Software Test Plan

Yannick Verschueren

November 2014

Document geschiedenis

Versie	Datum	Auteur/co-auteur	Beschrijving
1	November 2014	Yannick Verschueren	Eerste versie

Inhoudstafel

1 Introductie

1.1 Domein

Het Software Test Plan behandelt alle methoden, gebruiken en standaarden die van toepassing zijn bij het testen van het "WiseLib" project. Het domein van de testen beschreven in dit plan beschouwt:

1. Het testen van de funtionele vereisten,beschreven in het SRS
2. Controleren van de gebruikers vereisten
3. Verzekering van code kwaliteit
4. Unit testen en verwijderen van bugs

1.2 Doel

Het doel van dit document is om een overzicht te geven van alle strategieën, procedures, activiteiten en taken die instaan voor het testen van het project. Het document volgt de IEEE standaard voor Software en Systeem Test Documentatie.[1]

1.3 Objectieven

1.3.1 Primair objectief

Software projecten bestaan uit verschillende onderdelen: software, hardware en interfaces. De testing procedures verder beschreven in dit document behandelen elk onderdeel met variërende mate.

Software De geïmplementeerde software is de basis van het project en wordt dus met bijhorende itensiteit getest. Hierdoor wordt een aanzienlijk deel van de tijd gespendeerd aan het testen van de code.

Hardware Het systeem draait op twee verschillende types hardware. Het client gedeelte werkt op het apparaat van de gebruiker en is dus buiten het bereik van de tester van het systeem. De server waarop het systeem draait is nogmaals niet in het bezit van het team en is dus geen verantwoordelijkheid voor de tester.

Interface De functionaliteit van de interface wordt door zowel test manager als webmaster getest.

De test procedures zorgen ervoor dat in het systeem:

1. De functionele verseisten vervuld zijn.
2. De gebruikers het systeem correct en zoals beoogd kunnen gebruiken. Dit houdt in dat de voorgeschreven use-cases op een correcte manier door de applicatie worden afgehandeld.

[1] `http://standards.ieee.org/findstds/standard/829-2008.html`

1.3.2 Secundair objectief

Het secundair objectief in het testen van een systeem is het opsporen en behandelen van alle problemen en risico's, deze problemen delen met het team achter het project en het oplossen van deze problemen op een correcte manier.
Hierbij ligt de nadruk op het efficiënt en correct communiceren van problemen tussen leden van het team. Dit document beschrijft alle tools en platformen die gebruikt worden om problemen te melden en op te lossen.

2 Referenties

- Software Configuration Managment Plan
- Software Design Document

3 Definities, acroniemen en afkortingen

3.1 Acroniemen

SPMP Software Project Management Plan

SRS System Requirement Specifiaction

STD Software Test Document

SDD Software Design Document

3.2 Definities

Integriteit niveau is een indicatie van de relatieve belangerijkheid van een software onderdeel tot de stakeholders of opdrachtgevers.

Unit test methode om softwaremodulen of stukken broncode (units) afzonderlijk te testen.

Integratie test fase waarin individuele software modules gecombineerd worden en getest worden als een groep.

Traceerbaarheidsmatrix deze matrix koppelt de test-cases aan de correcte use-cases

4 Integriteit niveau's

Elk onderdeel van het systeem heeft een bepaald integriteits niveau (zie definities in sectie 3). In geval van het falen van dit onderdeel zegt de integriteit in hoeverre dit het gehele systeem zal beïnvloeden en hoe belangerijk het oplossen van het probleem is. Hoe hoger het niveau, hoe meer tests zullen worden uitgevoerd.
Er bestaan drie niveaus

1. De functionele vereisten.(Hoge prioriteit)

2. De gebruikers vereisten of use-cases. (Gemiddelde prioriteit)

3. Aanvullende functionaliteiten. (Lage prioriteit)

De functionele vereisten hebben de hoogste graad van belangerijkheid aangezien zij opgelegd zijn door de opdrachtgevers. Deze functionaliteiten moeten aanwezig zijn in de applicatie en moeten feilloos werken. De gebruikers vereisten zijn de vereisten opgelegd door de ontwikkelaars en zijn afgeleid uit de functionele vereisten. Deze vereisten hebben een lager niveau aangezien ze niet noodzakelijk zijn voor de opdrachtgevers. De aanvullende functionaliteiten hebben de laagste graad aangezien zij weinig of geen belang hebben in het correct functioneren van de applicatie.

5 Test processen

Het Mocha [2] framework wordt gebruikt in combinatie met Should.js [3] als algemeen testplatform.

De intensiteitsgraad en strengheid in het uitvoeren en documenteren van test procedures is evenredig met het intergriteitsgraad. Hoe lager de graad, hoe lager de intensiteit en strengheid van de test procedures.

De test methodologie wordt besproken in het Quality Assurance onderdeel van het SPMP.

5.1 Unit testing, integratie testing en validatie testen

5.1.1 White-box tests

Zowel de unit tests als de integratie tests zijn white-box tests, zij worden geïmplementeerd met kennis van het gehele systeem. Zij testen de interne structuren en werking van het programma.

