Software Project Management Plan
WiseLib

Wout Van Riel
Yannick Verschueren
Arno Moonens
Mathieu Reymond
se2-1415

19 April 2015

Contents

Versie	Auteur	Commentaar
version 0.1	Wout Van Riel	Eerste versie
version 0.2	Wout Van Riel	Eerste versie verbetert
version 1.0	Wout Van Riel	SPMP aangepast met behulp van de standaard
version 1.1	Wout Van Riel	Tweede versie
version 1.2	Wout Van Riel	Nakijken op spellingsfouten
version 2.0	Wout Van Riel	Derde versie
version 3.0	Wout Van Riel	Vierde versie

Table 1: Revision Chart

1 Introductie

1.1 Projectbeschrijving

1.1.1 Doel en objectieven

Dit project heeft als doel een webtoepassing te creëren waar men publicaties op kan plaatsen en beheren. Het project is onderdeel van het vak Software Engineering dat gegeven wordt door Ragnhild Van Der Straeten [10] aan de Vrije Universiteit Brussel. De vereisten die gegeven zijn voor dit project, worden beschreven in het SRS.

1.1.2 Veronderstellingen en beperkingen

Veronderstellingen We gaan er van uit dat de toepassing op verschillende toestellen gaat werken, zodat mensen overal aan hun publicaties kunnen geraken. Na deze iteratie werkt de toepassing op een computer en een mobiel toestel,dit zal in latere iteraties worden verbeterd.

Beperkingen

- Het systeem moet werken onder Wilma [3] en een up-to-date browser.
- GitHub [4] moet gebruikt worden als repository.
- Enkel JavaScript, HTML5, CSS en bijbehorende open-source frameworks en bibliotheken mogen worden gebruikt als programmeertaal of technologie.
- Enkel vrije software mag gebruikt worden, zowel voor het eindproduct als voor hulpmiddelen.
- Het systeem moet eenvoudig kunnen geïnstalleerd worden, op een standaard manier.
- De grafische gebruikersinterface moet aantrekkelijk en eenvoudig zijn.

1.1.3 Project Deliverables

In onderstaande tabellen staan de data wanneer de documenten 2 en de code 3 worden afgeleverd en wanneer presentaties 4 over het project 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

1.2 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].

2 Referentiemateriaal

References

[1] *IEEE Std 1058-1998*

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

[3] *Wilma* `http://wilma.vub.ac.be`

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

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

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

[7] *ShareLaTeX* `https://www.sharelatex.com`

[8] *Crockords code conventions* `http://javascript.crockford.com/code.html`

[9] *Google Javascript Style Guide* `https://google-styleguide.googlecode.com/svn/trunk/javascriptguide.xml`

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

[11] *Jens Nicolay* jnicolay@vub.ac.be

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

3 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
VUB	Vrije Universiteit Brussel

Table 5: Ancroniemen

4 Project organisatie

4.1 Externe interfaces

Het project is opgedragen door de VUB in het kader van de lessen "Software Engineering". De cliënten zijn Ragnhild Van Der Straeten [10] en assistent Jens Nicolay [11]. De applicatie zal initieel op Wilma [3] staan, een multifunctionele Linux-server die wordt onderhouden door Dirk van Deun [12].

4.2 Interne structuur

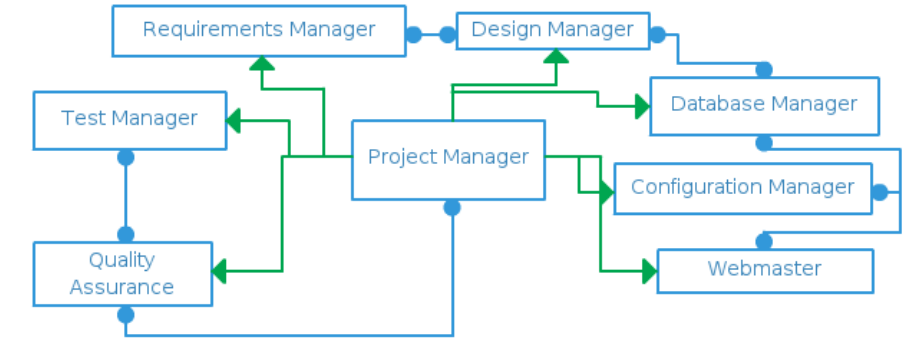

Figure 1: Interne organisatie

De groene pijlen duiden aan wie toezicht houdt over wie.
De blauwe pijlen duiden aan tussen wie er veel gecomuniceerd wordt.

