Project Hathor

Project code: **13**

Date completed: **11**/6/2013
Auteur: Nick Jole

Version: **v** 1.0
Status: **Final**

Document ID: **5**

Afstudeerverslag
Fontys Hogeschool ICT

Gegevens student:

Naam:	N. Jole
Studentnummer:	2148551
Afstudeerrichting:	HBO ICT Software Engineering (Voltijd)
Afstudeerperiode:	11 / 2 / 2013 t/m xx / 06 / 2013 (85 Dagen)

Gegevens bedrijf:

Naam bedrijf/instelling	RES Software
Afdeling	Research and Development
Plaats	‘s-Hertogenbosch

Gegevens bedrijfsbegeleider:

Naam	N. Tanis
Functie bedrijfsbegeleider	Senior Software Developer

Gegevens docentbegeleider:

Naam:	R. van der Heijden

Gegevens verslag:

Titel afstudeerverslag:	Project Hathor
Datum uitgifte afstudeerverslag:	11 / 06 / 2013

Getekend voor gezien door bedrijfsbegeleider:

Datum:

De bedrijfsbegeleider,

Document History

Revisions

Version	Status	Date	Changes
0.3	Concept	15-4-2013	Eerste versie
0.4	Concept	26-4-2013	Technische sprint toegevoegd
0.5	Concept	3-5-2013	Totaal bijgewerkt en Literatuur lijst bijgewerkt.
0.6	Concept	21-5-2013	Technische sprint toegevoegd.
0.7	Concept	23-5-2013	Totale verandering scriptie aan de hand van feedback.
0.8	Concept	25-5-2013	Alle onderdelen ingevuld.
0.9	Concept	31-5-2013	Grammatica gecontroleerd en uitleg uitgebreid waar nodig.
1.0	Final	11/6/2013	Finalized

Approval

This documents needs to be approved by the following persons:

Version	Date of approval	Name	Function	paraph
1.0		Ralph van Roosmalen	Supervisor	
1.0		René van der Heijden	Project Assurance	

Distribution

This document has been distributed to:

Version	Send Date	Name	Function
0.4	3-5-2013	Ralph van Roosmalen	Supervisor
0.5	6-5-2013	René van der Heijden	Project Assurance
0.9	31-5-2013	Ralph van Roosmalen	Supervisor
0.9	31-5-2013	René van der Heijden	Project Assurance
1.0	11/6/2013	Ephorus	Plagiaat tool

Voorwoord

Deze scriptie is geschreven als afronding voor de bachelor studie ICT Software Engineering aan Fontys Hogescholen te Eindhoven. Tijdens de periode dat ik bij RES Software heb afgestudeerd, heb ik veel geleerd over .NET en ASP.NET.

Op het moment dat ik de opdracht van RES Software tegen kwam in het stage systeem van Fontys leek het een grote opdracht omdat ik ASP.NET moest gaan gebruiken. Door de hulp van mijn begeleider viel het leren van ASP.NET heel erg mee, en kon ik er na een paar weken goed mee omgaan.

Daarom wil ik in eerste instantie Niels Tanis, mijn technische begeleider vanuit RES Software bedanken voor zijn hulp en advies op de momenten dat ik vast zat. Maar ook voor zijn kritische blik op de inhoud van mijn opdracht en zijn begeleiding op het gebied van ASP.NET.

Vanuit school wil ik René van der Heijden bedanken voor het beantwoorden van mijn vragen over het proces, en het nakijken en feedback geven op mijn scriptie.

Daarnaast wil ik ook Ralph van Roosmalen bedanken voor het mogelijk maken van deze stage, en wil ik Bart Withaar bedanken voor zijn informatie over de interpretatie van de database.

Ten slotte wil ik iedereen bedanken die mijn scriptie heeft nagelezen en de taal heeft nagekeken. Met speciaal dank aan Inge Luchtenberg voor haar begeleiding in en feedback op de schrijfstijl van mijn scriptie.

Zonder deze mensen zou het voltooien van deze scriptie nooit zijn gelukt.

Bedankt voor jullie inzet.

Nick Jole

Management samenvatting

Doel van dit document

Het doel van dit document is om de procesgang van het project vast te leggen. Dit document is de leidraad om de begeleiders en assessors inzicht te geven in de werken die de afstudeerder verricht op het bedrijf.
De twee belangrijkste redenen voor dit document zijn:

- Om er voor te zorgen dat het project voldoende is uitgeschreven voor de jury.
- Om inzicht te geven in de werken die de afstudeerder verricht op het bedrijf, zodat deze door de jury in voldoende mate kunnen worden beoordeeld.

Motivatie opdracht

RES Software ontwikkelt software volgens de agile principes. Een van de pilaren van agile ontwikkeling is automatisering van testen. Om deze automatische testen eenvoudig te kunnen maken heeft RES Software een Automated Test Platform (ATP) ontwikkeld. Met ATP kunnen de ontwikkelaars en testers snel en effectief testen bouwen, die dan op elke configuratie van virtuele machines kunnen worden gedraaid.
De volgende stap in het automatiseren van testen is de automatische rapportage van de test runs. Toen het afstudeerproject begon, moest het hoofd van de testers dagelijks een half uur met de hand de resultaten bij elkaar zoeken. Deze resultaten werden dan verstuurd via email naar de persoon die de test had gestart. Het was arbeidsintensief om de email dagelijks op te stellen waardoor afdelingen liever op oude manieren, die wel directe feedback gaven, terug vielen. Dit project is een oplossing voor hun probleem. De resultaten zijn direct zichtbaar op een dashboard, die door het gehele bedrijf hangen, en daarnaast ook via een website inzichtelijk zijn.
Een bijkomstigheid is dat een website zichtbaar is op andere locaties, zoals Roemenië en de USA. Hierdoor zijn de beperkingen van het huidige dashboard, dat alleen zichtbaar is binnen het bedrijf in Nederland, verleden tijd.

Globale aanpak opdracht

De globale aanpak van de opdracht is:

- Vind alle ASP.NET Stacks;
- Onderzoek welke ASP.NET Stacks het beste werken;
- Design de software met de gekozen ASP.NET Stack;
- Maak een Proof of Concept van de design;
- Realiseer de design;
- Lever de software op aan de organisatie.

Inhoud

Samenvatting

Het afstuderen heeft plaatsgevonden binnen het bedrijf RES Software. RES Software is een ICT bedrijf dat met haar software IT organisaties meer productief maakt door het afleveren van IT diensten. Omdat RES Software andere bedrijven productiever wil maken, kijkt RES Software zelf ook naar zijn eigen productiviteit.

Om productief te zijn heeft RES Software het Automated Test Platform (ATP) ontwikkeld om haar ontwikkellaars snel efficiënte testen te laten bouwen en uitvoeren. Het ATP is op zich heel productief, alleen is de rapportage die uit het ATP Center komt alles behalve gemakkelijk te interpreteren. Het hoofd van de testers moet dagelijks met de hand alle nieuwe testen bij elkaar zoeken en deze tests worden dan gemaild naar de testers die de test hebben gestart. Om dit probleem op te lossen is er gekozen voor een dashboard dat de informatie van het ATP op elk tijdstip inzichtelijk maakt.

Deze scriptie beschrijft de weg naar een goed functionerend dashboard. Om het dashboard te ontwikkelen is er gebruik gemaakt van de Scrum-methodiek, waarbij dagelijks een stand-up meeting was en wekelijks een demonstratie van het product.

Voordat er begonnen werd aan het ontwikkelen van het dashboard is er onderzoek gedaan naar de manieren waarop het dashboard ontwikkeld kon worden binnen ASP.NET. Uit dit onderzoek is gekomen dat de beste manier voor RES Software ASP.NET MVC 4 was met D3.JS voor de grafieken. Daarbij worden technieken als Responsive Web Design toegepast zodat de website zich kan aanpassen aan de afmetingen van het scherm.

Het project verliep niet zo soepel als gehoopt. Dit kwam doordat er foute data in de ATP database aanwezig was waardoor er heel veel vragen waren over de inhoud van de database. Daarbij waren de prestaties van de database traag, waardoor eerst de performance van de database verbeterd moest worden voordat er verder kon worden ontwikkeld. Nadat de performance acceptabel was ging het project voorspoedig en zonder al te grote tegenslagen.

Tot slot is een van de aanbevelingen om binnen RES Software kritisch naar alle database-gerelateerde bewerkingen te kijken. Omdat dit de stagiair veel tijd heeft gekost en deze problemen van te voren hadden kunnen worden opgelost. Ook is ASP.NET MVC 4 de toekomst op het gebied van web ontwikkeling binnen .NET.

Summary

The graduation took place within the company RES Software. RES Software is an ICT company that makes software to enable IT organizations to be more productive through the delivery of IT services and through workspace management. Because RES Software wants to increase the productivity of other companies, it also looks at its own productivity.

To be productive, RES Software has developed the Automated Test Platform (ATP), to quickly build and run effective tests. Although the ATP in itself is very productive, the reporting that is delivered by the ATP Center is anything but easy to interpret. The head of the testers must gather all new tests by hand daily and then mail the gathered information to the testers that have started the test. To solve this problem, RES Software decided to develop a dashboard that makes the information of the ATP available at any given time.

This paper describes the road to a well-functioning dashboard. The method that is used to develop the dashboard is Scrum. There was a daily stand-up meeting and there was a weekly demo of the product.

Before development on the dashboard started, research was made on the ways in which the dashboard could be developed in ASP.NET. The outcome of the research was that the best way to develop the dashboard would be ASP.NET MVC 4 and D3.JS for the graphs. Techniques as Responsive Web Design are applied so that the website can adapt to the size of the screen.

The project did not go as smoothly as hoped. This was because the ATP database contained erroneous data, so there were a lot of questions about how to interpret the contents of the database. In addition, the performance of the database was very sluggish, which first had to be improved before further development could be done. After the performance was acceptable, the project went smoothly and without major setbacks.

In conclusion, one of the recommendations for RES Software is to look critically at all database-related operations, because this has cost the student a lot of time which could have been solved in advance. Also ASP.NET MVC 4 is the future in the field of web development in. NET.

Verklarende Woordenlijst

Definitie	Uitleg
ASP.NET	Een manier om schaalbare websites te ontwikkelen.
Asynchroon	Communicatiemethode, waarbij er tegelijk gegevens worden opgevraagd.
Automated Test Platform	Een test platform waar de ontwikkelaars automatische testen op kunnen maken en deze dan dagelijks kunnen laten uitvoeren.
Database view	Een opgeslagen SQL query.
Dependency injection	Het inladen van een klasse die niet expliciet staat vermeld in de code. Waardoor de implementatie van een klasse kan veranderen zonder de bron code aan te passen.
Exception throws	Het opwerpen van een fout waardoor het programma tot een halt komt.
File Watch	Houdt in de gaten wanneer een bestand of directory is gewijzigd.
Framework	Een raamwerk dat veel, vaak herhalende taken, wegneemt van de ontwikkelaar en deze makkelijker bereikbaar maakt voor de ontwikkelaar.
Interface	Een overzicht van methoden en eigenschappen die in een klasse staan.
Mockups	Een schets van hoe de GUI er uit komt te zien als het klaar is.
NuGet	Een package management tool om zonder veel zoekwerk afhankelijkheden toe te voegen aan een project.
Out parameters	Een manier om meerdere return waarden terug te geven uit een methode.
Querystring	Het deel van de URL achter het vraagteken waar data aan wordt meegegeven voor de web applicatie.
Responsive Web Design	Een aanpak voor de opbouw van een website waardoor deze altijd probeert een zo optimaal mogelijke kijkervaring te geven.
RESTFul	Een manier om URL's op te bouwen zodat deze makkelijk schaalbaar zijn.
Schaduwkopie	Een back-up maken van een live omgeving zodat er geen data verloren gaat als er tijdens de ontwikkeling iets fout gaat.
SQL Query	Een commando dat door een database kan worden uitgevoerd om gegevens op te vragen of aan te passen.
Uniform Resource Locator (URL)	Een web adres dat een referentie is naar een resource ergens op een server.
User Requirements Specification (URS).	Een document waar de eisen in staan die door de opdrachtgever zijn opgegeven.
Windows Forms	De user interface die bij .NET wordt geleverd.
Extensible Markup Language (XML)	Een manier om data bij te houden en te versturen.

1 Inleiding

RES Software legt veel waarde in het goed testen van een project voordat het wordt vrijgegeven. Om het testen sneller en effectiever te laten verlopen is er binnen RES Software een Automated Test Platform (ATP) ontwikkeld. Door het ATP kunnen ontwikkelaars en testers snel en effectief testen bouwen en laten uitvoeren op elke willekeurige configuratie van Virtuele Machines. Echter is de rapportage van de test resultaten binnen het ATP center (De GUI van het ATP) nogal beperkt. De resultaten moeten nog met de hand bij elkaar worden gezocht in de log files. Hier wilde RES Software graag een oplossing voor.

Dit project is opgezet om een dashboard te ontwikkelen voor het ATP. Zodat de test resultaten van het ATP beter inzichtelijk worden en deze ook door management kunnen worden gezien. Ook wil RES Software kunnen vaststellen of er een stijgende of dalende trend is in het slagen van testen.

Deze scriptie geeft de procesgang weer hoe Project Hathor door de afstudeerder is ontwikkeld. De scriptie bevat de volgende hoofdstukken:

Hoofdstuk 2: Algemene informatie over het bedrijf en organisatiestructuur.
Hoofdstuk 3: Een gedetailleerde beschrijving van de afstudeeropdracht.
Hoofdstuk 4: De aanpak en de planning van de afstudeeropdracht.
Hoofdstuk 5, 7, 8: De 3 fasen van TSP met de daarbij horende uitleg.
Hoofdstuk 6: Het onderzoek van de ASP.NET Stacks.
Hoofdstuk 9: De conclusie van het gehele project en welke aanbevelingen er zijn.
Hoofdstuk 10: Het proces en de bevindingen tijdens de looptijd van het project.

In de bijlagen is het onderzoek en het Project Initiation Document opgenomen.

2 Bedrijfsbeschrijving

2.1 Algemeen

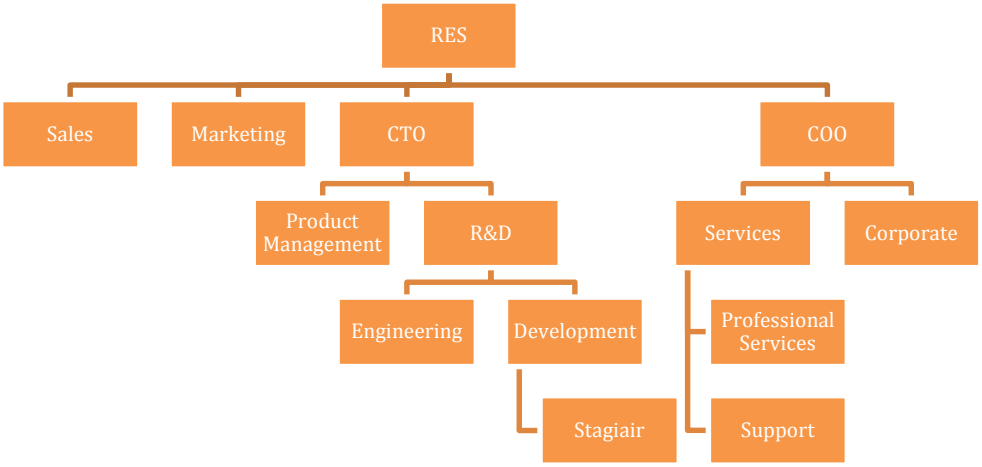
RES Software is opgericht in 1999 in 's-Hertogenbosch en ontwikkelt software aan de hand van agile principes voor organisaties die IT diensten leveren. Het doel van RES Software is het efficiënter maken van organisaties door snelle en makkelijke levering van hun IT diensten. Hierdoor kan het personeel van de organisatie efficiënter en sneller hun werk doen.

Sinds de oprichting van RES Software is het bedrijf doorgegroeid tot een internationaal bedrijf dat werk biedt aan 180 werknemers. Het hoofdkwartier is gevestigd in 's-Hertogenbosch (NL) en Philadelphia (US). RES Software heeft kantoren in UK, Norway en Frankrijk. De ontwikkeling van software gebeurt in verschillende onderzoeks- en ontwikkelcentra in Boston (US), 's-Hertogenbosch (NL) en Bucharest (RO).

De organisaties die de producten van RES Software gebruiken zijn organisaties die een groot IT netwerk hebben waar veel applicaties op worden gebruikt. Dit zijn bedrijven zoals IBM, Philips, Connexxion, Leids Universitair Medisch Centrum en CANADIAN IMPERIAL BANK OF COMMERCE.

RES Software werkt samen met bedrijven als Citrix en VMWare om de producten van RES Software zo goed mogelijk te integreren met de bestaande software architectuur.

2.2 Organisatiestructuur



Figuur 1: Organisatiestructuur

Binnen RESSoftware is de CEO (Chief Executive Officer) verantwoordelijk voor 4 afdelingen, Sales, marketing, CTO en COO.

De namen van de afdelingen sales en marketing spreken voor zich. Binnen deze afdelingen vindt de verkoop en promotie van producten plaats.

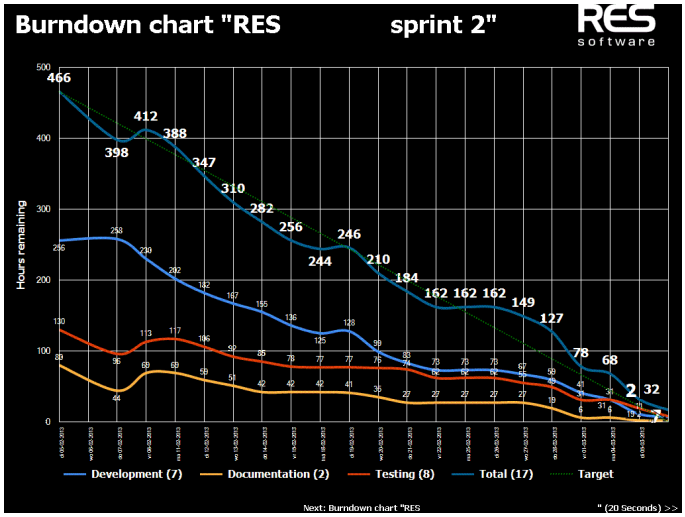
CTO (Chief Technology Officer) is de afdeling die er op toeziet dat er voldoende voortgang in de producten zit en problemen in de producten oplost. De afstudeeropdracht is uitgevoerd binnen Development, dat deel uitmaakt van Research & Development (R&D). R&D is verantwoordelijk voor het onderhouden (Engineering) en het doorontwikkelen van een applicatie (Development).R&D valt, samen met Product Management, onder de CTO van het bedrijf.

De afdeling COO (Chief Operating Officer) verleent service en support aan klanten. Service in de vorm van online learning, workshops, trainingen en certificeren. Support wordt online, zoals door middel van een forum, en telefonisch aan klanten gegeven.

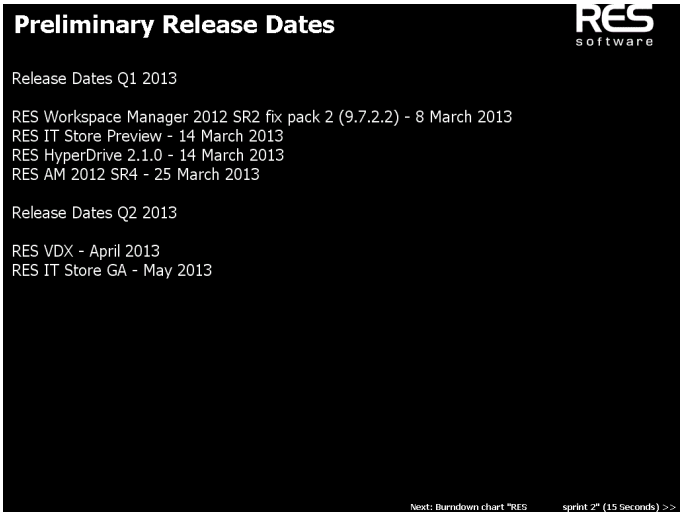
3 De opdracht

3.1 Motivatie opdracht

Aan het begin van het afstudeerproject was er al een dashboard aanwezig, zie **Figuur 2** en **Figuur 3**, op de schermen die op verschillende plaatsen in het gebouw hangen. Het dashboard laat de Scrum planning en de release dates van de producten van RES Software zien. Echter was dit dashboard aan het verouderen en had het dashboard beperkingen. RES Software wilde dit dashboard graag vervangen door een nieuw dashboard die de beperkingen niet meer heeft. Dit dashboard moest online via een website kunnen worden opgeroepen zodat het de beperkingen van een desktop applicatie, zoals het alleen zichtbaar zijn binnen de vestiging in 's-Hertogenbosch, tot het verleden toebehoren.



Figuur 2: Burndown Chart



Figuur 3: Release Dates

Op het nieuwe dashboard wil RES Software ook nieuwe data laten zien. De nieuwe data die zichtbaar moet worden gemaakt komt vanuit het Automatische Test Platform. Deze data moet weergeven hoe goed projecten lopen en waar nog fouten zitten in de projecten die binnen het ATP draaien. Aan het begin van het afstudeer project moest het hoofd van de testers dagelijks een half uur met de hand de resultaten bij elkaar verzamelen. Deze resultaten werden dan verstuurd via email naar de persoon die de test had gestart. Deze manier van werken was daarom ook heel arbeidsintensief waardoor afdelingen liever op oudere test manieren terug vielen die wel direct de informatie gaven.

Door middel van de afstudeer opdracht wil RES Software meer accelereren op het gebied van Web Development. RES Software wil graag in de toekomst investeren vanwege de verschuiving van desktop applicaties naar web applicaties. Met de afstudeeropdracht wil RES Software een goed basis project hebben voor ontwikkeling in ASP.NET waardoor RES Software de volgende projecten in ASP.NET kan bouwen in plaats van Windows Forms.

