Wiselib

Wout Van Riel
se2-1415

16 November 2014

Contents

Versie	Auteur	Commentaar
version 0.1	Wout Van Riel	Eerste versie
version 0.2	Wout Van Riel	Eerste versie verbeteren

Table 1: Revision Chart

1 Introductie

1.1 Projectbeschrijving

Dit project heeft als doel een webtoepassing te creëren waar men publicaties op kan plaatsen en beheren. De toepassing zal in meerdere iteraties ontwikkeld worden. Het project is onderdeel van het vak Software Engineering dat gegeven wordt door Ragnhild Van Der Straeten [7] aan de Vrije Universiteit Brussel.

1.2 Evolutie van het SPMP

De eerste versie van het SPMP wordt ingeleverd op 5 november 2014. Daarna zal bij elke iteratie een geüpdate versie van het SPMP worden meegegeven. De geschiedenis van het SPMP is gegeven in tabel 1. De verantwoordelijkheid van dit document ligt bij de project manager. Dit SPMP volgt voornamelijk de standaard IEEE 1058-1998 [1].

1.3 Project Deliverables

In onderstaande tabellen staan de data wanneer de documenten 2 en de code 3 zullen worden afgeleverd en wanneer de presentaties 4 van het project zullen worden gegeven.

Datum	**Documenten**
Woensdag 05/11/2014	inleveren SPMP
Woensdag 19/11/2014	eerste versie documenten
Maandag 15/12/2014	Einde iteratie 1: opleveren documenten
Dinsdag 03/03/2015	Einde iteratie 2: opleveren documenten
Maandag 20/04/2015	Einde iteratie 3: opleveren documenten
Vrijdag 15/05/2015	Einde iteratie 4: finale oplevering

Table 2: Documenten

Datum	**Code**
Maandag 15/12/2014	Einde iteratie 1: opleveren code
Dinsdag 03/03/2015	Einde iteratie 2: opleveren code
Maandag 20/04/2015	Einde iteratie 3: opleveren code
Vrijdag 15/05/2015	Einde iteratie 4: finale oplevering

Table 3: Code

Datum	Presentaties
Vrijdag 19/12/2014	presentatie 1
Woensdag 11/03/2015	presentatie 2
Woensdag 22/04/2015	presentatie 3
Woensdag 20/05/2015	finale presentatie

Table 4: Presentaties

De documenten die telkens afgeleverd worden, zijn de volgende:

- Software Project Management Plan (SPMP)
- Software Test Plan (STD)
- Software Requirements Specification (SRS)
- Software Design Document (SDD)
- Minutes van alle vergaderingen

1.4 Referentiemateriaal

References

[1] *IEEE Std 1058-1998*

[2] *Team Website* `http://wilma.vub.ac.be/~se2_1415/`

[3] *GitHub Repository* `https://github.com/wiselib`

[4] *Teamwork* `https://wiselib.teamwork.com`

[5] *Slack* `https://wiselib.slack.com`

[6] *ShareLa-eX* `https://www.sharelatex.com`

[7] *Ragnhild Van Der Straeten* rvdstrea@vub.ac.be

[8] *JensNicolay* jnicolay@vub.ac.be

[9] *Dirk van Deun* dirk@dinf.vub.ac.be

1.5 Defenities en Acroniemen

Acroniem	Betekenis
SPMP	Software Project Management Plan
SRS	Software Requirements Specification
STP	Software Test Plan
SDD	Software Design Document
SQAP	Software Quality Assurance Plan
SCMP	Software Configuration Management Plan

Table 5: Ancroniemen

2 Project organisatie

2.1 Externe Interfaces

Het project is opgedragen door de VUB in het kader van de lessen "Software Engineering". De cliënten zijn Ragnhild Van Der Straeten [7] en assistent Jens Nicolay [8]. Later kan de basis van het project worden uitgebreid naar geleerden binnen en buiten de VUB. De applicatie zal initieel op Wilma staan, een multifunctionele Linux-server die wordt onderhouden door Dirk van Deun [9].

