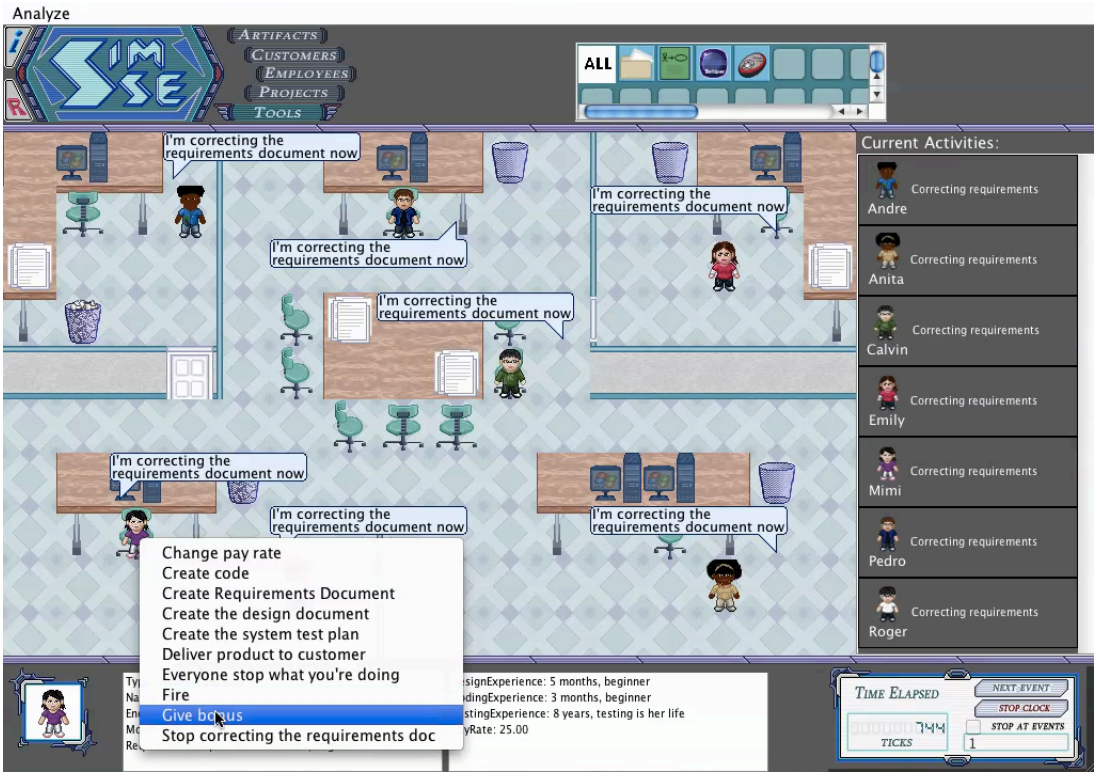
SimSE [12] is een single player serious game waarin de speler de rol van projectmanager van een team van ontwikkelaars overneemt. Net als SESAM [18] bestaat SimSE uit een modelbouwer en een simulatie omgeving [12]. Met behulp van de modelbouwer kunnen onderwijzers zelf scenario's opstellen door medewerkers, artefacten, projecten, klanten, tools en de activiteiten en regels die hun gedrag bepalen te definiëren. Hierdoor kan een onderwijzer het spel aanpassen aan de lessen die hij zijn studenten wil leren. Dit model kan vervolgens in de simulatie omgeving uitgevoerd worden als het daadwerkelijke spel.

Op de website van SimSE worden enkele door de makers van SimSE gegenereerde spellen aangeboden. Voor mijn onderzoek gebruik ik de daar te vinden waterval versie van het spel, welke hieronder dan ook nader besproken zal worden.

Als we SimSE beschouwen aan de hand van het serious games framework van Yusoff et al. [4] kunnen we het spel onderverdelen in de volgende componenten:

Capability

Het doel van SimSE is de speler praktijkervaring met het software ontwikkelingsproces te laten opdoen door hem verschillende situaties in het software ontwikkelingsproces voor te schotelen welke inzicht in en een reactie op de kwestie vereisen. [12] Hiermee zit het spel op het *evaluation* niveau van Bloom's taxonomie van leerdoelen [17], aangezien de speler spelenderwijs de verschillende regels die het spel bevat en het effect die deze hebben op het software ontwikkelingsproces leert door het spelverloop te analyseren en hier conclusies uit te trekken waardoor hij hierna een beargumenteerde keuze kan maken.



Figuur 2: De SimSE simulatie omgeving

Instructional content

De volgende belangrijke software engineering lessen zijn gemodelleerd in de waterval versie van SimSE:

- Volg nauwgezet het gebruikte software ontwikkelingsproces model. In het geval van het waterval model betekent dit dat de speler sequentieel het opstellen van requirements, het maken van een ontwerp, implementatie, integratie en testen moet laten uitvoeren.
- Verricht aan het eind van elke fase kwaliteitscontrole activiteiten (bijvoorbeeld reviews of inspecties) en corrigeer eventueel gevonden fouten.
- Zonder een goed software ontwerp verloopt de integratie fase moeizaam.
- De productiviteit van ontwikkelaars varieert sterk afhankelijk van hun individuele vaardigheden. Het afstemmen van de taken aan de vaardigheid en motivatie van de beschikbare ontwikkelaars verhoogt de productiviteit.
- Hoe meer ontwikkelaars aan een taak werken, hoe groter de communicatie overhead. Hoewel de taak eerder af zal zijn, kost dit meer inspanning door een grotere vereiste aan communicatie tussen ontwikkelaars.
- Financiële prikkels leiden tot een hogere productiviteit. In SimSE is dit geïmplementeerd door de mogelijkheid medewerkers bonussen of een opslag te geven.
- Software code inspecties zijn effectiever als ze zo snel mogelijk na het schrijven van de code worden uitgevoerd.
- Hoe beter een test is voorbereid, hoe groter het aantal gevonden fouten.
- Het gebruik van software engineering tools resulteert in een hogere productiviteit.
- Tijdens de ontwikkeling komt het vaak voor dat nieuwe requirements naar boven komen aangezien deze niet duidelijk waren totdat delen van het systeem ontworpen of geïmplementeerd waren.

Intended learning outcomes

SimSE is bedoeld om in combinatie met reguliere colleges gebruikt te worden om studenten praktijkervaring met het software ontwikkelingsproces te laten opdoen door de speler zelf te laten omgaan met de concepten die tijdens het college besproken zijn [12].

Na het spelen van SimSE kunnen spelers software ontwikkelingsprocessen in een praktijksituatie analyseren en beargumenteerd eventuele verbeteringen voorstellen. Dit komt overeen met het *evaluation* niveau in Bloom's taxonomie van leerdoelen [17] aangezien de student niet alleen geacht wordt de situatie te analyseren maar hij ook zelf als projectmanager keuzes moet kunnen maken en deze keuzes moet kunnen onderbouwen.

Game attributes

Een van de dingen die SimSE apart zet van andere software engineering serious games is de grafische interface [14]. De speler kan als projectmanager zijn virtuele medewerkers aansturen door rechts op ze te klikken en uit het contextmenu een actie uit te kiezen die ze moeten uitvoeren. Voorbeelden van zulke acties zijn het maken, controleren en verbeteren van requirements, design en het test plan.

Tijdens het verloop van het spel kunnen er ook gebeurtenissen optreden waardoor de speler gedwongen wordt een keuze te maken. In de waterval versie van SimSE treed bijvoorbeeld gemiddeld $\frac{2}{3}$% van de tijd de *requirements effect rule* op om te simuleren dat tijdens de ontwikkeling vaak nieuwe requirements naar boven komen. Op het moment dat deze regel van effect wordt laat het spel weten dat de klant de requirements heeft veranderd en moet de speler een keuze maken over zijn reactie op deze gebeurtenis.

Learning activity

De *learning activity* van SimSE is het managen van een virtueel software ontwikkelingsteam dat de opdracht krijgt een bepaald project uit te voeren. Door de verschillende fases van het ontwikkelingsproces mee te maken leert de student spelenderwijs verschillende software engineering concepten kennen.

Reflection

In de simulatie omgeving van SimSE heeft de speler de beschikking over een analyse/uitleg tool. Hiermee kan hij kijken welke regels van toepassing zijn op de simulatie en dus wat het resultaat van de beschikbare acties is. Bovendien kan de speler hier zien hoe lang SimSE suggereert dat elke fase uit het software ontwikkelingsproces moet zijn, waardoor de speler direct kan controleren of zijn project aan het uitlopen is door zijn keuzes. Ook bied deze tool de speler de mogelijkheid om alternatieve tijdlijnen aan te maken en zo het resultaat van verschillende beslissingen te bekijken. Deze uitleg tool is toegevoegd aan SimSE naar aanleiding van onderzoek waarin studenten opmerkten dat ze graag meer informatie zouden hebben gehad over hoe hun score berekend was en wat er fout ging [14]. Door deze functionaliteit is reflectie door de speler op elk moment dat hij dat wenst mogelijk.