De Should.js bibilitotheek wordt gebruikt bij de unit en intergratie tests van de geschreven code.
Het Mocha framework zorgt ervoor dat alle tests op een correcte manier worden uitgevoerd. Een simpel commando voert alle test uit die zich in de "tests" map bevinden. Deze map heeft een indeling gelijkaardig aan de structuur van de applicatie code.

5.1.2 Black-box tests

Onder de black-box tests vallen alle test procedures die geen kennis hebben van de interne werking van het systeem. Hun hoofddoel is het testen van de functionaliteiten. Hieronder vallen:

Component interface tests testen de handeling van data tussen verschillende modules van de applicatie

[2] `http://mochajs.org/`
[3] `https://github.com/tj/should.js`

Systeem tests verifiëren dat de applicatie voldoet aan de vereisten

Acceptatie tests worden uitgevoerd door de cliënt

Deze tests maken weinig of geen gebruik van test platformen, maar worden getest door actief gebruik van (een deel van) de applicatie.

5.2 Test procedures

Unit tests vormen de basis van het gehele test proces en zullen dus blijvend worden uitgevoerd gedurende de ontwikkeling van het systeem. De eerste unit tests worden geschreven voordat het design, beschreven in het SDD, wordt geimplementeerd. Elke klasse beschreven in het SDD zal voor de beduidende methodes een test functie bevatten die de correcte werking van de methode zal verzekeren.

Het schrijven van de tests gebeurd met de Should syntax en wordt geplaatst in de test directory. De tester kan de test manueel oproepen of kan gebruik maken van het Mocha framework om alle testen met een commando uit te voeren. Een derde methode van testen is wanneer code naar GitHub (zie SCMP) wordt geduwd. Hierbij wordt de correcte functie opgeroepen en zal afhankelijk van het resultaat van de test de code op GitHub staan.

5.3 Test Cases en traceerbaarheidsmatrix

Dit deel van het test document beschrijft de test cases behorend bij de use case beschreven in deel vier van het SDD. De test cases gebruiken dezelfde naam als de bijhorende use case.

5.3.1 Test Cases

Definitie : Het systeem : het testing framework

Naam	Account aanmaken.
Samenvatting	De gebruiker maakt een nieuw account aan.
Tests	

Test 1 Het systeem gebruikt een willekeurig e-mailadres uit de database om een nieuw account aan te maken. De test moet terug geven dat het account niet is aangemaakt.

Test 2 Het systeem maakt een account met een uniek email adres maar met naam en voornaam aanwezig in de database. De test moet een lijst terug geven met correcte naam/voornaam combinatie.

Test 3 Het systeem maakt een compleet uniek account aan. De database wordt gecontroleerd om de nieuwe gebruiker, alle informatie moet overeen stemmen met de ingevulde gegevens.

Naam	Inloggen.
Samenvatting	Een gebruiker logt zich in.
Tests	

Test 1 Het systeem gebruikt een verkeerd e-mailadres om zich aan te melden. De test moet terug geven dat inloggen niet mogelijk is.

Test 2 Het systeem gebruikt een correct e-mailadres en een fout wachtwoord. De test moet terug geven dat inloggen niet mogelijk is.

Test 3 Het systeem logt in met correcte gegevens uit de database. Een correct RegisterdUser (zie SDD) object wordt aangemaakt.

Naam	Publicatie uploaden.
Samenvatting	Een gebruiker zet zijn publicatie online.
Tests	

Test 1 Het systeem probeert zowel een pdf als een bibtex bestand te uploaden, van het bestand zijn enkele publicatie velden gekend. De test kijkt of de velden correct worden aangevuld met de informatie uit de bestanden.

Test 2 Na een ingevuld publicatie formulier, eenderzijds door het systeem via analyse van een bestand anderzijds manueel door het systeem (aparte test) kijkt de test of de publicatie aanwezig is in de database.

Naam	Publicatie opzoeken.
Samenvatting	De gebruiker zoekt publicaties op volgens bepaalde criteria.
Tests	

Test 1 Fase 1: Het systeem zoekt publicaties met enkele willekeurige criteria. De test controleerd of er publicaties worden terruggeven.
Fase 2: Indien er publicaties werden teruggegeven worden deze gecontroleerd of ze bij de ingegeven criteria horen.

Naam	Publicatie toevoegen aan gebruiker zijn bibliotheek.
Samenvatting	De gebruiker moet een publicatie die hij gevonden heeft met het systeem kunnen toevoegen aan zijn eigen bibliotheek.

Tests

Test 1 Het systeem meldt zich aan met correcte gegevens van het test account. Een willekeurige publicatie wordt opgezocht en toegevoegt. Zowel de database als het account worden gecontroleerd of de nieuwe publicatie is toegevoegd.

5.3.2 Traceerbaarheidsmatrix

De matrix zal uitgewerkt worden in verdere iteraties van het document. Er bestaan nog niet voldoende test cases om de matrix op te stellen en de tests aan de functionaliteiten te koppelen.