2.2 Interne structuur

De teamleden vullen verschillende rollen in. De rollen en hun functies zijn:

- Project Manager
  - Leidt het team
  - Contactpersoon van het team
  - Bemiddelaar tijdens meningsverschillen
  - Verantwoordelijk voor de vergaderingen
  - Maakt SPMP en onderhoudt het
- Configuration Manager
  - Beheert de tools en libraries die gebruikt worden
  - Beheert de GitHub
  - Maakt het SCMP en onderhoudt het
- Database Manager
  - Beheert de database
  - Maakt en onderhoudt het SDD samen met de Design Manager
- Design Manager
  - Maakt en onderhoudt het SDD
  - Beslist en beheert het design van de toepassing en de database
  - Communiceert met de cliënt over het design
- Test Manager
  - Zorgt dat het programma aan voldoende testen wordt onderworpen
  - Documenteert en rapporteert de resultaten van de testen
  - Maakt een test database aan
  - Maakt en onderhoudt het STD

- Quality Assurance
  - Zorgt voor de kwaliteit van de code
  - Is verantwoordelijk voor de kwaliteit van de applicatie
  - Zorgt ervoor dat aan alle nodige requirements voldaan zijn
  - Maakt en onderhoudt het SQAP
- Requirements Manager
  - Maakt en onderhoudt het SRS
  - Beslist de prioriteiten van de requirements
  - Communiceert met de cliënt over de requirements
- Webmaster
  - Maakt en onderhoudt de website

In de onderstaande tabel staan de functies en de personen die verantwoordelijk en back-up zijn voor deze functies.

Rol	Verantwoordelijke	Back-up
Project Manager	Wout Van Riel	Mathieu Reymond
Design Manager	Mathieu Reymond	Yannick Verschueren
Requirements Manager	Mathieu Reymond	Yannick Verschueren
Test Manager	Yannick Verschueren	Arno Moonens
Quality Assurance	Yannick Verschueren	Arno Moonens
Database Manager	Arno Moonens	Sam Vervaeck
Config Manager	Sam Vervaeck	Wout Van Riel
Webmaster	Sam Vervaeck	Wout Van Riel

Table 6: Rolverdeling

Er wordt gecommuniceerd in de groep met behulp van slack [5] en mail.

2.3 Werkplanning

In volgende Gant-chart 1 is de werkplanning tot de volgende deadline te zien.

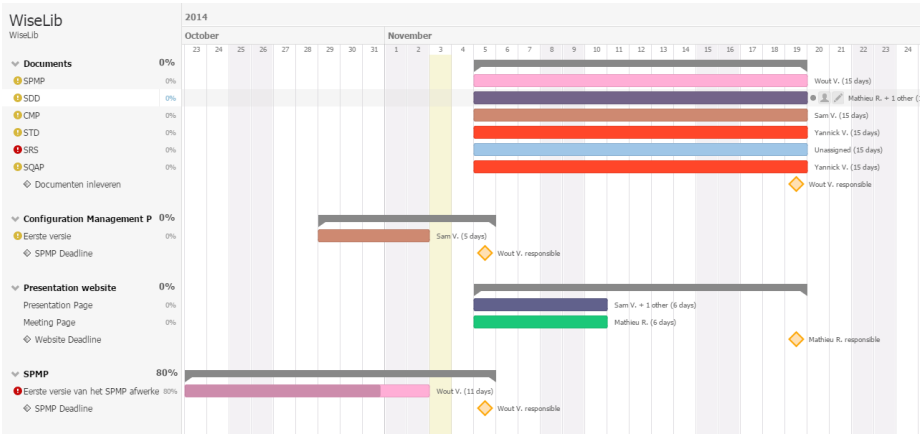

Figure 1: Gant chart

3 Management proces

3.1 Controle plan

3.1.1 Requirements controle

Om een overzicht te houden van alle requirements zal er op de site [2] een requirements dashboard worden opgezet. Deze tabel zal de requirements bijhouden en hun prioriteit. Als een requirement is afgewerkt, wordt de requirement doorstreept in de tabel. Mogelijke aanpassing van de requirements worden tussen de requirements manager en de cliënt besproken. Als er een requirement moet veranderen, dan moet de requirements manager dit als punt op de agenda zetten voor de volgende vergadering en past hij het SRS aan.

