# Assignment 4

Robin bakker, Robert Kraaijeveld, Cees-Jan Nolen, William de Visser, Steven Schenk – INF2C

Table of Contents

[Assignment 4 1](#_Toc434231490)

[A: Risk Analysis & Counter measures 3](#_Toc434231491)

[B: Configuration Management Plan 7](#_Toc434231492)

[C: Process Improvement 18](#_Toc434231493)

[D: Reflectie 19](#_Toc434231494)

# A: Risk Analysis & Counter measures

**Risico Analyse**

Bij een project komen veel risico's kijken. Denk bijvoorbeeld aan het wegvallen van een teamlid of hardware die niet werkt zoals het moet. Daarom is het belangrijk om deze risico's op te nemen in een document waarin de gevolgen van de risico worden beschreven en wat te doen als het risico werkelijkheid wordt.

Zoals Confucius, een chinese filosoof, eens zei: *"Succes is altijd afhankelijk van een nauwkeurige voorbereiding en zonder die voorbereiding is falen een feit."*

De Risico-Analyse zal in vier stukken verdeeld worden. Als eerste de indentificatie, hierbij worden de risico's op een rijtje gezet, genummerd en het type risico bepaalt.

Ten tweede de analyse van het risico, hierbij wordt de kans op het risico en de gevolgen van het risico bepaald. Ten derde de planning, Hierbij moet worden bepaald hoe elk risico word aangepakt, door het te vermijden of de effecten op het project ervan te minimaliseren. En als laatste het controleren van het risico hierbij worden mogelijk nieuwe risico's toegevoegd of risico's veranderd.

Dus even kort samengevat de risico-analyse is verdeeld in 4 stukken:

* Identificatie
* Analyse van de risico's.
* Planning
* Controleren

**Identificatie**

Er zijn 6 verschillende risico's: Technologische risico's, menselijke risico's, organisationele risico's, tools risico's, requirements risico's en schattings risico's .

We zullen na de tabel per kopje de risico's, die uitleg nodig hebben, uitleggen.

|  |  |  |
| --- | --- | --- |
| Risico type | Mogelijke Risico | Nummer |
| Technologische | Server stopt met werken. | 1 |
| Menselijke | Teamlid wordt ziek.  Teamlid stopt met de opleiding. | 2  3 |
| Organisationele | De product owner verandert. | 4 |
| Tools | De benodigde programmeerprogamma's zijn opeens niet meer gratis voor studenten. | 5 |
| Requirements | Product Owner wil requirements veranderen die veel programmeer- of designwerk kost. | 6 |
| Schattings | De grootte/tijd van een taak is onderschat. | 7 |

*Tabel 1*

Technologische

We lenen van school een server, de server staat op school. Het probleem is dat de kans bestaat dat de servers niet werken, de toegang gewijgerd wordt of de server crasht, waardoor er opnieuw aan de beheerder moet worden gevraagd om de server te resetten of opnieuw toegang te verlenen.

Menselijke

De kans bestaat natuurlijk altijd dat een teamlid ziek wordt. Maar hierbij hebben we het over de ziektes die het werk langdurig belemmeren.

Één van de teamleden, William de Visser, twijfelt erg of hij wel thuis hoort bij de opleiding Informatica. De kans bestaat dat hij stopt met de opleiding en dan is er een teamlid minder. Dit zorg voor meer werklast op de andere teamleden wat kan leiden tot vertraging.

Organisationele

De kans bestaat natuurlijk altijd dat de product owner verandert. Dit kan verschillende redenen hebben. De verandering van de product owner zal niet een grote impact hebben maar zal wel tot introductie leiden wat tijd kost.

Tools

De programmeer programma's die wij momenteel gebruiken zijn Clion en IntelliJ met laravel. Als die opeens niet meer gratis te gebruiken zijn moeten we overstappen naar een ander programma wat enige introductie zal vereisen. Dit zal veel tijd kosten, tenzij we betalen voor de programma's, maar dat zal dan moeten afgewogen worden tegen de tijd die het anders kost.

