Software Project Management Plan
for
WiseLib

Wout Van Riel
Mathieu Reymond
Sam Vervaeck
Yannick Verschueren
Arno Moonens

October 2014

Contents

Versie	Auteur	Commentaar
version 0.1	Wout Van Riel	Eerste versie

Table 1: Revision Chart

1 Introductie

1.1 Project Overzicht

Het project heeft de naam WiseLib meegekregen omdat het een bibliotheek van wijsheid en kennis moet worden op het net.

WiseLib heeft als doel een webtoepassing te zijn waar onderzoekers publicaties op kunnen plaatsen en beheren. Met behulp van deze webtoepassing zullen de onderzoekers overal aan hun publicaties kunnen. Om het nog toegankelijker te maken zal er ook ondersteuning zijn voor de mobiele gebruiker.

1.2 Project Deliverables

WiseLib biedt de gebruiker de volgende mogelijkheden aan:

- Publicaties uploaden en downloaden
- Beheren van eigen publicaties
- Andere publicaties consulteren
- Mogelijkheid om de site mobiel te gebruiken

Om die op tijd te kunnen verwezenlijken, zullen de deadlines van de kalender in tabel 2 gevolgd worden.

Datum	**To Do**
Woensdag 05/11/2014	inleveren SPMP
Woensdag 19/11/2014	eerste versie documenten
Maandag 15/12/2014	Einde iteratie 1: opleveren code en documenten
Vrijdag 19/12/2014	presentatie
Dinsdag 03/03/2015	Einde iteratie 2: opleveren code en documenten
Woensdag 11/03/2015	presentatie
Maandag 20/04/2015	Einde iteratie 3: opleveren code en documenten
Woensdag 22/04/2015	presentatie
Vrijdag 15/05/2015	Einde iteratie 4: finale oplevering
Woensdag 20/05/2015	finale presentatie

Table 2: Kalender

De documenten die telkens mee moeten 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

1.3 Evolutie van het SPMP

De project manager kijkt elke week het SPMP na. Als er nood is aan aanpassing dan zal deze de aanpassingen doen en zal hij de revision chart 1 aanvullen.

1.4 Referentiemateriaal

References

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

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

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

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

[5] *Trello* `https://trello.com/wiselib`

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

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

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

1.5 Defenities en Acroniemen

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

Table 3: 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 deze basis uitgebreid worden naar geleerden binnen en buiten de VUB. De communicatie met de cliënten gebeurt langs mail.

2.2 Interne structuur

Er zijn verschillende rollen die moeten opgevuld worden. De rollen en hun functies zijn:

- Project Manager
  - Leidt het team
  - Contactpersoon van het team
  - Bemiddelaar tijdens meningsverschillen
  - SPMP
- Configuration Manager
  - Beheert de tools en libraries die gebruikt worden
  - Beheert de GitHub
  - SCMP
- Database Manager
  - Beheert de database
  - SDD
- Design Manager
  - Webdesign
  - Product design
  - SDD
- Test Manager
  - Zorgt dat het programma aan voldoende tests wordt onderworpen
  - STD
- Quality Assurance
  - Zorgt voor de kwaliteit van de code
  - SQAP

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

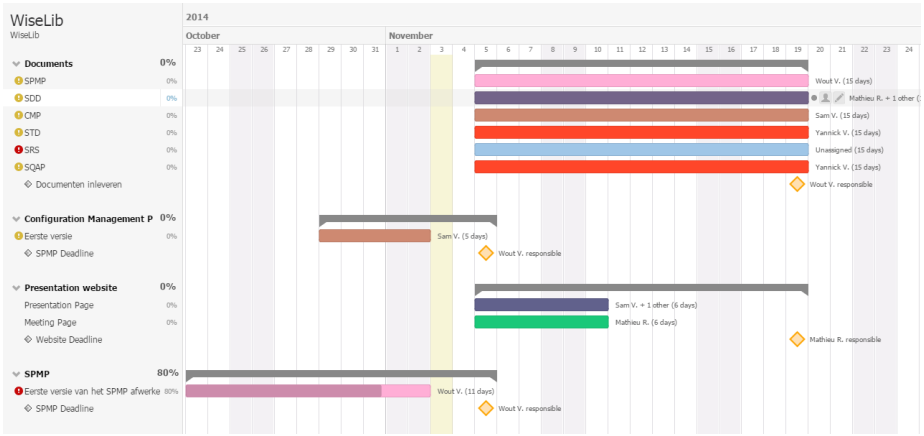
Rol	Verantwoordelijke	Back-up
Project Manager	Wout Van Riel	Mathieu Reymond
Test Manager	Yannick Verschueren	Arno Moonens
Design	Mathieu Reymond	Yannick Verschueren
Quality Assurance	Yannick Verschueren	Arno Moonens
Database Manager	Arno Moonens	Sam Vervaeck
Config Manager	Sam Vervaeck	Wout Van Riel