Games genre

SimSE valt in het genre van *simulation games*, spellen waarbij een bepaald aspect van de realiteit gesimuleerd wordt. Binnen het genre van simulation games is SimSE in te delen in de categorie *management games*.

Game mechanics

De speler moet als projectmanager leiding geven aan zijn team om een software engineering project te voltooien. Hiervoor krijgt hij de beschikking over een bepaalde hoeveelheid tijd en geld. Management activiteiten zijn onder andere het aannemen en ontslaan van medewerkers, het toewijzen van taken, voortgangsbewaking en het aanschaffen van hulpmiddelen. In het algemeen zal het volgen van goede software engineering praktijken leiden tot positieve resultaten, terwijl argeloos negeren van deze praktijken tot mislukking van het voltooien van het project zal leiden.

Game achievement

Aan het eind van het spel krijgt de speler een score tussen de 0 en de 100, afhankelijk van de compleetheid van de verschillende artefacten (requirements document, design document, code en test plan) en of het project op tijd en binnen het budget af was.

5.2 Observatie

De transcripties van de acties en gebeurtenissen in SimSE van de verschillende proefpersonen zijn te vinden in appendix C. Er zijn in totaal 7 virtuele medewerkers. Tussen haakjes staat bij elke actie aangegeven hoeveel medewerkers betrokken waren bij die specifieke actie.

5.3 Enquête

De antwoorden die de proefpersonen op de enquête gaven zijn samengevat in de onderstaande tabel. Zie appendix A voor een uitgebreide versie van de vragen. Alle antwoorden zijn op een schaal van 1 tot 5.

Vraag	**1**	**2**	**3**	**4**	**5**	**Gemiddelde**	**Standaardafwijking**
Hoe leuk?	0	0	3	3	0	3,5	0,5
Hoe makkelijk?	0	1	4	1	0	3	0,58
Versterkt kennis over SE collegemateriaal?	1	1	1	3	0	3	1,15
Onderwijst nieuwe proces kennis?	2	2	0	2	0	2,3	1,25
Onderwijst SE proces in het algemeen?	0	3	1	2	0	2,8	0,9
SimSE als onderdeel van het SE vak?	0	2	2	1	1	3,2	1,07
Als optioneel onderdeel?	1	1	3	1	0	2,7	0,94
Als verplicht onderdeel?	1	2	0	3	0	2,8	1,21

5.4 Scenario discussie

Het aantal unieke annotaties van correcte argumenten bij de discussies van de SimSE spelers zijn opgeteld en hieronder te zien. Zie appendix D voor de transcripties van de discussies en bijbehorende annotaties.

Proefpersonen	**Aantal annotaties**
P1 & P2	6
P3 & P4	5
P5 & P6	3
Gemiddelde	4.67

Voor de controlegroep van discussiepartners die SimSE niet hebben gespeeld zien de scores er als volgt uit:

Proefpersonen	**Aantal annotaties**
P7 & P8	4
P9 & P10	3
P11 & P12	2
Gemiddelde	3

Een eenzijdige t-toets geeft met een significantie van 0.05 aan dat de groep van proefpersonen die SimSE had gespeeld gemiddeld meer relevante software ontwikkeling kennis wist te gebruiken tijdens de discussie over het scenario.

6 Discussie

Door de argumenten die de proefpersonen gaven tijdens het spelen van SimSE was het duidelijk dat ze verschillende concepten die ze in software engineering colleges geleerd hebben kunnen toepassen in SimSE. Quotes zoals

'*het moet eigenlijk allemaal via waterval hè, dan mag je niet terug naar vorige stappen*'

kwamen bij elke speler wel voor en de meeste spelers voerden de verschillende stappen in het waterval model in de juiste volgorde uit. Wel was het zo dat veel spelers -ondanks opmerkingen als de bovenstaande- toch op een laat moment (b.v. tijdens het programmeren) nog artefacten uit eerdere fasen (zoals bv. het requirements document) aanpasten. Een aantal van deze fouten kunnen ook te wijten zijn aan het feit dat dit de eerste keer was dat verschillende proefpersonen SimSE speelden en dat ze allemaal een relatief korte tijd gespeeld hebben. Aan het begin van het spel reageerden spelers nog positief op nieuwe wensen van de klant:

'*de klant gaf ons nieuwe requirements, nou dan gaan we die toevoegen aan de software*'

maar na enkele keren het requirements document aangepast te hebben naar aanleiding van nieuwe wensen van de klant hadden ze door dat dit zo eindeloos door zou kunnen gaan:

'*ja, hey, ze blijven bezig*'
'*dat gaat wel echt eindeloos zo door*'

De volgende keer dat zij SimSE zouden spelen zouden zij deze kennis al hebben en wellicht andere keuzes maken.

Doordat ik gebruik maakte van de standaard waterval versie van SimSE in plaats van een model dat speciaal aangepast was om aan te sluiten op de collegestof is het te verwachten dat de onderwijseffectiviteit van een SimSE model dat dit wel is hoger zal zijn. Het gebruik van het waterval model verklaart waarschijnlijk ook waarom de spelers in de enquête aangaven geen nieuwe proceskennis geleerd te hebben. De makers van SimSE trekken de zelfde conclusie bij hun onderzoek met het waterval model [14].

De spelers maakten weinig gebruik van de functionaliteit van de analyse tool om meer te weten te komen over de regels van het spel. Dit is bijvoorbeeld te zien aan spelers die alle 7 virtuele medewerkers de zelfde taak laten uitvoeren omdat ze niet doorhebben dat door communicatie overhead en extra fouten van onervaren medewerkers dit een negatief effect kan hebben op het eindresultaat, iets wat ze hadden kunnen weten door de analyse tool te gebruiken. Een mogelijke oplossing hiervoor zou het toevoegen van een *advisor* (zoals beschreven op pagina 3.1.1) zijn, die de speler uitleg geeft over een relevante regel van het spel als het spel merkt dat de speler te erg op het verkeerde pad zit.

Ook maakten alle proefpersonen geen gebruik van de 'alternative timelines' functionaliteit van de SimSE analyse tool, waarmee verschillende mogelijkheden in alternatieve realiteiten uitgeprobeerd kunnen worden. Misschien zou deze functionaliteit wel gebruikt worden als het makkelijker te vinden en te gebruiken was, zoals *undo* en *redo* knoppen in de menubalk.

De discussies over het scenario hebben aangetoond dat 2 weken na het spelen van SimSE de proefpersonen significant meer software ontwikkeling concepten gebruiken bij het analyseren van een casus dan studenten die geen SimSE hebben gespeeld. Dit betekent dat SimSE de speler kennis bijbrengt die hij vervolgens buiten het spel kan toepassen. Wel is het zo dat een totaal van 6 discussies redelijk weinig is om met grote zekerheid uitspraken hierover te kunnen doen.

De resultaten van de enquête komen overeen met de resultaten van enquêtes die bij eerder onderzoek naar SimSE [14, 15, 16] zijn afgenomen. Ook in Nijmegen vinden de studenten SimSE leuk om te spelen en niet te moeilijk. Verder is er veel verschil van mening over de vraag of SimSE gebruikt zou moeten worden bij het vak software engineering, met een deel van de proefpersonen grote voorstanders terwijl een deel fel tegen is. Het gemiddelde van deze vragen ligt hierdoor ongeveer een half punt lager dan bij de resultaten van de vergelijkbare enquête bij eerder onderzoek [14] maar door de hoge standaardafwijking en het lage aantal proefpersonen hebben deze resultaten weinig waarde.

Ik denk dat meer proefpersonen deze enquête zou helpen resultaten te verkrijgen waarmee met grotere zekerheid iets kan worden gezegd over hoe de speler SimSE ervaart.

7 Conclusie

In dit artikel werd gekeken naar de mogelijkheid om SimSE als hulpmiddel bij software engineering colleges te gebruiken.

De eerste deelvraag, die gedefinieerd was als *Hoe kunnen we SimSE beschouwen als een serious game?* is beantwoord door SimSE te analyseren met behulp van het serious games framework van Yusoff et al. [4].

Om de vraag *In hoeverre gebruiken studenten software engineering kennis bij het spelen van SimSE?* zijn 6 studenten geobserveerd bij het spelen van SimSE. Hieruit bleek dat basis software engineering kennis -zoals het waterval model- direct te gebruiken is in SimSE.

Uiteindelijk kwam de vraag *In hoeverre kan de kennis die wordt opgedaan tijdens het spelen van SimSE buiten het spel toegepast worden?* aan bod. Door de proefpersonen die SimSE hadden gespeeld in groepjes van 2 te laten discussiëren over een software engineering casus en een controlegroep het zelfde te laten doen bleek dat de studenten die SimSE hadden gespeeld de kennis die ze daarbij hadden opgedaan konden gebruiken bij het analyseren van de casus.