3.1.2 Planning controle

Er zijn algemene milestones en deadlines die tijdens de vergaderingen worden afgesproken. Iedereen is verantwoordelijk voor zijn eigen deadlines. Op elke vergadering vraagt de project manager aan iedereen de stand van zaken. Als er iemand zijn deadline niet kan halen, dan zal er besproken worden of er iemand anders mee kan helpen of dat de deadline kan opgeschoven worden. Mist iemand zijn deadline zonder geldige reden, dan zal de project manager die persoon daar over aanspreken en zal hij het vermelden op de volgende vergadering. De planning wordt met Teamwork [4] beheerd.

3.1.3 Rapportering

Documenten en code Op het einde van elke iteratie worden documenten, source code en (eventueel) andere artefacten opgeleverd. Afspraken in verband met deze opleveringen:

- Alle documenten en source code (inclusief unit tests) worden per mail afgeleverd als een enkele zipfile, met als naam se2-iterM, waarbij M het nummer van de iteratie is (voor eerste versie van documenten geldt M = 0). De aanlevering gebeurt ten laatste voor 9u00 's ochtends op de dag van de deadline.
- Alle documenten en source code worden worden overeenkomstig getagd/gebranchd (se2-iterM) in de repository.
- Andere artefacten (zoals executables) worden apart aangeleverd (direct, of via een link, in de opleveringsmail) en vermelden duidelijk de overeenkomstige iteratie in de bestandsnaam.
- De mail van de oplevering bevat een bondig overzicht (lijstje) van wat er precies opgeleverd wordt.

Presentatie Een presentatie duurt een half uur per groep en wordt ingevuld door een aantal sprekers, met 2 sprekers per groep. Alle groepsleden moeten minimum een keer presenteren. De volgende zaken worden besproken of gedemonstreerd:

- een demo van de toegevoegde functionaliteit ten opzichte van de vorige iteratie: gebruik requirements dashboard, en toon een testrapport (aantal tests, aantal succesvol)
- analyse van de ontmoette obstakels en de genomen beslissingen
- bespreking van de functionaliteiten die aan bod zullen komen in de volgende iteratie
- bespreking van eventuele obstakels, risico's, etc. in de volgende iteratie
- overzicht van de architectuur en design van de applicatie
- bespreking van de statistieken zoals de tijd per taak en per persoon en van de eventuele vertragingen (plus oplossingen om deze zo klein mogelijk te houden en te vermijden in de toekomst)

3.2 Risico Management Plan

3.2.1 Eén van de teamleden is ziek of verlaat het team

- Kans: 4
- Impact: 6
- Prioriteit: 7
- Oplossing: De Back-up neemt de positie van het teamlid over
- Verantwoordelijkheid: Project Manager

3.2.2 Slechte communicatie binnen de groep

- Kans: 6
- Impact: 5
- Prioriteit: 7
- Oplossing: Er worden algemene afspraken gemaakt over de communicatie
- Verantwoordelijkheid: Project Manager

3.2.3 Een deadline wordt gemist

- Kans: 2
- Impact: 9
- Prioriteit: 7
- Oplossing: Alle voortgang wordt op github geplaatst, elk teamlid geeft elke week zijn status op de vergadering
- Verantwoordelijkheid: Project Manager

3.2.4 Miscommunicatie tussen de cliënt en het team

- Kans: 2
- Impact: 7
- Prioriteit: 7
- Oplossing: De requirements manager houdt contact met de cliënt en bespreekt onduidelijkheden met de cliënt
- Verantwoordelijkheid: Requirements Manager

3.2.5 Plotselinge verandering in de requirements

- Kans: 2
- Impact: 5
- Prioriteit: 5
- Oplossing: Proberen de requirement in de bestaande code toe te voegen
- Verantwoordelijkheid: Requirements Manager