Table 4: Rolverdeling

Er wordt gecommuniceerd in de groep met behulp van slack [4] en mail.

2.3 Werkplanning

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



3 Management proces

3.1 Werkplanning

3.2 Controle plan

3.2.1 Requirements controle

Om een overzicht te houden op alle requirements zal er op de site [1] een requirements dashboard worden opgezet. Deze tabel zal de requirements bijhouden en hun status per iteratie weergeven. Voor elke requirement zullen de belangrijkste statistieken (prioriteit, moeilijkheidsgraad, ...) terug te vinden zijn.

3.2.2 Planning controle

Voor de planning maken we gebruik van Teamwork [3]. Het laat toe om milestones en deadlines vast te leggen voor het hele team. Bovendien is het ook mogelijk voor een enkel teamlid om zichzelf persoonlijke milestones en deadlines geven. Deze gegevens kunnen dan doorgestuurd worden met behulp van een iCal-feed naar de agenda van dat teamlid.

3.2.3 Raportering

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

- Alle documenten en source code (inclusief unit tests) worden per mail afgeleverd als een enkele zipfile, met als naam seN-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[1].
- Alle documenten en source code worden worden overeenkomstig getagd/gebranchd (seN-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.

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:

[1] Zie de kalender)

- 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)

3.3 Risico Management Plan

3.3.1 Persoonlijk risico's

Het persoonlijke risico zijn de goede en slechte eigenschappen van elk teamlid. Door deze te kennen, kunnen de taken beter verdeeld worden en kan iedereen op de juiste manier gestimuleerd worden. De analyse is terug te vinden in onderstaande tabel.

Arno Moonens
Positief
Heeft een basis in CSS, HTML en Javascript
Heeft Information Systems gevolgd
Negatief
Is stil

Mathieu Reymond
Positief
Heeft een matige kennis van CSS en HTML
Kan goed met deadlines werken
Heeft Machine Learning en Information Systems gevolgd
Perfectionistisch
Negatief
Heeft geen kennis van Javascript
Perfectionistisch

Sam Vervaeck
Positief
Heeft veel ideeën voor het project
Heeft een goede kennis van CSS, HTML en Javascript
Heeft al ervaring met servers

Heeft Machine Learning en Information Systems gevolgd
Perfectionistisch
Negatief
Kan zich verliezen in details
Is niet goed met deadlines
Perfectionistisch
Yannick Verschueren
Positief
Kent Javascript
Is goed met deadlines
Is goed met taal
Negatief
Kent geen HTML en CSS
Is laks
Wout Van Riel
Positief
Heeft een basis in HTML en CSS
Heeft managment ervaring
Graphic design
Negatief
Kent geen Javascript
Is laks

Table 5: Persoonlijke Risico Analyse

3.3.2 Project risico's

Aan elk risico wordt een waarde meegegeven dat de kans weergeeft dat het probleem zich gaat voordoen tijdens het project en er wordt ook een waarde meegegeven aan de impact die het probleem gaat hebben op het project. De waarden die deze parameters kunnen aannemen liggen tussen één en negen.

Risico	Kans	Impact	Oplossing
Meningsverschillen	9	4	Bemiddeling of stemming
Database model staat functies systeem niet toe	4	7	Database model veranderen
Bugs	9	5	Verantwoordelijke aanduiden

Table 6: Project Risico's

4 Technisch proces plan

4.1 Methodes, Tools en Technieken

Wij zijn verplicht om in dit project enkel en alleen JavaScript, HTML5 en CSS te gebruiken. Marco-talen, zoals onder meer SASS en CoffeeScript, mogen in dit project niet niet aanwezig zijn.