4.3 Rollen en verantwoordelijkheden

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 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

5 Leidinggevende werkwijze plan

5.1 Start-up plan

5.1.1 Personeelsbezetting

De groep die aan dit project werkt, is samengesteld door professor Raghnild Van Der Straeten [10] en haar assistent Jens Nicolay[11]. In totaal bestaat de groep uit vijf personen en er is geen mogelijkheid om nieuwe werkkrachten in te huren. Er kan altijd een persoon wegvallen uit de groep. Dit kan door persoonlijke redenen of omdat hij niet in de groep kan functioneren. Als iemand uit de groep wil gaan, zal dit intern en met de professor besproken worden. 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	
Config Manager	Wout Van Riel	
Webmaster	Wout Van Riel	

Table 6: Rolverdeling

5.1.2 Plan om middelen te werven

Er wordt enkel gebruik gemaakt van vrije software. Dit project heeft geen budget om mee te werken en zal dus dienst doen op vrije software. Het is ook een vereiste die is meegegeven met het project.

5.1.3 Trainingsplan

In onderstaande tabel staan de onderdelen waar het personeel in getraind moeten worden. Naast elk onderdeel staat de wijze hoe het personeel opgeleid zal worden.

LaTeX	Tutorials en documentatie
Javascript	Tutorials en documentatie
HTML	Tutorials en documentatie
CSS	Tutorials en documentatie

Table 7: Trainingstabel

5.2 Werkplan

5.2.1 Werk activiteiten

Voor de tijd die aan de code wordt besteed, houden we de tijden bij met behulp van een timer op Teamwork[5]. Telkens als iemand aan het project werkt, moet hij zijn tijd opnemen en komt de tijd online zodat we een overzicht krijgen van de tijd die besteed is aan het project. Soms wordt er vergeten de tijd op te nemen dus wordt er een schatting gegeven die zo dicht mogelijk bij de werkelijke tijd ligt.

Iteratie 1 Omdat er geen tijden zijn bijgehouden tijdens deze iteratie wordt in onderstaande tabel enkel de geschatte duur van alle activiteiten weergeven en kan de werklast per persoon niet worden getoond.

Activiteit	**Verantwoordelijke**	**Geschatte duur**	**Document**
Project beheren	Project Manager	30u	SPMP
Kwaliteit	Quality Assurance	25u	SQAP
Testen	Test Manager	20u	STD
Design	Design Manager	25u	SDD
Vereisten	Requirements Manager	30u	SRS
Configuratie	Configuration Manager	25u	SCMP
Implementatie	Iedereen	50u pp	Code

Table 8: Werk activiteiten iteratie 1

Iteratie 2 In onderstaande tabel staat het werkschema van iteratie 2. In de afbeelding onder de tabel staat de werkplanning van deze iteratie gegeven in een gant-chart.

Wout Van Riel	Bibtex uploaden	5u
	Verslagen uitschrijven	1u
	SPMP updaten	5u
Arno Moonens	Registratie	9u
	Front-end verbeteren	9u
	Persoonlijke publicatielijst	3u
Yannick Verschueren	PDF analyse	15u
	Tests schrijven	9u
	STD	3u
	SQAP	2u
Mathieu Reymond	DBManager	9u
	Paper toevoegen	10u
	Account aanpassen	3u
	SDD	3u
	SRS	3u
Sam Vervaeck	Configuratie	10u

Table 9: Werk activiteiten iteratie 2

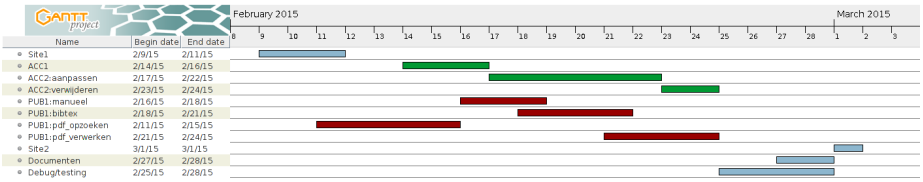

Figure 2: Planning iteratie 2

Iteratie 3 In onderstaande tabel staat het werkschema van iteratie 3. In de afbeelding onder de tabel staat de werkplanning van deze iteratie gegeven in een gant-chart. Deze iteratie is opgedeeld in twee sprints waarbij de eerste sprint van 11/3 tot 29/3 duurde en de tweede sprint van 29/3 tot 19/4 duurde. Tijdens deze iteratie is Sam Vervaeck weggevallen en hebben alle leden een paar van Sam zijn taken overgenomen.