Requirements

Als de product owner opeens een belangrijke requirement wilt veranderen die veel tijd kost om aan te voldoen en een grote verandering bevat dan zal er tijd moeten worden vrijgemaakt om de verandering door te voeren.

Schattings

Het kan gebeuren dat er een taak, onderdeel van een requirement, wordt onderschat. Als dat gebeurt kan er een tekort aan tijd ontstaan.

**Analyse van de risico's**

Aan de hand van de voorgaande Identificatie kan er nu een tabel worden gemaakt waarin de effecten worden gedefinieerd. Deze is gesorteerd van Serieus naar Insignificant.

|  |  |  |
| --- | --- | --- |
| Risico | Mogelijkheid | Effect |
| Server stopt met werken. (1) | Hoog | Serieus |
| Teamlid stopt met opleiding. (3) | Gemiddeld | Serieus |
| De benodigde programmeerprogamma's zijn opeens niet meer gratis voor studenten. (5) | Laag | Serieus |
| Teamlid wordt ziek. (2) | Gemiddeld | Draagbaar |
| Product Owner wil requirements veranderen die veel programmeer- of designwerk kost. (6) | Laag | Draagbaar |
| De product owner verandert. (4) | Low | Draagbaar |
| De grootte/tijd van een taak is onderschat. (7) | Gemiddeld | Insignificant |

*Tabel 2*

**Planning**

Dit is de belangrijke fase, hierin wordt bepaald wat te doen. Bij ieder risico wordt een strategie bedacht. Want: *"Het is beter om te voorkomen dan te genezen".* Aan de hand van tabel 2 en veel overleg en afspraken wordt de strategie bepaald. Met de strategie kan het risico worden voorkomen of geminimaliseerd.

|  |  |
| --- | --- |
| Risico | Strategie |
| Server stopt met werken. | Omdat het hierbij om een schoolproject gaat hoeft het niet per se op een echte dedicated server te draaien. De product owner hoeft het product maar een paar keer te zien. We zouden dus ook gewoon een virtual machine kunnen draaien op een van onze laptops en het daarop laten zien. De software komt uiteindelijk toch niet echt tot productie. |
| Teamlid stopt opleiding. | Als iemand daarover twijfelt, wordt dit ver van te voren aangegeven. Ook duidelijk maken wat je taken zijn zodat de workload zo goed mogelijk verdeeld kan worden onder de overgebleven teamleden. |
| Programmeer programma's niet meer gratis. | Meteen een goede vervanging van het programma regelen. Goed overleg met de PO en strakke en duidelijke deadlines stellen. Hulp vragen aan docenten die ermee werken kan ook helpen. |
| Teamlid ziek. | Goede afspraken maken als het gebeurt van wat hij doet en wat hij niet doet. Alles wat hij niet kan doen verdelen onder de overige teamleden. |
| PO wil requirements veranderen. | Allereerst er achter komen wat er precies anders wordt. Goed doorvragen hierbij! Daarna één of meerdere prototype's maken, hierbij wordt al snel duidelijk wat de PO precies bedoelt. Als je zeker weet wat er gemaakt moet worden, moet de workload van de desbetreffende requirement verdeeld worden onder de teamleden. |
| PO veranderd. | Een zo duidelijk mogelijke introductie geven waardoor er weinig communicatie-fouten ontstaan die negatieve invloed kunnen hebben op ons cijfer, |
| Grootte/tijd onderschat. | Als het bekend is dat het niet gaat lukken meteen naar de PO want hij is degene die het cijfer moet geven. |

*Tabel 3*

\*\*PO = Product Owner

**Controleren**

Wat nu nog heel belangrijk is, is het bijhouden van de risico's. Situatie's veranderen veel dus de risico's ook. Na ieder inlever moment bij de product owner zullen we met de groep een kleine vergadering houden waar we ieder risico apart behandelen.