De hoofdvraag van dit onderzoek -*Wat is de onderwijseffectiviteit van SimSE?*- is dan ook te beantwoorden met dat SimSE concepten uit het software engineering college kan demonstreren waardoor studenten na het spelen beter in staat zijn deze concepten te gebruiken.

Hierdoor kan SimSE een nuttige toevoeging zijn bij verschillende software engineering vakken.

Referenties

[1] Ernst & Young, *Onderzoeksresultaten ICT Barometer over ICT-projecten*, 2010

[2] M. Zyda, *From Visual Simulation to Virtual Reality to Games*, 2005

[3] T. Susi, M. Johannesson, P. Backlund, *Serious Games - An Overview*, 2007

[4] A. Yusoff, R. Crowder, L. Gilbert en G. Wills, *A Conceptual Framework for Serious Games*, 2009

[5] E. Adams, *The Designer's Notebook: Educational Games Don't Have to Stink!*, 2005

[6] K. Corti, *Games□based Learning; a serious business application*, 2006

[7] D. Michael en S. Chen, *Proof of Learning: Assessment in Serious Games*, 2005

[8] P. Naur en B. Randell, *Software Engineering: Report of a conference sponsored by the NATO Science Committee*, 1968

[9] W. Royce, *Managing the Development of Large Software Systems*, 1970

[10] R. Pressman, *Software Engineering: A practitioner's Approach*, 2005

[11] `http://www.ics.uci.edu/~emilyo/SimSE/`

[12] E. Navarro en A. van der Hoek, *SimSE: an interactive simulation game for software engineering education*, 2004

[13] E. Navarro en A. van der Hoek, *Software Process Modeling for an Educational Software Engineering Simulation Game*, 2004

[14] E. Navarro en A. van der Hoek, *Design and Evaluation of an Educational Software Process Simulation Environment and Associated Model*, 2005

[15] E. Navarro en A. van der Hoek, *Comprehensive Evaluation of an Educational Software Engineering Simulation Environment*, 2007

[16] E. Navarro en A. van der Hoek, *Multi-Site Evaluation of SimSE*, 2009

[17] B. Bloom, M. Engelhart, E. Furst, W. Hill en D. Kratwohl, *Taxonomy of educational objectives: the classification of educational goals; handbook. Cognitive domain* 1956.

[18] A. Drappa en J. Ludewig, *Simulation in Software Engineering Training*, 2000

[19] H. Sharp en P. Hall, *An Interactive Multimedia Software House Simulation for Postgraduate Software Engineers*, 2000

[20] A. Jain en B. Boehm, *SimVBSE: Developing a Game for Value-Based Software Engineering*, 2005

[21] A. Dantas, M. Barros en C. Werner, *A Simulation-Based Game for Project Management Experiential Learning*, 2004

[22] C. Lewis, J. Rieman, *Task-centered user interface design - A practical Introduction*, 1993

[23] D. Cotton en K. Gresty, *Reflecting on the think-aloud method for evaluating e-learning*, 2006

[24] B. Paras en J. Bizzocchi, *Game, Motivation and Effective Learning: An Integrated Model for Educational Game Design*, 2005

A Enquête voor spelers van SimSE

1. Hoe leuk vond je SimSE?
   (*1 = helemaal niet leuk, 5 = heel leuk*)
2. Hoe makkelijk vond je SimSE?
   (*1 = heel moeilijk, 5 = heel makkelijk*)
3. Vind je dat SimSE je kennis over het software engineering collegemateriaal versterkt?
   (*1 = helemaal niet, 5 = absoluut*)
4. Vind je dat SimSE je nieuwe proces kennis bijbrengt?
   (*1 = helemaal niet, 5 = absoluut*)
5. Vind je dat SimSE je het software engineering proces in het algemeen bijbrengt?
   (*1 = helemaal niet, 5 = absoluut*)
6. Vind je dat SimSE een onderdeel van het vak Software Engineering moet worden?
   (*1 = helemaal niet, 5 = absoluut*)
7. Als een optioneel onderdeel?
   (*1 = helemaal niet, 5 = absoluut*)
8. Als een verplicht onderdeel?
   (*1 = helemaal niet, 5 = absoluut*)

B Scenario voor discussie opdracht

Je bent de manager van een software ontwikkelingsteam bij een groot ICT consultancy bedrijf dat werkt volgens het waterval model.

Jullie zijn bezig met een project voor het korps landelijke politiediensten om een systeem te maken waarmee agenten makkelijker en sneller bekeuringen en dossiers kunnen invoeren, wijzigen en delen.

Terwijl jullie aan het programmeren zijn komt er een nieuwe regering, met daarbij een nieuwe minister van justitie. Als onderdeel van zijn installatie verdiept zijn ministerie zich in de voortgang van het project. Doordat er de laatste tijd regelmatig verhalen over mislukte ICT projecten bij de overheid in het nieuws zijn, wil hij zich ervan verzekeren dat dit bij dit project niet gebeurt. Hij komt erachter dat enkele aannames die in het project zijn gedaan niet stroken met de plannen van de nieuwe regering.

Je baas stelt voor een aantal externe ontwikkelaars in te huren om te zorgen dat het systeem niet al verouderd is voordat het opgeleverd wordt.

1. Welke risico's loopt dit software ontwikkelingsproject?
2. Hoe zou je deze situatie aanpakken?

C Transcripties van de observaties

Proefpersoon P1

tijd	actie / *event*	relevante opmerkingen
11:05	create requirements document (7)	'je kunt nog geen andere dingen doen als je nog geen requirements hebt gemaakt, anders zou ik deze [werknemers met weinig requirements ervaring] iets anders laten doen'
		'ik heb hier nog een paar mensen maar goed die kunnen natuurlijk nog niks gaan doen'
		' eigenlijk zou je ze natuurlijk het beste allemaal op de requirements kunnen zetten'
		'weet je wat, ik zet ze gewoon allemaal op requirements, we zien wel wat er gebeurt'
13:30	stop creating requirements document (1)	
	create design document (1)	'deze is goed in designen en de requirements zijn al bijna klaar dus dan laat ik haar alvast designen'
13:35	*finished creating requirements document*	
15:55	review requirements document (1)	
	review design document (3)	
16:00	*new requirements*	'nou goed, dan moeten ze maar weer verder gaan met het maken van het requirements document
19:20	stop creating design document (1)	'deze was aan het designen maar gaat nu ook maar de requirements corrigeren'
	stop reviewing design document (2)	'het design schiet toch niet op, want dat gaat toch weer veranderen'
	review requirements document (1)	
	correct requirements document (2)	
	create design document (2)	'deze twee zijn toch niet goed, dus laat ik ze maar designen'
22:50	stop reviewing requirements document (2)	
	stop correcting requirements (2)	
	stop reviewing design document (1)	
	stop creating design document (2)	
	review requirements document (7)	
24:05	stop reviewing requirements document (1)	

	correct requirements document (1)	'ja weet je wat, we zitten nu op iets meer dan een kwart van de tijd, nu is het wel goed'
24:40	stop reviewing requirements document (1)	
	correct requirements document (1)	
26:20	stop reviewing requirements document (5)	'met iedereen aan het reviewen, dat gaat eindeloos door'
	correct requirements document (5)	
27:20	stop correcting requirements (1)	
	create design document (2)	
27:25	*finished correcting requirements*	
28:15	create design document (4)	
	create system test plan (1)	'deze vindt designen helemaal niet leuk, dus dan lijkt het me een slecht idee om hem dan te laten designen, dus dan kan die vast het test plan maken'
28:20	*new requirements*	'we hebben niet meer tijd gekregen, dus dat is dan jammer voor hem'
28:45	*new requirements*	'nu hebben we wel een beetje meer tijd gekregen'
30:00	stop creating design document	
	create requirements document	
30:10	*new requirements*	
31:25	create design document (1)	
	stop creating design document (1)	
	review requirements (1)	
32:00	*finished creating system test plan*	
33:10	review system test plan (1)	
	stop creating design document (2)	
	review design document (1)	
	review requirements document (1)	
33:15	*finished reviewing system test plan*	
34:25	create code (1)	'er zijn nog wel wat fouten maar we moeten wel een beetje gaan opschieten nu'
35:10	stop reviewing requirements document	'deze is bezig met reviewen [van de requirements] en hij vind een hele boel, als hij nu stopt met reviewen dan vind hij ook niks meer'
	correct requirements document	'ik geloof dat ik niet helemaal waterval aan het volgen ben, maar ja...'