3.2 Doelstelling

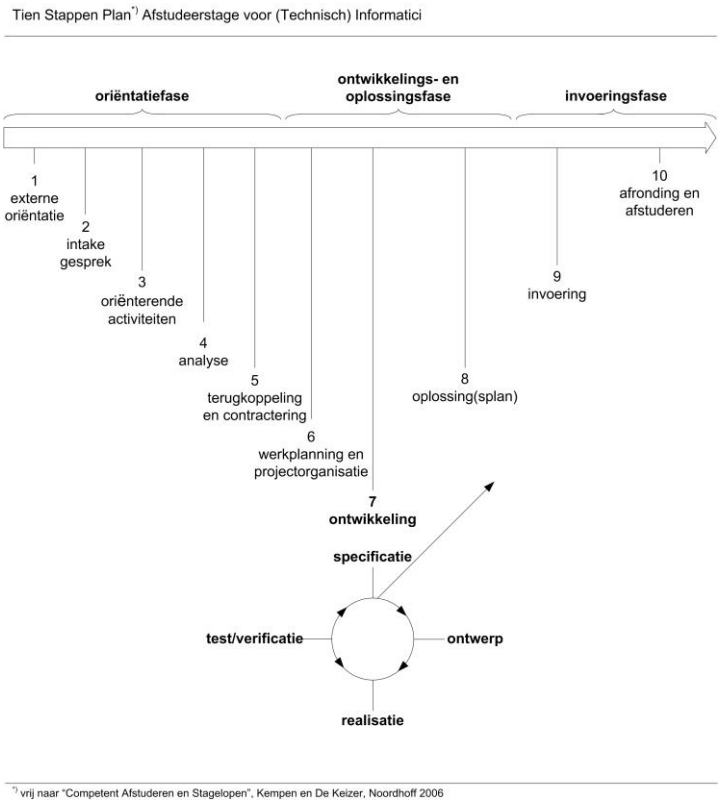
De afstudeeropdracht heeft als doelstelling dat er een automatisch rapportagesysteem voor het ATP wordt opgeleverd die bij voorkeur gebouwd is in ASP.NET, HTML5 en C#. De manier waarop de data getoond dient te worden is nog niet onderbouwd en er is nog geen onderzoek naar gedaan. Daarom zal er in de afstudeerperiode gestart worden met onderzoek naar de manier waarop de testresultaten inzichtelijk kunnen worden gemaakt voor de teams die binnen RES Software opereren. Vervolgens zal, met behulp van een agile project aanpak, de software zelfstandig worden ontworpen, gerealiseerd en gedocumenteerd.

Een tweede doelstelling van de afstudeeropdracht is dat RES Software graag meer inzicht wil krijgen in de werkwijze van ASP.NET. Dit inzicht wordt verkregen door de afstudeer opdracht in ASP.NET te laten ontwikkelen. Hierdoor heeft RES Software een basis project dat de teams kunnen gebruiken om hun eigen web applicaties op te bouwen.

4 Aanpak en planning

4.1 Tien Stappen Plan

Een onderdeel van het afstuderen is methodisch afstuderen. Om methodisch af te studeren is er gekozen voor het Tien Stappen Plan (TSP). Aan de hand van **Figuur 4** is het afstuderen onderverdeeld in drie hoofd fasen: Oriëntatiefase, ontwikkelings- en oplossingsfase, en de uitvoeringsfase. Elke fase is onderverdeeld in een aantal sub fasen die gedefinieerd zijn volgens TSP. Alle fasen worden uitgevoerd in de agile methodologie die standaard is binnen RES Software.



Figuur 4: Tien Stappen Plan

4.1.1 Oriëntatiefase

De focus van de oriëntatiefase ligt op het opstarten van het project. Tijdens deze fase zijn er verschillende interviews afgenomen van het personeel van RES Software. De interviews zijn voornamelijk gericht op welke functionaliteit het personeel graag wil zien in het dashboard. Het PID is het document dat uit deze fase is ontstaan.

Nadat het PID is goedgekeurd kan het onderzoek gestart worden. Maar voor het onderzoek gestart kon worden moest er een User Requirements Specification (URS) en een GUI worden gemaakt. Voor het opstellen van een URS moeten eerst de eisen van de opdrachtgever en het personeel worden geïnventariseerd. Als de eisen bekend zijn kunnen ze in de URS worden genoteerd. Een GUI, die aan de in de URS beschreven eisen voldoet, kan daarna worden ontworpen. Zodra de GUI is voltooid kan er begonnen worden aan het onderzoek.

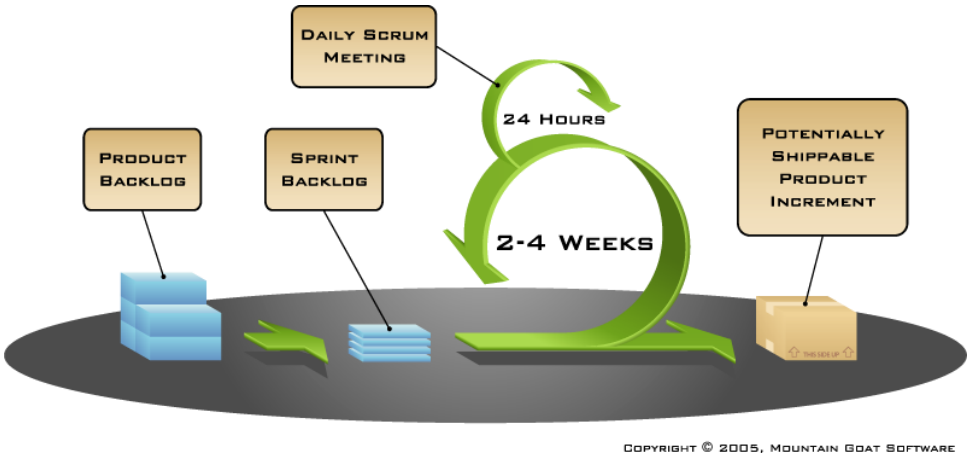
MoSCoW is een methode om prioritering van eisen en wensen aan te geven. Aan elke requirement wordt een van de MoSCoW (Must have, Should have, Could have, Would like) letters toegekend. De prioriteit van een requirement wordt bepaald op basis van hoe graag de opdrachtgever de requirement ziet terug komen in het eindproduct. Daarom wordt er eerst gefocust op de requirements die een M hebben. Daarna worden de rest van de letters afgewerkt om zo tot een eindresultaat te komen die de opdrachtgever wil zien.

Tijdens het onderzoek worden voornamelijk de onderzoeksvragen beantwoord. Dit gebeurt door middel van literatuuronderzoek dat voornamelijk zal bestaan uit het zoeken op het internet. Er wordt een onderzoeksdocument bijgehouden waarin de bevindingen worden genoteerd. Nadat de vragen in voldoende mate zijn beantwoord, is het onderzoek klaar en wordt er een conclusie getrokken. Aan de hand van deze conclusie wordt dan het dashboard ontwikkeld.

4.1.2 Ontwikkelings- en oplossingsfase

In de ontwikkelings- en oplossingsfase wordt de ontwikkeling van het project uitgevoerd. In deze fase worden typische software engineer taken uitgevoerd zoals het ontwerpen van de software, het realiseren van de ontworpen software, en het documenteren van de software. Het ontwikkelen gaat via Scrum en aan het einde van elke Scrum sprint is een demonstratie. In deze demonstratie wordt het dashboard getoond, zoals het er uitziet op dat moment, en besproken wat de stand van zaken is.

De ontwikkel methode die voor het realiseren van het project wordt gebruikt is Scrum (**Figuur 5)**. Scrum is een iteratief management-framework dat vaak gebruikt wordt om op een gestructureerde manier software te ontwikkelen (James, -). Scrum is een iteratieve methode waarmee er in een korte periode, rond 2 weken, een planning wordt gemaakt. Deze korte periode wordt een sprint genoemd en in de sprint wordt een iteratie van het product gerealiseerd. Tijdens de sprint is er dagelijks een daily stand-up meeting waar alle teamleden vertellen wat ze de dag ervoor hebben gedaan en wat ze vandaag gaan doen. Ook is aan het einde van elke sprint een korte demonstratie en wordt de voortgang besproken met de opdrachtgever en daarna samen te bepalen welke werkzaamheden de volgende sprint bevat.



Figuur 5: Scrum Proces

De voornaamste reden om voor Scrum als ontwikkelmethode te kiezen is dat Scrum binnen RES Software als standaard ontwikkelmethode geld. Hierbij is het ook verstandig om wat extra aandacht aan deze methode te besteden omdat er in de toekomst wel vaker mee gewerkt zal worden. Bovendien is de opdracht eenvoudig op te delen in iteraties, omdat de opdracht bestaat uit het maken van een dashboard met meerdere schermen.

Omdat Scrum vaak in een team wordt gebruikt en een stand-up meeting meestal binnen het team wordt gehouden is er voor gekozen om de afstudeerder in een team te plaatsen voor de stand-up meeting. Binnen dit team werkt ook de technische begeleider van de afstudeerder waardoor deze dagelijks de voortgang van het project kan horen.

4.1.3 Invoeringsfase

In de invoeringsfase wordt het dashboard klaar gemaakt voor beheer. Dit houdt in dat van het dashboard een installatiebestand wordt gemaakt en dat alle bevindingen van de ontwikkeling worden genoteerd in twee handleidingen. In de eerste handleiding wordt ingegaan op de technische kant en hoe er websites toegevoegd kunnen worden aan het project. In deze handleiding wordt ook uitgelegd hoe de beheerder de instellingen van het dashboard kan veranderen. In de tweede handleiding wordt er uitgelegd wat de gebruiker allemaal kan met het dashboard en hoe de functies van het dashboard zijn te gebruiken.

5 Oriëntatiefase

Tijdens de oriëntatiefase is er gewerkt aan het PID. De opdracht is samen met de opdrachtgever besproken en verduidelijkt. Hier is ook bepaald op welke onderdelen van het dashboard de afstudeerder zich ging richten. Het project heeft hier ook een naam gekregen: Project: Hathor. Deze naam is gekozen aan de hand van de Egyptische godin van de schoonheid Hathor. Omdat dit project het huidige dashboard mooier moest maken. Ook is het de godin van de mijnwerkers die naar ertsen mijnen. Het eindproduct van het project mijnt ook, alleen dan naar data in de vorm van test resultaten.

Een van de dingen waar de afstudeerder zich op gaat richten is welke ASP.NET stack op dit moment het beste is om te gebruiken. En wat er allemaal met deze ASP.NET stack kan worden gedaan.

Een goed onderbouwde keuze van stack is uiteraard erg belangrijk. Aangezien hier het hele project op wordt ontwikkeld. Een foute stack keuze kan betekenen dat de website minder goed onderhoudbaar wordt. Daarom wordt er in de ontwikkelingsfase ook uitgebreid onderzoek naar gedaan. Voor een goed onderzoek naar de ASP.NET stack moet er een document worden gemaakt waarin alle voordelen en nadelen van elke stack worden doorgelicht. Aangezien de keuze erg belangrijk is, wordt er voor elke stack ook een kleine POC gemaakt om de functionaliteit te testen die er behaalt dient te worden.

Aan de hand van wat er beschreven wordt in het onderzoeksdocument en de uitkomsten van de POC wordt er een keuze gemaakt voor de stack. Deze stack wordt dan gebruikt om het uiteindelijke project te realiseren.

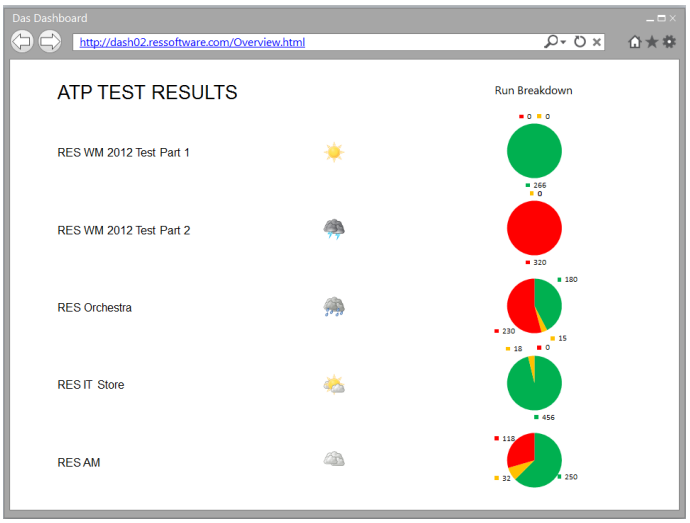
Er is een eerste opzet gemaakt van de planning voor het project. Hierin zijn de hoofdfasen van het Tien Stappen Plan terug te zien en de datums wanneer deze fasen afgerond zouden moeten zijn. De mijlpalen van het project zijn in de planning ook terug te zien samen met wanneer specifieke onderdelen afgerond dienen te zijn.

Er zijn een aantal risico’s geschat voor het Hathor project. Een groot risico van het project was dat de afstudeerder niet zo veel ervaring had met ASP.NET en hoe hierin alles is opgezet. Samen met de technische begeleider zijn hier een paar afspraken over gemaakt als de afstudeerder echt vast zou komen te zitten.

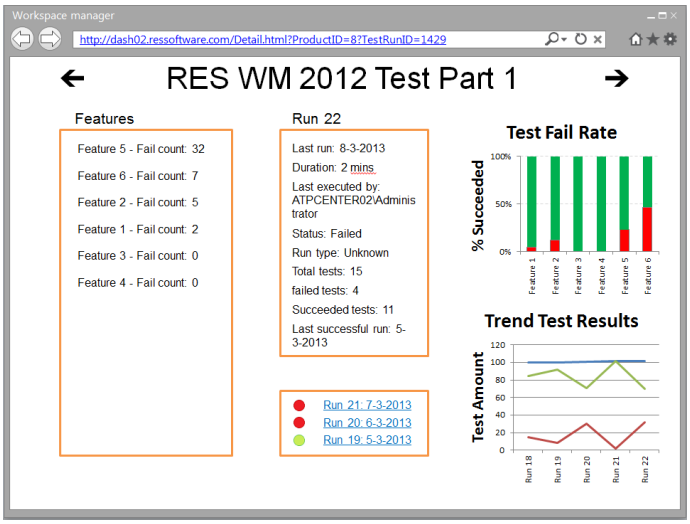
Verder zijn er afspraken gemaakt tussen de afstudeerder en de begeleiders met betrekking tot de voortgang van het project. Iedere dag is er een stand-up meeting waar er wordt verteld wat iedereen die dag heeft gedaan en wat er de komende dag gaat gebeuren. Hierbij doet de afstudeerder mee in de meeting van het Automation Manager team waar ook de technische begeleider aan deelneemt. De hoofdpunten die in de stand-up meeting worden besproken worden genoteerd in een dagboek dat wekelijks naar de docent begeleider wordt verstuurd.

Nadat het PID af was, is begonnen aan het URS. In het URS zijn eerst de requirements opgenomen van de opdrachtgever. Toen deze waren uitgeschreven is er bij de afdelingshoofden, die het project gaan gebruiken, nagevraagd of ze deze requirements acceptabel vonden. Hieruit zijn meningen van verschillende personen naar voren gekomen die het niet eens waren met de requirements zoals ze eerder waren opgesteld. Aan de hand van deze meningen zijn de requirements aangepast, die daarna door alle afdelingshoofden zijn goedgekeurd.

Op het moment dat het URS af was zijn GUI Mockups gemaakt van de schermen. Deze zijn van belang in het onderzoek omdat er ook wordt onderzocht welke soorten grafieken er gebruikt gaan worden.



Figuur 6: Overzichtsscherm



Figuur 7: Detail Scherm

Het overzichtsscherm (**Figuur 6**) is opgebouwd uit blokken die elk één product voorstellen. Elke blok is weer opgebouwd uit drie onderdelen: De naam van het product, een indicatie of de laatste vijf testen van het product goed verliepen en een cirkeldiagram die de laatste test run wat meer gedetailleerd laat zien. Het overzichtsscherm is het scherm dat zichtbaar wordt gemaakt op de dashboards die door het gebouw hangen.

Door in het overzichtsscherm op een blok te klikken gaat men naar het Detailscherm (**Figuur 7**). Dit scherm is ervoor bedoeld om in een browser te bekijken en geeft de gebruiker meer informatie over een test run van het aangeklikte product. In het linker gedeelte van het scherm is per feature te zien hoeveel testen niet succesvol waren. Op deze features kan door worden geklikt om steeds specifieker de fouten te kunnen zien.

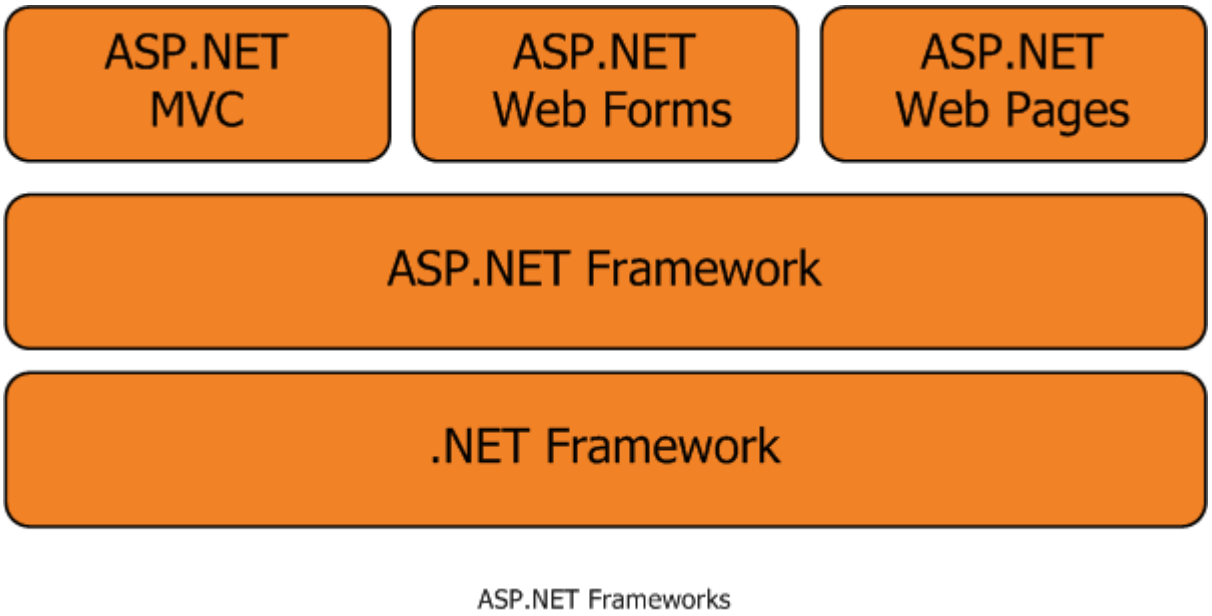
6 Onderzoek

Voordat er is begonnen aan de realisatie van Project Hathor, is er onderzoek gedaan. De hoofdredenen om het onderzoek uit te voeren zijn:

1. Het verrichten van onderzoek is een onderdeel van de afstudeeropdracht.
2. RES Software wilde weten welke stack voor de uitvoering van het project het beste was. Dit met de achterliggende gedachte dat ze meer op web gebied willen ontwikkelen.
3. Technische kennis verwerven voor de afstudeerder zodat deze de software kan realiseren en problemen van te voren al kan zien aankomen.

6.1 Het onderzoek

De software binnen RES Software wordt vooral ontwikkeld in .NET. Met de huidige markt komt er steeds meer behoefte aan web gebaseerde oplossingen. Hier wil RES Software op inspelen door middel van het ontwikkelen van projecten in ASP.NET. Hierdoor is de scope van het project ook op ASP.NET gevallen. RES Software wenst graag een start project te hebben dat als basis gebruikt kan worden om volgende web projecten te kunnen realiseren binnen het bedrijf. Binnen ASP.NET zijn er drie grote stacks aanwezig, zoals zichtbaar in **Figuur 8,** namelijk: Web Forms, MVC 4 en Web Pages. Binnen deze stacks wordt gekeken welke stack op dit moment het beste is om websites in te maken.



Figuur 8: ASP.NET stacks

6.2 Onderzoeksvragen

De hoofdvraag binnen het onderzoek is:

- Welke ASP.NET stack is het beste om te gebruiken voor het realiseren van het dashboard?

De deelvragen binnen het onderzoek zijn:

- Welke ASP.NET stacks zijn er?
  - In welke mate zijn de ASP.NET stacks bruikbaar?
  - Kunnen de ASP.NET stacks worden uitgebreid?
  - Zijn er extra frameworks die gebruikt kunnen worden?

- Hoe kunnen de ASP.NET stacks worden getest?
  - Welke test tool kits kunnen gebruikt worden?
  - Zijn de stacks gemakkelijk te testen?

- Welke stack heeft de beste tools om de data in het dashboard te laten zien?
  - Welke types grafieken zijn er?
  - Welk type grafiek presenteert de data het beste?
  - Welk type framework kan er gebruikt worden om de grafieken te laten zien?

6.3 Welke ASP.NET stacks zijn er?

Tijdens het onderzoeken kwam al snel aan het licht dat Web pages bijna geen ondersteuning had op professionele fora. Op websites als stackoverflow waren de topics over Web Pages nogal gering in aantal ten op zichtte van Web Forms en MVC. Hierdoor was het vinden van informatie over Web Pages een groot probleem. De enige website die goede informatie heeft is w3schools. Deze website is dan ook als leidraad gebruikt voor het onderzoek want de website heeft uitgebreide uitleg over alle 3 de stacks.

Web Pages (Microsoft, -) (w3schools, -)
Web Pages is de eerste van de drie stacks voor het creëren van ASP.NET websites en web applicaties.

Figuur 9: Web Pages Logo

Web Pages is de simpelste van de drie stacks en geeft een simpele en makkelijke manier om HTML, CSS, JavaScript, en server code te combineren.