Daarbij zal bepaald worden of het risico nog wel relevant is, zo niet dan gaat het van de lijst. Ook moet er gekeken worden naar nieuwe risico's. Als we op een nieuw risico stuiten zal hier meteen weer een risico-analyse en een planning voor gemaakt moeten worden. Het kan ook gebeuren dat de gevolgen van het risico veranderen dit moet ook meteen vastgelegd worden en een strategie bij bedacht worden.

Als een risico plaatsgevonden heeft is het belangrijk, voor de toekomst, om vast te leggen hoe er gehandeld is bij het risico, wat de gevolgen waren van het risico en vooral wat er beter kan als het risico nog een keer plaats vindt. Door deze reflectie kan er in de toekomst beter omgegaan worden met de desbetreffende problemen.

# B: Configuration Management Plan

**1. Introductie**

Dit document beschrijft het Configuration Management Plan (CMP). Dit CMP stelt de technische en administratieve leiding en controle voor het beheer van configuratieonderdelen, bijbehorend aan Project56, die geplaatst zijn onder configuratiecontrole. ***This document defines the project’s structure and methods for:***

* Identifying, defining, and baselining configuration items (CIs);
* Controlling modifications and releases of CIs;
* Reporting and recording status of CIs and any requested modifications;
* Ensuring completeness, consistency, and correctness of CIs; and
* Controlling storage, handling, and delivery of CIs.

Terwijl het project vordert, bepaalde delen van dit document zullen regelmatig geüpdatet worden.

**2. Overview**

Het project omvat een system dat input krijgt van een gebruiker in de vorm van .csv files. Deze moeten snel uitgelezen worden door het input proces en doorgestuurd worden naar het processing proces. Hier wordt de date opgeslagen en verwerkt in rapporten die de gebruiker kan downloaden op de output webpage. Mijlpalen om te bereiken zijn:

* Een webpage die .csv files kan uploaden en rapporten kan downloaden.
* Een filereader die de data leest en doorstuurt.
* Een verwerker die de ruwe data omzet in rapporten.

De stakeholder is het bedrijf CityGis vertegenwoordigd door de Product Owner (PO) dhr. Abbadi. Deze PO overlegt met ons over welke items en details er in het project moeten worden opgenomen.

**2.1 Beschrijving van het systeem**

Het system is een system dat input files ontvangt van de gebruiker. Deze files moeten verwerkt worden in rapporten e.d. welke de gebruiker weer kan downloaden als output. Hierbij moet het systeem op een server werken, waarbij wij gebruik maken van een Ubuntu server. Momenteel is er nog weinig historie aan het product qua veranderingen en aanpassingen aangezien we nog maar korte tijd bezig zijn met de implementatie van het project oftewel het programmeren ervan.

**2.2 Architectuur van het systeem**

De geplande system architectuur is een input-process-output architectuur. Hierbij wordt bij de input en process de programmeertaal C++ gebruikt. Dit wegens de hoge snelheid van verwerken die mogelijk is in die taal. Bij het output gedeelte, oftewel de webpage waarbij data gedownload kan worden, wordt gebruik gemaakt van het Laravel framework wat werkt op PHP. De applicatie taal is nog niet officieel vastgelegd, maar het zal Nederlands of Engels worden, of beiden.

**3. Aannames / Beperkingen / Risico’s**

**3.1 Aannames**

Aannames zullen voorkomen om de reden dat het project nog in ontwikkeling is en de implementatieperiode in de opstartfase. Een aantal aannames:

* De gebruiker (CityGis) laat de software draaien op een Ubuntu/Linux server.
* De gebruiker (klant van CityGis) heeft een van de volgende webbrowsers:
  + Google Chrome
  + Internet Explorer 9 of hoger
  + Mozilla Firefox
* De gebruiker (zowel CityGis als de klant van CityGis) heeft een actieve internetverbinding.
* De gebruiker (klant van CityGis) heeft een pdf-reader.

