Test Approach

Get Connected 2012/2013 - QBA

Auteurs: David de Hoop
Daniël Koek
Olger Alphenaar
Martijn van Delden
Graham Tjin Liep Shie
Groep nr.: 1

Versie: v1.0
Status: final
Datum: 29 november 2012

Hogeschool van Amsterdam

Versiebeheer

Ver.	Status	Datum	Auteur(s)	Wijzigingen
0.1	Concept	24-11	David de Hoop	
0.2	Concept	25-11	David de Hoop, Daniël Koek	Toevoegingen over documentatie en beveiliging.
0.3	Concept	26-11	Òlger Alphenaar	Toevoeging H5: Proceskwaliteit.
0.4	Final	28-11	David de Hoop, Martijn van Delden	Aangepast, H5 verwijderd, concretiseren.
1.0	Final	29-11	Martijn van Delden	Grammaticale correctie.

Goedkeuring

	Uitvoering		Controle		Goedkeuring	
Ver.	**Naam**	**Datum**	**Naam**	**Datum**	**Naam**	**Datum**

1. Samenvatting

Dit rapport beschrijft de wijze waarop ons softwaresysteem voor dit project getest zal gaan worden en wat de resultaten hiervan zijn. De uitkomst van dit rapport vormt de basis voor de kwaliteitscontrole die in de bèta fase zal worden uitgevoerd, voorafgaande aan de oplevering van het softwaresysteem.

In dit rapport zullen de dynamische softwarekwaliteit alsmede de statische softwarekwaliteit worden getest. Verder zal er zeer diepgaand gekeken worden naar de beveiliging van de software. Er zal met verschillende scenario's uitvoerig worden getest of het softwaresysteem in de meeste browsers veilig is. Er zullen ook verschillende aanvallen worden uitgevoerd op het systeem om te kunnen testen of er nog beveiligingslekken in het systeem zitten.

Pas als uit de kwaliteitscontrole blijkt dat het systeem veilig is en naar behoren werkt zal deze worden opgeleverd. U als opdrachtgever kan dus aan de hand van dit rapport en een positieve uitkomst daarvan zien dat uw systeem uitvoerig is getest en gecontroleerd. Dit wil echter niet zeggen dat het systeem ook honderd procent fout vrij is. Het is namelijk onmogelijk een onfeilbaar systeem te ontwikkelen met de tijd en middelen die wij voorhande hebben.

Wij zullen echter wel zorgen dat de belangsrijkste en meest vitale componenten van het systeem zo goed mogelijk werken en dat de privacy van de gebruikers gewaarborgd is. De focus wat betreft de beveiliging ligt dan ook op het zo goed mogelijk versleutelen van privacy gevoelige gegevens van gebruikers, en het voorkomen van aanvallen die het systeem van binnen uit aantasten.

2. Inhoudsopgave

3. Statische Softwarekwaliteit

De statische kwaliteit van een softwaresysteem bestaat uit de opbouw van de code. Is deze volgens de standaard? Worden de syntax regels gehandhaafd? Verder kijken wij bij statische softwarekwalitiet ook naar de hoeveelheid code, de overzichtelijkheid van de code en de hoeveelheid en kwaliteit van het commentaar.

3.1 Aanpak

Om de statische kwaliteit van het op te leveren softwaresysteem te meten, stellen wij als eerste een rapport met feiten op. Dit doen wij met behulp van PHP Code Sniffer i.c.m. PHP Depend, die werken met het PHP Pear framework. Met Code Sniffer is het mogelijk om syntax fouten op te sporen en om te kijken of de code aan een bepaalde PHP standaard voldoet. Met PHP Depend kunnen we een rapport opstellen met cijfers over het aantal regels code, commentaar en functies. Dit is vergelijkbaar met SourceMonitor voor Java. De uitkomst van deze twee rapporten is leidend voor de verdere aanpak van de statische softwarekwaliteit. Verder zullen wij JSLint gebruiken om de stukken JavaScript die wij gebruiken te standaardiseren.

Aan de hand van dit rapport zal de code volgens een bepaald patroon worden opgebouwd (code beautifying). Dit verhoogt de leesbaarheid van de code. Verder zal er commentaar worden toegevoegd indien uit het rapport is gebleken dat het percentage te laag is en zal er per file een changelog worden bijgehouden, in de file zelf, om een zo goed mogelijk overzicht van het versiebeheer te geven. Nadat alle code opnieuw is opgebouwd en eventuele overbodige code is verwijderd zal er een syntaxcontrole plaatsvinden. Deze gaat na of er per programmeertaal aan alle statische regels en eisen word voldaan, die voor de desbetreffende programmeertaal gelden. Uiteindelijk zullen dus bovengenoemde tools weer opnieuw gebruikt worden om te kijken of de code aan alle bovenstaande eisen voldoet.