		'ik probeer min of meer het waterval model te volgen maar als iemand iets niet wil of niet kan dan is het niet handig om hem dat toch te laten doen, en ja op een gegeven moment moet je toch verder gaan met je project want we zijn al veel te lang bezig met [het requirements en design] stuk'
35:15	*new requirements*	
35:20	*finished creating design document*	
43:00	stop reviewing requirements document (1)	
	stop correcting requirements document (1)	
	review design document (1)	
	create code (1)	
	inspect code (3)	
44:45	stop reviewing design document (1)	
	create code (1)	
45:20	*finished creating code*	
46:15	correct code (3)	
47:35	stop reviewing requirements document (1)	
	correct system test plan (1)	
47:40	*new requirements*	
47:45	*new requirements*	'we hebben niet heel veel tijd erbij'
48:00	*new requirements*	
49:05	stop correcting code (1)	
	create code (1)	
49:10	*finished creating code*	
20:45	stop inspecting code (3)	
	integrate code (6)	
	create system test plan (1)	
20:50	*finished correcting code*	
51:25	stop correcting system test plan (1)	
51:30	*finished creating system test plan*	
52:00	integrate code (1)	
52:10	*finished integrating code*	

54:45	create design document (7)	'als je gaat designen moet je daarna weer opnieuw gaan testen en dan de code weer aanpassen dus dan blijf je eindeloos bezig; maar goed, we kunnen wel het design document helemaal afmaken natuurlijk'
54:50	*finished creating design document*	
55:15	review design document (7)	
55:25	*new requirements*	
55:30	*new requirements*	
56:30	system test (7)	
57:10	stop reviewing design document (7)	
58:35	stop system test (7)	
	correct code (7)	'nu hebben we nog 20 [clockticks] om fouten eruit te halen die in onze system test zijn gevonden'
58:40	*new requirements*	
60:00	deliver project to customer	'we zitten al enigszins over tijd'

score: 56

Proefpersoon P2

tijd	actie / *event*	relevante opmerkingen
05:00	create requirements document (4)	
	create design document (3)	
06:05	*new requirements*	
07:30	review requirements document (1)	'ik laat deze kerel wel even de review doen, want hij is een expert [in requirements]'
08:00	correct requirements document (1)	
08:10	*new requirements*	
09:30	stop reviewing requirements document (1)	
	stop creating design document (1)	
	review design document (1)	'zij heeft ervaring met design, dus dan kan die mooi het design document reviewen'
10:20	stop creating design document (1)	
	create requirements document (1)	'ik denk dat ik misschien eerst wel eens even iemand van het design af moet halen om die mee te laten werken aan het requirements document, dan is dat in ieder geval klaar'
10:30	*finished correcting requirements document*	
10:20	create requirements document (1)	'omdat hij klaar is [met corrigeren van het requirements document] moet hij nu verder gaan met het maken van het requirements document'
	stop reviewing design document (1)	
	correct design document (1)	'het is onderhand wel duidelijk dat [in het design document] fouten zitten, dus die kan ze nu gaan corrigeren'
11:25	*finished correcting design document*	
11:40	create design document (1)	'als ze klaar is het met corrigeren van het design document kan ze verder gaan met het maken van het design document'
11:45	*finished creating requirements document*	
14:05	create design document (4)	'dit is een expert op het gebied van designen dus die moet dat an natuurlijk gaan doen'
		'ik weet alleen niet meer wat er allemaal naast elkaar kon, in feite mocht je volgens mij niet zozeer dingen naast elkaar doen in het waterval model'
		'dus eigenlijk had dit net al niet zo gemogen'

	stop creating design document (1)	
	review requirements document (1)	'ik laat het haar nu nu nakijken omdat zij meer ervaring heeft'
14:25	*new requirements*	'kan ik ook zeggen: fuck jou customer want het requirements document is al af, we gaan het niet meer doorvoeren?'
15:10	stop creating design document (1)	
	correct requirements document (1)	'ik vind dat Calvin toch maar weer fouten moet gaan repareren in de requirements'
15:50	correct requirements document (1)	
15:55	*new requirements*	
16:10	*finished correcting requirements document*	
18:35	create design (1)	'de mensen die klaar zijn kunnen natuurlijk weer verder werken aan het maken van het design'
	stop reviewing requirements document (2)	
	create code (1)	'ik denk dat ik deze kerel wel vast aan het coden laat, dat is dan misschien wel niet volgens het waterval model, maar...'
19:00	stop reviewing design document (1)	
	correct design document (1)	
19:25	*finished correcting design document*	
19:30	*new requirements*	
22:00	review requirements document (1)	
	create requirements document (1)	
22:10	*finished creating requirements document*	
22:25	stop reviewing requirements document (1)	
	correct requirements document (1)	
22:30	*finished correcting requirements document*	
22:50	create code (1)	
		'ik ben al bijna op de helft [van de tijd] maar het is nog niet half af'
23:20	stop creating code (1)	
	create design document (1)	
24:55	*finished creating design document*	
26:35	create code (3)	'jij bent snel met coden, dus jij moet sowieso aan code werken'

	create system test plan (1)	'zij is heel goed met testen en testen is haar leven dus die gaat vast beginnen met het test-plan want als de code al een beetje klaar is dan kun je daar een testplan voor gaan maken'
	inspect code (3)	
28:05	*new requirements*	
29:30	stop inspecting code (3)	
	correct code (2)	
	create system test plan (1)	
29:35	*finished creating code*	
30:20	integrate code (2)	
30:25	*finished correcting code*	
30:45	integrate code (3)	
30:50	*finished creating system test plan*	
31:25	review system test plan (2)	
31:50	*new requirements*	
33:00	stop reviewing system test plan (1)	
	stop integrating code (1)	
	correct system test plan (2)	
35:10	stop reviewing system test plan (1)	
	create code (1)	
35:15	*finished creating code*	
35:35	integrate code (1)	
35:40	*finished correcting system test plan*	
36:50	inspect code (3)	
37:20	stop integrating code (2)	
	correct code (2)	
38:00	*new requirements*	'ik zit ook bijna aan de uren'
38:25	deliver product to customer	

score: 30

Proefpersoon P3

tijd	actie / *event*	relevante opmerkingen
07:45	create requirements document (7)	'ik heb toch nog niets anders voor ze te doen, ik laat ze gewoon allemaal hieraan werken'
09:05	*finished creating requirements document*	
12:10	review requirements document (7)	
13:05	*new requirements*	'jeej, de klant gaf ons nieuwe requirements, nou dan gaan we die toevoegen aan de software'
13:25	correct requirements document (7)	
14:00	*new requirements*	
14:10	create requirements document (7)	
14:15	*finished creating requirements document*	
14:20	*new requirements*	'dat gaat wel echt eindeloos zo door'
		'ik weet niet of het me lukt om het echt netjes af te handelen, ik ben bang dat ik toch wel aan de code beginnen moet terwijl de requirements nog niet af zijn, dat is eigenlijk niet wat ik wil'
15:15	*new requirements*	
15:25	*finished correcting requirements*	
15:30	*finished reviewing requirements*	
16:25	correct requirements document (7)	
16:30	*finished correcting requirements document*	
19:50	review requirements document (7)	
	create design document (7)	
19:55	*finished reviewing requirements document*	
20:00	*new requirements*	
20:40	stop creating design document (1)	
	correct requirements document (1)	
20:45	*finished reviewing requirements document*	
20:50	review requirements document (1)	
20:55	*finished reviewing requirements document*	
21:00	create design document (1)	
21:05	*new requirements*	
21:15	stop creating design document (1)	
	correct requirements document (1)	
21:20	*finished correcting requirements document*	

21:25	review requirements document (1)
21:30	*finished reviewing requirements document*
21:45	create design (1)
22:05	*new requirements*
22:15	stop creating design (1)
	correct requirements document (1)
22:20	*finished correcting requirements document*
22:25	review requirements document (1)
22:35	*finished reviewing requirements document*
22:50	*new requirements*
	create code (2)
23:40	correct requirements document (1)
23:45	*finished correcting requirements document*
23:55	review requirements document (1)
24:00	*finished reviewing requirements document*
	new requirements
24:10	correct requirements document (1)
24:15	*finished correcting requirements document*
24:20	review design document (1)
24:40	review requirements document (3)
24:45	*finished reviewing requirements document*
25:30	stop creating design document (1)
	correct design document (5)
25:35	*finished correcting design document*
25:55	correct design document (1)
26:00	*finished correcting design document*
26:25	create code (3)
26:35	correct requirements document (1)
26:40	*finished correcting requirements document*
26:45	review requirements document (1)
26:50	*finished correcting requirements document*
27:00	create code (1)
	correct design (1)
27:05	*finished correcting design*