**3.2 Beperkingen**

1. Serverkwaliteit
2. Environment van de eindgebruiker
3. Beschikbaarheid van resources.

1: Een van de beperkingen die we kunnen ondervinden, en momenteel al hebben ondervonden, is de betrouwbaarheid en beschikbaarheid van de server. Momenteel hebben we een server gehuurd op school, maar de betrouw- en beschikbaarheid van deze server is erg laag. Vaak krijgen we geen toegang of is er niet genoeg recources qua geheugencapaciteit.

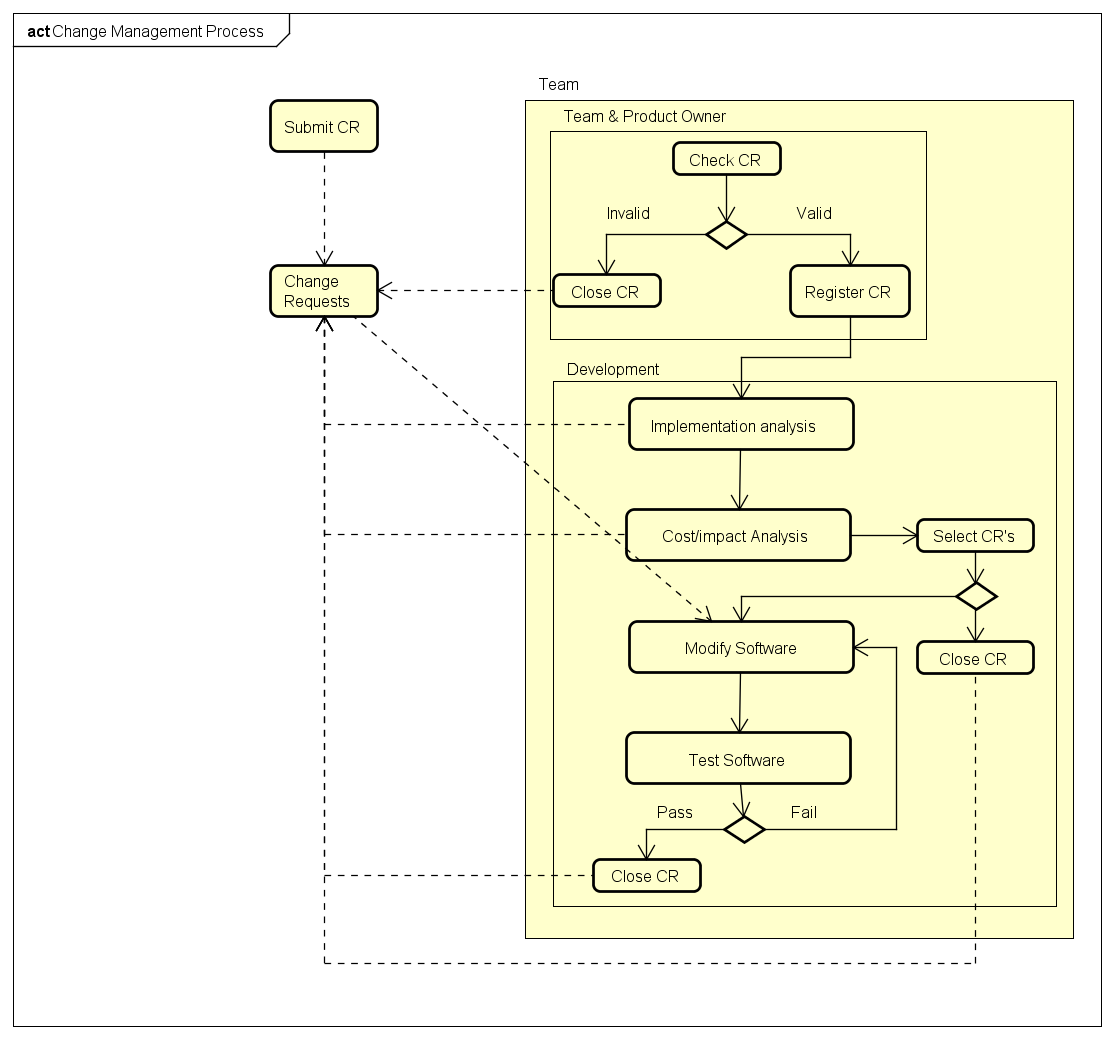
2: Wanneer de gebruiker een erg oude environment heeft kunnen wij niet garanderen dat alles werkt naar behoren. Wanneer de gebruiker een erg oude internetbrowser heeft, kan het voorkomen dat bepaalde functies op de webpage niet ondersteunt worden door de browser. Ook moet de gebruiker de beschikking hebben over een pdf-viewer aangezien onze rapport bestandsformaat hoogst waarschijnlijk een pdf formaat zal worden.