3.2.6 Server is tijdelijk offline

- Kans: 1
- Impact: 8
- Prioriteit: 8
- Oplossing: Een mirror server opzetten
- Verantwoordelijkheid: Configuration Manager

3.2.7 Onenigheid tussen de teamleden

- Kans: 4
- Impact: 6
- Prioriteit: 8
- Oplossing: Bemiddelen tussen de verschillende leden
- Verantwoordelijkheid: Project Manager

3.2.8 Database model staat de functies van het systeem niet toe

- Kans: 4
- Impact: 7
- Prioriteit: 8
- Oplossing: Het database model veranderen
- Verantwoordelijkheid: Database Manager

4 Technisch proces plan

4.1 Proces model

Er zal gebruik gemaakt worden van een iteratieve ontwikkeling. Dit model is gekozen vanwege de agenda waar we in verschillende iteraties code moeten afleveren. Het is ook gekozen omdat het makkelijker is om requirements per iteratie te verdelen. Zo wordt beslist aan het begin van elke iteratie welke requirements uitgewerkt zullen worden en zal het einde van de iteratie als deadline dienen voor de documenten en code.

4.2 Methodes, Tools en Technieken

In dit project worden enkel JavaScript, HTML5 en CSS gebruikt. Marco-talen, zoals onder meer SASS en CoffeeScript, zullen niet aanwezig zijn in dit project. Er wordt wel gebruik gemaakt van open-source softwarebibliotheken en softwareframeworks. De tools die gebruikt worden, zijn terug te vinden in het Configuration Management Plan **??** in tabel **??**.

Alle documentatie van het project worden op de GitHub [3] van WiseLib geplaatst.

4.3 Software Documentatie

Bij elke iteratie moeten er een aantal documenten mee doorgestuurd worden. De documenten zijn de volgende:

- Software Project Management Plan (SPMP)
  - Software Quality Assurance Plan (SQAP)
  - Software Configuration Management Plan (SCMP)
- Software Test Plan (STP)
- Software Requirements Specification (SRS)
- Software Design Document (SDD)
- Minutes van alle vergaderingen
- Code documentatie

5 Software Quality Assurance Plan

5.1 Doel

Dit document behandelt de processen en methodes die gebruikt worden tijdens de quality assurance van het project "WiseLib". Dit document volgt de IEEE standaarden voor Quality Assurance Plannen. [1]

5.2 Referentiedocumenten

SCMP Software Configuration Managment Plan

5.3 Management

5.3.1 Organisatie

De Quality Assurance wordt verzorgd door Yannick Verschueren, tevens de Test Manager, en Arno Moonens. Hierbij speelt Yannick de hoofdrol en zal Arno slechts invallen in geval van nood. De Quality Assurance Manager heeft als verantwoordelijkheid dat de gevraagde functionaliteiten van het project correct werken, en dat de geproduceerde code geen bugs meer bevat. Aangezien software-ontwikkeling verdeeld wordt over het gehele team zal er dus grote afhankelijkheid zijn tussen de QA manager en de overige teamleden.

5.3.2 Taken

Hoewel het Quality Assurance proces gedurende het hele project actief is, zal er na de voltooiing van een functionaliteit een periode van verhoogde activiteit volgen. Tijdens deze periode wordt er nagegaan of de functionaliteit op een correcte manier functioneert. Tussen deze verschillende periode zal er uiteraard ook gewerkt aan bugs en algemene correctheid van de code.

5.3.3 Verantwoordelijkheden

Aangezien het QA team uit één persoon bestaat, is de manager verantwoordelijk voor alle taken die kwaliteit verzekeren.

De manager heeft verschillende verantwoordelijkheden en komen allemaal terug op de concrete taken die het QA proces definiëren.

- Het bedenken en oprichten van de quality assurance procedures , standaarden en specificaties.
- Het herzien van de vereisten van de klant en voor zorgen dat deze na gegaan worden.
- Samenwerken met productie team om kwaliteits vereisten op te stellen.