Enkele voordelen opgesomd in willekeurige volgorde: (w3schools, -)

- Het is gemakkelijk te leren en te begrijpen.
- Het is gebouwd rond Single Web Pages van ASP.NET.
- Het lijkt veel op andere talen zoals PHP en klassieke ASP
- Het kan server scripts aan die in Visual Basic of C # zijn geschreven.
- Het geeft de ontwikkelaar de controle van HTML, CSS en JavaScript.

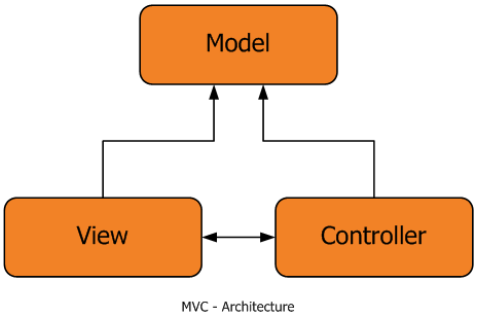
Web Forms (Microsoft, -)
Web Forms is de tweede van de drie stacks voor het creëren van ASP.NET websites en web applicaties.

Web Forms is gebaseerd op hetzelfde 'event driven' systeem dat wordt gebruikt bij Windows desktop applicaties.

Enkele voordelen opgesomd in willekeurige volgorde: (Vaibhav, 2008) (w3schools, -)

- Het is heel handig voor Rapid Application Development (RAD)
- Het heeft een uitstekende ondersteuning voor de designer van Visual Studio
- Het heeft een grote standaard bibliotheek aan componenten
- Het heeft componenten die met grote hoeveelheden data kunnen omgaan.
- Het heeft dezelfde 'event driven' system als Windows desktop applicaties waardoor het gemakkelijk kan worden geleerd door ontwikkelaars die dit ooit al eens gebruikt hebben.

MVC (Microsoft, -)
MVC is de derde en laatste van de drie stacks voor het creëren van ASP.NET websites en web applicaties.
MVC is de afkorting voor Model View Controller. Deze architectuur wordt door veel ontwikkelaars gebruikt in OOP. Zie **Figuur 10** voor een overzicht hoe MVC werkt.



Figuur 10: Opbouw van MVC

Enkele voordelen opgesomd in willekeurige volgorde: (Vaibhav, 2008) (w3schools, -)

- Het geeft volledige controle over HTML en CSS;
- Het genereert beter leesbare HTML in de ontwerper en de browser;
- Het geeft een bijna perfecte scheiding tussen de code en de view;
- Het kan gemakkelijk getest worden;
- Het kan meerdere view engines gebruiken inclusief view engines van derden;
- Het gebruikt RESTFul als standaard;
- Het heeft geen View State;
- Het geeft een overzichtelijke view page;
- Het werkt naadloos met JavaScript frameworks zoals JQuery.

Er is één groot framework dat kan toegevoegd worden aan Web Forms en MVC. Het framework heet Model View Presenter (MVP) (Boodhoo, 2006) en is in de basis een mix van MVC en Web Forms. MVP wil het beste uit twee stacks combineren en deze dan toegankelijk maken zodat de ontwikkelaars van beide stacks er mee kunnen werken.

MVP heeft een passieve view waar men probeert geen logica in te stoppen. Ook is het de “man in the middle” die communicatie tussen de view en model bijhoud. De presenter stelt ook alle eigenschappen van de weergave in. Terwijl MVC andersom werkt en de eigenschappen van de weergave vult met de data uit het model.

6.3.1 Conclusie

ASP.NET bestaat al een geruime tijd en heeft een grote hoeveelheid volgers in elke stack. Aangezien alle stacks nog altijd actief ontwikkeld worden zijn ze nog allemaal te gebruiken. Als er wordt gekeken naar hoe bruikbaar ze zijn valt Web Pages al meteen af, omdat er over Web Pages maar weinig informatie te vinden is.

De twee stacks die over blijven zijn Web Forms en MVC. Het oordeel van welke stack beter is, kan hier echter niet geveld worden. De keuze tussen de stacks is of het project snel moet af zijn (Web Forms) of dat het een project is dat er mooi uit moet zien (MVC). Daarnaast moet er ook gekeken worden of de ontwikkelaar kan omgaan met HTML / CSS.

Beide stacks hebben een gemeenschappelijke component. Voor beide is er een grote bibliotheek met uitbreidingen en mogelijkheden beschikbaar. Web Forms met alle aangepaste componenten en MVC met zijn view engines en grote hoeveelheden van JavaScript bibliotheken.

6.4 Hoe kunnen de ASP.NET stacks worden getest?

Al snel bleek dat het grote probleem van ASP.NET de slechte testbaarheid is. Omdat de meeste unit testers zijn gebouwd voor het testen van code en niet voor UI functionaliteit. Hiervoor zijn toen nieuwe manieren gezocht om de front-end te testen. Een van deze manieren is de Selenium tool kit die totale controle geeft over het front-end testen. Daarna is er nog een lijstje gemaakt met back end tool kits. Op het moment dat deze bekend waren is gekeken hoe gemakkelijk de stacks te testen waren. Hieruit kwam al snel naar voren dat MVC vele malen beter en makkelijker te testen was dan Web Forms.

Front-End Testen

Er zijn veel test tool kits die de front-end automatisch kunnen testen. Een van de betere test tool kits is Selenium. (Tester, 2008)

Selenium heeft 4 grote projecten (Selenium, -) die totale controle geven over het front-end testen. De projecten van Selenium zijn:

- Selenium IDE
- Selenium Remote Control
- Selenium WebDriver
- Selenium Grid

De IDE en WebDriver zijn de projecten die het meest bruikbaar zijn voor de ontwikkelaar De andere twee zijn er meer voor de tester die wil gaan testen over meerdere virtuele machines (VM) met daarop verschillende soorten besturingssystemen.

Back End Testen (Kojevnikov, 2008)
Voor het testen van de back-end van het programma bestaan er meerdere test platforms. De grotere test platforms voor unit tests zijn:

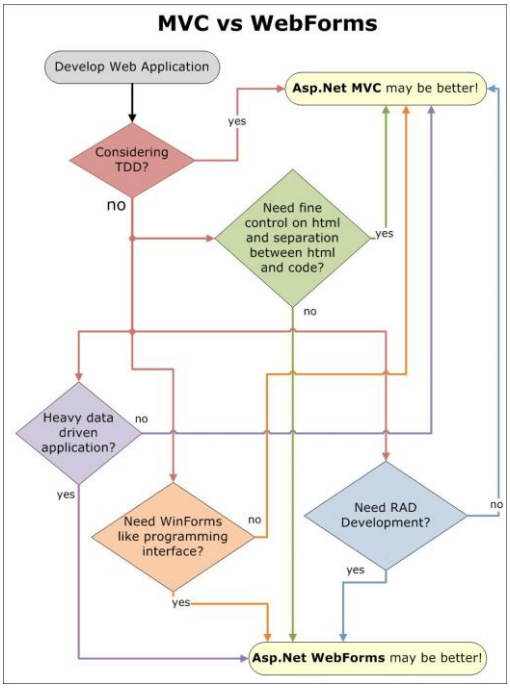
- NUnit
- MSTest
- XUnit

Er zijn ook een paar ondersteunende platforms zoals:

- NMock
- RhinoMock
- EasyMock

Elk professioneel bedrijf wil zijn software getest hebben voordat het wordt vrijgeven. Als er in het eindproduct een bug zit, kan dit zware consequenties hebben. Dit kan onder andere vermeden worden door het eindproduct goed te testen. RES Software is niet anders dan de rest en wil daarom dat de software altijd ondersteund wordt door unit tests.

De beste manier om een zo hoog mogelijke dekkingsgraad te verkrijgen met unit tests is om Test Driven Development (TDD) uit te voeren. Om goed te kunnen unit testen moet elk onderdeel van de stack in isolatie worden getest. Dit moet dus liefst ook zonder een webserver kunnen. In **Figuur 11** is zichtbaar gemaakt door (Vaibhav, 2008) wat de keuzes zijn in verband met keuzen tussen MVC en Web Forms. Hier is de eerste vraag of je TDD gaat gebruiken voor het ontwikkelen van je applicatie.



Figuur 11: Beslissing schema

6.4.1 Conclusie

Omdat er gebruik gemaakt wordt van Resharper is het van belang welke test tool kit er wordt gebruikt. Hierdoor is de standaard MSTest meer dan voldoende om de back end te testen. Als er iets niet door MSTest kan worden opgelost dan kan er ook gemakkelijk een ander tool kit worden gebruikt om de test toch nog tot een goed einde te brengen.

De stack die hier als winnaar naar voren komt is MVC. Dit vanwege de volgende redenen:

- Elk deel van MVC kan in isolatie worden getest
- Er zijn geen extra frameworks nodig om MVC zijn back end te testen.
- Er kan ontwikkeld worden in TDD
- MVC is goed test baar met Selenium

6.5 Welke types grafieken zijn er?

Een grafiek kan een grote hoeveelheid vormen hebben, De meest voorkomende vormen zijn (Wikipedia, 2013):

- Histogram
- Bar / Column chart
- Pie chart
- Line chart

Bij het zoeken naar de goede grafiek tool kit is er gekeken naar wat je met een grafiek kunt en op welke manier de grafieken zichtbaar worden gemaakt. Er werd ook gekeken naar welke software de grafieken het beste en mooiste liet zien. Na enig zoekwerk was er een lijst met grafiek software gevonden (Wikipedia, 2013). Deze lijst was een goed beginpunt omdat deze nog recentelijk was bijgewerkt. Hieruit is dan onderzocht wat elke grafieksoftware kon en hoeveel die zou gaan kosten.

Tabel 1 is een gedetailleerd overzicht van wat elke software kan.

Naam	Area	Bar	Column	Line	Pie	Legenda	Mengbaar
amcharts	N%SM	N%SM3D	N%SM3D	NMG	NM3DBD	✓	✓
CanvasJS	N%SM	N%SM	N%SM	NM	NMBD	✓	✓
dc.js	×	N%SM	N%SM	NM	NMD	✓	✓
D3js	N%SM	N%SM	N%SM	NMG	NMD	✓	✓
Highcharts	N%SM	N%SM	N%SM	NM	NMD	✓	✓
JqChart	NM	NSM	NSM	NM	NM	✓	✓
InfoVis	SM	SM	SM	×	SM	✓	×
KoolChart	NSM	NSM3D	NSM3D	NM	NSM3DBD	✓	✓
extJS	N%SM	NSM	NSM	NM	NSMBD	✓	✓
SenchaC	N%SM	NSM	NSM	NM	NSMBD	✓	✓
TechOctave	NM	NSM	NSM	NM	NMD	✓	✓
TeeChart	NSM3D	NSM3D	NSM3D	NM	NM3DBD	✓	✓
ZinoUI	NM	NM	NM	NM	NMD	✓	✓

Afkorting	Naam
N	Normaal
%	Percentueel
S	Gestapeld
3D	3D
G	Gaten
M	Meerdere Series
B	Gebroken
D	Donut

Tabel 1: Overzicht wat grafiek software kan

6.5.1 Conclusie

Het aantal soorten grafieksoftware is groot en de prijzen gaan van gratis tot $1000+. Alle frameworks hebben hun eigen kenmerken om hun data te laten zien. De keuze viel op D3.js omdat het veel mogelijkheden heeft. De hoeveelheid mogelijkheden komt omdat D3 niet alleen gefocust is op grafieken. Als ontwikkelaar kunnen er eigen dingen aan toe worden gevoegd omdat D3 open source is. Daarom is D3.js het breedste en het nuttigste framework van alle die er getest zijn. Het gebruikt de alle mogelijkheden van een browser door via JavaScript SVG tags aan te maken waar dan de grafieken in worden getekend. Daarbij valt het ook nog onder de BSD licentie waardoor het gebruik ervan gratis is.

6.6 Algemene Conclusie

Door het onderzoek zijn de drie deelvragen beantwoord. De antwoorden zijn terug te lezen in de conclusies van de hoofdstukken die hier voor staan. Hier is nog een verkorte versie.

Welke ASP.NET stacks zijn er?

- ASP.NET Web Pages
- ASP.NET Web Forms
- ASP.NET MVC 4

Hiervan kan Web Pages al worden weg gestreept, omdat er bijna geen informatie over te vinden is, en blijven Web Forms en MVC 4 over.

Hoe kunnen de ASP.NET stacks worden getest?
Beide stacks kunnen getest worden, alleen voor Web Forms is er meer moeite nodig. De stack die hier als beste naar voren komt is MVC. Vooral omdat de ontwikkelaar er Test Driven Development mee kan doen.

Welke stack heeft de beste tools om de data in het dashboard te laten zien?
Beide stacks kunnen de data even goed weergeven. Alleen is JavaScript makkelijker te gebruiken in MVC dan in Web Forms. Als D3.JS gebruikt gaat worden is de keuze meer geneigd naar MVC.

Nu de deelvragen voldoende zijn beantwoord kan ook de hoofdvraag worden beantwoord.

Welke ASP.NET stack is het beste om te gebruiken voor het realiseren van het dashboard?
De beste stack die gebruikt kan worden voor een nieuw project op dit moment is ASP.NET MVC4. De reden hiervoor is de uitbreidbaarheid van de stack op zowel front-end als back-end niveau. De ontwikkelaar kan zijn eigen view engine bouwen, kan de website naar hartenlust aanpassen en opbouwen en kan zijn eigen model gebruiken. Ook is MVC zeer goed te testen en kan er gewerkt worden volgens Test Driven Development.

Het enige nadeel van MVC is dat de eerste productie van een website langzamer is ten opzichte van Web Forms. Als de website eenmaal klaar is werkt de website sneller en kan de website ook makkelijker worden aangepast en uitgebreid. Omdat alles los van elkaar staat en met unit tests kan worden gecontroleerd.

Wat is het effect op het afstudeerproject?
Het effect dat de uitkomst van dit onderzoek heeft op het afstudeerproject is groot. Want de uitkomst betekent dat het gehele afstudeerproject in ASP.NET MVC4 gaat worden ontwikkeld.

7 Ontwikkelings- en Oplossingsfase

Nadat het onderzoek is afgerond, is gestart met de ontwikkelingsfase. De ontwikkelingsfase is onderverdeeld in verschillende kleinere Scrum sprints die één week zullen duren. Aan het einde van een sprint wordt een demonstratie gegeven aan de opdrachtgever. Na de demonstratie wordt er in een kort gesprek besproken of het product voldoet aan de eisen en of de gestelde doelen zijn behaald.

7.1 Technische Sprint 1

De eerste sprint was voornamelijk bedoeld om een eerste opzet van het project te maken en een overzichtsscherm te verkrijgen. Deze sprint duurde twee weken omdat de hele structuur moest worden ontworpen en daarna moest worden gerealiseerd.

Als eerste is er een klassendiagram met bijbehorend architectuurdocument gemaakt. Het architectuurdocument is volgens de n-layer richtlijnen (Microsoft N-Layer, 2013) opgesteld. Waardoor iedereen, die met de richtlijnen bekend is, het document kan lezen. Tijdens het maken van het architectuurdocument gaf de begeleider het advies om Managed Extensibility Framework (MEF) te gaan gebruiken in het project. Door MEF kan er dependency injection plaatsvinden en is het niet meer nodig dat de ontwikkelaar een statische referentie maakt naar een klasse, maar dat de ontwikkelaar een referentie maakt naar de interface van de klasse. Voor de interface zoekt MEF dan een implementatie en geeft deze dan aan de klasse. Hierdoor kan MEF gemakkelijk nieuwe implementaties inladen zonder dat de oorspronkelijke code wordt aangepast.

Hierna is begonnen aan de realisatie van het overzichtsscherm. Het overzichtsscherm wijkt op een aantal punten af van het originele design omdat het originele design geen rekening houdt met verschillende schermafmetingen. Hier moet rekening mee gehouden worden omdat het dashboard veel groter is dan het gemiddelde scherm van de ontwikkelaar. Er is daarom gebruik gemaakt van Responsive Web Design (Marcotte, 2010). Aan de hand van de responsive manier is de lijst nu in los staande blokken veranderd. De blokken kunnen door de browser los over het scherm worden bewogen. Voordeel hiervan is dat meerdere blokken niet meer zo worden geplaatst dat ze buiten het scherm vallen. Natuurlijk heeft dit ook zijn limieten. Een van de limieten is dat de blokken gedeeltelijk buiten beeld vallen als het scherm smaller is dan de breedte van een blok. Maar dit zou ook gebeuren als de website voor een specifieke schermafmeting zou zijn gemaakt.

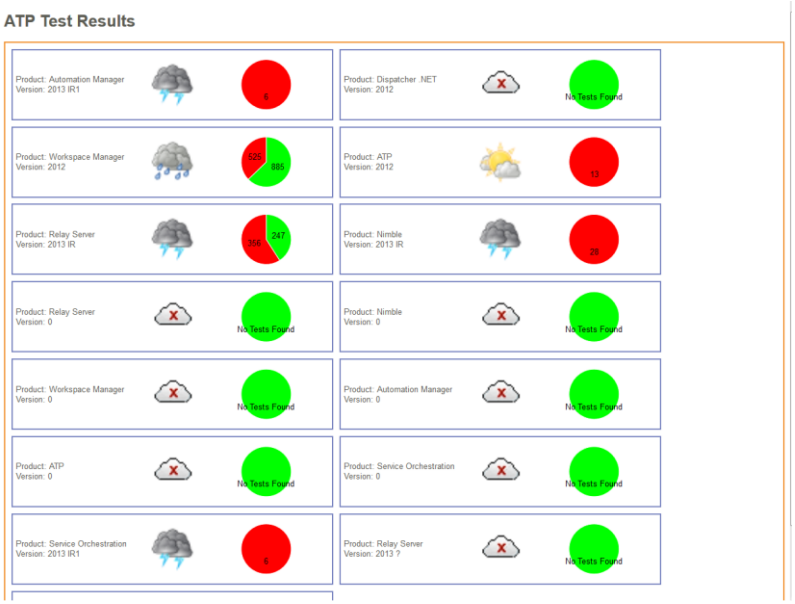
De cirkeldiagrammen die een overzicht geven van de laatste run worden asynchroon ingeladen via d3.js. (MacCaw, 2011) Dit heeft als voordeel dat de overzichtspagina sneller kan worden getoond, waardoor de gebruiker minder lang hoeft te wachten op zijn web pagina. Het heeft een positief effect op het gebruik van de pagina en gebruikers zullen daardoor meer gebruik gaan maken van de web variant van het dashboard.

Toen de GUI in grote lijnen af was kon de tijdelijke data vervangen worden met data uit de echte database. Dit ging echter niet omdat de database op een afgeschermd intern netwerk staat. Daarbij is het ook niet aan te raden om op een live database applicaties te ontwikkelen.

Er is daarom gekozen om een zogenoemde schaduwkopie te maken van de database en deze op een andere server te plaatsen. Omdat deze server kleiner was, is er gekozen om alleen de tabellen te kopiëren die ook daadwerkelijk gebruikt gingen worden. Het voordeel van een schaduwkopie is dat, als er iets fout gaat, de hele database weer terug kan worden gezet naar de beginstatus. Hierdoor konden er aanpassingen gemaakt worden in de database. Door bij te houden welke aanpassingen worden gemaakt, kunnen de aanpassingen later op de live database worden doorgevoerd. Hierna kunnen de programma's die de database gebruiken worden aangepast. Aangezien het laatste veel tijd en geld kost is het niet wenselijk om bestaande structuren te veranderen. De veranderingen zijn daarom vooral op de database views doorgevoerd.

In de bespreking van de demonstratie kwam al snel naar voren dat de opdrachtgever toch iets wil kunnen instellen aan het programma. Hiervoor is er een XML opzet gemaakt die tijdens een latere sprint wordt ingebouwd.

Figuur 12 is een screenshot van hoe het overzichtsscherm er uitziet aan het einde van sprint 1. De blokken structuur van de mockups met de drie onderdelen is gebleven. Alleen is de witruimte tussen de onderdelen flink verkleind zodat er meer blokken op het scherm passen.



Figuur 12: Overzichtsscherm sprint 1

7.2 Technische Sprint 2

De tweede sprint ging voornamelijk om het realiseren van het eerste detailscherm. Dit scherm laat de details van een test run van een product zien. De GUI maken ging, net als bij het overzicht, met behulp van responsive design en asynchroon laden via d3.js. Echter, toen de tijdelijke data werd vervangen door data uit de database, viel het op dat het laden van de grootste pagina, met de meeste resultaten, 30 seconden duurde. 10 seconden voor de eerste site en twee keer 10 seconden voor de grafieken.

De tijden voor het laden van het dashboard bleken een groot probleem. De beginwaarden waren als volgt:

```
Start:
Time get product: 1000 Ms
Time get run: 4687 Ms
Time get runs: 7354 Ms
End Time: 10456 Ms
```

Deze tijden zijn niet echt wenselijk voor een website. De get product tijd gaf al een grote hint wat er fout was, omdat dit een SQL Query was op een tabel die amper 50 entries had. De oplossing was om primary keys toe voegen, die verloren waren gegaan onder het kopiëren, aan de afgeslankte database. Hierdoor werden de wacht tijden die de website nodig had om te laden al meer acceptabel namelijk:

```
Primary Keys:
Time get product: 1 Ms
Time get run: 521 Ms
Time get runs: 1105 Ms
End Time: 1109 Ms
```

De impact van de primary keys op de snelheid was direct zichtbaar in het detailscherm en het overzichtsscherm. Het laden van de content in de website ging stukken sneller. Er waren echter nog wel uitschieters op het gebied van laadtijden. Bijvoorbeeld als er vijf browsers tegelijk content opvroegen dan waren de browsers op elkaar aan het wachten. Dit is niet echt wenselijk gedrag waardoor er eens goed naar de database views gekeken moest worden. De views bleken moeilijke query's te zijn zonder dat de uitkomsten werden opgeslagen.