3: De resources die we krijgen van school zijn soms erg gelimiteerd om toe te passen tot het project. Een voorbeeld hiervan is de beschikbare ruimte op de server die we van school huren. De ruimte hierop is erg laag en we lopen constant tegen het limiet aan.

**3.3 Risico’s**

De risico’s die mogelijk zijn, is merge problemen bij de version control. Wij maken gebruik van GitHub, en daarbij kan iedereen zijn versie toevoegen. Meestal gaat dit goed maar wanneer meerdere personen op hetzelfde moment de code gaan updaten naar de globale repository kan dit merge problemen geven.

**4. Configuration Management Approach**

**4.1 Methods & Tools**

*Figuur 1: Change Management Process*

Tabel 1: Configuration Management Processes

| Process | Tools & Techniques |
| --- | --- |
| Change Management Process | Zoals zichtbaar in de diagram hierboven bijgevoegd. Daarin is te zien dat wanneer de klant een request instuurt, ons team (en Product Owner) er over gaan overleggen en of die geldig is. Daarna gaat ons team analyses uitvoeren en op basis daarvan worden de requests geselecteerd en die worden uitgewerkt. |
| Quality management | Ons team checkt elkaars code en de kwaliteit en correctheid ervan. Bovendien zal er uitvoerig getest gaan worden. |

**4.2 Rollen & Verantwoordelijkheden**

In ons team zijn er nog geen rollen en verantwoordelijkheden toegedeeld. De architectuur is een stuk minder complex in ons team omdat we bijvoorbeeld geen klantenservice of CCB hebben maar alleen een development team. Hiermee verandert daardoor ook het Change Management Process, wat een stuk anders en minder complex eruitziet door het gebrek aan diverse teams. Configuration Management Administration

**4.3 Configuration Identification**

De Configuration Items (CI’s) worden doorgenomen door ons team en de product owner. Wanneer we denken bepaalde items te willen opnemen in het systeem gaan we analyseren of dit mogelijk is. We doen een implementatieanalyse en een soort kostenanalyse. Bij de implementatieanalyse gaan we analyseren welke onderdelen van het systeem aangepast moeten worden. De kostenanalyse baseert zich daarop zodat de kosten berekent kunnen worden, in ons geval is dat tijd en in praktijk op de arbeidsmarkt moet ook de extra kostprijs voor de klant berekent worden. Wij kijken of het efficiënt is om het toe te voegen en of het nut heeft. Wanneer een belangrijker item minder kost, krijgt die prioriteit.*.*

**4.4 Naming Standard: Baselines**

Onze naming standard van Baselines zal als volgende gaan werken. Het ziet er in basale formulevorm zo uit: x.y.z. In eerste instantie hebben we de versienummer. Hiermee bedoelen we welk systeem het is, en aangezien dit het eerste systeem is zal het eerste systeem een 1.y.z versie zijn.. Wanneer er een nieuw systeem gecreëerd wordt zal dit 2.y.z heten. De Y wordt gedefinieerd door middel van het aantal grote mijlpalen in deze systeemversie. Wanneer een eerste grote mijlpaal bereikt wordt zal de verse van het eerste systeem 1.1.z zijn. Deze mijlpalen worden door het team van tevoren vastgelegd welke dat zullen zijn. De Z houdt de versienummer van de mijlpaal in. Wanneer het development team opnieuw updatet met code zal de Z aangepast worden. Aangezien een eerste mijlpaal nooit in een keer bereikt zal worden, ziet een realistisch voorbeeld er zo uit: 1.1.24.