27:10	create code (1)	
27:45	*finish creating code*	
28:00	inspect code (7)	
28:20	stop reviewing design document (1)	
		'volgens mij kan ik eindeloos fouten aan het zoeken blijven'
31:25	stop inspecting code (1)	
	integrate code (1)	
31:30	*finished inspecting code*	
32:00	integrate code (6)	
32:10	*finished integrating code*	
32:30	correct code (7)	
33:00	*new requirements*	
33:10	*finished correcting code*	
34:00	correct requirements document (1)	
	create system test plan (6)	
34:05	*finished correcting requirements document*	
34:40	correct design document (1)	
34:50	*finished correcting design document*	
35:00	*finished creating system test plan*	
35:20	review system test plan (7)	
		'het moet eigenlijk allemaal via waterval hè, dan mag je niet terug naar vorige stappen'
36:10	*finished reviewing system test plan*	
36:30	correct system test plan (7)	
36:50	*finished correcting system test plan*	
37:15	deliver project to customer	

score: 65

Proefpersoon P4

tijd	actie / *event*	relevante opmerkingen
02:50	create requirements document (5)	'zij hebben veel requirements ervaring en één onervaren mag erbij zodat ze ervan kan leren'
04:00	*new requirements*	
05:00	review requirements document (1)	'hij is daar wel goed in, dus hij mag dat gaan doen'
	create design document (6)	'als ze niks te doen hebben kunnen ze wel vast een design gaan maken, het kost ook geld als ze niks doen'
05:25	stop creating design document (2)	
05:45	correct requirements document (2)	
05:50	*new requirements*	
06:00	stop correcting requirements document (1)	
	stop creating design document (4)	
07:15	review requirements document (3)	
	review design document (1)	
	correct requirements document (1)	
07:40	*new requirements*	
09:25	correct requirements document (1)	
09:35	*new requirements*	
11:00	stop reviewing design document (1)	'Het design schiet toch niet op zo. Misschien moeten we het design even laten voor wat het is'
	stop reviewing requirements document (2)	
	correct requirements document (2)	
11:30	*new requirements*	
12:00	*finished correcting requirements document*	
	finished reviewing requirements document	
12:45	create requirements document (3)	'de requirements zijn nog niet compleet'
	create design document (3)	
	review design document (1)	
12:50	*finished creating requirements document*	
13:20	correct design document (3)	
13:40	*finished correcting design document*	
14:00	create design document (3)	

14:45	stop creating design document (1)	
	correct design document (1)	
15:35	correct design document (1)	
16:05	*new requirements*	
16:40	review requirements document (1)	
17:30	stop correcting design document (1)	
	correct requirements document (1)	
	finished reviewing requirements document	
18:00	*finished creating design document*	
18:30	correct design document (3)	
	finished correcting requirements document	
18:35	correct design document (1)	
18:45	*finished correcting design document*	'nu kunnen we met het volgende traject bezig'
20:30	create code (4)	
	create system test plan (3)	'zij haat coding, dus ze mag het test plan gaan maken'
		'zij is een beginner in coding en testing is her life, nou dan moet zij zeker gaan testen'
20:35	*new requirements*	
21:10	correct design document (1)	
	done creating test plan	
22:50	review system test plan (2)	
	inspect code (3)	
23:10	*finished creating code*	
23:35	inspect code (1)	
23:45	*new requirements*	
24:40	stop inspecting code (3)	'is het af als ik het goed genoeg vind?'
	stop reviewing system test plan (2)	
	stop reviewing design document (2)	
25:00	integrate code (6)	
	correct system test plan (1)	'deze heeft een hekel aan coding dus ik heb nog wel wat anders voor haar te doen'
27:00	stop integrating code	
	correct code (6)	
27:05	*finished correcting code*	

27:50	integrate code (6)	
27:60	*finished integrating code*	
29:05	system test (7)	
	correct design document (1)	
30:30	correct code (2)	'volgens mij heb ik het niet goed gedaan als [er zoveel fouten in zitten]'
31:45	stop system test (6)	
	stop correcting design document (1)	
	correct code (4)	
33:05	deliver project to customer	'ik lever hem gewoon zo in'

score: 53

Proefpersoon P5

tijd	actie / *event*	relevante opmerkingen
04:30	create requirements document (1)	
05:40	create requirements document (1)	
06:30	create requirements document (1)	
07:45	*new requirements*	
08:40	*finished creating requirements document* 7	
12:15	create design document (1)	
12:40	create design document (1)	
12:55	*new requirements*	
14:35	review requirements document (1)	
15:50	*new requirements*	
16:35	*new requirements*	
17:05	*finished creating design document*	
		'hoe pak je dat eigenlijk op, die nieuwe requirements? Met het waterval model kun je toch eigenlijk niet terug?'
19:40	correct requirements document (2)	'ik ga gewoon lekker door'
19:55	stop reviewing requirements document (1)	'het lijkt me lastig om de fouten eruit te halen en tegelijkertijd ook te reviewen'
20:45	*finished correcting requirements document*	
22:00	review design document (2)	
23:50	stop reviewing design document (2)	
	correct design document (1)	'ik vind het wel best zo, stop met reviewen dan ga ik wel de fouten eruit halen'
23:55	*new requirements*	
24:15	correct design (1)	
24:45	*finished correcting design document*	'door stampen naar de volgende fase, even wat code genereren'
26:30	create code (3)	
26:35	*new requirements*	
26:50	*new requirements*	
27:05	*finished creating code*	
30:05	inspect code (3)	
30:10	*new requirements*	
30:25	*new requirements*	

30:40	*new requirements*	
31:00	correct code (3)	
32:15	stop inspecting code (3)	'dat is wel genoeg [gevonden errors in de code] zo'
32:30	*new requirements*	
32:35	*finished correcting code* 7	
33:10	integrate code (3)	
33:15	*new requirements*	
33:20	*finished integrating code*	
34:00	create system test plan (2)	
34:20	*finished creating system test plan*	
34:40	*new requirements*	
35:20	*new requirements*	
36:30	system test (2)	
37:10	*finished system test*	
39:00	correct code (2)	
39:40	*new requirements*	
39:45	*new requirements*	
39:55	*finished correcting code*	
40:40	deliver product to customer	'alles is 90% of over de 90% compleet dus ik vind het wel best'

score: 58

Proefpersoon P6

tijd	actie / *event*	relevante opmerkingen
06:20	create requirements document (2)	'het requirements document moet je eerst hebben voordat je überhaupt iets kan designen'
06:30	*new requirements*	
07:35	review requirements document (2)	
07:40	*new requirements*	'ja, hey, ze blijven bezig'
08:45	correct requirements document (1)	
	create design document (2)	
09:05	*new requirements*	'ik denk, eh, dat kan natuurlijk een half uur doorgaan met die requirements'
12:50	*new requirements*	
13:30	*finished correcting requirements document*	
14:35	stop reviewing requirements document (1)	
	create requirements document (1)	
15:40	stop creating requirements document (1)	
	review requirements document (1)	
15:45	*finished creating requirements document* 7	
17:30	stop reviewing requirements document (1)	
	create design document (3)	
	correct requirements document (1)	
19:00	stop reviewing requirements document (1)	
	review design document (1)	
19:05	*new requirements*	
20:10	review requirements document (1)	
20:40	stop creating design document (1)	
	correct design document (1)	
20:55	*new requirements*	
21:05	*finished creating design document*	
22:10	review design document (1)	
	create code (1)	
	create system test plan (1)	
22:15	*finished correcting requirements document*	
22:35	create code (1)	
22:40	*finished reviewing requirements document*	

23:00	create code (1)
24:00	stop reviewing design document (1)
	correct design document (1)
24:05	*finished correcting design document*
25:40	create code (1)
25:40	create system test plan (1)
25:50	*finished creating system test plan*
27:10	review system test plan (2)
27:15	*new requirements*
28:05	stop reviewing system test plan (1)
	create requirements document (1)
28:10	*finished creating requirements document*
29:05	correct system test plan (1)
	stop creating code (1)
	correct design document (1)
29:30	*finished creating code*
	finished correcting design document
30:50	inspect code (3)
	review requirements document (1)
30:55	*finished reviewing requirements document*
31:05	correct requirements document (1)
31:10	*finished reviewing system test plan*
31:45	correct code (1)
31:50	*finished correcting requirements document*
32:30	integrate code (1)
	stop correcting code (1)
	inspect code (1)
	stop inspecting code (1)
	correct code (1)
32:35	*finished correcting system test plan*
32:45	integrate code (1)
34:00	stop inspecting code (3)
	create system test plan (1)
34:05	*finished creating system test plan*

34:45	integrate code (2)	
	correct code (1)	
	review design document (1)	
34:50	*finished correcting code*	
35:30	integrate code (1)	
35:35	*finished integrating code*	
36:00	inspect code (4)	
	system test (1)	
36:05	*new requirements*	
36:35	stop inspecting code (1)	
	review requirements document (1)	
36:40	*finished reviewing requirements document*	
37:10	system test (1)	'hij heeft super veel test ervaring'
	stop reviewing design document (1)	
	create system test plan (1)	
37:15	*finished creating system test plan*	
37:40	review design document (1)	
	correct design document (1)	'dat is niet handig, het programma is bijna klaar en ik heb nog fouten in het design'
37:45	*new requirements*	
38:00	correct requirements document (1)	
	create requirements document (1)	
38:05	*finished correcting requirements document*	
38:10	*finished creating requirements document*	
38:15	review requirements document (1)	
38:20	*finished reviewing requirements document*	
38:50	correct requirements document (1)	
38:55	*finished correcting requirements document*	
39:10	create design document (1)	
39:15	*finished creating design document*	
		'het begint onderhand uit de klauwen te lopen met geld'
40:05	correct design document (1)	
	correct code (1)	
40:20	*finished correcting design document*	