[1] `http://users.csc.calpoly.edu/~jdalbey/205/Resources/IEEE7301989.pdf`

- Standaarden voor kwaliteit opzetten.
- Er voor zorgen dat standaarden worden nagegaan.
- Documentatie procedures opvolgen.

5.4 Documentatie vereisten

De documenten beschreven in de Project Deliverables volgen de IEEE standaarden en zijn geschreven in correct algemeen Nederlands. De documenten worden aangemaakt met behulp van Latex [2]

5.5 Standaarden, gebruiken, conventies en metrieken

5.5.1 Standaarden

Documentstandaarden De documenten gebruiken de IEEE standaarden [3]

Codering standaarden Codering standaarden en conventies zijn gebaseerd op Crockords code conventions en de Google Javascript Style Guide.Deze conventies worden verder besproken in deel vijf van het Software Design Document.

Commentaar standaarden Tijdens het schrijven van code zal gebruik gemaakt worden van JSDoc [4], om code te documenteren. Dit is een van de niet-functionele vereisten van het project en werd opgelegd door de klant.

5.5.2 Gebruiken

Naast het testen van de applicatie door algemeen gebruik, zal het testing-framework (zie 9) gebruikt worden om correctheid van de geimplementeerde functionaliteiten te garanderen doorheen de verschillende iteraties van de applicatie.

5.6 Test

Niet van toepassing (SVVP (Software Verification an Validation Plan) niet vereist)

5.7 Probleem rappotering

De teamleden kunnen problemen in zowel de software alsook in software-ontwikkelingsprocessen op verschillende manieren rapporteren.

[2] `http://www.latex-project.org/`
[3] `http://www.ieee.org/publications_standards/publications_standards_index.html`
[4] `http://usejsdoc.org/`

GitHub heeft een geïntegreerde issue-tracker. Dit maakt het onder meer mogelijk om naar bepaalde wijzigingen te refereren in het issue zelf, of omgekeerd om issues te sluiten bij het doorsturen van een wijziging.

Slack bevat een "bugs" kanaal waarin alle fouten van het systeem in detail kunnen besproken worden.

Teamwork bevat alle officiële rapporten met statistieken over de problemen.

5.8 Tools, technieken en methodologieën

Het QA proces gebruikt Mocha [5] als testing-framework. Dit framework zal in combinatie met Chai[6] zorgen voor taalprimitieven in JavaScript die het mogelijk maken om behaviour-driven tests neer te schrijven en uit te voeren. Er zijn drie verschillende interfaces voor Chai, maar wij beperken ons in dit project tot Should[7].

Het gebruik van het framework wordt in het Software Test Document uitgebreider uitgelegd.

5.8.1 Methodologie

Tijdens de implementatie van de applicatie wordt gebruik gemaakt van het Agile ontwikkelingsmodel. Dit is een test-driven ontwikkelingsmodel, dit betekent dat unit tests worden geimplementeerd voordat code wordt geschreven. Deze unit tests falen in het begin maar zullen naarmate het project vordert de geschreven code testen. Uiteraard worden ook de testen onderhouden en zullen zij mee evolueren met het verloop van het project.

5.9 Code Beheer

Deze specificatie vormt deel van het SCMP (Software Configuration Managment Plan).

5.10 Media Beheer

Deze specificatie vormt deel van het SCMP.

5.11 Leverancier beheer

Niet van toepassing. Software wordt nagekeken door de opdrachtgevers.

[5] `https://github.com/mochajs`
[6] http://chaijs.com
[7] http://chaijs.com/guide/styles/#should

6 Software Configuration Management Plan

Onder de term "Software Configuration Management" verstaan we alle ondersteunende tools en processen die het makkelijker maken om wijzigingen aan de software aan te brengen en bij te houden.

Dit document legt verder het beheer van de software configuration vast in dit project, alsook de rol van de Configuration Manager.