Wout Van Riel	References	8u
	Verslagen uitschrijven	1u
	SPMP updaten	5u
Arno Moonens	Persoonlijke publicatielijst	2u
	Publicatiepagina	5u
	Personenpagina	4u
	Front-end aanpassen	9u
	Internationalization	8u
Yannick Verschueren	Auteurs die niet bestaan toevoegen	5u
	Opzoeken van research-papers	8u
	Testing	10u
	Ranking	8u
	STD	2u
	QA en SQAP	5u
Mathieu Reymond	DBManager	15u
	Opzoeken van research-papers	7u
	Affiliations	12u
	Persoonlijke bibliotheek	4u
	SDD	2u
	SRS	1u

Table 10: Werk activiteiten iteratie 3

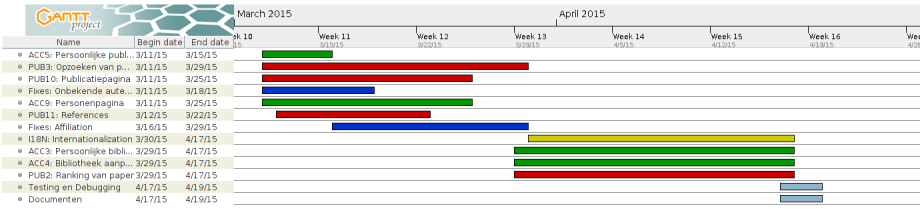

Figure 3: Planning iteratie 3

5.3 Controle plan

5.3.1 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 aanpassingen 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.

5.3.2 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 worden opgeschoven. 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 [5] beheerd.

5.3.3 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)

5.4 Risico Management Plan

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

Slechte communicatie binnen de groep

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

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

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

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

Server is tijdelijk offline

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

Onenigheid tussen de teamleden

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

Database model staat de functies van het systeem niet toe

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

Merge-conflicten in de master branch in Git

- Kans: 9
- Impact: 3
- Prioriteit: 7
- Oplossing: Er wordt een verantwoordelijke aangesteld die de pull requests moet goedkeuren
- Verantwoordelijkheid: Quality Assurance

Onvoldoende kennis van gebruikte tools en frameworks

- Kans: 8
- Impact: 4
- Prioriteit: 8
- Oplossing: Men zoekt extra informatie op over de tools of het framework, of men vraagt uitleg aan andere teamleden.
- Verantwoordelijkheid: Configuration Manager

6 Technische werkwijze plan

6.1 Werkwijze 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 gemakkelijker 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.

6.2 Methodes, tools en technieken

Javascript	Front-end en bacek-end
HTML	Front-end
CSS	Front-end
LaTeX	Documentatie
MariaSQL	Database
GulpJS	Taak-automatisatie
GitHub	Versiebeheersysteem
Travis CI	Testen

Table 11: Methodes, tools en technieken

6.3 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

7 Software Quality Assurance Plan

7.1 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]

7.2 Referentiedocumenten

SCMP Software Configuration Managment Plan

7.3 Management

7.3.1 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 softwareontwikkeling verdeeld wordt over het gehele team zal er dus grote afhankelijkheid zijn tussen de QA manager en de overige teamleden.

7.3.2 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.

Naast algemene controle van de code, valt ook de verzorging van de documenten onder de taak van de Quality Assurance manager.

7.3.3 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 nagegaan worden.

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

- Samenwerken met het productie team om kwaliteit vereisten op te stellen.
- Standaarden voor kwaliteit opzetten.
- Er voor zorgen dat standaarden worden nagegaan.
- Documentatie procedures opvolgen.

7.4 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]

7.5 Standaarden, gebruiken, conventies en metrieken

7.5.1 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 Configuration Management Plan.

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.

7.5.2 Gebruiken

Naast het testen van de applicatie door algemeen gebruik, zal het testing-framework (zie 8) gebruikt worden om correctheid van de geimplementeerde functionaliteiten te garanderen doorheen de verschillende iteraties van de applicatie.

7.6 Probleem rappotering

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

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.

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

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

Teamwork bevat alle officiële rapporten met statistieken over de problemen.

7.7 Tools, technieken en methodologieën

Het QA proces gebruikt Mocha [5] als testing-framework voor server code. 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]. Client code wordt getest door middel van Karma in cobminatie met het Mocha framework[8].

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

7.7.1 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 geïmplementeerd 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] `https://github.com/mochajs`
[6] http://chaijs.com
[7] http://chaijs.com/guide/styles/#should
[8] `http://karma-runner.github.io/`

8 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.

8.1 SCM-beheer

8.1.1 Organisatie

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

8.1.2 SCM verantwoordelijkheden

De Configuration Manager moet zorgen voor de nodige documentatie en uitleg over de processen die vastgelegd zijn in dit plan, zodat alle teamleden de richtlijnen correct kunnen toepassen.

8.1.3 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.

8.2 SCM-activiteiten

8.2.1 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 Wilma-server.

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[9]. 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

Ook de applicatie wordt beschouwd als een CI. Het SSD gaat dieper in op de architectuur van de applicatie. Zowel de client en de server zullen in hetzelfde CI zitten, omdat ervaring uitwijst dat dit het onderhoud drastisch kan verminderen.