Buiten deze beperkingen is het team vrij om eenderwelke softwarebibliotheek of software-framework te gebruiken.

Alle documenten moeten geplaatst worden op GitHub [2] van WiseLib.

4.2 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

4.2.1 Documentatie van Broncode

5 Software Quality Assurance Plan

5.1 Doel

Dit document behandeld de processen en methodes die gebruikt worden in de quality assurance van het project "WiseLib". Dit document volgt de IEEE standaarden voor Quality Assurance Plannen. [2]

5.2 Referentiedocumenten

SCMP Software Configuration Mangment Plan

5.3 Management

5.3.1 Organisatie

De Quality Assurance wordt beheerd door Yannick Verschueren 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 verige teamleden.

5.3.2 Taken

Hoewel het Quality Assurance proces gedurende het hele project loopt, zal er na de voltooiing van een functionaliteit een periode 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 algemeene 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.

5.4 Standaarden, gebruiken, conventies en metrieken

5.4.1 Standaarden

Documentstandaarden
De documenten gebruiken de IEEE standaarden [3]

[2] `http://users.csc.calpoly.edu/~jdalbey/205/Resources/IEEE7301989.pdf`
[3] `http://www.ieee.org/publications_standards/publications_standards_index.html`

5.4.2 Gebruiken

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

5.5 Audits

Aangezien het hele team aan de implementering van het systeem is elke vergadering waarin geschreven code wordt besproken een zowel functionele, fysieke als in-process audit.

5.6 Probleem rappotrering

Het rapporteren van problemen in zowel software als software ontwikkelings processen worden op verschillende manieren gerapporteerd.

Slack het gebruikte communicatie platform bevat een "bugs" kanaal waarin problemen en bugs worden gepost.

GitHub heeft een ingebouwde issue-tracker.

Trello wordt gebruikt om grotere problemen te melden die de voortgang van de implementatie zouden kunnen hinderen.

5.7 Tools, technieken en methodologieën

Het QA proces gebruikt Mocha [4] als testing-framework. Dit framework zal in combinatie met Chai[5] 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*[6] beperkt.

[4] Mocha Homepage
[5] http://chaijs.com
[6] http://chaijs.com/guide/styles/#should

6 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.

Deze sectie legt de rol van de configuration manager verder vast, alsook enkele van de taken die de configuration manager dient te vervullen.

6.1 Verantwoordelijkheden

Het is de primaire taak van de configuration manager om alle ondersteunende tools en processen te overzien en te verbeteren.

6.1.1 Documentatie

De configuration manager is niet alleen verplicht om al deze processen en tools uit te werken en te verbeteren, maar ook gebruikersdocumentatie te voorzien. De gebruikersdocumentatie moet voor elk teamlid verstaanbaar zijn, maar moet ook niet meer uitgebreid zijn dan nodig is.

De primaire bron van informatie zal altijd terug te vinden zijn in dit document. Verdere informatie zal verder worden uitgelegd op GitHub: ofwel in de wiki, ofwel in één of meerdere README-bestanden in de repository zelf. De locatie van de documentatie hangt af van welke plaat het beste erbij past.

Andere manieren van uitleg geven, zoals mondeling overleg, zijn ook altijd mogelijk en worden ten zeerste aangeraden.

6.1.2 Softwarevereisten

De configuration manager moet trachten om geen beperking te leggen op de programmeeromgeving die door elk teamlid gebruikt wordt. Op die manier is ieder teamlid vrij om zijn eigen vertrouwde omgeving gebruiken, en kan iedereen zo optimaal mogelijk werken.

Het is wel mogeljk dat bepaalde tools onbeschikbaar zijn op bepaalde platformen, maar in dat geval is de configuration manager verplicht op een redelijk alternatief aan te bieden op datzelfde platform.

6.2 Vastgelegde activiteiten

Wij hebben in dit project gekozen voor een "Separation of Concerns"-aanpak. Dit geldt niet alleen voor de software zelf maar ook bij de tools die worden gebruikt tijdens de ontwikkeling. Als gevolg hiervan is er geen centraal platform, hoewel er wel enkele hulpmiddelen (voornamelijk Slack) instaan om informatie te aggregeren.

6.2.1 Communicatie

Alle informele communicatie verloopt via Slack [4]. De formele communicatie verloopt via Teamwork.com [3]. Op die website zullen ook alle deadlines en verslagen komen te staan, zoals al eerder in dit verslag werd besproken.