41:00	correct system test plan (2)	
	review system test plan (1)	
41:05	*new requirements*	
41:10	*finished correcting system test plan*	
41:45	stop system test (1)	
	create requirements document (1)	
	review requirements document (1)	
	correct design document (1)	
41:50	*finished reviewing system test plan*	
42:05	correct system test plan (1)	
42:10	*finished creating requirements document*	
42:15	*finished correcting design document*	
42:25	create design document (1)	
42:30	*finished creating design document*	
42:45	create system test plan (1)	
42:50	*finished creating system test plan*	
43:00	correct design document (1)	
43:05	*finished correcting design document*	
43:55	create code (1)	
	correct code (1)	
	stop system test (1)	
	system test (1)	
44:00	*finished creating code*	
44:20	integrate code (1)	
44:25	*finished integrating code*	
44:45	correct code (1)	
45:20	system test (1)	
	correct design document (1)	
		'ik moet een beetje in de gaten houden dat we niet over het geld heen gaan'
46:10	*new requirements*	
46:55	create design document (1)	
	correct requirements document (1)	
	correct design document (1)	
47:00	*finished correcting requirements document*	

47:05	correct design document (1)	
47:10	*finished creating design document*	
47:55	create code (2)	
48:00	*finished creating code*	
48:40	*finished creating design document*	
49:10	deliver project to customer	'volgens mij zitten we nu ook redelijk over de tijd'

score: 49

D Transcripties van de discussies

Legenda

waterval	De volgorde van het waterval model en dat je niet terug kunt naar een vorige fase wordt genoemd.
looptijd	Het risico dat het project gaat uitlopen wordt genoemd.
kosten	Het risico dat de geraamde kosten voor het project overschreden worden wordt genoemd.
requirements	Het risico dat de klant keer op keer met nieuwe requirements komt wordt genoemd.
communicatie	Communicatie problemen bij grote project teams worden genoemd.
negeren	Doorgaan zonder de nieuwe eisen te implementeren wordt genoemd.
controle	Review of inspectie van de al bestaande artefacten (requirements, design, code, tests) wordt genoemd.

Groep 1: Twee SimSE spelers

P1	sowieso is het altijd een risico als een minister zich gaat verdiepen in jouw project	
P2	ambtenaren, ambtenarij, dat is het grootste risico hier	
P1	Uiteraard, dat sowieso maar dan gaat die minister van alles verzinnen en uiteindelijk heeft hij dat allemaal verzonnen en moet alles weer anders	*requirements (+)*
P2	Nou, ik denk inderdaad dat dat het grote probleem hier zal zijn, dat die kerel iedere keer halverwege denkt: ooh, maar dit gaat niet goed dus dat gaan we veranderen, dat is natuurlijk geen waterval methode	*requirements (-), waterval (+)*
P1	nee, inderdaad	
P2	ik denk dus dat hij iedere keer nieuwe requirements zal gaan opstellen die dan ...	*requirements (+)*
P1	... ergens halverwege het project geïntegreerd gaan moeten worden met wat er al bestaat	
P2	ja, ja dat	
P1	en dan moeten die dus externe mensen inhuren	
P2	ja	
P1	om de nieuwe requirements te doen, maar goed dan komen er weer nieuwe requirements en hebben ze weer nieuwe mensen nodig om de nieuwe requirements te verwerken	
P2	maar waarom zouden de oude mensen niet met de nieuwe requirements ...	
P1	omdat het meestal zo is dat er dan nieuwe requirements bij komen maar dat wil niet zeggen dat de er oude requirements af gaan	
P2	Ja, oké	
P1	dus je krijgt steeds meer	
P2	en waarom falen ICT projecten? Dat is natuurlijk omdat ze heel duur zijn -wat het dan natuurlijk wordt want je moet steeds meer externe ontwikkelaars hebben- en dat de requirements uiteindelijk niet stroken met wat de klant als verwachting had, nee, dat je de verkeerde dingen oplevert, niet zozeer dat de requirements niet stroken maar dat je het verkeerde oplevert omdat er ergens een stukje miscommunicatie was	*kosten (+), communicatie (+)*
P1	ja, of dat er gewoon dingen in staan die niet kunnen omdat ze elkaar uitsluiten.	
P2	maar de risico's die wij dus zien zijn dus vooral dat de requirements iedere keer aangepast zullen worden en dat je dat dan maar weer op moet zien te vangen in je projectteam en dat dat eigenlijk niet kan in de waterval ontwikkelmethode. En ambtenarij natuurlijk	*requirements (+), waterval (+)*
P1	ja, ambtenarij is meer de oorzaak van het hele gebeuren natuurlijk. Waarschijnlijk is het ook nog zo'n minister die geen flauw idee heeft van wat hij nu eigenlijk aan het aansturen is	
P2	nee, dat zeker. De inhoudelijke kennis is natuurlijk weer nul.	

P2	Hoe zouden we deze situatie aanpakken? Ik denk dat ik vooral zou verwachtingen zo proberen te sturen naar datgene waar ik al mee bezig ben, zodat je zo min mogelijk hoeft aan te passen aan je requirements en dergelijke om die kerel tevreden te stellen en om eigenlijk stiekem zo weinig mogelijk invloed uit laat oefenen op wat er nu uiteindelijk opgeleverd wordt, dus hem het idee geven dat hij heel veel invloed heeft, maar dat je ondertussen gewoon lekker doorgaat met je ontwikkeltraject en dat je hem dan zo een beetje probeert te sturen dat z'n verwachtingen uiteindelijk overeen komen met wat je daadwerkelijk aan het opleveren bent	
P1	ja, dus je probeert meer de klant aan te passen dan de requirements.	
P2	ja, maar dat zou ik eigenlijk in iedere situatie doen.	
P1	ja, je moet hem inderdaad wel iedere keer echt voor de keuze stellen: je wilt dit nieuwe erbij, we hebben niet tijd voor de nieuwe en de oude dingen, dus wat moet eraf, wat gaan we niet doen?	
P2	ja	
P1	dan komen ze volgens mij wel snel tot de ontdekking dat niet alles kan.	
P2	ja, en op een gegeven moment moet je gewoon zeggen van ja, sorry kerel, maar de requirements fase zijn we voorbij en dit is nou gewoon de implementatie van wat je ooit gezegd hebt en we kunnen niet meer wijzigen, sorry, en het dan op een nette manier verpakken en hem tevreden houden en zo.	*negeren (+)*
P1	ja, maar dan krijg je altijd: "ja, maar het moet"	
P2	of je doet er gewoon langer over dan 4 jaar of zo, dan heb je een kans dat die kerel weer weg is en dat je een nieuwe minister krijgt, dat kan ook nog	*looptijd (-)*
P1	met waterval moet dat zeker lukken	
P2	absoluut	

Groep 2: Twee SimSE spelers

P3	oké, dus even samenvatten: we hebben hier een systeem dat in ontwikkeling is en opeens verandert de opdrachtgever, het nieuwe ministerie komt er, met nieuwe ministers en dan zijn die opeens bang dat dit project niet goed afkomt	
P4	ja, ze zijn bang dat ze iets verkeerds op gaan leveren. Dus de requirements worden aangepast. Dus dan is de eerste vraag: welke risico's loopt dit software ontwikkel project. Nou, ze lopen natuurlijk het risico dat het niet voldoet aan de specificaties, tenzij die opnieuw worden geëvalueerd	*controle (+)*
P3	ja, maar dat is natuurlijk lastig, want je hebt natuurlijk op andere aannames je project begonnen	
P4	ze gaan waarschijnlijk nu uit hun uren lopen, als de code nog moet worden gereviewed en herzien waarschijnlijk	*looptijd (+), controle (+)*
P3	je moet dan eigenlijk als je in het waterval proces zit het hele ding opnieuw doorlopen hè	*waterval (+)*
P4	ligt er natuurlijk aan hoe ingrijpend het is, maar schijnbaar hebben ze dus extra ontwikkelaars nodig om binnen korte tijd het ding alsnog op te kunnen leveren, want ik neem aan dat ze een capaciteit probleem hebben. Het eerste risico lijkt me dus de kosten, het tweede risico is natuurlijk dat je te laat gaat leveren	*kosten (+), looptijd (+)*
P3	of gewoon dat het systeem nooit afkomt, omdat het nooit aan de nieuwe eisen voldoet	*requirements (+)*
P4	ja, ik weet niet wat de looptijd is, want als er straks een derde kabinet komt begin je natuurlijk weer bij square one	*looptijd (-)*
P3	het loopt eigenlijk gewoon het zelfde gevaar als heel veel andere gefaalde software projecten. Precies hetzelfde dreigt hier ook te gebeuren: dat het gewoon niet afkomt, verkeerd afkomt	
P3	Hoe zou je de situatie aanpakken? Ik zou vooral in gesprek gaan en proberen uit te vinden in hoeverre die nieuwe aannames wel misschien overeen kunnen komen met de oude aannames of of de nieuwe aannames minder belangrijk zijn	
P4	ik sowieso denk ik eerst even kijken welke van de aannames die niet kloppen welke al wel en welke nog niet geïmplementeerd zijn. Als je dat in kaart hebt gebracht kun je gaan kijken hoeveel uren het kost om die dingen te verbeteren en uiteindelijk kun je dan gaan bepalen of je ze gaat verbeten of niet. Dan pas zou ik de ontwikkelaars gaan bij huren.	*controle (+)*
P3	Maar je zou hier toch ook kunnen stellen dat het niet de schuld van het ICT-bedrijf is dus daarom eigenlijk zouden we ook kunnen zeggen: nou wij hebben die opdracht gekregen...	
P4	je kan ook zeggen, we doen het niet maar...	*negeren (-)*
P3	maar goed, dan verlies je je project dat is niet echt een optie en je wil het wel zo doen dat iedereen tevreden is	
P4	ik denk dat je gewoon een prioriteitenlijst moet opstellen: wat zijn de belangrijkste wijzigingen, zijn ze essentieel	
P3	uitvoerbaar	