Er is daarom besloten om Schemabound views te maken. Schemabound views zijn in grote lijnen hetzelfde als de materialized views van Oracle, maar de werkwijze van een Schemabound view is heel anders dan die van een normale view. Bij een normale view wordt bij elke query de view opnieuw gegenereerd. Bij een Schemabound view wordt de view gegenereerd als er een verandering is op de inhoud van de tabellen die de view gebruikt. De view wordt opgeslagen in een tabel en fungeert exact hetzelfde als een normale tabel. Hierdoor kunnen indices op de views worden aangemaakt die het zoeken door de views versnellen. Dit was ook meteen zichtbaar:

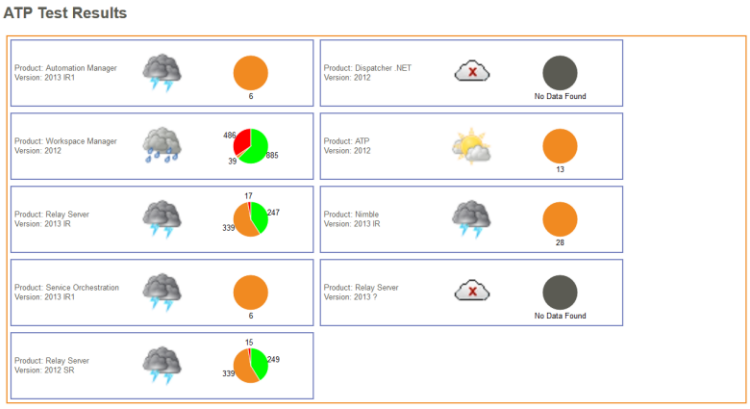
```
New views:
Time get product: 1 Ms
Time get run: 40 Ms
Time get runs: 106 Ms
End Time: 140 Ms
```

Hierna zijn er verschillende selenium tests gemaakt om te valideren of alle data correct werd weergegeven. Dit bleek niet het geval te zijn omdat features zonder tests niet worden weer gegeven. Door eerst de features op te halen en daar testen aan te koppelen, is dit opgelost. Het koppelen ging in eerste instantie niet naar verwachting. Alleen de eerste test werd gekoppeld en de rest werd genegeerd. Na onderzoek bleek dat Entity Framework de query goed uitvoerde, maar het converteren naar objecten ging fout. Dit kwam doordat het Entity Framework een cache had waar corrupte data in stond. Nadat dit cache was uitgezet, was alle data goed en de pagina sneller geworden.

```
Cache off:
Time get product: 3 Ms
Time get run: 24 Ms
Time get runs: 39 Ms
End Time: 56 Ms
```

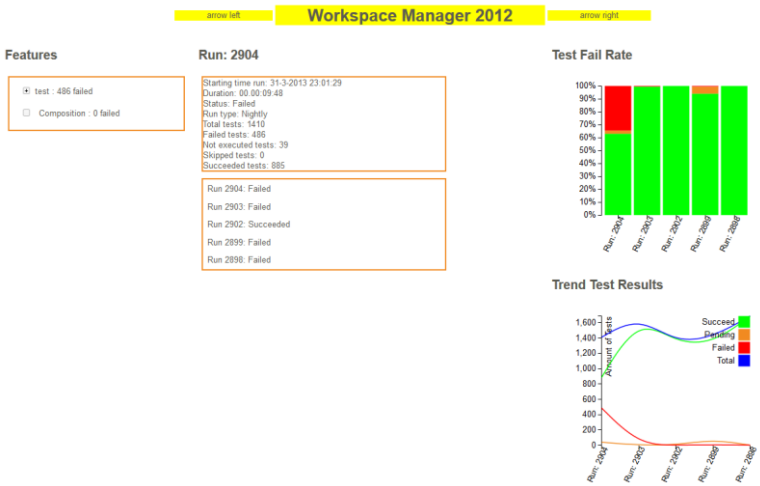
Nadat dit probleem was opgelost stond er nog steeds foutieve data op de schermen. Dit probleem is ontstaan doordat de parentID's niet worden bijgewerkt binnen het ATP Center als er hier testen in worden gekopieerd. Hiervoor is een bug report aangemaakt en hopelijk wordt dit opgelost in de volgende versie van ATP Center.

Figuur 13 is een screenshot van hoe het overzichtsscherm er uitziet aan het einde van sprint 2. Er zijn niet veel verschillen tussen sprint 1 en 2 qua uiterlijk. Alleen is de data, die naar de cirkeldiagrammen gaat, nu anders opgebouwd, zodat nu zichtbaar kan worden gemaakt hoeveel testen niet zijn uitgevoerd.



Figuur 13: Overzichtsscherm sprint 2

Figuur 14 is een screenshot van hoe het detailscherm er uitziet aan het einde van sprint 2. Links is de boom structuur aanwezig waar men kan door klikken naar het volgende detailscherm. In het midden is er meer gedetailleerde informatie zichtbaar en kan er worden door geklikt naar oudere test runs. De grafieken geven al data weer die relevant is voor de test run.



Figuur 14: Test Run scherm sprint 1

7.3 Technische Sprint 3

De derde sprint ging voornamelijk om het inbouwen van de profielen. Met de profielen kan direct worden beïnvloed welke producten zichtbaar zijn in het overzichtsscherm. Dit is gerealiseerd door een XML bestand waar de beheerder profielen in kan aanmaken die via de querystring van de browser kunnen worden gebruikt. Hiermee heeft de beheerder alle vrijheid om profielen toe te voegen, te veranderen en te verwijderen.

Het XML bestand van de profielen wordt enkel en alleen ingelezen als het dashboard wordt gestart. Als er een profiel wordt veranderd moet de server opnieuw worden opgestart. Dit is normaal niet wenselijk maar in deze sprint is deze manier van inlezen voldoende.

Aangezien het aanpassen van het XML bestand niet in een gecontroleerde omgeving gebeurd kan het zijn dat er fouten in zitten. Als er fouten in zitten of als het bestand is verwijderd, maakt het dashboard een nieuw bestand. Dit bestand wordt dan gemaakt op basis van een sjabloon dat in het dashboard is ingebouwd. Er wordt een back-up gemaakt van het foute bestand zodat de informatie die daar in staat niet verloren gaat.

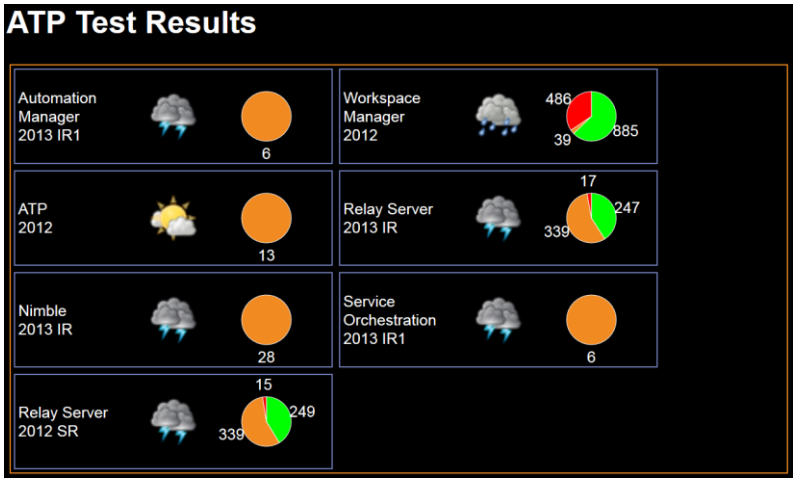
Hieronder staat een gedetailleerde uitleg wat er allemaal kan met de profielen XML bestand.

```
<Profile>
  <Name> </Name>
  <ShowChildrenOnly> </ShowChildrenOnly>
  <ShowEmpty> </ShowEmpty>
  <RefreshTime> </RefreshTime>
  <HighContrast> </HighContrast>
  <BigLetters> </BigLetters>
  <Products>
    <Include>
      <Product>
        <ID></ID>
        <Name> </Name>
        <Version></Version>
      </Product>
    </Include>
    <Execlude>
      <Product>
      </Product>
    </Execlude>
  </Products>
  <PresentationURLS>
    <URL></URL>
  </PresentationURLS>
</Profile>
```

Naam	Uitleg
Name	De unieke naam van het profiel
ShowChildrenOnly	Of het overzicht alleen de kinderen van het product laat zien.
ShowEmpty	Of het overzicht lege producten moet weergeven.
RefreshTime	De refresh tijd dat de site zichzelf refreshed. Als er URL's zijn in PresentationURLS dan is het refresh tijd naar de volgende URL.
HighContrast	Of het dashboard in high contrast moet verschijnen.
BigLetters	Of het overzicht vergrote letters moet gebruiken.
Products	Een lijst met producten.
Include	Een lijst met producten die in het overzicht moeten zichtbaar zijn. Als deze lijst leeg is of niet bestaat dan worden alle producten zichtbaar gemaakt.
Execlude	Specifieke producten die weg worden gehaald uit het overzicht.
Product	Een product
ID	Het ID van een product Als er geen ID wordt gegeven dan kan er een Naam en Versie worden gegeven.
Name	De naam van het Product
Version	De versie van het Product
PresentationURLS	Een lijst met URL die het profiel af gaat.
URL	Een URL die het profiel af gaat.

Aan het einde van de sprint zag het dashboard er als volgende uit:

Figuur 15 is een screenshot van hoe het overzichtsscherm er uitziet aan het einde van sprint 3. In deze screenshot zijn verschillende instellingen van het profiel terug te zien. De eerste instelling is de HighContrast. Dit wordt weergegeven als zwart met witte letters in plaats van wit met zwarte letters, zoals zichtbaar in **Figuur 13**. De tweede instelling is de BigLetters instelling, deze maakt alle tekst wat groter zodat de tekst goed zichtbaar is. De derde instelling is ShowEmpty, hierdoor worden lege producten weg gefilterd.



Figuur 15: Overzichtsscherm sprint 3

7.4 Technische Sprint 4

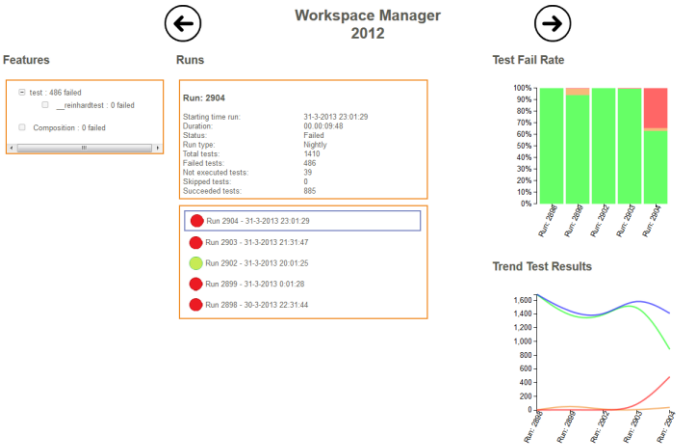
De vierde sprint ging voornamelijk om het realiseren van het tweede detailscherm. Dit scherm zoomt in op een feature van een product en laat een overzicht zien van alle scenario's die de feature heeft. Bij het maken van het eerste detailscherm is er rekening gehouden met het feit dat er nog een tweede detailscherm komt. Het tweede detailscherm komt er ongeveer hetzelfde uit te zien als het eerste detailscherm. Hierdoor kan het tweede detailscherm dezelfde layout gebruiken als het eerste detailscherm. Op deze manier is het tweede detailscherm in minder dan een dag gerealiseerd en getest.

De rest van de week zijn er vooral details opgelost die samen toch een grote lijst vormde. Een van deze details was het gebruik maken van een File Watch om veranderingen in profiles.XML waar te nemen. Door de File Watch te gebruiken kunnen de waarden binnen het programma worden bijgewerkt op het moment dat er iets aan de XML wordt veranderd. Hierdoor hoeft de beheerder niet de server opnieuw op te starten of op een knop te drukken om de XML opnieuw in te lezen.

Het programma is robuuster gemaakt door exception throws te vervangen door out parameters. Out parameters zijn bijna hetzelfde als return statements en kunnen objecten uit een methode terug geven. Alleen worden de out parameters niet impliciet terug gegeven. Hierdoor kan de methode meerdere out parameters vullen en dan bijvoorbeeld een boolean terug geven om aan te geven of de methode goed of fout is gegaan.

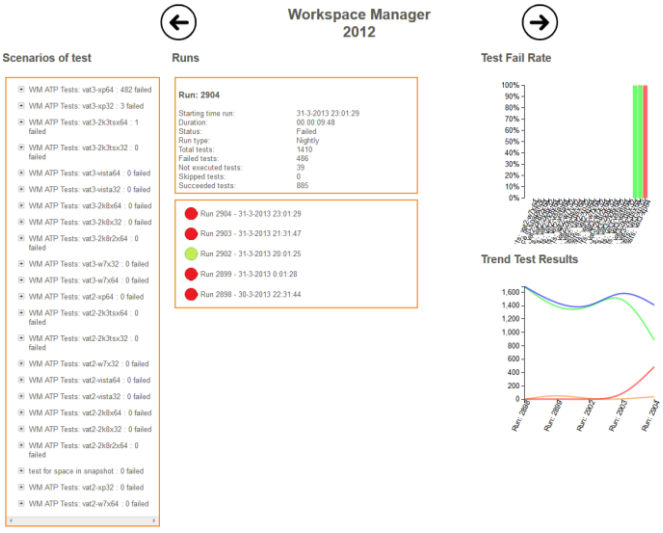
In de vorige demonstratie is gebleken dat nergens in het detailscherm zichtbaar was in welke run men zich bevond. Dit is opgelost door een blauwe rand rondom de huidige run, in het overzicht van beschikbare runs, te plaatsen waardoor in een oogopslag zichtbaar was in welke run men zich bevond. Waar eerder de tekst "failed" of "succes" werd gebruikt, om te laten zien of een run wel of niet succesvol is verlopen, wordt nu een rood of groen bolletje geplaatst.

Figuur 16 is een screenshot van hoe het detail scherm er uitziet aan het einde van sprint 4. In dit scherm is vooral aan de opmaak gewerkt. Alles ziet er wat schoner uit en de grafieken laten een legenda zien als de gebruiker met de muis over een staaf of over een lijn gaat. Met de twee pijlen kan er tussen runs worden genavigeerd.



Figuur 16: Test Run scherm sprint 4

Figuur 17 is een screenshot van hoe het feature scherm er uitziet aan het einde van sprint 4. In grote lijnen is dit scherm hetzelfde als het scherm uit Figuur 16 alleen is er meer ingezoomd op één feature. De enige verandering is dat aan de linker kant een lijst met scenario's te zien is van de feature.



Figuur 17: Feature scherm sprint 4

7.5 Technische Sprint 5

In de vijfde en tevens laatste technische sprint is er vooral gekeken naar de requirements. Eén van de requirements was om een gedetailleerd foutenrapport op te stellen en die te versturen naar de relevante personen. Hiervoor is er een automatisch mail systeem gemaakt waar dagelijks een mailtje wordt uitgestuurd naar een lijst van personen. Deze lijst van personen wordt via een XML mail lijst opgehaald waar ook de producten in staan die gemaild moeten worden. In dit XML bestand staan ook de instellingen voor de Mailer.

```
<Settings>
  <FromAddress> </FromAddress>
  <Host> </Host>
  <Port> </Port>
</Settings>
<MailList>
  <Mail>
    <Name> </Name>
    <SendAround></SendAround>
    <EmailList>
      <Email> </Email>
    </EmailList>
    <ProductIDList>
      <ProductID></ProductID>
    </ProductIDList>
  </Mail>
</MailList>
```

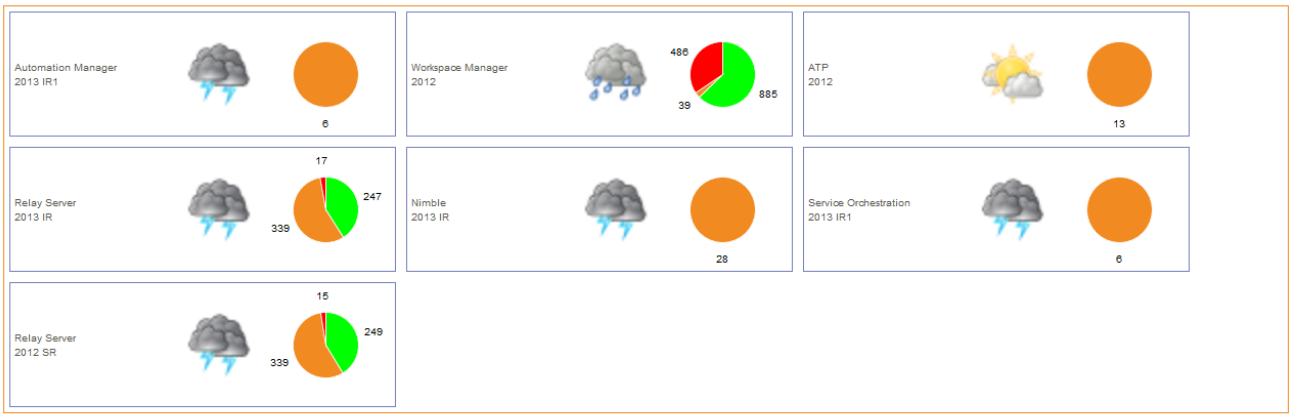
Naam	Uitleg
Settings	De instellingen van de mailer
FromAddress	Het adres waar de e-mail vanaf verzonden wordt.
Host	Het SMTP host adres van de e-mail server.
Port	De SMTP port van de e-mail server.
MailList	Een lijst van mails.
Mail	Een mailing lijst.
Name	De naam van de mail lijst.
SendAround	Het tijdstip wanneer de mail ongeveer wordt verstuurd.
EmailList	Een lijst van e-mails
Email	Een e-mail waar de mail wordt heen verstuurd.
ProductIDList	Een lijst van product ID.
ProductID	Een product ID die wat in de mail wordt verwerkt.

Nadat de Mailer af was miste er nog wat gebruiksvriendelijkheid. Dit gaf echter een probleem met de indeling van de site. Want waar laat je een “Back to Overview” knop zonder dat de structuur van de site wordt veranderd. Om dit probleem op te lossen werd gekeken hoe andere websites dit oplosten. Het viel al snel op dat de meeste websites hun logo in de header van de site hadden staan. Als er op dit logo wordt geklikt, gaat de gebruiker terug naar de hoofdpagina van de website. De oplossing van dit probleem is dus om het logo van het bedrijf links in de header van het dashboard te plaatsen. Door op dit logo te klikken kan terug worden gegaan naar het overzichtsscherm.

De querystring van de website begon langzaam maar zeker steeds groter te worden. Om deze wat in te korten is het profiel uit de querystring gehaald en in een cookie geplaatst. Dit had verschillende voordelen zoals:

- Geen “Profile=” meer in de querystring die elke keer doorgegeven moet worden
- Profile wordt eenmaal ingesteld aan het begin
- Profile kan ingesteld worden via het instellingen scherm

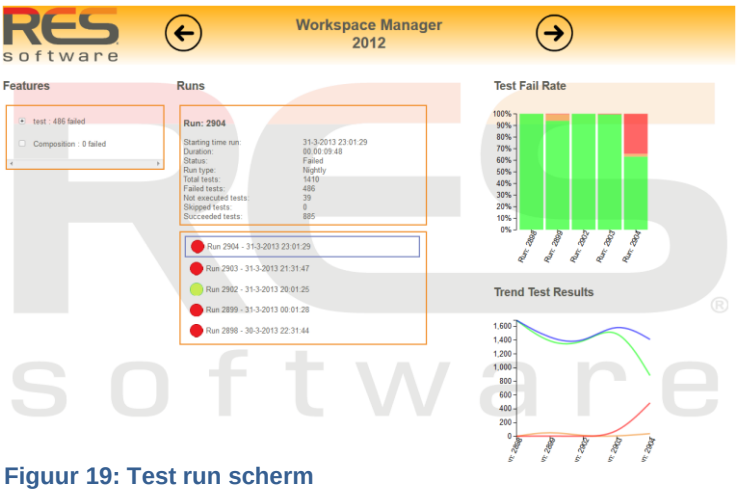
Als laatste waren de detail schermen nog wat statisch en vast. Dit is opgelost met Responsive Web Design toe te passen op de onderdelen van het detail scherm. Hierdoor kan de website nu gemakkelijk gelezen worden op een mobiel, tablet en pc.



Figuur 18: Overzichtsscherm

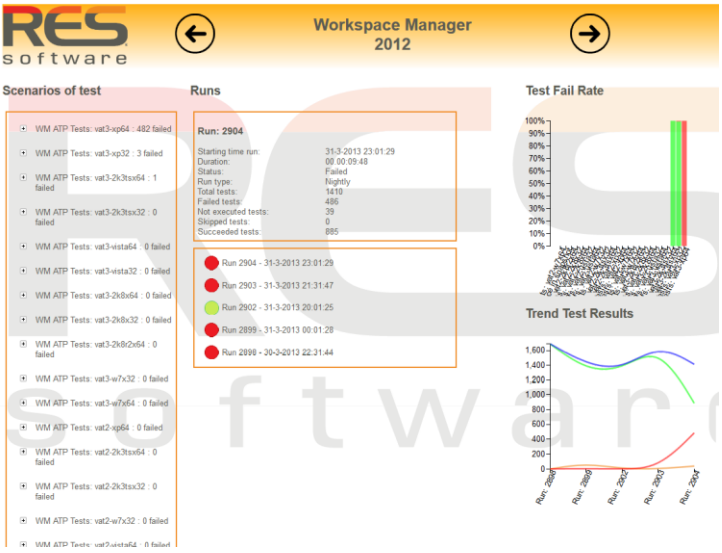
Figuur 18 is de uiteindelijke versie van het dashboard. Hier zijn de blokken goed op zichtbaar en heeft het profiel de lege waarden verwijderd.