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

8.3 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	Versienummer	Omschrijving
GulpJS	3.8.10	Taak-automatisatie
Git	1.9.3	Versiebeheersysteem
Travis CI	n.v.t.	Tests
pm2	0.12.5	Deployment

8.3.1 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.

VCS-branchingmodel

Het branching model van elke CI volgt in grote lijnen Git-flow[10] en zijn extensie GitHub-flow[11]. 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 zijn verplicht om hun lokale history te **rebasen** om overbodige merges te vermijden. Dit is enkel verplicht indien de branch die gerebased moet worden minder dan 5 commits achter loopt op zijn parent. In de andere gevallen is het toegelaten om een gewone *merge* te gebruiken, hoewel de rebase de voorkeur heeft waar mogelijk.

Indien twee of meerdere teamleden met eenzelfde bug of feature bezig zijn en ze deze wensen publiek te maken, dan zijn ze toegestaan om het project te **forken** om hun eigen versie toegankelijk te maken voor de andere teamleden

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

zonder de branches overbodig ingewikkeld te maken. De test manager en de configuration manager zijn beiden verantwoordelijk voor het goedkeuren of afwijzen van wijzigingen die terug naar de masterbranch moeten. Hiervoor kunnen de andere teamleden een *pull request* op GitHub zelf openen.

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.

8.3.2 Beheer van uitgaven en opleveringen

Elke CI wordt gekenmerkt door een eigen tempo in het ontwikkelingsproces. 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 met een script[12] dat ook de gezondheid van het proces op de server monitort.

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.

8.4 Code-standaarden

Om overzichtelijke, uniforme en duidelijke code te hebben is het nodig om een set van conventies te volgen. Deze richtlijnen zijn gebaseerd op Crockords code

[12] https://github.com/Unitech/pm2

conventions[8] en de Google Javascript Style Guide[9].

8.4.1 Taalregels

Variabelen

Alle variabelen worden door middel van "var" gedeclareerd, om te voorkomen dat de globale omgeving vervuild wordt.

Statements

- Elke lijn heeft maximum een simpele statement.
- ";" wordt altijd gebruikt om het einde van een statement aan te duiden.

Methoden en eigenschap definities

De verkozen stijl voor methodes is:

```
Foo.prototype.bar = function() {
    /* ... */
};
```

De verkozen stijl voor andere eigenschappen is om ze in de constructor te initialiseren:

```
/** @constructor */
function Foo() {
    this.bar = value;
}
```

Multiline string literals
Schrijf nooit strings op meerdere lijnen.

```
var myString = 'A rather long string of English text, an error message \
               actually that just keeps going and going';
```

In plaats daarvan worden ze geconcateneerd.

```
var myString = 'A rather long string of English text, an error message ' +
    'actually that just keeps going and going';
```

8.4.2 Stijlregels

Benamingen

Gebruik *camelCase* voor het benoemen van variabelen, functies, en andere identifiers, behalve voor de naam van een bestand.

- functionNamesLikeThis
- variableNamesLikeThis
- ClassNamesLikeThis
- EnumNamesLikeThis
- methodNamesLikeThis
- CONSTANT_VALUES_LIKE_THIS
- foo.namespaceNamesLikeThis.bar
- filenameslikethis.js

8.4.3 Code formattering

Indentatie Omdat "TAB" verschillende resultaten geeft afhankelijk van het gebruikte programma wordt deze niet gebruikt. Indentatie gebeurt met vier spaties.

Gekrulde haakjes De gekrulde haakjes worden op dezelfde lijn als wat ze openen gezet.

```
if (something) {
    // ...
} else {
    // ...
}
```

Witruimten Er zijn geen witruimten tussen linker -en rechter haakjes.

```
var arr = [1, 2, 3];  // No space after [ or before ].
```

Na "," wordt er altijd een witruimte gezet.

8.4.4 Databaseregels

Identifiers Alle identifiers worden vermeld in *snake_case*. Ze worden ook steeds met kleine letters geschreven.

- account
- first_name

Primary key Deze worden steeds "id" genoemd. Dit is kort, simpel en makkelijk te onthouden.

Foreign key De velden in een *foreign key* tabel moeten een combinatie zijn van de naam van de gerefereerde tabel en van de gerefereerde velden.

```
CREATE TABLE publication_written_by_person (
  publication_id        bigint NOT NULL REFERENCES publication(id),
  person_id      bigint NOT NULL REFERENCES person(id),
  CONSTRAINT publication_written_by_person_pkey PRIMARY KEY (publication_id,
  person_id));
```

8.5 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.

Elk lid 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.