**4.5 Configuration Control**

De CI’s worden onder configuration control geplaatst wanneer een CI op GitHub wordt geplaatst en het aan onze Definition Of Done voldoet (Scrum).

**4.6 Configuration Integrity**

Niet van toepassing.

1. Overzicht van aanpassingen

Tabel 3: Overzicht van aanpassingen

| Versie nummer | Datum | Auteur/beheerder | Beschrijving van de aanpassing |
| --- | --- | --- | --- |
| 1.0.0 | 15-10-2015 | Cooperatio | Start project |
| 1.1.0 | 22-10-2015 | Cooperatio | DirListener en parser af. |
| 1.1.1 | 29-10-2015 | Cooperatio | Verbetering van de dirListener en parser. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Afkortingen

Tabel 4: Afkortingen

| Acronym | Literal Translation |
| --- | --- |
| **CCB** | Configuration (or Change) Control Board |
| **CI** | Configuration Item |
| **CM** | Configuration Management |
| **PMP** | Project Management Plan |
| **PPA** | Project Process Agreement |
| **SDMP** | System Development Management Plan |
| **DOD** | Definition Of Done |
| **PO** | Product Owner |

1. Woordenlijst

Tabel 5: Woordenlijst

| Term | Definition |
| --- | --- |
| **Baseline** | (1) A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures. (2) A document or a set of such documents formally designated and fixed at a specific time during the life cycle of a configuration item. (3) Any agreement or result designated and fixed at a given time, from which changes require justification and approval.  (IEEE Std. 610-12-1990) A baseline is a configuration identification formally designated and applicable at a specific point in the life cycle of a configuration item. |
| **Build** | An operational version of a system or component that incorporates a specified subset of the capabilities that the final product will provide. (IEEE Std. 610-12-1990) |
| **Configuration** | The functional and physical characteristics of hardware or software as set forth in technical documentation or achieved in a product. (IEEE Std. 610-12-1990) |
| **Configuration Audit** | A functional configuration audit is conducted to verify that the development of a configuration item has been completed satisfactorily, that the item has achieved the performance and functional characteristics specified in the functional and allocated configuration identification, and that its operational and support documents are complete and satisfactory. A physical configuration audit is conducted to verify that a configuration item, as built, conforms to the technical documentation that defines it. (IEEE Std. 610-12-1990) |
| **Configuration Control** | An element of CM, consisting of the evaluation, coordination, approval or disapproval, and implementation of changes to configuration items after formal establishment of their configuration identification. (IEEE Std. 610-12-1990) |
| **Configuration (or Change) Control Board (CCB)** | A group of people responsible for evaluating and approving or disapproving proposed changes to configuration items, and for ensuring implementation of approved changes. (IEEE Std. 610-12-  1990) |
| **Configuration Identification** | An element of CM, consisting of selecting the configuration items for a system and recording their functional and physical characteristics in technical documentation. (IEEE Std. 610-12-1990) |
| Configuration Item (CI) | An aggregation of hardware, software, or both, that is designated for configuration management and treated as a single entity in the configuration process. (IEEE Std. 610-12-1990) |
| Configuration Management (CM) | A discipline applying technical and administrative direction and surveillance to identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements. (IEEE Std. 610-12-1990) |
| Configuration Status Accounting | An element of CM, consisting of the recording and reporting of information needed to manage a configuration effectively. This information includes a listing of the approved configuration identification, the status of proposed changes to the configuration, and the implementation status of approved changes. (IEEE Std. 610-12-1990) |
| Product | A physical entity (e.g., a piece of hardware or software) or artifact (e.g., a document) that is created by someone or some process. |

1. Referenced Documents

Niet van toepassing.

1. Approvals

Niet van toepassing.

# C: Process Improvement

Process improvents zijn op de volgende manier in ons project opgenomen.