Als alle code goed is opgebouwd, de verschillende soorten code (talen) zoveel mogelijk gescheiden zijn, de hoeveelheid code tot een minimum is beperkt en de gehele syntax klopt, zal de statische softwarekwaliteit als voldoende worden beoordeeld.

3.1.1 Documentatie

Statische kwaliteitseisen stellen we ook aan in onze documentatie. Alle documentatie moet volgens ITopia standaarden zijn geschreven en bewaard in .pdf formaat. Alle documentatie moet verder zijn voorzien van een adequaat bijgehouden versiebeheer en in correct Nederlands en/of Engels zijn geschreven. Ook moeten alle teamleden hun persoonlijke logboek goed bijhouden en deze minimaal twee keer per week updaten. Dit wordt allemaal nagekeken en als ieder teamlid aan de eisen voldoet zal de statische kwaliteit van de documentatie als voldoende worden beoordeeld.

4. Dynamische Softwarekwaliteit

De dynamische softwarekwaliteit van ons systeem bestaat voor het grootste gedeelte uit een juiste werking van de functionele requirements. Alle onderdelen en functies van het systeem moeten werken zonder dat er tijdens de uitvoering fouten optreden. Verder mag het ook niet voorkomen dat functies vatbaar zijn voor incidentele fouten. Onder dynamische softwarekwaliteit valt ook de beveiliging van het softwaresysteem samen met de gebruiksvriendelijkheid van het systeem.

4.1 Aanpak

Eén van de belangrijkste onderdelen van de dynamische softwarekwaliteit is de werking van alle functies. Alle use cases zullen worden getest d.m.v. unit testing. Hierbij worden alle use cases afzonderlijk getest. Unit testen is heel intensief omdat ook hiervoor weer scripts geschreven moeten worden die de unit tests automatiseren. Wij gebruiken PHP Unit, tevens onderdeel van het PHP Pear framework, om een groot gedeelte van het unit testen te automatiseren. PHP Unit bevat een heleboel functies om form inputs, database scripts en dergelijke te testen zonder dat wij deze zelf hoeven te schrijven.

Naast de unit tests door middel van scripts zal Graham binnen ons team alle use cases met de hand aflopen naar aanleiding van de user manual om te kijken of alle use cases ook echt werken zoals beschreven staat en om te kijken hoe het systeem omgaat met menselijke interactie, en dus ook menselijke fouten, die met geautomatiseerde unit testing niet zijn af te vangen. Deze menselijke tests van de use cases vormen ook de basis voor de gebruiksvriendelijkheidtests.

Om de gebruiksvriendelijkheid van het softwaresysteem verder te testen en tevens incidentele fouten en bugs op te sporen doen wij een open Alfa test waarbij willekeurige personen van uiteenlopende leeftijden en geslacht binnen onze doelgroep het volledige systeem kunnen gebruiken. Wij zullen de gebruiksvriendelijkheid dus toespitsen op de leeftijdscategorieën binnen onze doelgroep. Als deze personen het systeem getest hebben vragen wij om een feedbackformulier in te vullen waar zij op- of aanmerkingen over verschillende aspecten van het systeem kunnen aangeven. Denk hierbij aan snelheid, aantal benodigde handelingen, duidelijkheid van de stappen, etc. Aan de hand van deze feedback kunnen wij indien nodig nog tweaks of opmaak technische aanpassingen doorvoeren voor de oplevering. Deze tests zullen beginnen tijdens de Afla oplevering waar er tijd is voor de verschillende mensen van de OBA en gasten van ITopia om het systeem op zowel pc als tablet (en smart Phone) te gebruiken en een feedbackformulier in te vullen. We zullen hier in de eerste week van de beta fase mee verder gaan om extra data te verzamelen.

4.1.1 Beveiliging

Een belangrijk aspect van ons softwaresysteem is de beveiliging, en dan specifiek de beveiliging van privacy gevoelige gegevens, binnen ons systeem. Om de beveiliging te testen zullen wij zelf een aantal van de meest gebruikte cyberaanvallen en exploits gebruiken en uitvoeren op ons systeem en aan de hand van die resultaten de beveiliging van het systeem aanpassen. Deze tests vallen ook onder de unit tests en zijn er puur op gericht het systeem zoveel mogelijk te “slopen” om zo te kunnen bekijken hoe en wanneer er de meeste schade aan het systeem wordt aangebracht en zo dit te kunnen afvangen voor de oplevering van het beta product.