Figuur 19 is de uiteindelijke versie van het test run scherm. Dit scherm is nog wat levendiger gemaakt door een watermerk toe te voegen, de header een oranje kleurverloop te geven en het logo, waardoor de gebruiker terug naar het overzichtsscherm kan gaan, in de linker boven hoek te plaatsen.



Figuur 19: Test run scherm

Figuur 20 is de uiteindelijke versie van het featurescherm. Deze bouwt voort op het vorige scherm waarin alle veranderingen qua uiterlijk ook zijn doorgevoerd.



Figuur 20: Feature scherm

Figuur 21 is het test run scherm, zoals het er uitziet op het formaat van een tablet. Hier is goed zichtbaar wat Responsive Web Design doet met de website. De twee grafieken verplaatsen zich van rechts op het scherm naar onder op het scherm. Hierdoor past nog steeds alles netjes op één scherm en hoeft de gebruiker niet onnodig te scrollen.

Aangezien het featurescherm verder bouwt op het test run scherm is het verschuiven van de grafieken ook meteen van toepassing op het featurescherm.

Figuur 21: Tablet versie

Figuur 22 is het test run scherm zoals het er uitziet op het formaat van een mobiel. Alle onderdelen van het scherm zijn nu onder elkaar gezet. Het scrollen beperkt zich alleen tot omlaag scrollen en geeft dezelfde informatie als in de schermen hierboven.

Figuur 22: Mobile versie

8 Invoeringsfase

Ten tijde van het schrijven van deze scriptie was deze fase nog in volle gang. Daarom wordt in dit hoofdstuk samengevat wat er nog moet gebeuren om deze fase af te ronden.

Deze fase gaat hoofdzakelijk om het documenteren van de opdracht zodat de kennis niet verloren gaat. Er zullen hier twee hoofddocumenten gemaakt worden, namelijk de Gebruikshandleiding en Technische handleiding. Deze zullen de hoofdstukken bevatten zoals hieronder is beschreven.

Gebruikshandleiding

- Hoe kan het dashboard gebruikt worden
- Welke features heeft het dashboard
- Wat kan er veranderd worden aan de instellingen van het dashboard
- Instellingen als iets fout gaat

Technische handleiding

- Wat kan er met de templates van de Mailer
- Hoe zet je een ASP.NET MVC4 project op
    - Waar moet op gelet worden
    - Wat en hoe te gebruiken
        - Razor
        - D3.js
        - MEF MVC4
        - TFS build server
        - URL Redirector
- Dashboard deploy handleiding

Daarnaast worden problemen, die zich voordoen tijdens het publiceren van de website, opgelost en gedocumenteerd. Hier kan dan bij volgende web projecten rekening mee gehouden worden. Een van de problemen was dat de JavaScripts niet geminimaliseerd werden door de TFS build server. Dit is opgelost door een aantal extra waarden mee te sturen naar de build server waardoor die de Release profiel draaide i.p.v. het Debug profiel.

9 Conclusies en aanbevelingen

RES Software had het probleem dat het ATP slecht inzichtelijk was voor de testers die het systeem gebruikten. Dit probleem zouden ze graag zien opgelost met een dashboard waar de testers snel en gemakkelijk de informatie van hun testen konden aflezen. Hiervoor is er onderzoek gedaan naar ASP.NET en grafieken software om dit dashboard te kunnen realiseren. Uit het onderzoek is gebleken dat ASP.NET MVC en D3.JS de beste software zijn om het dashboard mee te realiseren. Dit had als gevolg dat het hele project ook is uitgevoerd met deze frameworks.

Het project is na het onderzoek uitgevoerd in ASP.NET MVC4 waarbij de grafieken asynchroon door D3.JS werden getoond. Dit heeft als voordeel dat er een deel van de server load naar de browser van de gebruiker wordt verplaatst. Door dit te verplaatsen kan de web pagina sneller worden getoond aan de gebruiker, wat weer tot een betere gebruikerservaring leidt.

Daarbij is ASP.NET MVC4 een goede keuze gebleken om het probleem op te lossen. Het doet precies wat de ontwikkelaar wil en heeft geen verborgen functionaliteiten waar de ontwikkelaar geen invloed op heeft.

Tijdens de afstudeerstage zijn alle doelen bereikt. RES Software heeft een dashboard waarop een overzicht van de test runs zichtbaar is. In het overzicht kan worden doorgeklikt naar detailschermen, waar de uitkomsten in detail kunnen worden bekeken. Een bijkomstigheid is dat het project een web project is en het overal, waar een internet verbinding is, weergegeven kan worden, zoals op een computer, mobiele telefoon of tablet. Natuurlijk hebben een mobiele telefoon en tablet een aantal beperkingen waaronder de grootte van het scherm. Op deze beperking is ingespeeld met Responsive Web Design waardoor de site in stukken is opgedeeld die zich ten opzichte van elkaar kunnen verplaatsen. Hierdoor hoeft de gebruiker niet opzij te scrollen alleen maar omlaag.

Er zijn echter nog wat aanbevelingen:

- Op het huidige dashboard staan scrum burndown grafieken van de huidige sprints. Deze zijn niet meer zichtbaar als het nieuwe dashboard online wordt gezet. Daarom is het verstandig deze toe te voegen aan het nieuwe dashboard zodat deze dan ook zichtbaar zijn in de browser.
- Maak één instellingen scherm waar de gebruiker profielen kan aanpassen, toevoegen, verwijderen ook de instelling van het huidige profiel kan instellen.
- Gebruik Selenium tests om te controleren of ook alle data goed in de browser wordt weer gegeven. Dit is vaak zeer handig gebleken omdat op deze manier snel fouten in de achterliggende code opgespoord konden worden.
- Zet Selenium tests en I/O tests uit in de automatische build server van TFS. Deze falen altijd en geven een fout beeld van hoe goed het project werkt.
- Test de front-end altijd met de hand testen ook al heeft de website veel Selenium tests. Omdat Selenium niet de “look and feel” van een website kan testen.
- Gebruik schemabound database views om meer performance uit de database te halen. De views die aanwezig zijn in de huidige database hebben een zeer slechte performance.
- Zet de cache van het Entity Framework uit. Caching wil je absoluut niet hebben als je er geen 100% controle over hebt. Tijdens dit project heeft de cache voor meer dan genoeg corrupte data gezorgd.

10 Afronding en evaluatie

Als backend-webdeveloper die graag met Java, Java-EE en GWT werkte was ik erg benieuwd naar hoe het was om met ASP.NET te werken.

10.1 MVC

Op het moment dat ik met de afstudeerstage startte, verwachtte ik dat ASP.NET nog flink achter zou liggen op Java-EE en dat ik alle voordelen van Java-EE zoals dependency injection, GWT en JavaScript niet kon gebruiken. Ik ben echter blij dat ik hier een foute aanname heb gedaan.

Het standaard project van MVC 4 had al een zeer mooie GUI dat ook nog eens via Responsive Web Design standaarden was gebouwd. Ook zaten JQuery en Modernizer er standaard al in. Voor de backend waren veel geschikte onderdelen waar ik mij helemaal op kon uitleven. In het begin miste ik wel een aantal onderdelen zoals dendency injection, Apache Maven om alle afhankelijkheden op te lossen en een build server zoals Jenkins. Dit was even wennen want door die technieken was mijn productiviteit vele malen omhoog gegaan in de laatste jaren.

Nadat ik had vastgesteld dat deze technieken niet standaard in ASP.NET MVC zaten, ben ik er naar op zoek gegaan. Al snel bleek dat Microsoft op dit vlak flink bezig was met verbeteren. Hun versie van Maven is NuGet, dit is niet zo uitgebreid als Maven en werkt ook heel anders maar het gaf mij wel de mogelijkheid om zonder veel problemen bibliotheken toe te voegen aan het project. Na wat zoeken in NuGet en vragen aan mensen binnen het bedrijf waren er al snel oplossingen gevonden voor de technieken die ik miste. Voor dependency injection was er MeF waar ik later nog een addon bij vond, ‘MeF for MVC’. Hoewel deze niet hetzelfde aanbood als de dependency injection van Java-EE kon ik hier wel al goed mee werken. Nog iets wat mij snel opviel was dat er maar maximaal 1 object aan een view kon worden meegegeven. Daarbij waren dingen als een sessie gewoon niet bestaand binnen MVC. Dit vond ik een grote beperking in de functionaliteit van MVC en het web.

10.2 Responsive Web Design

Ik heb zelf nooit echt tijd gestoken in het design en de ‘look and feel’ van een website. Ik kende wel de basis van HTML 5/CSS 3 maar daar hield het dan ook mee op. Daarom was het front end fatsoenlijk krijgen een grote uitdaging voor mij. Daarbij had ik de lat voor mij flink hoog gelegd met D3.js aangezien die puur op HTML leunde.

Het werken met deze technologieën heeft mij flink veel nieuwe inzichten gegeven over hoe gemakkelijk / moeilijk het is om een goed functionerende en modern ogende site te hebben. Daarom ben ik ook zeer tevreden met het eind resultaat van de opdracht. Het werkt zoals ik het voor ogen had in het begin en is het hier en daar zelfs mooier geworden dan ik van mijzelf had verwacht. Hierdoor is de stage een echte aanvulling geweest voor mijn kennis op dit gebied.

10.3 Cross-Browser

Dit is altijd een heikel thema als je het over website hebt. Is de site wel Cross-Browser? Dit zal altijd een probleem blijven zolang browsers als Internet Explorer blijven bestaan. Als je eenmaal alles werkend hebt op Firefox, werkt het bijna altijd ook op alle andere browsers. Behalve Internet Explorer, die heeft alles anders ingebouwd. Hierdoor zijn er veel regels dubbel in het CSS omdat de HTML 5 standaarden nog niet allemaal zijn ingebouwd in Internet Explorer. Toch is het gelukt om de site ook op Internet Explorer goed te laten draaien. Hierbij was Selenium een echte hulp aangezien je met Selenium ook kan testen of de CSS goed is gebruikt door de browser, en of de JavaScript naar behoren werkte.

10.4 TSP

Tijdens het afstuderen vond ik TSP meer een last dan een aanvulling. TSP is naar mijn mening meer bedoeld voor business doeleinden dan software doeleinden. Regelmatig had ik moeite met het uitvoeren van bepaalde onderdelen van het TSP. Een goed voorbeeld is dat er nergens in de TSP planning voor Technische Informatica een stap staat waar het onderzoek in dient uitgevoerd te worden. Ik heb daarom het onderzoek op de voor mij meest logische plek neergezet.

10.5 Competenties

Een onderdeel van het onderwijs binnen Fontys zijn de competenties. Deze competenties worden gebruikt om te kijken of de student gereed is om tot de beroepsmarkt toe te treden. Tijdens het afstuderen heb ik van een aantal competenties kunnen aantonen dat ik deze meer dan voldoende beheers.

Ik heb *professioneel handelen* aangetoond, onder andere toen ik er achter kwam dat de foutieve data, die zichtbaar was in de GUI, niet door mij was ontstaan. Dit probleem zou ik nooit kunnen verhelpen maar moest in ATP worden opgelost. Hierover zijn een aantal gesprekken geweest met de makers van ATP. In deze gesprekken heb ik feedback gegeven op het ATP en hoe deze verbeterd kon worden.

Daarnaast heb ik ook *methodisch handelen* aangetoond door meer personen te interviewen dan alleen mijn opdrachtgever. Na een aantal gesprekken met de testers en het hoofd van de test afdeling was het al snel duidelijk dat ze meer verwachtten. Hierdoor is het project een bruikbare oplossing gaan bieden die ook gebruikt gaat worden door de testers en niet na het afstuderen in een stoffige hoek wordt gegooid en waar niet meer naar wordt gekeken.

De competenties *ontwerpen* en *realiseren* zijn flink aangetoond. De opdrachtgever is zeer tevreden met het eindresultaat en het is, met hier en daar een kleine verandering, bijna perfect met wat er in de GUI Mockups staat. Hierbij is het project gerealiseerd in ASP.NET waar ik aan het begin van de afstudeerstage weinig kennis van had.

10.6 Eindresultaat

Over het eindresultaat van het afstudeerproject ben ik erg tevreden. Het dashboard heeft alle functionaliteit waar ik op gehoopt had en werkt exact zoals ik dat verwacht van een hedendaagse website.

Het dashboard geeft alle informatie van een test run weer op een manier die niet snel verveelt. Door het gebruik van kleuren en iconen is de site ook niet zo saai als die met alleen tekst zou zijn. Dat de site daarbij dan ook nog eens op elk soort device leesbaar is vind ik zelf een grote plus. Ik ben daarom ook zeer tevreden over de GUI van het dashboard

11 Literatuurlijst

ali62b. (2010, 1 25). *difference between model view controller (mvc) and asp.net web application ?* Opgeroepen op 3 18, 2013, van The Official Microsoft ASP.NET Forums: http://forums.asp.net/t/1518156.aspx/1

Block, G. (2010, 8 17). *design patterns - What are MVP and MVC and what is the difference?* Opgeroepen op 3 19, 2013, van Stack Overflow: http://stackoverflow.com/questions/2056/what-are-mvp-and-mvc-and-what-is-the-difference/101561#101561

Boodhoo, J.-P. (2006, 8 -). *Design Patterns: Model View Presenter*. Opgehaald van MSDN Magazine: http://msdn.microsoft.com/en-us/magazine/cc188690.aspx

Chateaux Software. (2013, 4 24). *Working with SQL Server Schema Bound Views*. Opgehaald van Chateaux Software: http://www.chatsoft.com/aboutus/articles/schema-bound-views.asp

Ext.NET. (-, - -). -. Opgehaald van Ext.NET: http://www.ext.net/

Fassett, W. (2011, 4 26). *Why does the ASP.Net Web Forms model "suck"?* Opgeroepen op 3 12, 2013, van Stack Overflow: http://stackoverflow.com/questions/46031/why-does-the-asp-net-web-forms-model-suck/234385#234385

Fink's, G. (20009, 1 16). *Deciding When to Use ASP.NET MVC Framework*. Opgeroepen op 3 12, 2013, van Gil Fink's Blog: http://beta.blogs.microsoft.co.il/blogs/gilf/archive/2009/01/16/deciding-when-to-use-asp-net-mvc-framework.aspx

Garrett, J. J. (2005, 2 18). *Ajax: A New Approach to Web Applications*. Opgehaald van Adaptive Path: http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications

Haack, P. (2008, 1 19). *Everything You Wanted To Know About MVC and MVP But Were Afraid To Ask*. Opgeroepen op 3 1, 2013, van Haacked: http://haacked.com/archive/2008/06/16/everything-you-wanted-to-know-about-mvc-and-mvp-but.aspx

James, M. (-, - -). *Scrum Cheat Sheet from DZone Refcardz*. Opgehaald van Dzone: http://refcardz.dzone.com/refcardz/scrum#refcard-download-social-buttons-display

JuiceUI. (-, - -). *Supercharge ASP.NET Web Forms with jQuery UI*. Opgehaald van JuiceUI: http://juiceui.com/

Kojevnikov, A. (2008, 11 4). *unit testing - NUnit vs. MbUnit vs. MSTest vs. xUnit.net*. Opgehaald van Stack Overflow: http://stackoverflow.com/a/261156/1047155

MacCaw, A. (2011, 11 2011). *Asynchronous UIs - the future of web user interfaces*. Opgehaald van alexmaccaw: http://alexmaccaw.com/posts/async_ui

Maclean, M. (2013, 4 12). *D3 Tips and Tricks*. Opgehaald van LeanPub: https://leanpub.com/D3-Tips-and-Tricks

Marcotte, E. (2010, 5 25). *Responsive Web Design · An A List Apart Article*. Opgehaald van alistapart: http://alistapart.com/article/responsive-web-design

Math Is Fun. (-, - -). *Bar Graphs*. Opgehaald van Math Is Fun: http://www.mathsisfun.com/data/bar-graphs.html

Math Is Fun. (-, - -). *Histograms*. Opgehaald van Math Is Fun: http://www.mathsisfun.com/data/histograms.html

Math Is Fun. (-, - -). *Line Graphs*. Opgehaald van Math Is Fun: http://www.mathsisfun.com/data/line-graphs.html

Math Is Fun. (-, - -). *Pie Chart*. Opgehaald van Math Is Fun: http://www.mathsisfun.com/data/pie-charts.html

McCafferty, B. (2007, 10 16). *Model View Presenter with ASP.NET*. Opgeroepen op 3 18, 2013, van CodeProject: http://www.codeproject.com/Articles/14642/Model-View-Presenter-with-ASP-NET

Microsoft. (-, - -). *MVC*. Opgeroepen op 3 11, 2013, van The Official Microsoft ASP.NET Site: http://www.asp.net/mvc

Microsoft. (-, - -). *Web Forms*. Opgeroepen op 03 11, 2013, van The Official Microsoft ASP.NET Site: http://www.asp.net/web-forms

Microsoft. (-, - -). *Web Pages*. Opgeroepen op 3 11, 2013, van The Official Microsoft ASP.NET Site: http://www.asp.net/web-pages

Microsoft N-Layer. (2013, 4 10). *Chapter 5: Layered Application Guidelines*. Opgehaald van MSDN: http://msdn.microsoft.com/en-us/library/ee658109.aspx

NS, S. (2013, 3 20). *ASP.NET MVC 4*. Opgeroepen op 11 19, 2012, van CodeProject: http://www.codeproject.com/Articles/470107/ASP-NET-MVC-4-Part-1-Introduction

Selenium. (-, - -). *Introduction — Selenium Documentation*. Opgeroepen op 3 29, 2013, van SeleniumHQ: http://docs.seleniumhq.org/docs/01_introducing_selenium.jsp#

Selenium. (-, - -). *Selenium IDE Plugins*. Opgeroepen op 3 29, 2013, van SeleniumHQ: http://docs.seleniumhq.org/projects/ide/

Selenium. (-, - -). *Selenium Projects*. Opgehaald van SeleniumHQ: http://docs.seleniumhq.org/projects/

SeleniumHQ. (-, - -). *Selenium IDE Plugins*. Opgehaald van SeleniumHQ: http://docs.seleniumhq.org/projects/ide/

SeleniumHQ. (-, - -). *Selenium WebDriver*. Opgehaald van SeleniumHQ: http://docs.seleniumhq.org/projects/webdriver/
Tester, T. A. (2008, 8 8). *Selenium Training*. Opgehaald van The Automated Tester: http://www.theautomatedtester.co.uk/selenium_training.htm
Vaibhav. (2008, 12 16). *Choosing Between WebForms and MVC*. Opgeroepen op 3 13, 2013, van Habitually Good: http://blog.gadodia.net/choosing-between-webforms-and-mvc/
w3schools. (-, - -). *ASP.NET MVC Introduction*. Opgehaald van w3schools: http://www.w3schools.com/aspnet/mvc_intro.asp
w3schools. (-, - -). *ASP.NET Web Forms Tutorial*. Opgehaald van w3schools: http://www.w3schools.com/aspnet/aspnet_intro.asp
w3schools. (-, - -). *ASP.NET Web Pages Tutorial*. Opgehaald van w3schools: http://www.w3schools.com/aspnet/webpages_intro.asp
Wertheim, D. (2011, 7 20). *Why the ExpectedExceptionAttribute sucks!* Opgeroepen op 3 26, 2013, van Daniel Wertheim: http://daniel.wertheim.se/2011/07/20/why-the-expectedexceptionattribute-sucks/
Wikipedia. (2013, 3 2). *Chart*. Opgeroepen op 3 25, 2013, van Wikipedia: http://en.wikipedia.org/w/index.php?title=Chart&oldid=541719896
Wikipedia. (2013, 2 21). *Wikipedia*. Opgehaald van List of charting software: https://en.wikipedia.org/wiki/List_of_charting_software
Wilson, B. (2012, 11 14). *xUnit.net - Unit testing framework for C# and .NET (a successor to NUnit) - Home*. Opgehaald van xUnit.net: http://xunit.codeplex.com/wikipage?title=Comparisons&ProjectName=xunit

12 Bijlagen

Project: Hathor / initiation Phase

(Project Initiation Document)

Project code: 13

Date completed: 25/3/2013
Auteur: Nick Jole

Version: 1.1
Status: Final

Document ID: 001
Filename: Project Initiation Document v1.1.docx

Document History

Revisions

Version	Status	Date	Changes
0.1	Concept	20-02-2013	Started
0.2	Concept	21-2-2013	Project management / Language correction
0.3	Concept	22-2-2013	Process feedback
1.0	Final	1-3-2013	Process feedback / finalized
1.1	Final	19-3-2013	Process feedback for approval

Approval

This documents needs to be approved by the following persons:

Version	Date of approval	Name	Function	paraph
1.0		Ralph van Roosmalen	Supervisor	
1.0		René van der Heijden	Project Assurance	

Distribution

This document has been distributed to:

Version	Send Date	Name	Function
0.2	21-02-2013	Ralph van Roosmalen / Niels Tanis	Supervisor
0.3	22-02-2013	René van der Heijden	Project Assurance
1.1	25-03-2013	Ralph van Roosmalen / Niels Tanis	Supervisor
1.1	25-03-2013	René van der Heijden	Project Assurance

Executive Summary

Goal of this document

The goal of this document is to define the project: this is the base for managing the project and the assessment of the success to complete the project.
The two most important rules for the use of this document are:

- To be sure that the project has a healthy base before the steering committee is invited to commit to the project
- To serve as the base document that enables the steering committee and project manager to monitor and test the progress of the project. And also to judge the questions around the validity of the project during its execution.

Motivation

RES Software develops software according to agile principles. One of the pillars of agile development is automation of tests. To make those automatic tests RES Software has developed an Automated Test Platform (ATP). With ATP the developers and testers can build tests effectively and fast. The tests can be executed on any possible configuration by using virtual machines.
The next step is the automation of reporting the results of the automated tests. Currently, the results of the tests are being processed manually. The results are then emailed to the owner of the tests and other stakeholders.
The current way of working is very labor intensive and should be replaced with an automatic system. This system should show the test results that the development teams and stakeholders would like to see. The data should be made visible in such a way that any team member can see it. (Whether in the Netherlands, US or Romania.)