6.1 SCM-beheer

6.1.1 Organisatie

Sam Vervaeck is de hoofdverantwoordelijke van het Software Configuration Management Plan. In het geval dat hij zijn rol niet kan vervullen zal Wout Van Riel zijn taken overnemen.

6.1.2 SCM verantwoordelijkheden

6.1.3 Externe beperkingen op dit plan

Het is een vereiste van de klant dat GitHub als ontwikkelingsplatform wordt gebruikt door het team. Alle repositories moeten toegankelijk zijn voor de klant, zodat deze de voortgang van het project kan bijhouden.

Verder is mogelijk om eenderwelke tool en software-library te gebruiken, zolang deze open-source is en binnen de context van dit project valt. Bij twijfel moet de Configuration Manager een van de klanten contacteren.

6.2 SCM-activiteiten

6.2.1 Configuration-items

Het onderstaand stuk bevat een lijst van configuration-items (ofwel "CI's") die in dit project aan bod komen.

Presentatie-site

Deze mini-website bevat alle informatie betreffende de ontwikkeling van het project. De klant kan op ieder gewenst moment deze website raadplegen om de huidige staat van het project in te zien.

De website probeert zoveel mogelijk informatie te "aggregeren" vanuit de ontwikkelingstools. Ontbrekende informatie moet handmatig worden aangevuld.

- Een dashboard dat de voortgang van de requirements van het project weergeeft
- Een lijst van teamleden met hun functie
- De documenten en rapporten die moeten worden opgeleverd

- Een wegwijzer voor tools die ook consulteerbaar zijn voor de klant, zoals GitHub

Voor de implementatie en het onderhoud van de website zijn zowel de Configuration Manager als de Design Manager verantwoordelijk. Van ieder teamlid wordt echter verwacht dat hij de website controleert en updatet indien nodig. De uiteindelijke verantwoordelijkheid ligt hiervoor bij de projectmanager.

De website is handgeschreven. De laatste versie van de website bevindt zich telkens op GitHub en zal automatisch worden gedeployed naar de

Documenten en rapporten

Alle documenten worden onder versiebeheer bijgehouden op GitHub. Het is de teamleden toegestaan om ShareLaTeX te gebruiken, aangezien deze tool voor iedereen beschikbaar is en makkelijker is om mee te werken. De configuration manager is dan verantwoordelijk voor de synchronisatie tussen ShareLaTeX en GitHub. Het is hem vrij of hij dit manueel doet, of hij hiervoor een automatisatie-script gebruikt.

Server-configuratie

De standaardinstellingen van de server zullen in een apart CI opgenomen worden, onder de naam *dotfiles*. Tot deze CI behoren alle globale configuratiebestanden voor de applicatie zelf en voor ondersteunende software op de server zoals Vim[8]. In latere fasen kan dit artefact eventueel uitgebreid worden om configuratie-files te bevatten voor specifieke services. In dat geval zal dit artefact hernoemd worden naar *server-config*.

Applicatie

De applicatie zelf wordt in twee grote stukken onderverdeeld: een *server* en een *client*. Het SSD gaat dieper in op de architectuur van beide delen. De twee CI's zullen wel duidelijk van elkaar te onderscheiden zijn. Ontwikkeling aan de client en de server kan bijgevolg los van elkaar gebeuren.

6.3 SCM-hulpmiddelen

Ieder teamlid is vrij om zijn eigen editor en besturingssyteem te gebruiken, maar er wordt wel verondersteld dat er een minimale shell-omgeving beschikbaar is. Concreet moet alle software op zijn minst kunnen draaien op de laatste versies van Windows, Mac OS X en Ubuntu LTS.

Applicatienaam Omschrijving	Versienummer
GulpJS Automatisatie van taken	3.8.10
Git Versiebeheersysteem	1.9.3
Travis CI Tests uitvoeren en deployment	2014-11-13.17-08

[8] `http://www.vim.org`

6.3.1 Controle van de configuratie

Om het ontwikkelingsproces zo vlot mogelijk te laten verlopen krijgt ieder teamlid vrije toegang tot de repositories. Er wordt wel verondersteld dat de configuration manager alle wijzigingen nauw opvolgt, zodat hij makkelijk een foute toepassing van een tool of proces kan corrigeren.