*De robuustheid*: Op dit punt hebben wij gezorgd voor het opvangen van de onverwachte fouten die in het systeem kunnen voorkomen. Deze worden aan de gebruiker getoond ter informatie. De software zelf zal blijven functioneren.

*Begrijpelijkheid*: Bij dit punt hebben wij goed nagedacht om voor de gebruiker een eenvoudig systeem te maken. De gebruiker kan na twee dagen gebruik te hebben gemaakt van het systeem, volledig ermee overweg.

*Zichtbaarheid*: Tijdens het maken van de software is er duidelijk zichtbaar wat de status is van de huidige ontwikkelingen. Dit doordat er veel wordt uitgeschreven naar externe bronnen.

*Acceptabel*: De gedefinieerde vraag van de klant naar de software is duidelijk en daarnaast goed bruikbaar om deze aan de ontwikkelaars door te geven zodat deze de vraag van de klant kunnen beantwoorden door middel van het leveren van software.

*Ondersteuningsmogelijkheden*: Verschillende software programma’s / tools zullen worden gebruikt voor het ondersteunen van het ontwikkelen van de software. Te denken valt aan verschillende IDE’s maar daarnaast ook programma’s om de vraag van de klant om te zetten in verschillende diagrammen.

*Betrouwbaarheid*: De software wordt voordat deze wordt opgeleverd getest door verschillenden instanties waardoor eventuele product fouten kunnen worden opgevangen voordat deze in de officiële product versie komen.

Met de bovenstaande punten hopen wij de software in ons project te ontwikkelen. Deze punten zijn opgenomen om er voor te zorgen dat het ontwikkelproces zo goed mogelijk en daarnaast ook zo efficiënt mogelijk verloopt.

# D: Reflectie

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Theorie en praktijk vs. Eerdere kennis**

De (analyse-)cursussen/lessen die wij als groep al collectief hadden gehad voor het maken v.d. assignments was het volgende:

- Design en Usability

- Testen

- UML

- Relationele databases

- DFD’s en systeemmodellen

Hierbij was de ervaring die wij opgedaan hadden in UML en DFD’s het meest bruikbaar bij het maken van de assignments. Tevens was het feit dat wij al veel UML (en soms ook DFD-ish) modellen hebben gebruikt in onze 2 eerdere projecten en daarbuiten zeker een positieve invloed.

Tijdens het maken van de opdrachten waren de 2 grootste problemen die wij vaak hadden met het maken van modellen e.d. dat wij (a de syntax van de modellen niet 100% kenden en (b dat wij vaak niet goed wisten wanneer bepaalde modellen handig te gebruiken zijn.

Het niet goed kennen van de syntax van de modellen leek ons aanvankelijk niet zo’n heel erg groot probleem: Als de lezer de bedoeling van het model maar snapt is het toch goed? Dit bleek echter veel belangrijker te zijn dan gedacht. Als een model namelijk ook maar 1 element van de syntax fout heeft, is het hele model eigenlijk al geen model meer volgens de standaard (bv. UML) waar het naar gemaakt is. Dit leidt ertoe dat je als ontwikkelaar bij ieder model wat ook maar een klein beetje afwijkt een eigen legenda moet schrijven, wat het voordeel van het gebruiken van een standaard ondergraaft.

Case in point was bij ons ons Use-case overview diagram. Wij hadden dit diagram geschreven met alle symbolen die voor een Use-case diagram gemaakt worden, terwijl wij echter een overzicht wouden geven van alle Use case diagrammen! Doordat wij de syntax van 2 modellen door elkaar houden ontstond er veel verwarring. Samengevat hebben wij hiervan dus geleerd dat het goed gebruiken van de syntax van een model dat naar een bepaalde standaard gemaakt is veel belangrijker is dan dat het lijkt: Het spaart namelijk veel extra werk en verwarring uit.