Global approach

The global approach of this project is.
The useable ASP.NET frameworks should be mapped and the framework with the best results should be implemented. For this the following activities will be done:

- Map all the ASP.NET frameworks
- Investigate which ASP.NET frameworks should be used
- Make a software design
- Make tests and Proof of Concept of the design
- Realize the design in a development environment
- Deliver the software to the organization

Overall costs and time

The overall time of the project will be 5 months, starting 11-02-2013 and ending 21-06-2013. The budget of the project will be 3500 euro, consisting of the wage of the intern and the operational costs of the internships laptop.

Table of contents

Introduction

Goal of this document

This document has been prepared to provide all relevant basic information and principles of the project so that the project can be controlled correctly. The goal of this document is to define the project and to serve as a base for the management of the project and the assessment of the success of the project.

This Project Initiation Document (PID) treats the next fundamental aspects of the projects:

- What do we aim to achieve with the project?
- Why is it important to achieve these objectives?
- Who is involved in managing the project and what are their roles and responsibilities?
- How and when will the measures that are discussed in this PID be realized?

This document will be used:

- To be sure that the project has a healthy base before the steering committee is invited to commit to the project
- To serve as the base document that enables the steering committee and project manager to monitor and test the progress of the project. And also to judge the questions around the validity of the project during its execution.

Background

Context & Motivation

Automated Test Platform (ATP) has been build and is being used by developers and tests. The next step is to have the results shown in an easy to use and nice looking dashboard. This dashboard should be in ASP.NET.

The goal of the project is to find a framework that makes it possible to build an open, maintainable, easily extendable dashboard. It should also be possible to add extra reporting information to the dashboard in the near future, like the burndown graph of a development team.

The Current Situation

Currently there are dashboards all through the department. These dashboards show the scrum progress, some basic test results of the current automated test platforms and the build state of their projects. Additionally, also support tickets that are open, longest open tickets and traffic information during the afternoon.

These dashboards are programmed in VB6 and are getting outdated. RES Software is expanding the department to other locations (US, Romania), which adds another facet to the company. This means that everything inside the department should also be accessible from other locations. It is not possible to use the current dashboard outside the development department in the Netherlands because of technical limitations. The current dashboard has no functionality to be shown on other locations than the building in which it is hosted.

Project Definition

Goals of the project

The goal of the project is to develop a new dashboard for the ATP results. The dashboard should also be easily extendable for further implementation of features. This has to be more automatic than the current dashboard, be able to run on their general company servers and provide more detail of the ATP results. The exact requirements will be identified during the research for RES Software and will be written out in the requirements document.

This project is also the graduation project of Nick Jole, and therefore regulations of the Fontys University of Applied Science must also be complied with to ensure this project will lead to a successful graduation.

RES Software main development language is .NET. Therefore they have a strong preference to make the dashboard in ASP.NET. This is because they want the new dashboard to be accessible from the web, so that teams from all around the globe can see the dashboard.

Advantages

The advantages of this project are that RES Software does the first step into using new technology for their internal programs. By this project they can find out all of the advantages of the new stacks that are out in the field. This information can be used to select the stack that they can use to develop new Web software. Furthermore, they will also have a new dashboard that can be used from multiple locations.

End Result

The end results will include:

- ATP Dashboard general research document;
- ATP Dashboard for RES Software research document;
- Test results of acceptance tests;
- Installable application for the web server;
- Source code of the application;
    - Including unit tests
    - Build environment

There will also be an answer on the following Research Questions:

- Which ASP.NET stack should be used to realize the dashboard looking at its extendibility, testability and what is best for the future?
- How extendable are the ASP.NET stacks?
- Which stack has the best toolset to present the data needed for the dashboard?
- How can the stack be tested?

This end product also provides a dashboard made in the chosen ASP.NET stack. But its exact specifications of what the end product must be capable of will be defined at a later stage in the project.

Chosen solution or approach

This project will be divided in the 3 main phases: Initiation Phase, Research and Solution Phase, and Implementation Phase. Each of the main phases has its own sub phases defined by the Ten Steps Plan. All phases will be executed in the agile methodology used by RES Software.

Initiation Phase

In this phase the initiation of the project is started. The whole phase is situated around the startup of the project. This includes chats with personnel of RES Software about their vision on the project and an identification of potential problems. This document is the result of the Initiation phase with all the relevant data contained inside it.

Research and Solution Phase
In this phase the actual research of the project is done. The main sub-phase is the Depth Research. In the Depth Research there will be searched to answer the questions listed above. Because when the questions are sufficiently answered, the research part is done.

The approach on answering the research questions will be described in the next sections.

The first part of the research will be about the display of the dashboard. Therefore the demands of the company and its workers should be researched. The demands will be researched through interviews with the heads of each department. But also the normal workers will be interviewed if necessary. The demands will be the guidelines to research for the data that should be displayed in the dashboard. The findings will be written out in a User Requirements Specification (URS).

The data will be displayed in a GUI which should present the data in the best way possible. Therefore the GUI will need some features that will present the data in the best way possible. These features need to be researched in a literature research on the internet, and a survey around the work floor on the way in which they like to see the data. The findings of this research will be written out in an overview on how to display the data.

If the requirements are written out and the overview with all the ways to display data is complete, the next step will be to have a brainstorm about how to combine and display all the data on the dashboard. This brainstorm will result in multiple GUI sketches that will be presented to the heads of each department. Eventually, a final GUI sketch will be made.

When the GUI sketch is complete there will be a thorough search on what stacks there are for Asp.net. This will then result in an overview of stacks that are available for ASP.NET.

Now that there is an overview of all the available stacks, the search of the best stack can begin. To find the best stack, 3 things need to be looked at: extensibility, ease of work and the speed of work.

For the extensibility part there will be a literature research on how each stack can produce the display of data found by the previous research. This will then result in a matrix of frameworks that can be used by each stack.

For the ease and speed of working in a stack part there will be some interviews with workers that worked with the stacks. Based on the information at hand, a simple model will be made with its corresponding unit tests. Then the model will be used in each stack to make Proof Of Concepts (POC's). This will then serve as a base to compare the stacks to each other.

The conclusion of the research will then be formed. The conclusion will hold several solutions to the problem. These solutions will then be presented to the company, which then chooses the solution that works best for them. This will then result in a recommendation that will be written out in a report.

Then a Thesis will be created to include any knowledge obtained during the length of the project, and also a final presentation will be made to be presented for the jury.

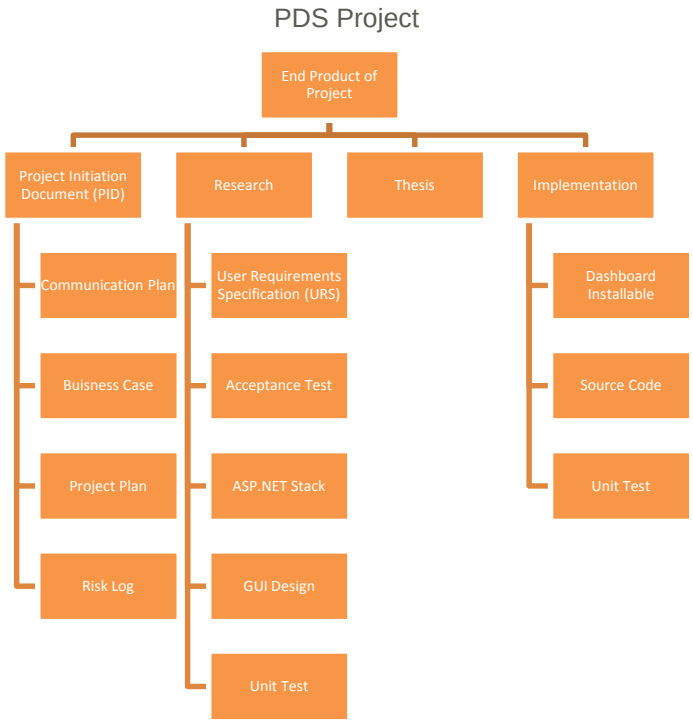
Implementation Phase
In this phase it is all about the realization of the product. The product will be designed, created and tested inside the chosen stack and then will be written out in the documentation. The documentation will contain a User Requirements Specification and a Software design.

Scope of the project

The scope of the project is a dashboard that must be researched and realized in the 85 working days set out by the regulations of the University.

During the project the target departments will be Development, Engineering and Testing. These groups benefit most of the project. They will also be my main source of information on what they want to see inside the project. But all other groups inside RES Software will see the project and can be asked for their view on the project if needed.

Product Breakdown (PB) on project level



Project budget

The budget of the project is 3500 euro, but can be increased if really necessary.
For a detailed breakdown of the project costs, please refer to Appendix B business case.

Required Resources and Dependencies

To bring the project to a good end the next resources are required:

- A laptop / desktop and needed infrastructure;
- Explanation of programs that are needed;
- Microsoft office Word;
- Microsoft office Excel;
- Microsoft office Visio;
- Microsoft office Project;
- Visual Studio;
- Resharper 7;
- ATP Test account;
- VMware;
- Microsoft SQL Server;
- Internet access;
- Email access;
- Telephone;
- Bureau accessories.

Exclusions

This project will not cover:

- Any electrical or mechanical installation;
- Any help given to colleagues;
- Any non-related work.

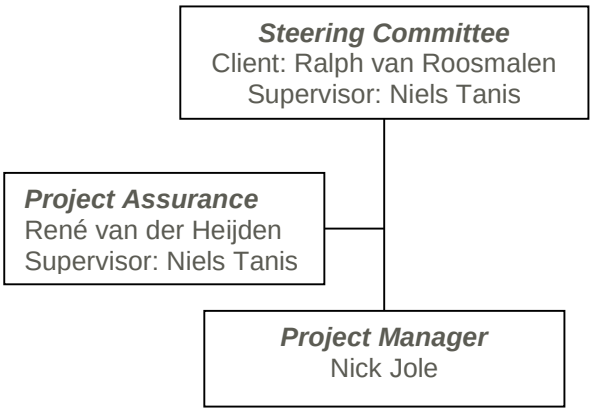
Preconditions

The preconditions of the project are that any documents sent for feedback will be responded to within a week. If the person can't reply to mail and give the requested feedback, an acknowledgement will be sent as soon as possible to confirm a new date of reply.

Assumptions

RES Software gives sufficient guidance / support in the time that the Internship runs.

Project Organization Structure



Project Manager

Role description

The Project Manager is the intern and will be carrying out the project. His task will involve the following: conducting research, writing a PID, writing an internship report, and the execution and implementation of the product.

Project-related tasks

- Conducts the research;
- Makes a technical design of the desired end result;
- Realizes and implements the product.

Specific responsibilities

- Writes the Project Initiation Document (PID)
- Sends the PID to the Client and Project Assurance and processes the feedback of the documents to a final version
- Writes the Thesis and asks on a regular basis to provide feedback on the Thesis.
- Sends the Thesis to the Client and the Project Assurance and processes the feedback given in the document to a final version.

Project Assurance

Role description

The project assurance should monitor progress and coordination between POP and the Internship. In addition, the project assurance guides the client in making the internship report and gives feedback on it.

Project-related tasks

The project assurance isn't involved in the project.

Specific responsibilities

- Makes after receipt of the PID an appointment for an initial interview with the Intern and the Client;
- Discusses and provides feedback to the PID;
- Visits during the internship period the company twice and discusses the progress to date on base of the delivered products;
- Keeps track of the internship based on the logs / blogs and looks it the student gets the chance to realize his competences.
- Supervises the student in writing and also gives feedback on his internship report and professional products.
- Provides feedback to the Internship Coordinator about the company, the assignment and the professional product.

Client

Role description

The client approves the documents of the intern over the duration of the internship. He also gives comments on made products.

Project-related tasks

- Discuss with Project Manager about the state of the project;
- Give feedback over the design and how to improve it;
- Give feedback on intermediates;
- Give feedback on the end concept Thesis.

Specific responsibilities

- Receives the intern at the start of the internship and introduces him / her in the company;
- Ensures that the work performed is according to the objective of the internship;
- Is an expert on the subject of the product and helps the intern to get an as good as possible result, he/she supervises the intern also in the preparation of the PID;
- Is present at the company visits and discusses with the project assurance and the intern the social and professionally operation;
- Consults with the student and / or project assurance if necessary;
- Signs the Thesis for seen;
- Fills the business review and signs the filled in day justification form, does the assessment interview at the end of the internship with the intern and couples the business review back to the project assurance.

Supervisor

Role description

The supervisor helps with the solving of project related problems the intern may have over the duration of the internship.

Project-related tasks

- Discuss with Project Manager about the state of the project;
- Give feedback over the design and how to improve it;
- Give feedback on intermediates;
- Give feedback on the end concept Thesis.

Specific responsibilities

- Receives the intern at the start of the internship and introduces him / her in the company;
- Ensures that the work performed is according to the objective of the internship;
- Is an expert on the subject of the product and helps the intern to get an as good as possible result, he/she supervises the intern also in the preparation of the PID;
- Is present at the company visits and discusses with the project assurance and the intern the social and professionally operation;
- Consults with the student and / or project assurance if necessary;
- Signs the Thesis for seen;
- Completes the business review and signs the completed day justification form, does the assessment interview at the end of the internship with the intern and couples the business review back to the project assurance.

Project Management

Reporting

Party: / Rapport:	Client Ralph van Roosmalen	Project Assurance: René van der Heijden	Project manager: Nick Jole	Supervisor: Niels Tanis
PID	O, A, D, G	O, A, D, G		O, A, D, G
Progress Report	I	I		I
Thesis	O, A, D, G	O, A, D, G		O, A, D, G

Legend:

O	Draft	**A**	Advise	**I**	Received for information
T	Test	**D**	Distribute/archive	**G**	Approve

Progress Monitoring

Consultation	Present	Frequency	point of time	Goal	Subject	Notes
Scrum Backlog Meeting	Niels Tanis, Nick Jole	Once every two weeks	Mon 8.00-10.00	Make subprojects concrete	Progress, Interfaces, Open Issues, Risks	Project Manager
Sprint demo meeting	Niels Tanis, Ralph van Roosmalen, Nick Jole	Once every two weeks	To be defined	Review progress of product and give input	Progress	Project Manager
Phase final assessment	Niels Tanis, Ralph van Roosmalen, Nick Jole	After Each Phase	To be defined	Review last phase Approve next phase	See if all the products are delivered.	Project Manager
Scrum Meeting	R&D, Nick Jole	Once every day	9:30-9:45	Goals	See if the plan is kept and if there are any major issues	Project Manager

Progress Report

Every day there is a scrum meeting with the Automation Manager Development team. The notes of these meetings are written down on a daily basis. The collection of these notes will form the progress report and will be sent by email on the first working day of each week, before 10:00.

Risk Management

See appendix D risk log for a detailed view of the risk management.

Quality Control

Product Quality

The product quality is described below per Phase and in relation with the project.

- Initiation Phase:
  A PID with its appendixes the project dependencies, costs, risks and a project plan.
- Research and Solution Phase:
  A report with frameworks, stacks, and tests aimed at making a proper and well-working dashboard which will then be selected from for implementation.
  Also a Thesis that includes any knowledge obtained during the length of the project.
  And also a final presentation will be made to be presented to the jury.
- Implementation Phase:
  A technical document containing all technical documentation made in the research and implementation phase.
  A working and fully tested web application that can run on the servers.

Process Quality

The process quality is assured by the following:

- Weekly sending of a logbook to the project assurance with details of the daily progress.
- Daily scrum meeting.

Appendix A: Communication Plan

This communication plan appoints all parties that have an interest in this project (may it be positive or negative) and the way in which they are connected to the project and which communication forms are used. It involves parties and communication outside the formal project management as described in the PID.

Communication Plan

Appendix A Project Initiation Document

Document History

Revisions

Version	Status	Date	Changes
0.1	Concept	19-02-2013	Started
0.2	Concept	22-02-2013	Updated Stakeholders
1.0	Final	01-03-2013	Finalized
1.1	Final	19-3-2013	Process feedback for approval

Approval

This documents needs to be approved by the following persons:

Version	Date of approval	Name	Function	paraph
1.0		Ralph van Roosmalen	Supervisor	
1.0		René van der Heijden	Project Assurance	

Distribution

This document has been distributed to:

Version	Send Date	Name	Function
0.1	21-02-2013	Ralph van Roosmalen / Niels Tanis	Supervisor
0.2	22-02-2013	René van der Heijden	Project Assurance
1.1	25-03-2013	Ralph van Roosmalen / Niels Tanis	Supervisor
1.1	25-03-2013	René van der Heijden	Project Assurance

Introduction

This Communication Plan appoints all parties that have an interest in this project (may it be positive or negative) and the way in which they are connected to the project and which communication forms are used. It involves parties and communication outside the formal project management as described in the PID.

Stakeholders of the project

Who	On behalf of	Interest	Communication form (*)
Niels Tanis	Development	Looks for new ways of programming.	Consult, Inform, Direct
Dennis Smits	Development	Looks for new ways of programming. Seeing results of his tests for the team.	Accept, Consult.
Ralph van Roosmalen	RES Software	Insight into test results of ATP	Accept, Inform, Direct.
Bart Withaar	Testing	Insight into test results of ATP	Consult
Daan van Osch	Engineering	Insight into test results of ATP	Consult
Bram van den Berg	Testing	Seeing results of his tests for the team	Consult

Communication Channels

From	To	Information	Medium	Frequency of data
Intern	Company Supervisor	Questions, Consult	Email, Talk	No structural way
Company Supervisor	Intern	Feedback	Email, Talk	After PID and Execution
Intern	Company Supervisor, Project Assurance	Intermediates	Email, Talk	After each step in the internship process
Intern	Company Supervisor, Project Assurance	PID	Email	1 times concept 1 times final on 1-3-2013
Intern	Company Supervisor	URS	Hyperdrive	No structural way Final on 12-3-2013
Intern	Company Supervisor	GUI	Hyperdrive	No structural way Final on 22-3-2013
Intern	Company Supervisor	Research Document	Hyperdrive	No structural way Final on 5-4-2013
Intern	Company Supervisor	Software Design	Hyperdrive	No structural way Final on 19-4-2013
Intern	Company Supervisor	Tests	Hyperdrive	No structural way Final on 3-5-2013
Intern	Company Supervisor	Demo	Hyperdrive	Every 2 weeks. Started after tests are complete.
Intern	Company Supervisor, Project Assurance	Concept and final Thesis	Email	3 times concept 1 times final on 7-6-2013

Communication with school

From	To	Information	Send Date
Intern	Project Assurance	logbook	Every week Monday.
Intern	Project Assurance	PID	1 times concept 1 times final on 1-3-2013
Intern	Project Assurance	Concept and final Thesis	3 times concept (first before 17-4-2013) 1 times final on 7-6-2013
Intern	Project Assurance	Questions of course internship	No structural way

Information of involved persons

Company Supervisor

Name: Niels Tanis
E-mail address: N.Tanis@ressoftware.com
Telephone Number: +31 (73) 6 22 89 17 (Work)
+31 6 37 33 61 82 (Mobile)

Client

Name: Ralph van Roosmalen
E-mail address: R.vanRoosmalen@ressoftware.com
Telephone Number: +31 (73) 6 22 89 58 (Work)
+31 6 50 81 64 25 (Mobile)

Project Assurance

Name: René M.J.W. van der Heijden
E-mail address: r.vanderheijden@fontys.nl
Telephone Number: 08 85 07 10 70 (Work)

Intern

Name: Nick Jole
E-mail address: N.Jole@student.fontys.nl
Telephone Number: +31 6 13 36 39 81

Appendix B: Business Case

This document contains the considerations to start the current project. It forms the justification of the project and for this reason will be reviewed by the Steering Committee. The substantiation of the project will be evaluated based on the Business Case. It describes the expected investment costs against the expected befits, savings, and project risks. The cost-benefit analysis is substantiated by an investment proposal. The business case starts with a description of the considerations made of the project.

Business Case

Appendix B - Project Initiation Document

Document History

Revisions

Version	Status	Date	Changes
0.1	Concept	19-02-2013	Started
0.2	Concept	22-02-2013	Process feedback
1.0	Final	01-03-2013	Finalized
1.1	Final	19-3-2013	Process feedback for approval

Approval

This documents needs to be approved by the following persons:

Version	Date of approval	Name	Function	paraph
1.0		Ralph van Roosmalen		
1.0		René van der Heijden		

Distribution

This document has been distributed to:

Version	Send Date	Name	Function
0.1	21-02-2013	Ralph van Roosmalen / Niels Tanis	Supervisor
0.2	22-02-2013	René van der Heijden	Project Assurance
1.1	25-03-2013	Ralph van Roosmalen / Niels Tanis	Supervisor
1.1	25-03-2013	René van der Heijden	Project Assurance

Introduction

This document contains the considerations to start the current project. It forms the justification of the project and for this reason will be reviewed by the Steering Committee. The substantiation of the project will be evaluated based on the Business Case. It describes the expected investment costs against the expected benefits, savings, and project risks. The cost-benefit analysis is substantiated by an investment proposal. The business case starts with a description of the considerations made about the project.

Reasons

At RES Software they develop software according to agile principles. One of the pillars of agile development for RES Software is the automation of tests. To make those automatic tests RES Software has developed an Automated Test Platform (ATP). With the environment of ATP the developers can build tests effectively and fast. The automatic tests then produce allot of results which gets written to a database.
The next step is the automation of reporting the results of the automated tests. Currently the results of the tests are being processed manually. The results are then emailed to the respective owners of the tests.
The current way of working is very labor intensive and should be replaced with an automatic system. This system should show the test results that the development teams and stakeholders would like to see. This data should be made visible in such a way that any team member can see it. (Whether in the Netherlands, US or Romania.)

Considered Alternatives

The only alternative is to keep on doing it by hand. This isn't really an option, because it already takes 30 minutes to get the data for 2 test runs. And the time needed will increase by every new test run.