Het branching model van elke CI volgt in grote lijnen Git-flow[9] en zijn extensie GitHub-flow[10]. Voor elke nieuwe functionele requirement of bugfix wordt een nieuwe branch gemaakt. Er gelden enkele regels voor de naamgeving van de branch:

- enkel kleine letters en cijfers
- streepjes ("-") als scheidingsteken
- prefix alle features met *req-*
- prefix alle bugfixes met *fix-*
- een korte maar goede beschrijving zodat de logs duidelijk zijn

Teamleden mogen hun lokale history rebasen om overbodige merges te vermijden. Dit is geen verplichting en wordt enkel aangeraden aan personen die vertrouwd zijn met het concept van rebasing.

Indien twee of meerdere teamleden met eenzelfde bug of feature bezig zijn, moeten ze hun branchnaam prefixen met *gebruikersnaam-*. Bij een conflict heeft de Test Manager het laatste woord over welke versie teruggaat naar de hoofdbranch.

Er bestaat geen aparte productie-branch. In de plaats daarvan geven tags aan welke versies op de masterbranch uitvoerig gecontroleerd werden. Deze tags hebben altijd de vorm van *se2-iterN-M*, waarbij *M* bij elke productie-release. De manier waarop deze tags worden aangemaakt en beheerd staat in het volgende punt beschreven.

6.3.2 Beheer van uitgaven en opleveringen

Elke CI wordt gekenmerkt door een eigen tempo in het ontiwkkelproces. Om die reden verschilt het release/deployment-proces van iedere tool lichtjes.

Voor de applicatie is het nodig dat de productieversie van zowel de client als de server grondig getest is geweest en geen fouten bevat. Om die reden zal de Test Manager handmatig aanduiden welke versies gereleased kunnen worden. Deze releases worden dan automatisch gedeployed.

[9] http://nvie.com/posts/a-successful-git-branching-model/
[10] http://scottchacon.com/2011/08/31/github-flow.html

De Configuration Manager haalt na iedere iteratie de documenten rechtstreeks van ShareLaTeX en plaatst ze in het versiebeheersysteem. Na elke wijziging wordt er dus automatisch een nieuwe release gemaakt. De releases bevatten de documenten in PDF-vorm, zodat deze automatisch kunnen worden gedeployed naar de presentatie-website.

De presentatie-website is minder complex. Bovendien is het minder belangrijk dat de website volledig foutloos is, maar is het wel belangrijk dat snelle updates mogelijk zijn. Daarom wordt voor de presentatie-website de release-fase overgeslagen en rechtstreeks gedeployed vanuit de meest recente versie.

Onsderstaande tabel vat deze configuratie samen.

CI-naam	Release	Deployment
Server	na goedkeuring	bij nieuwe release
Client	na goedkeuring	bij nieuwe release
Documenten	nieuwe wijziging op de hoofdbranch	bij nieuwe release
Presentatie-site	bij iedere iteratie	bij nieuwe versie
Serverconfiguratie	bij iedere iteratie	n.v.t.

6.4 Onderhoud van het SCM-plan

De Configuration Manager is verplicht om aanpassingen in het beheer van de software configuratie grondig bij te houden. Hij dient bij het begin van elke iteratie dit plan grondig te herzien en aan de passen aan de verwachtingen van die (en volgende) iteraties.

Aangezien net zoals de software zelf het ook goed mogelijk is dat de ondersteunende tools of processen in de toekomst moeten veranderen, is het altijd mogelijk om wijzigingen aan dit Configuration Management Plan aan te brengen. De enige vereiste is dat het hele team hiermee akkoord gaat nadat de Configuration Manager de wijzigingen heeft voorgelegd.

Iedereen van het team heeft ook altijd de mogelijkheid om wijzigingen aan het Configuration Management plan voor te stellen. In dat geval geldt voor dat teamlid dezelfde regels als voor de Configuration Manager.