P4	en ja, dan moet je bekijken of het überhaupt zin heeft om die code te verbeteren of dat we beter opnieuw kunnen beginnen	
P3	ja, en dan natuurlijk al die inschattingen, moet je het hele waterval model nog een keertje helemaal terug naar scratch? Of kun je misschien met een kleine verandering al heel veel bereiken? Ook voor een regering moet het eigenlijk duidelijk zijn dat ze niet helemaal een lopend project overhoop kunnen gooien met nieuwe eisen en dan zeggen van: oh ja, het komt niet af. Dan moet de overheid ook wel een beetje bereid zijn te begrijpen dat er al een project gaande was en dat het niet gewoon opnieuw gestart kan worden	*waterval (+)*
P4	ja, natuurlijk, hij stelt daar zo dat hij wil niet dat de projecten mislukken omdat ze in hun algemeenheid mislukken dus ja dat is natuurlijk misschien niet de meest ideale drijfveer	
P3	ja, dat is sowieso een beetje een tegenspraak he	
P4	ja, hij wil niet dat ze mislukken maar eigenlijk is het al mislukt, want je kan het al niet meer leveren binnen de geraamde kosten	*kosten (+)*
P3	ja, door de nieuwe aannames. Dan moet je dus de aannames managen en de verwachtingen managen en kijken in hoeverre je aan die nieuwe verwachtingen kunt voldoen	
P4	ja	
P3	dat zou betekenen: een paar van die doelen misschien wel halen, een paar niet	
P4	ja, maar je moet, om even terug te vallen op die software, ook de hele test rommel, systemen en alles moet je herschrijven	*waterval (+), controle (+)*
P3	dus eigenlijk moet je opnieuw beginnen met de waterval	*waterval (+)*
P4	ja, dat is dan de vraag natuurlijk, je weet niet wat er aangepast moet worden en hoe ingrijpend het is. Als je weet wat de plannen zijn en wat je dus moet aanpassen dan kun je zeggen van: begin je opnieuw, pas je het aan of pas je het gedeeltelijk aan en heb je daadwerkelijk die ontwikkelaars nodig. Dat kun je pas doen als je alles geïnventariseerd hebt.	*controle (+)*

Groep 3: Twee SimSE spelers

P5	oké, welke risico's loopt dit software ontwikkelingsproject? Nou, ik denk omdat je werkt met de waterval methode, dat dan toch op een gegeven moment de requirements veranderen dus ja, dat past eigenlijk niet helemaal	*waterval (+)*
P6	en die kun je inderdaad niet tijdens het project aanpassen	*waterval (+)*
P5	dus ja, ik denk dat dat wel echt een groot probleem kan zijn. De oplossing die zij zeggen van externe ontwikkelaars inhuren, ja oké, ja je schiet er niks mee op of je moet het hele project echt opnieuw willen gaan doen. Heb jij daar verder nog iets aan toe te voegen?	
P6	ja, lastig. Maar hoe bedoelen ze met welke risico's loopt dit software ontwikkelingsproject? Op het moment dat er externe ontwikkelaars worden ingehuurd of op het moment dat die wijzigingen door de overheid zijn doorgevoerd?	
P5	ja, ik persoonlijk denk beiden natuurlijk, het zijn de twee risico's en die externen hoeven natuurlijk geen risico te zijn als je gewoon goede mensen inhuurt. Ik denk persoonlijk dat ik gewoon zou kijken van: hey wat moet er nou daadwerkelijk veranderen en zijn het gewoon dingen die we niet vitale functionaliteit die we gewoon weg kunnen laten of moeten we nu echt opnieuw beginnen? Maar ja, als je opnieuw begint heb je eigenlijk alweer een mislukt IT project.	*negeren (+)*
P6	ja, je kunt beter gewoon iets opleveren en het naderhand weer aanpassen aan de nieuwe regelgeving. Maar ja ik bedoel als je zo'n systeem ontwerpt moet je natuurlijk altijd rekening houden met externe factoren, zoals wetgeving die verandert. Dat zie je bijvoorbeeld ook met zorgverzekeraars. Zij bouwen complexe systemen, maar moeten rekening houden met regel- en wetgeving, die bij wijze van spreken ieder jaar kan veranderen.	*requirements (+), negeren (+)*
P5	dus gewoon doorgaan en...	
P6	ja, misschien bij zorgverzekeraars iets van informatie en kennis inwinnen	
P5	dus externe ontwikkelaars halen bij zorgverzekeraars of mensen met ervaring daar en dan...	
P6	Zij hebben waarschijnlijk vaker te maken gehad met dat soort ontwikkelingen	
P5	oké, duidelijk, ben ik het mee eens	

Controlegroep 1: twee proefpersonen die SimSE niet gespeeld hebben

P7	Welke risico's loopt het software ontwikkelingsproces? Nou, als die terwijl het waterval proces eigenlijk al bezig is, de requirements wil veranderen dan gaan er dingen mis.	*waterval (+)*
P8	dan gaat het waterval proces kapot en krijg je dus heel veel uitloop in tijd en gaat het dus heel veel meer geld kosten	*waterval (+), looptijd (+), kosten (+)*
P7	maar sowieso, omdat de requirements misschien al geïmplementeerd zijn of gedeeltelijk bekeken zijn en er dan veranderingen worden aangebracht dan betekent het dat die functies van de software misschien stuk gaan.	
P8	ja.	
P7	ja, of niet goed uit worden gevoerd.	
P8	ja, niet zoals ze bedoeld zijn	
P7	ja, inderdaad	
P8	ja, dat is het risico	
P7	inderdaad	
P7	nou, hoe zouden we deze situatie aanpakken?	
P8	ja, ik zou wel even goed kijken naar wat er nou precies gedaan is en wat er dan zodanig veranderd is dat de aanpassing noodzakelijk wordt gemaakt en dan ja, ik denk toch maar aanpassen want anders heb je inderdaad een softwarepakket wat dan toch al verouderd is en dan heeft het toch niet haar functionaliteit	
P7	nou, ik zou denk ik kijken hoeveel het systeem afwijkt van de gewenste situatie en en als het meevalt gewoon lekker doorgaan en anders kijken hoever men al is en dan kan men beter opnieuw beginnen of met een andere werkwijze gaan werken.	
P8	maar als het niet klopt met de plannen van de nieuwe regering zal er toch best wel iets meer dan een paar kleine dingetjes aangepast moeten worden.	
P7	ja, inderdaad. Het zou ook kunnen dat je zeg maar een soort extra module toevoegt die de fouten corrigeert.	
P8	ja, gewoon een update!	
P7	ja, dus eerst doorgaan en dan achteraf een aantal dingen bijstellen. Als het kleine regels zijn...	*negeren (+)*
P8	maar dan heb je het waterval proces dus niet meer zoals het hoort. Maar ja, dat is ook zelden dat dat goed doorlopen wordt.	*waterval (+)*
P7	ja, daar weten we alles van	