Benefits

- Less time needed to read out the ATP
- Overview of all results in 1 screen
- Detailed breakdown in just 1 click

Risks

Risk	Measure	Chance	Impact
Server with source code crashes	Make backup on other server	Low	High
Production Server Crashes	Backups	Medium	Medium
ATP Stops writing logs to the database	Contact ATP team why the results aren't written to the database.	Low	Medium
ATP writes incorrect data to the database	Manual data check every month.	Medium	Medium

Time and Costs

The duration of the project is 5 months. In these months the costs of the project will be defined by the wage of the intern and the operational costs of the machine.

Proposal of Investment

Assumptions

The following calculation is based on the following assumptions:
rate: 8%

Investment Calculation

	Year 0	Year 1	Year 2	Year 3	Year 4	Year 5
Costs						
Project Costs	2.500	-	-	-	-	-
Operational Costs	100	200	200	200	200	200
Other costs	500	-	-	-	-	-
Total Costs	**3.100**	**200**	**200**	**200**	**200**	**200**

revenues						
Direct revenues / savings	-	-	-	-	-	-
Indirect revenues	1.500	3.000	3.200	3.400	3.600	3.800
Other revenues	-	-	-	-	-	-
Total revenues	**1.500**	**3.000**	**3.200**	**3.400**	**3.600**	**3.800**

Cash flow	1.600-	2.800	3.000	3.200	3.400	3.600
Discount factor by 8%	100%	93%	86%	79%	74%	68%
Discounted cash flow	1.600-	2.593	2.572	2.540	2.499	2.450

Return on Investment (ROI)	14.400
Net present value (NVP)	**11.054**

Appendix C: Project plan (MS Project)

This plan describes the project set out against the time. It shows when products have to be delivered and when the phases are done.

	Task Name	Duration	Start	Finish
1	**Initiation Phase**	**15 days**	**Mon 11-2-13**	**Fri 1-3-13**
2	External Oriëntation	2 days	Tue 12-2-13	Wed 13-2-13
3	Intake Interview	1 day	Thu 14-2-13	Thu 14-2-13
4	orienting interviews	1 day	Fri 15-2-13	Fri 15-2-13
5	Analyse	2 wks	Mon 18-2-13	Fri 1-3-13
6	PID	1 wk	Mon 18-2-13	Fri 22-2-13
7	Feedback / contracting	1 wk	Mon 25-2-13	Fri 1-3-13
8	Process Feedback PID	1 wk	Mon 25-2-13	Fri 1-3-13
9	PID	0 days	Fri 1-3-13	Fri 1-3-13
10	**Research Phase**	**50 days**	**Mon 4-3-13**	**Fri 10-5-13**
11	Plan Work and organize project	2 days	Mon 4-3-13	Tue 5-3-13
12	Depth Research	5 wks	Wed 6-3-13	Tue 9-4-13
13	URS	1 wk	Wed 6-3-13	Tue 12-3-13
14	URS	0 days	Tue 12-3-13	Tue 12-3-13
15	GUI Research	1 wk	Mon 11-3-13	Fri 15-3-13
16	GUI Sketch	1 wk	Mon 18-3-13	Fri 22-3-13
17	GUI	0 days	Fri 22-3-13	Fri 22-3-13
18	Find Stacks	1 wk	Wed 20-3-13	Tue 26-3-13
19	Extend research	3 days	Wed 20-3-13	Fri 22-3-13
20	Speed research	2 wks	Mon 25-3-13	Fri 5-4-13
21	Stack	0 days	Fri 5-4-13	Fri 5-4-13
22	Solution Plan	4 wks	Mon 8-4-13	Fri 3-5-13
23	**First Technical Sprint**	**10 days**	**Mon 8-4-13**	**Fri 19-4-13**
24	Software Design	1 wk	Mon 8-4-13	Fri 12-4-13
25	First Version Of Dashboard	1 wk	Mon 15-4-13	Fri 19-4-13
26	**Second Technical Sprint**	**5 days**	**Mon 22-4-13**	**Fri 26-4-13**
27	First Version of Detail Screens	1 wk	Mon 22-4-13	Fri 26-4-13
28	**Third Technical Sprint**	**5 days**	**Mon 29-4-13**	**Fri 3-5-13**
29	Add profile functionallity	1 wk	Mon 29-4-13	Fri 3-5-13
30	Thesis	10 wks	Mon 1-4-13	Fri 7-6-13
31	Thesis	0 days	Fri 7-6-13	Fri 7-6-13
32	**Implementation Phase**	**35 days**	**Mon 6-5-13**	**Fri 21-6-13**
33	**Fourth Technical Sprint**	**5 days**	**Mon 6-5-13**	**Fri 10-5-13**
34	Lose scrum items that remain	5 days	Mon 6-5-13	Fri 10-5-13
35	**Documentary Sprint**	**2 wks**	**Mon 13-5-13**	**Fri 24-5-13**
36	Document Profile	1 wk	Mon 13-5-13	Fri 17-5-13
37	Document Overvie\	1 wk	Tue 14-5-13	Mon 20-5-13
38	Document Detail Screens	1 wk	Mon 20-5-13	Fri 24-5-13
39	**Prepare for deployment**	**20 days**	**Mon 27-5-13**	**Fri 21-6-13**
40	Rounding	4 wks	Mon 27-5-13	Fri 21-6-13
41	Product	0 days	Fri 21-6-13	Fri 21-6-13

Appendix D: Risk log

This document contains all the risks and solutions of them. But also keeps track of when risks actually occur and what the solution to the risk then was.

Project: Hathor

Risk Log

ID	Cat.	Brief Description	Proximity	Chance	Impact	Overall	PRACT	Action(s)	State	Owner	Submitter	Identification Date	Date Updated
1	T/O/I	Team Foundation Server Crashes	always	L	H	😐	C	Consult Supervisor / Teacher	0	RES Software	Nick		
2	J/R	Research gets refused	25/02/2013	M	M	😐	C	Consult Supervisor / Teacher	-	Nick	Nick	19/02/2013	21/02/2013
3	J/R	PID gets refused	27/02/2013	M	M	😐	R	Write a solid PID	0	Nick	Nick		
4	T/O/I	Production Server Crashes	always	M	M	😐	R	Make backup of the server	0	RES Software	Nick		
5	T/O/I	ATP stops writing logs to the database	always	L	M	🙂	A		0	RES Software	Nick		
6	T/O/I	ATP writes incorrect data to the database	always	M	M	😐	A		0	RES Software	Nick		
7	T/O/I	Views don't get implemented	25/04/2013	M	H	☹	C	Consult ATP team		Nick	Nick	25/04/2013	25/04/2013
8													
9													
10													
11													
12													
13													
14													
15													
16													
17													
18													
19													
20													
21													
22													
23													
24													
25													
26													
27													
28													
29													
30													
31													
32													
33													
34													
35													
36													
37													
38													
39													

Project: Hathor

(Research Document)

Project code:	13
Date completed:	28/3/2013
Auteur:	Nick Jole
Version:	1.0
Status:	Final
Document ID:	6
Filename:	Research Document.docx

Contents

1 Problem Definition

1.1 Main Question

- Which ASP.NET stack should be used to realize the dashboard, looking at its extendibility, testability and future readiness?

1.2 Side Questions

- Which ASP.NET stacks are there?
  - How extendable are the ASP.NET stacks?
  - Can they be extended?
  - Are there extra frameworks that can be used?

- Which stack has the best toolset to present the data needed for the dashboard?
  - What types of charts are there?
  - Which type of chart displays the data best?
  - What type of framework can be used to display the charts?

- How can the stack be tested?
  - Which test toolkits can be used?
  - Can the stacks be easily tested?
  - What parts need to be tested?

2 Which ASP.NET Stacks are there?

2.1 How extendable are the ASP.NET stacks?

There are three main stacks (**Figure 1**) in ASP.NET: Web Pages, Web Forms, and MVC. A detailed description per stack is written below.

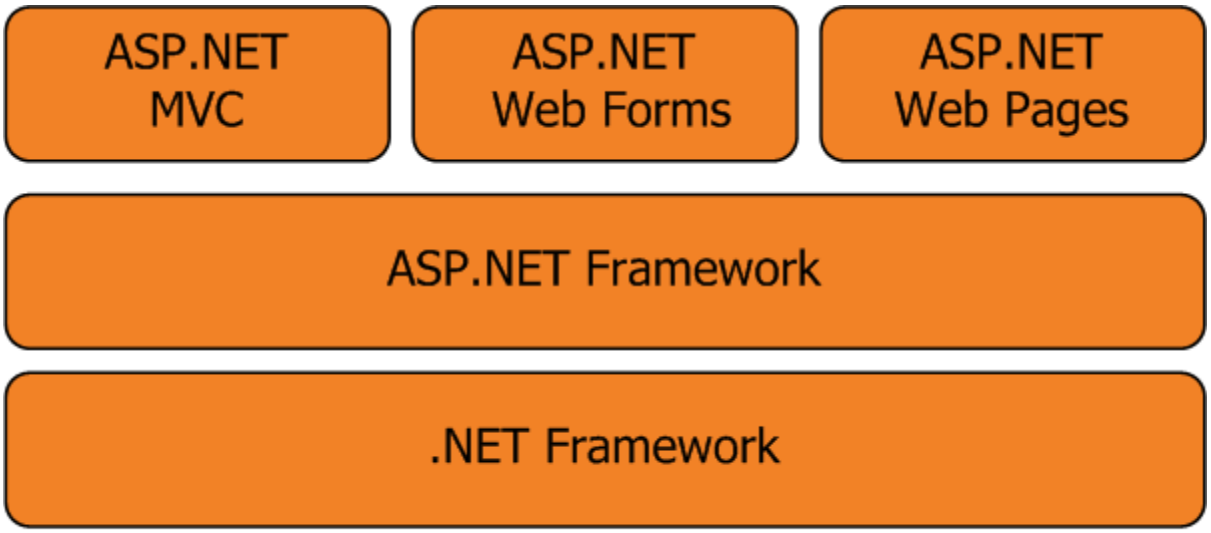


Figure 1: ASP.NET Stacks

Web Pages (Microsoft, -) (w3schools, -)
Web Pages is the first of the 3 stacks for creating ASP.NET web sites and web applications.

Figure 2: Web Pages Logo

Web Pages is the simplest stack out of the 3 and provides a simple and easy way to combine HTML, CSS, JavaScript, and server code.

The learning curve of this stack is rather low. Some basic knowledge of C# or VB and HTML / CSS is really a plus for Web Pages, but a hobbyist can learn this knowledge in a few days / weeks by watching some tutorials. Therefore it is perfect to start out with in ASP.NET and web development.

Web Pages runs with a special development tool from Microsoft called WebMatrix. WebMatrix is a lightweight development environment that is specially designed for Web Pages. It features almost the same toolset when a class or webpage is created in comparison with Visual Studio. Only it has no debugger, which makes error hunting a bit of a pain. And class files can only be created in special dedicated folders.

WebMatrix has some basic tools to quickly create, publish, and maintain a website. It has a database manager where the developer can add or delete databases. There are 3 types of database the developer can use: SQL Server, MySQL, and SQL Server CE. If there is a database, the tables and views can be added. The tables can also be populated by data if it's needed.

The connection to the database is also fairly easy. It only needs a database.open("<name>") in a script block on top of the view and the database connection is automatically made. The response gives a database that can be used to send SQL statements to. The return of those statements is a list of objects that can be used freely throughout the view of the site.

The project itself can be opened in Visual Studio, which gives the developer all possibilities that Visual Studio gives including the database manager of Visual Studio. It also gives a handy popup when the developer creates class files that hint on a missing map. The map is App_Code where custom class files can be placed if there are any. These classes can then be used in the view to make new functionality.

Because of the lack of the App_Code map at startup of the project and not being mentioned in the tutorials, on the main ASP.NET site a big hint is shown that it isn't normal to use class files in WebMatrix. Therefore it isn't

recommended to use this stack as an experienced programmer Object Oriented Programming (OOP). This is also underlined in multiple blogs (Vaibhav, 2008) where people only put web forms and MVC beside each other, never saying a word about web pages.

Some advantages listed in no particular order: (w3schools, -)

- It is easy to learn understand and use.
- It is built around the single web pages of ASP.NET.
- It is very similar to other languages like PHP and classic ASP
- It can do server scripting with either Visual Basic or C#.
- It gives the developer control of HTML, CSS, and JavaScript.

Web Forms (Microsoft, -)
Web Forms is the second of the 3 stacks for creating ASP.NET web sites and web applications.

Web Forms is based on the same event driven system that are used by Windows desktop applications. So anyone that ever made a desktop application will easily get accustomed to the way ASP.NET Web Forms works. There still isn't that much support for design patterns like Model-View-Controller or Model-View-Presenter, but there are possibilities to add those patterns to Web Forms.

Web Forms will take some more time to learn than Web Pages, because .NET has a large number of options and many components to get used to before a good application can be made in .NET. Also, the event driven structure needs to be understood and the same goes for the View State that can really get in the way.

One of the advantages of Web Forms is that it is seamlessly integrated in Visual Studio. It doesn't need an additional development tool and works quick and easy. By example: Web Forms gives a wizard to bind data sources to components. These components can be a grid view or a drop down box. It also has a large library with components that can be moved in the view using drag and drop. These components all have events that are wired up automatically to the server.

Another big difference between Web Forms and Web Pages is that most of the code is gone from the view. Every view has its own code behind file and designer file that respectively contains the code and design code. Therefore there is no longer a huge cluttered view file with code and events. Because of this separation the view is much cleaner and easier to read and maintain.

Web Forms is made from the perspective of productivity. The whole stack is built on how fast the developer can make a site and publish it. With its drag and drop functionality a developer can make a Prototype or PoC (Proof of Concept) in mere hours. This in turn has a downside and that is the control and performance of Web Forms.

The performance suffers because everything is generated on run time. Also the code behind files, where all the code goes, is coupled with the view. The code behind file can only be used by the view where it is derived from and thus can't be reused by other views. This causes some redundancy to the code behind files if there are parts of the site that almost do the same. Also, it sends the view state on every call to the server. This puts some load on the server, especially with a large view state.

The control is more or less done by the view state. The output of the html can't be changed unless the components are changed and the whole page always has a Form tag around it. The Form tag is something that most web developers who live by the html / css rules don't want to have around their whole page.

Strengths listed in no particular order: (Vaibhav, 2008) (w3schools, -)

- It is very good for RAD (Rapid Application Development).
- It has great designer support in visual studio.
- It has very rich control libraries.
- It has controls to easy get around with data-heavy applications.
  It uses about the same event driven model as windows forms which make web forms easy to pick up for developers used working with windows forms.

Weaknesses listed in no particular order: (Vaibhav, 2008)

- UI logic coupled with the code, and thus is hard to separate.
- Harder to unit test, so difficult to employ TDD.
- Heavy page sizes due to view state management.

MVC (Microsoft, -)
MVC is the third and last of the 3 stacks for creating ASP.NET web sites and web applications.

MVC is an abbreviation for Model View Controller which is widely used by developers in any object oriented programming language.

The MVC design pattern has 3 distinctive parts (**Figure 3**):
The Model (also called Business layer) represents the core of the application. The classes in the model are most of the time derived from a database, data warehouse or any other type of data storage structure.

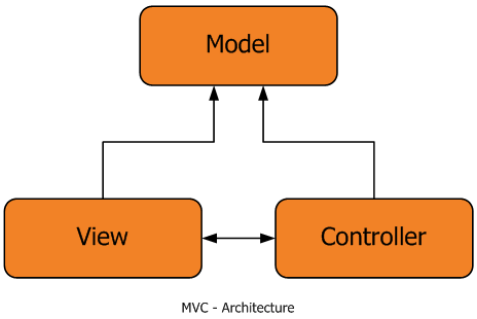


Figure 3: MVC Architecture

The View (also called Presentation or Display layer) is the visible part of the application. The screens are mostly derived from the models data and show the data of the model in the best way possible.

The Controller controls the flow of the application. It handles the user interaction with the application. It can send commands to update the models state and then notify the view that the model has been updated.

The MVC pattern has proven itself very useful in group development, because it separates the 3 logics from each other. There can be a group that realizes the model, a group that realizes the control, and a group that realizes the view of the application.

Because MVC separates all the parts of an application it makes testing and TDD (Test Driven Development) much easier. MVC doesn't need a webserver to test the back-end of the application, but still needs a webserver for front end testing. Unlike Web Forms and Web Pages, which need a webserver for both back-end and front end testing.

The power of MVC lies mainly in the control that the developer has on every part. Where Web Forms has its pre-defined controls, MVC has nothing but the asp.net methods, HTML, CSS, and JavaScript. This brings down the productivity and how fast there is a product to show. On the other hand, all developers experienced in HTML/CSS can put all their skills to work and show the product at the best of their abilities.

Because the HTML/CSS output of MVC can be modified to every need, technologies like AJAX (Asynchronous JavaScript and XML) (Garrett, 2005) can be used to get data dynamically from the server. By using such technologies the load on the servers will be considerably less than with web forms. MVC doesn't use View State, which further decreases the load on the servers. MVC also helps in getting used to techniques as RESTfull for URL routing and notation. This way it's easier to make sub sections like Account and Home. In this way the site has a clear structure, which helps the user to get around the site. And because the models and views are decoupled, there can be multiple views of the same model.

Some advantages listed in no particular order: (Vaibhav, 2008) (w3schools, -)
- It gives full control over HTML and CSS.
- It generates much more readable HTML in the browser.
- It separates the code and view nearly perfect.
- It is easier to test.
- It supports multiple view engines including third party engines
- It uses Restful by default.
- It has no View State.
- It has no cluttered view page.
- It works seamless with frameworks like JQuery.

Weaknesses listed in no particular order: (Vaibhav, 2008)
- Not event driven, so maybe difficult for people who know only Asp.Net Webforms to wrap their minds around it.
- Third party control library support is not that strong,
- No Viewstate (This is more a strength then a weakness)

2.2 Can they be extended?

All stacks can be extended with the libraries found in NuGet and extensions that are in general for Visual Studio and WebMatrix.

Web Pages

Because it is made for the starter or hobbyist that wants to start out in web development / ASP.NET, it only has few extensions. The main focus here lies on the templates that generate the page for the user.

It doesn't really add anything to a professional company developing web applications, so it is of no use to do any further research on this stack.

Web Forms

The front end extensions in Web Forms are mainly consisted of components. These extra components range from modern looking remakes of the basic components till smart implementations of everyday problems. The power here lies in the mass of third party libraries that companies offer, like Juice-UI (JuiceUI, -) and Ext.Net. (Ext.NET, -)

MVC

MVC is a framework that doesn't build on components. It uses normal HTML / CSS / JavaScript to generate the pages, which means that any JavaScript extension can be used. So it works perfectly fine with JavaScript libraries like JQuery, Sencha Ext JS, Modernizer and Backbone. Because of the usage of JSON and REST any real web developer can start out with MVC right away.

2.3 Are there extra frameworks that can be used?

MVP (Model View Presenter) (Boodhoo, 2006)

There is one mayor framework that can be added to Web Forms and MVC and that is MVP. It basically is a mix-up of both MVC and Web Forms trying to get the best out of both the stacks. It is also called a transition to go from Web Forms to MVC. The framework sacrifices some control on MVC's end and gains therefore some production from the Web Forms side.

On the other side, MVP still has components that can be dragged and dropped to generate HTML/CSS. And also the view is kept as simple as possible and contains zero logic. The presenter is a middle man that links up the view with the back-end and processes at every request.

2.4 Conclusion

ASP.NET has been around for some time and has its fair amount of followers in every stack, but the 2 main stacks for professional developers are Web Forms and MVC.

Each of the 2 stacks has its own strength why it should be used. While the power of Web Forms lies in its productivity, the power of MVC lies in the control the developer has on each part of the website.

However, both stacks have one thing in common and that is their extendability. Either Web Forms with all the custom made controls or MVC with its view engines and tons of JavaScript libraries.

3 Which stack has the best toolset to present the data needed for the dashboard?

3.1 What types of charts are there?

A chart can have a large variety of forms, but the four most common charts are: (Wikipedia, 2013)

- Histogram
- Bar / Column chart
- Pie chart
- Line chart

Which type of chart displays the data best?
Each chart has its own strengths and weaknesses. Therefore every chart will be looked at in more detail.

Histogram (Math Is Fun, -)
A histogram (**Figure 4**) is a visual representation of the distribution of data. It mostly is a comparison of data that have the same trait counted together. Like the amount of test that have 1-5 fails, 6-10 fails, 11-15 fails and so on.

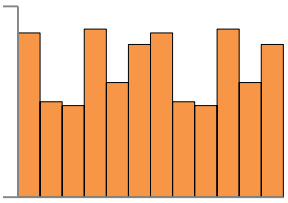


Figure 4: Histogram

The strength of this chart is that there is an overview of the amount compared to how often it happens. With this chart people can see which test should be looked at first. The weakness of this chart is that there isn't a distinctive test shown inside the chart.

Bar / Column Chart (Math Is Fun, -)
The bar chart (**Figure 5**) is a visual representation of the amount of data per category, like the number of tests per product that have failed in the last 4 days. A higher bar means more tests have failed.

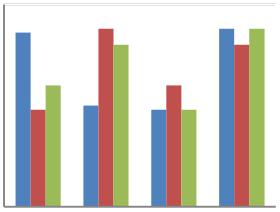


Figure 5: Column Chart

The strength of this chart is that it can group the data that needs to be displayed by category. And it can have multiple charts beside each other. The user can also easily see if the data has a trend of going up or down.

The weakness of this chart is that it is very global and mostly contains a larger number of tests.

Pie Chart (Math Is Fun, -)
The pie chart (**Figure 6**) is a visual representation of the amount of data set out against each other against a percentage scale. For example, if 300 out of 400 tests are completed successfully and 100 didn't, then 75% of the pie chart would be green and 25% of the chart would be red.