De issues zullen op GitHub [2] komen te staan. 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.

Het is de bedoeling om alle services zo goed mogelijk te integreren, zodat de teamleden niet onnodig moeten switchen tussen verschillende tools. Zo is ook de mailinglijst[7] geïntegreerd in Slack.

6.2.2 Source Control

Een van de belangrijkste ondersteunende middelen voor elk software project is revision control. Revision control maakt het mogelijk voor het team om bij te houden wie welke wijzigingen aan het project heeft gemaakt, op welk moment, en wat de staat van het project was voor de wijziging van kracht ging.

De tool (ook "Version Control System" genoemd) die hiervoor wordt gebruikt is door de opdrachtgever vastgelegd. Git[8] zal in combinatie met GitHub [2] gebruikt worden om alle code-reviews mee uit te voeren. Het wordt ook van elk teamlid verwacht dat vanaf de development-fase van start gaat alle issues in de tracker op GitHub zelf worden gezet.

De stuctuur en lay-out van de branches (het "Branching model") is voor een groot deel gebaseerd op Git-flow [**?**]. Dit source control management plan komt ook vaak terug in andere open-source projecten. Alle teamleden hebben er dus baat bij om vertrouwd te zijn met dit systeem, voor wanneer ze in contact komen met externe libraries of frameworks tijdens dit project.

Github-flow [**?**] is een versimpelde versie van dit model. Het leent veel van de concepten van Git-flow, met als voornaamste wijziging dat de *develop* en *master*-branch samengevoegd worden tot één branch. Door deze samenvoeging is er nagenoeg geen verschil tussen de productieversie en de developmentversie. Versies van het project die als stabiel worden aanzien krijgen een bijbehorende "tag" mee.

Om de kwaliteit van het ontwikkelproces verder te bevorderen zullen er nog enkele extra scripts[9] worden gebruikt die de ontwikkelaar zullen verbieden om wijzigingen te publiceren die niet doorheen alle tests geraken. Dit verplicht

[7] se2-1415@wilma.vub.ac.be
[8] http://git-scm.com
[9] `Git-hooks`http://git-scm.com/book/en/v2/Customizing-Git-Git-Hooks

iedereen nog meer om de tests goed te onderhouden, en zorgt ervoor dat het project zich nooit in een gebroken staat bevindt.

6.2.3 Continuous Integration

Onder continuous integration verstaan we de automatisatie van processen die tussen het aanbrengen van een wijziging aan de code en het *deployen* van de code naar de productieserver liggen.

Omdat sommige tools niet cross-platform werken — en dus bepaalde teamleden niet in staat zijn om sommige tools uit te voeren — zal een CI-server in werking genomen worden. Die server zal verantwoordelijk zijn om het project na te kijken op fouten, en kan eventueel het project verder uitrollen indien het werd goedgekeurd.

Een veelgebruikte service die een deel van deze processen kan automatiseren is Travis CI [**?**]. Deze service is vrij beschikbaar voor open-source projecten en integreert nauw samen met GitHub. Wanneer een bepaalde *commit* in GitHub getagd wordt, zal Travis CI alle tests uitvoeren op de laatste versie van het project.

6.2.4 Streamlinen van ontwikkelproces

Sommige tools moet de ontwikkelaar herhaaldelijk oproepen. Om de tijd die de ontwikkelaar hierin steekt kan worden verminderd door gebruik te maken van een "Build System". Dit is een volwaardig programma op zich dat instaat om de juiste tools op het juiste moment aan te roepen.

Er zijn twee Build Systems die naar voren komen in een vergelijking: Grunt[10] en Gulp[11]. Dit project gebruikt Gulp omwille van zijn versimpelde gebruikersinterface en de mogelijkheid om bepaalde taken asynchroon uit te voeren dankzij Orchestrator[12]. De snelle afhandeling van repetitieve taken kan de ontwikkelaars aanzienlijk wat tijd winnen, en zorgt ervoor dat ze zich nog meer op de code kunnen focussen.

6.2.5 Veranderlijkheid van afspraken

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 zijn motivatie voor de wijziging heeft uitgelegd.

[10] `http://gruntjs.com`
[11] http://gulpjs.com
[12] `http://orchestratorjs.org`

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