Controlegroep 2: twee proefpersonen die SimSE niet gespeeld hebben

P9	nou ja risico's, de meest logische is natuurlijk dat het gewoon uitloopt qua tijd en qua geld	*looptijd (+), kosten (+)*
P10	zeker omdat het een tijdje duurt voordat die externe ontwikkelaars helemaal ingewerkt zijn en waarschijnlijk een week nodig hebben voordat ze het programma kennen	
P9	ja, dan moet je nog inwerken	
P10	daardoor wordt het uiteindelijk waarschijnlijk duurder maar toch niet sneller	
P9	je hebt binnen een bedrijf op een gegeven moment consensus ontwikkeld over hoe je programmeert, dus het is een hele andere manier van uitwerken dus je moet ook heel sterk die architectuur bij externe ontwikkelaars bijbrengen	*communicatie (+)*
P10	ik weet ook niet zeker of er nog veiligheid problemen zijn als je iets voor de politie gaat ontwikkelen, dat je kan zeggen nou je mag alleen maar ontwikkelen als je ook echt te vertrouwen bent, als je dan externe mensen gaat inhuren dan moet je dat misschien eerst gaan checken en dan wordt het misschien lastig	
P9	ja, met de politie heb je allemaal clearance levels natuurlijk, maar goed in principe komt er natuurlijk niet echt informatie in te staan. Kijk, hoe je een proces-verbaal schrijft is niet zo'n probleem, maar de individuele proces-verbalen, die zijn privé, maar die heb je natuurlijk niet nodig tijdens het ontwikkelen, dan heb je gewoon proefdata	
P10	nee, zeker niet, maar stel je bouwt ergens een backdoor, dan kun je die achteraf allemaal lezen	
P9	ja, dat kan natuurlijk. Hoe zou je de situatie aanpakken? Ja, een geheimhoudingsplicht beschermt je niet tegen het inbouwen van backdoors natuurlijk	
P10	nee, daar heb je al de wet voor, maar als je je daaraan niet houdt dan eh....	
P9	ja, als je daar graag een backdoor wilt inbouwen, dan maakt een wet natuurlijk ook niet meer uit	
P10	ja, je kunt natuurlijk die code review door andere mensen nog laten doen	*controle (-)*
P9	ja, dat je gewoon altijd alles wat geprogrammeerd wordt door andere laat controleren. Ik heb trouwens laatst ergens gelezen dat het best wel schijnt te werken als je met twee programmeurs achter een computer zit, dus dat er eentje typt en dat je met z'n tweeën nadenkt. Dat lijkt heel eng, dat je dan minder productief bent, maar er worden zo veel minder fouten gemaakt dat het eigenlijk heel productief wordt. Dat zou je ook kunnen beschermen, dus dat je altijd een interne en een externe programmeur samen laat werken, dus dan zou je dat wel af kunnen vangen, tenzij de externe programmeur natuurlijk heel veel geld richting de interne programmeur schuift, maar...	
P10	maar ja, er is altijd een scenario te bedenken waar toch iets verkeerd gaat	
P9	ja, 100% veiligheid bestaat natuurlijk niet en volgens mij moet de politie daar ook rekening mee houden als ze zo'n project uitbesteden, want we zijn een groot ICT consultancy bedrijf dus het is al uitbesteed aan ons, dus de politie moet daar al rekening mee houden dat ze dat zelf ook nog een keer controleren. Even kijken, wat voor risico's hadden we nog meer? We hadden dus dat het uitloopt qua geld, tijd, ja hoe zul je dit aanpakken?	*looptijd (+), kosten (+)*
P10	veiligheid ook nog	

P9	ja, veiligheid hebben we net al een beetje besproken. Het inhuren van externe programmeurs is natuurlijk al een beetje en oplossing voor het uitlopen van tijd	
P10	ja, het is eigenlijk meer een compromis tussen tijd en geld	*looptijd (+), kosten (+)*
P9	ja	
P10	dus ja, ze moeten gewoon afwegen wat belangrijk is	
P9	ja, externe programmeurs, die zich helemaal moeten inlezen in het project ... je kan natuurlijk ook een aantal deelprojecten maken waar nog niet aan gewerkt is zodat ze gewoon vanaf scratch beginnen aan dat deelproject en dan hoeven ze zich ook niet helemaal in te lezen in het bestaande project	
P10	ja, er zijn zeker slimmere manieren die in te werken, of juist niet slimmere manieren die er uiteindelijk voor zorgen dat de hoeveelheid benodigde tijd groter of kleiner wordt	
P9	ja, als je systeem, als je architectuur gewoon een beetje modulair is opgebouwd dan geef je gewoon een API aan de ontwikkelaars en die hebben dan alleen dat om mee te werken en zo beperk je wel hoeveel ze moeten leren. Eh geld, ja het gaat sowieso meer geld kosten als je externe mensen inhuurt	*kosten (+)*
P10	ja, dan moet je gewoon die opdrachtgever vragen wat er belangrijker is, wil je het snel hebben of ietsje goedkoper...	
P9	ja, natuurlijk, ik weet niet zeker of de opdrachtgever de minister is maar als die minister iets wil aanpassen dan zal hij sowieso een handtekening moeten zetten onder een gewijzigd voorstel, er staat ergens een handtekening onder het plan, dat kun je niet zomaar veranderen	
P10	het is zeker ook afhankelijk van hoe ver je al bent in je project, als je nog aan het plannen bent dan kun je nog makkelijk ergens een stuk document gaan aanpassen en dan was het dat ook al zo'n beetje	
P9	ja, we zijn al aan het programmeren	
P10	ah, we zijn al aan het programmeren, ja dan wordt het waarschijnlijk best duur	*kosten (+)*
P9	dus je opdrachtgever zal toch wel even met je mee moeten kijken en opnieuw een handtekening zetten en dan kan je inderdaad voorleggen van, nou moet het dan af zijn of heeft u een flinke lading geld die we er tegenaan kunnen gooien	*looptijd (+), kosten (+)*

Controlegroep 3: twee proefpersonen die SimSE niet gespeeld hebben

P11	ja, welke risico's loopt dit software ontwikkelingsproject? Nou, ik denk dat de requirements niet meer kloppen en dat is het grote nadeel van het waterval model, dus, je kunt wel verder ontwikkelen, maar als je requirements niet meer kloppen dan is inderdaad het grote probleem dat het systeem al verouderd is voordat het wordt opgeleverd en, ja, wat zou je eraan kunnen doen? Ja toch maar weer terug naar de vorige stap denk ik, binnen waterval en zorgen dat alle requirements wel weer kloppen	*waterval (+)*
P12	ja, als je daarnaast ook nog eens een hoop nieuwe externe ontwikkelaars inhuurt, ten eerste klopt het dan nu waarschijnlijk al niet helemaal, daar moet je over gaan communiceren met mensen met mensen die het project niet van binnen en van buiten kennen, mij lijkt dat dat een drama wordt, om daar goed en netjes over te communiceren en dat allemaal er doorheen te krijgen. Ik weet niet hoe jij daar over denkt, discussiepartner?	*communicatie (+)*
P11	Dat denk ik ook	
P12	Het zou misschien wel kunnen dat je eerst een stap terug doet, heel die requirements opnieuw gaat bekijken. In feite doen je dan gewoon het project overnieuw en volgens mij als je dat doet dan kun je er wel weer externe ontwikkelaars bij nemen als je die weer vanaf een of andere fase eerder...	
P11	Als extra mensen, zeg maar	
P12	ja	
P11	maar dan nog zal je het moeten uitleggen	
P12	maar ik weet ook niet precies wat ze met externe ontwikkelaars bedoelen, ik bedoel als dat weer een heel team mensen is met een hele hiërarchie en weet ik het allemaal, misschien alleen programmeurs, dat je daar nog wel iets mee opschiet als je weer een stap terug doet en dan aan het werk gaat	
P11	ja, inderdaad, als het een nieuw team is dan schiet het niet op	
P12	Hoe zou je deze situatie aanpakken? Ja, waarschijnlijk is het het verstandigst om gewoon overnieuw te beginnen maar dat kost heel veel geld en dat willen ze meestal niet	*kosten (-)*
P11	nou, ten eerste zou ik mijn baas uitmaken voor een idioot	
P12	ja, dat die requirements niet kloppen	
P11	nee, je baas stelt voor een aantal externe ontwikkelaars in te huren	
P12	ja	
P11	dat lijkt me niet heel handig	
P12	ja, hoe zou je het aanpakken? Overstappen op sowieso een ander ding, dat Agile	
P11	Agile, ja dat kan	
P12	want het waterval model is volgens mij überhaupt prehistorisch	
P11	dat kan	
P12	ja, oké dan zou ik de situatie vooral niet aanpakken want ik ga ervoor zorgen nooit in die situatie terecht te komen	
P11	ja, maar stel? Puur hypothetisch	

P12 Hoe zou je het aanpakken? Ja, toch een stap terug doen. Het is niet anders, je kan wel doen alsof dat niet nodig is en met botte kop doorgaan en dan faalt uiteindelijk het hele project

P11 ja, je kunt beter teruggaan maar alles veranderen, dan heb je vertraging maar dan komt er uiteindelijk wel weer iets uit wat zou kunnen werken

P12 en misschien, wat wel waarschijnlijk schade beperkend zou kunnen zijn is wat je tot nu toe hebt ontwikkeld, op zoek gaan naar modules die daar niet door aangetast worden, die eventueel opzij zetten en hergebruiken

P11 ja