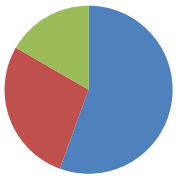


Figure 6: Pie Chart

The strength of this chart is that the last test run is in 1 rounded view and it's comparable with other runs.

The weakness of this chart is that there can only be one run in one chart at the same time. Unless the data that needs comparing is a total of the runs, which doesn't add anything to the whole.

Line Chart (Math Is Fun, -)
The line chart (**Figure 7**) is a visual representation of a series of data points connected by a straight line segment. This is more of a basic chart that has a big amount of variations and can almost do anything described by the charts above. But often it is used to show a trend of data. Like the amount of failed and succeeded tests of the past 10 runs.

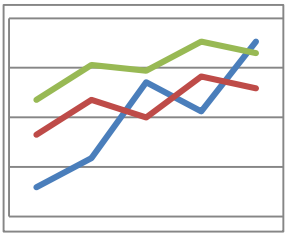


Figure 7: Line Chart

The strength of this chart is that a trend in the data can easily be seen, whether this is increasing or decreasing.

The weakness of this chart is that the user has no idea of the underlying patterns.

3.2 What type of framework can be used to display the charts?

There are many frameworks for charts using the full power of the browser. There are charting frameworks that use lots of different techniques to display the charts to the user the best way possible. The most common techniques to display charts are: HTML5/CSS3, JavaScript, PHP, and Flash. Because the dashboard will be realized in ASP.NET, there will also be a look at the frameworks that are specific to ASP.NET. To be able to choose the best framework for the dashboard, the table below displays the bigger frameworks and what they can and can't do.

Framework per language

Table 1 is a detailed breakdown of what a chart supports and what it doesn't. It also has a comment if there is something noticeable for the framework.

Name	HTML5	CSS3	JS	ASP.NET	SVG VML	Comments
amcharts	✓	✓	✓	×	✓	Some charts can use Images Line chart can change color with positive / negative
CanvasJS	✓	✓	✓	×	✓	
dc.js	✓	✓	✓	×	×	
D3js	✓	✓	✓	×	✓	If there is a need for something cool and out of the box, this is the package.
Highcharts	✓	✓	✓	×	✓	Changeable backgrounds
JqChart	✓	✓	✓	WF/MVC	✓	
InfoVis	✓	✓	✓	×	✓	Many trees and maps also Sunburst
KoolChart	✓	✓	✓	×	✓	Gauges / Images / matrix / …
extJS	✓	✓	✓	×	✓	Overkill javascript library
SenchaC	✓	✓	✓	×	✓	Overkill +1
TechOctave	✓	✓	✓	×	✓	
TeeChart	✓	✓	✓	✓	✓	Pictures for ASP
ZinoUI	✓	✓	✓	×	✓	

Table 1: Framework per language

Framework per chart

Table 2 is a detailed breakdown of what a chart supports and what it doesn't on chart base. It shows which framework there should be used best.

Name	Area	Bar	Column	Line	Pie	Legend	Mixable
amcharts	N%SM	N%SM3D	N%SM3D	NMG	NM3DBD	✓	✓
CanvasJS	N%SM	N%SM	N%SM	NM	NMBD	✓	✓
dc.js	×	N%SM	N%SM	NM	NMD	✓	✓
D3js	N%SM	N%SM	N%SM	NMG	NMD	✓	✓
Highcharts	N%SM	N%SM	N%SM	NM	NMD	✓	✓
JqChart	NM	NSM	NSM	NM	NM	✓	✓
InfoVis	SM	SM	SM	×	SM	✓	×
KoolChart	NSM	NSM3D	NSM3D	NM	NSM3DBD	✓	✓
extJS	N%SM	NSM	NSM	NM	NSMBD	✓	✓
SenchaC	N%SM	NSM	NSM	NM	NSMBD	✓	✓
TechOctave	NM	NSM	NSM	NM	NMD	✓	✓
TeeChart	NSM3D	NSM3D	NSM3D	NM	NM3DBD	✓	✓
ZinoUI	NM	NM	NM	NM	NMD	✓	✓

Table 2: Type of chart per framework

Abbreviation	Name
N	Normal
%	Percentage
S	Stacked
3D	3D
G	Gaps
M	Multi Series
B	Broken
D	Doughnut

Table 3 is the price that every framework asks for useage.
The common licensing is per developer (D) and per site (W). (S = Single, M = Multi, U = Unlimited)

Name	Price	Name	Price
amcharts	SW: €99 MW: €499 OEM: €999	InfoVis	MIT License
CanvasJS	1D: €299 5D: €799 10D: €1.499 UD: €2.999	KoolChart	SW: $490 2W: $880 5W: $1.830 MW: Cont
dc.js	Apache V2 License	extJS	1D: $595 5D: $2.885 20D: $11.185
D3js	BSD	SenchaC	1D: $995 5D: $4825 20D: $18.895
Highcharts	SW: $90 1D: $590 5D: $2.250 10D: $3.600 OEM: Cont	TechOctave	BLP: $200 CS: $340 All: $500
JqChart	JS 1D: $299 MVC 1D: $399 WF 1D: $399	TeeChart	.NET 1D: $759 IOS: $339 Android: $339 JS 1W: $129
		ZinoUI	1D/1W: $25 1D/UW: $40 UD/1W: $60 UD/UW: $105

Table 3: Prices per framework

3.3 Conclusion

The types and implementations of charts are sheer endless and prices for them range from free till $1000+. All of them have their own features and ways of showing the data. The choice fell on D3.js because of the endless possibilities with it. Because D3 isn't focused on charts, but works with HTML5's svg tag. D3 just supplies the JavaScript tools to change the svg tag and its component in the way the developer needs. It is also Open Source and thus problems with the framework can be solved by the person who encounters them.

4 How can the stack be tested?

4.1 Which test toolkits can be used?

Front End Testing

There aren't that many test toolkits that can automate front end testing. One of the better en most complete tools is Selenium and its components. (Tester, 2008)

Selenium has 4 big projects (Selenium, -) that give total control over the front end testing. The big projects are Selenium IDE, Selenium Remote Control, Selenium WebDriver, and Selenium Grid. The IDE and WebDriver are the projects that are most useful to a normal developer. The other 2 are more for the advanced tester who wants to do multi testing over multiple clients and programming languages.

SeleniumIDE **Figure 8** (SeleniumHQ, -) is an integrated development environment for selenium scripts. It is only useable in Firefox, which is the only downside to the whole project. For other browsers there is WebDriver, but more to that later. The power of IDE lies in that it doesn't need any source code to work. The plain HTML code of the site is enough for the IDE to work with.

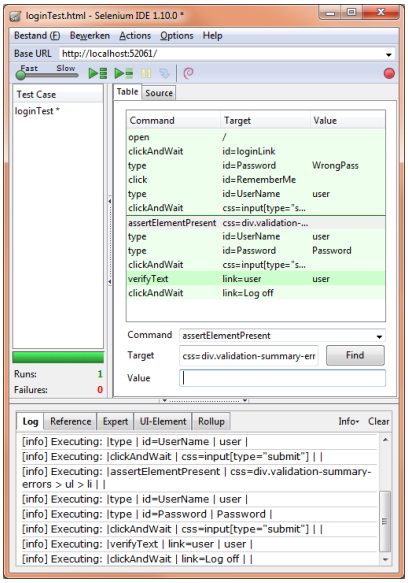


Figure 8: Selenium IDE

What Selenium does is to fake user input into the browser. This way the test does exactly the same as when a human tester would do it. So if the tester would make a Selenium script (a list of actions that the IDE should do) the test can be run automatically every time there is an update. So making testing of the front end much easier and faster.

With the built-in recording tool of the IDE, test making is a breeze. When the recording is started with the IDE, every step that is done in the browser is recorded. This way the IDE is highly productive and with all commands at the right-click of a button **Figure 9**, it can even do all the XPath searching work. If for example the text in a label needs to be tested, the tester can select the label and then right-click the selected part. The context menu can then be opened where the tester can click assertTextPresent. The IDE then automatically generates the path to the label and takes the current text of the label as testing reference.

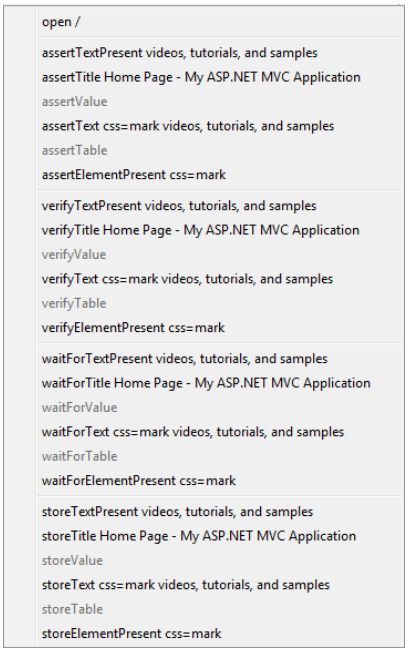


Figure 9: Choises of selenium IDE

But even when the IDE can automate the testing of the front-end, never leave out the human testing part. Because the IDE can't test the speed and look and feel of the page.

WebDriver (SeleniumHQ, -) is the project of Selenium that integrates with all the major development IDE's like Visual Studio, Netbeans, Eclipse, etc. WebDriver is the same as the IDE; only the tester needs to make unit tests with it. This has its upsides and downsides, but gives total control over Selenium.

The biggest upside of WebDriver is that all major browsers are supported. But only Firefox works out of the box. For the other browsers so called drivers need to be put in a folder and then added to the PATH variable of the system. The drivers also include Android and Iphone, which helps a lot for mobile testing.

Another upside is that with WebDriver the tester has full control over Selenium. The tester can make its own tests and doesn't need the HTML to be able to write them. There is only one requirement then and that is that the website must use the same ID's and classes as used in the test.

The downside is that there is no automatic recorder and that everything has to be written in hand. But since unit tests are mostly made by hand, this downside doesn't weigh heavily.

There is also screenshot functionality on the WebDriver which can make screenshots for the testers to evaluate and see if the site is displayed correctly.

The other 2 projects are more set out for remote and parallel testing and are made to be used inside continuous Integration Systems and other languages like Ruby, Python, Perl, etc. They can also be used in .Net to test the site on a remote server. These tests rely on the IDE API, which is different than the WebDriver API.

Back End Testing (Kojevnikov, 2008)
For the back-end testing there are more test platforms. The bigger platforms for unit testing are NUnit, MSTest, and XUnit. There are also some supporting frameworks like NMock, RhinoMock, and EasyMock.

MSTest (Wilson, 2012) is the default unit test of Visual Studio and is supplied with every copy of Visual Studio. MSTest has everything a tester needs to test a class on what it is expected to do, although some asserts aren't supported, like the Exception Assert and Type Asserts. These asserts can be made to work in MSTest, but the tester then has to take care of the comparison and then assert a Boolean.

NUnit (Wilson, 2012) has different assertions and attributes than MSTest. NUnit has Type Asserts, Exception Asserts, Repeat Properties, Culture Properties, and more. It also has its stand-alone runner where the tester can run the tests in. But it isn't shipped with Visual Studio and it needs an external runner to work properly. It also needs an plugin to work with Visual Studio's tester.

XUnit (Wilson, 2012) is an open source unit testing platform and has fewer options then the other 2 at the moment. But because its open source anyone can add new functionality to it when they need it and share it with the world.

There is also an extension of Visual Studio called Resharper. What it does to Visual Studio is just too big to write out. But Resharper has a component that manages the unit tests in Visual Studio. Although Resharper is a paid extension all the bigger test frameworks have a plugin for its test runner. Because of Resharper all frameworks can be used without a problem alongside each other making it easy to switch them out.

4.2 Are the stacks easy testable?

Every larger company wants its software to be tested before it's released to the public, because no company wants to release an application that is broken or malfunctioning, which could have been avoided by testing. This company is no different than the other bigger companies. Therefore they want the biggest part of their code covered with unit tests.

The best practice to cover most of the project with unit tests is to do Test Driven Development (TDD) Figure 10. To be able to do TDD the stack should be able to run separately without the need of any other component like a webserver.

The strongest part of the MVC pattern is that every part of MVC can be tested upon. The model and controller can be tested separately by unit tests. The tests don't need any other components like a webserver and thus the unit tests can be very simple. The View will always be a bit of a problem, because it needs a webserver to display itself. But with Selenium the unit tests aren't that complex and can be made fairly easy, provided that the HTML is written with good ID's.

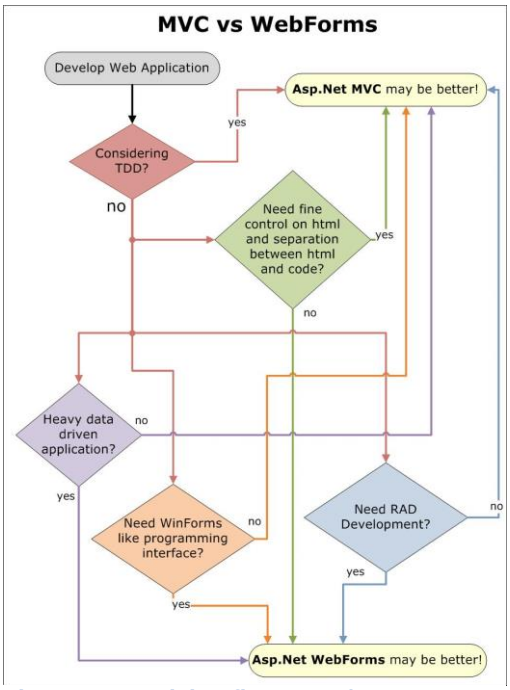


Figure 10: Decision flow MVC / WF

In opposite to MVC, Web Forms has a big problem where testing is concerned. Because the code and view are tightly weaved into each other, the only tests that can be run are Selenium tests. The Model View Presenter framework is something that can pull Web Forms apart and make them testable. But then there is the question: Why try to imitate MVC when there is a stack called MVC that offers the same but is fully developed with a library of additions?

4.3 What parts need to be tested?

The parts that need to be tested changes from project to project. If the project has a large number of fields that need to be filled in and has some custom security that needs to be tested, it is necessary to test the front-end thoroughly. If the project is something where the user can't really change anything, the front-end doesn't need that many tests. But the back-end always needs to be tested with unit tests. The testing of the back-end is critical to find bugs easily if there is any change on the dependencies or in the code itself.

4.4 Conclusion

Because of Resharper there isn't a problem which test platform the tester uses. Therefore MSTest is enough for the back-end testing of classes. If there is anything that can't be solved by MSTest, another platform can be used to make the test work.

The stack that wins here is MVC. This has some very simple reasons:

- Every part of MVC can be tested in isolation.
- There is no need for extra frameworks to be able to test.

5 Overall Conclusion

The best stack that a company should use if they start a new project is MVC, because it's highly extendable on every end of the stack. The developer can make its own view engine, its own custom view and put it in any model. MVC is also highly testable and doesn't need any extras to have a test run.

The only downside of the stack is that the production will go down in comparison to Web Forms. This is because MVC doesn't have any drag and drop components. But this doesn't weight heavy enough to stand against all positive sides of MVC.

6 Bibliography

ali62b. (2010, 1 25). *difference between model view controller (mvc) and asp.net web application ?* Retrieved 3 18, 2013, from The Official Microsoft ASP.NET Forums: http://forums.asp.net/t/1518156.aspx/1

Block, G. (2010, 8 17). *design patterns - What are MVP and MVC and what is the difference?* Retrieved 3 19, 2013, from Stack Overflow: http://stackoverflow.com/questions/2056/what-are-mvp-and-mvc-and-what-is-the-difference/101561#101561

Boodhoo, J.-P. (2006, 8 -). *Design Patterns: Model View Presenter*. Retrieved from MSDN Magazine: http://msdn.microsoft.com/en-us/magazine/cc188690.aspx

Chateaux Software. (2013, 4 24). *Working with SQL Server Schema Bound Views*. Retrieved from Chateaux Software: http://www.chatsoft.com/aboutus/articles/schema-bound-views.asp

Ext.NET. (-, - -). -. Retrieved from Ext.NET: http://www.ext.net/

Fassett, W. (2011, 4 26). *Why does the ASP.Net Web Forms model "suck"?* Retrieved 3 12, 2013, from Stack Overflow: http://stackoverflow.com/questions/46031/why-does-the-asp-net-web-forms-model-suck/234385#234385

Fink's, G. (20009, 1 16). *Deciding When to Use ASP.NET MVC Framework*. Retrieved 3 12, 2013, from Gil Fink's Blog: http://beta.blogs.microsoft.co.il/blogs/gilf/archive/2009/01/16/deciding-when-to-use-asp-net-mvc-framework.aspx

Garrett, J. J. (2005, 2 18). *Ajax: A New Approach to Web Applications*. Retrieved from Adaptive Path: http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications

Haack, P. (2008, 1 19). *Everything You Wanted To Know About MVC and MVP But Were Afraid To Ask*. Retrieved 3 1, 2013, from Haacked: http://haacked.com/archive/2008/06/16/everything-you-wanted-to-know-about-mvc-and-mvp-but.aspx

JuiceUI. (-, - -). *Supercharge ASP.NET Web Forms with jQuery UI*. Retrieved from JuiceUI: http://juiceui.com/

Kojevnikov, A. (2008, 11 4). *unit testing - NUnit vs. MbUnit vs. MSTest vs. xUnit.net*. Retrieved from Stack Overflow: http://stackoverflow.com/a/261156/1047155

MacCaw, A. (2011, 11 2011). *Asynchronous UIs - the future of web user interfaces*. Retrieved from alexmaccaw: http://alexmaccaw.com/posts/async_ui

Maclean, M. (2013, 4 12). *D3 Tips and Tricks*. Retrieved from LeanPub: https://leanpub.com/D3-Tips-and-Tricks

Marcotte, E. (2010, 5 25). *Responsive Web Design · An A List Apart Article*. Retrieved from alistapart: http://alistapart.com/article/responsive-web-design

Math Is Fun. (-, - -). *Bar Graphs*. Retrieved from Math Is Fun: http://www.mathsisfun.com/data/bar-graphs.html

Math Is Fun. (-, - -). *Histograms*. Retrieved from Math Is Fun: http://www.mathsisfun.com/data/histograms.html

Math Is Fun. (-, - -). *Line Graphs*. Retrieved from Math Is Fun: http://www.mathsisfun.com/data/line-graphs.html

Math Is Fun. (-, - -). *Pie Chart*. Retrieved from Math Is Fun: http://www.mathsisfun.com/data/pie-charts.html

McCafferty, B. (2007, 10 16). *Model View Presenter with ASP.NET*. Retrieved 3 18, 2013, from CodeProject: http://www.codeproject.com/Articles/14642/Model-View-Presenter-with-ASP-NET

Microsoft. (-, - -). *MVC*. Retrieved 3 11, 2013, from The Official Microsoft ASP.NET Site: http://www.asp.net/mvc

Microsoft. (-, - -). *Web Forms*. Retrieved 03 11, 2013, from The Official Microsoft ASP.NET Site: http://www.asp.net/web-forms

Microsoft. (-, - -). *Web Pages*. Retrieved 3 11, 2013, from The Official Microsoft ASP.NET Site: http://www.asp.net/web-pages

Microsoft N-Layer. (2013, 4 10). *Chapter 5: Layered Application Guidelines*. Retrieved from MSDN: http://msdn.microsoft.com/en-us/library/ee658109.aspx

NS, S. (2013, 3 20). *ASP.NET MVC 4*. Retrieved 11 19, 2012, from CodeProject: http://www.codeproject.com/Articles/470107/ASP-NET-MVC-4-Part-1-Introduction

Selenium. (-, - -). *Introduction — Selenium Documentation*. Retrieved 3 29, 2013, from SeleniumHQ: http://docs.seleniumhq.org/docs/01_introducing_selenium.jsp#

Selenium. (-, - -). *Selenium IDE Plugins*. Retrieved 3 29, 2013, from SeleniumHQ: http://docs.seleniumhq.org/projects/ide/

Selenium. (-, - -). *Selenium Projects*. Retrieved from SeleniumHQ: http://docs.seleniumhq.org/projects/

SeleniumHQ. (-, - -). *Selenium IDE Plugins*. Retrieved from SeleniumHQ: http://docs.seleniumhq.org/projects/ide/

SeleniumHQ. (-, - -). *Selenium WebDriver*. Retrieved from SeleniumHQ: http://docs.seleniumhq.org/projects/webdriver/

Tester, T. A. (2008, 8 8). *Selenium Training*. Retrieved from The Automated Tester: http://www.theautomatedtester.co.uk/selenium_training.htm

Vaibhav. (2008, 12 16). *Choosing Between WebForms and MVC*. Retrieved 3 13, 2013, from Habitually Good: http://blog.gadodia.net/choosing-between-webforms-and-mvc/

w3schools. (-, - -). *ASP.NET MVC Introduction*. Retrieved from w3schools: http://www.w3schools.com/aspnet/mvc_intro.asp

w3schools. (-, - -). *ASP.NET Web Forms Tutorial*. Retrieved from w3schools: http://www.w3schools.com/aspnet/aspnet_intro.asp

w3schools. (-, - -). *ASP.NET Web Pages Tutorial*. Retrieved from w3schools: http://www.w3schools.com/aspnet/webpages_intro.asp

Wertheim, D. (2011, 7 20). *Why the ExpectedExceptionAttribute sucks!* Retrieved 3 26, 2013, from Daniel Wertheim: http://daniel.wertheim.se/2011/07/20/why-the-expectedexceptionattribute-sucks/
Wikipedia. (2013, 3 2). *Chart*. Retrieved 3 25, 2013, from Wikipedia: http://en.wikipedia.org/w/index.php?title=Chart&oldid=541719896
Wilson, B. (2012, 11 14). *xUnit.net - Unit testing framework for C# and .NET (a successor to NUnit) - Home*. Retrieved from xUnit.net: http://xunit.codeplex.com/wikipage?title=Comparisons&ProjectName=xunit