Onze tweede veelvoorkomende fout was dat wij soms de verkeerde modellen in een bepaalde fase gebruikten. Ook dit was weer een geval waarin wij eerst dachten “Wat maakt dat nou uit” maar waarin wij later toch op deze gedachtegang terugkwamen. Het gebruiken van modellen met teveel detail in een vroege, nog relatief conceptuele fase kan tot teveel en/of foute aannames leidden, terwijl te conceptuele modellen tijdens bv. De implementatiefase kan leiden tot vaagheid en dubbele interpretaties over wat het systeem nou precies moet doen.

**Theorie en praktijk vs. Het Project**

De theorie (en de oefening die wij in de assignments kregen) is zeker goed van pas gekomen voor het project, vooral het goede gebruik van UML en DFD’s. Het kiezen en opzetten van architecturele structuren is in mindere mate gebruikt, aangezien wij voordat het project begon talen moesten kiezen, wat de architectuur van ons systeem al voor een groot deel dicteerde.

Ons gebruik van UML en DFD’s is zoals in het vorige kopje gezegd aanzienlijk verbeterd door de theorie en de assignments. Ook hebben wij geleerd om meer en gedetailleerder na te denken over de specificatie van Functionele requirements/User stories binnen een SRS: Functionele requirements moeten echt niet altijd heel specifiek zijn en user stories heten niet voor niets USER stories. Ook hebben wij geleerd dat het SRS goed bruikbaar is om aannames, dependencies en andere onzekerheden van het project op te vangen.

**Theorie en praktijk vs. leerdoelen**

Hieronder zijn de leerdoelen en de koppeling met de analyse-cursus van kwartaal 1 per groepslid weergegeven.

*STEVEN SCHENK*

Ik wou vooral meer leren over hoe ik de eisen van de klant kan interpreteren en deze om kan zetten in werkbare requirements. Alhoewel dit niet de de directe focus van deze cursus was heeft de feedback op ons SRS document zeker de kwaliteit van onze User- en (non)functionele requirements verbeterd, zoals u in het vorige kopje kon lezen.

*ROBERT KRAAIJEVELD*

Ik wou met deze cursus vooral meer leren over hoe ik DFD’s en andersoortige diagrammen kan gebruiken om een groter, uit meerdere componenten bestaand systeem zoals het systeem van Project56 kan modelleren. Niet alleen heb ik meer oefening hierin gehad gedurende deze cursus, maar ik heb vooral ook meer geleerd over welke modellen wanneer goed bruikbaar zijn; meer hierover staat onder “Theorie en praktijk vs. Eerdere kennis”.

*ROBIN BAKKER*

Mijn leerdoel was, samen met enkele andere groepsgenoten om mijn kennis van UML en DFD-modellen te verbeteren. Ik wou dit echter vooral kunnen doen binnen de ASTAH-modelleringsomgeving, aangezien dit programma veel gebruikt werd/word binnen ons project en zo’n beetje de De Facto standaard voor modelleren op de pc is. Door de vele oefening die ik binnen de assignments heb gekregen denk ik dat dit leerdoel aardig vervuld is!

*WILLIAM DE VISSER*

Ik wou in deze cursus vooral beter leren omgaan met UML modellen en ik wou meer oefening met dit soort modellen, onder andere omdat ik mijn UML tentamen van vorig jaar moet herkansen. In het taakverdelen vd. Opdrachten heb ik daarom ook expres alle UML-gerelateerde taken voor de assignments naar mij toe getrokken. K heb dankzij de feedback van mevr . Ubert meer kunnen leren over hoe UML-modellen wel en niet moeten worden gemaakt; onder “Theorie en praktijk vs. Eerdere kennis” kunt u meer lezen over op wat voor manier onze modellen-kunst is verbeterd.

*CEES-JAN NOLEN*

Ik wou vooral meer leren over hoe ik de eisen van de klant kan interpreteren en deze om kan zetten in werkbare requirements. Alhoewel dit niet de de directe focus van deze cursus was heeft de feedback op ons SRS document zeker de kwaliteit van onze User- en (non)functionele requirements verbeterd, zoals u in het vorige kopje kon lezen.