Afstudeerverslag

Datum: 06-06-2014
Van: Michael de Vreugd | 08090769
T.A.V.: E.M. van Doorn, G.M. Tuk
Versie: v1.0

Voorwoord

Dit verslag is geschreven met als doel om de lezer te informeren over mijn afstudeerproject. Het bevat informatie betreffende het gehele proces en alle daarbij gemaakte producten. Ik hoop u met dit verslag genoeg inzicht te bieden in deze aspecten van het project.

Het doorlopen van dit project heeft mij veel nieuwe inzichten gegeven. Zo heb ik een geheel nieuwe programmeertaal geleerd, met alle bijbehorende uniekheid van deze taal. Ik heb een diepgaand onderzoek uitgevoerd naar het hoe en wat van input devices en heb ik veel meer inzicht gekregen in het samenwerken binnen een team. Dit ondanks dat ik het project waar dit verslag over gaat zelfstandig heb uitgevoerd. Mijn enthousiasme over het vakgebied en de rol die ik daarin mag vervullen is alleen maar gegroeid en ik ben dan ook zeer benieuwd wat de toekomst mij zal brengen.

Als afsluiting wil ik nog wat mensen bedanken. Ik wil ten eerste Martin Middel bedanken voor het aanraden van mij als potentiële afstudeerder bij Liones. Ook wil ik hem bedanken voor de hulp en inzicht die hij verschaft heeft tijdens dit project. Verder wil ik Stef Busking en Bert Willems bedanken voor de feedback en gedeelde wijsheden tijdens dit project. Als laatste wil ik alle overige medewerkers van Liones bedanken voor de fijne werksfeer.

Michael de Vreugd
06/06/2014

Inhoudsopgave

1. Bedrijf / Organisatie 7

1.1 Bedrijf / Organisatie 7

1.2 Producten/diensten 7

1.3 Omvang 7

1.4 Klanten 7

2. Concrete werkzaamheden 8

2.1 Werkzaamheden 8

3. Het tot stand komen van het plan van aanpak 9

4. Het onderzoek 11

4.1 Het doel van het onderzoek 11

4.2 Het definiëren van de hoofdvraag en deelvragen 11

4.3 Vooronderzoek 13

4.4 Het onderzoek 14

4.4.1 “Wat zijn de verschillen tussen alle input methoden op de verschillende operating systems, browsers en devices?” 14

4.4.2 “Welke input methoden ondersteunen de concurrenten?” 19

4.4.3 “Welke van de input methoden zullen door de eindgebruikers van FontoXML gebruikt worden?” 21

4.4.4 “Hoe gaan code editors om met het verwerken van input methoden die gebruik maken van compositions, IME’s of een andere schrijfrichting?” 22

4.4.5 “Hoe gaan open source browsers om met het verwerken van input methoden die gebruik maken van compositions?” 25

4.4.6 Eindconclusie 25

5. Opstellen adviesrapport 26

6. Opstellen van requirements 27

6.1 Het doel 27

6.2 Activiteiten 27

7. Opstellen initieel ontwerp 29

7.1 Het doel 29

7.2 Activiteiten 29

8. Sprint 1: Basics en westers toetsenbord 31

8.1 Het doel 31
8.2 Activiteiten 31
9. Sprint 2: Mobile devices en IME's 36
9.1 Het doel 36
9.2 Activiteiten 36
9.2.1 Realisatie van event handling voor IME's 36
9.2.2 Complexiteit van event handling op iOS en afhandeling voor autocorrect 38
10. Sprint 3: iOS, output en intelligentie 40
10.1 Het doel 40
10.2 Activiteiten 40
10.2.1 Vervolg iOS en autocorrect 40
10.2.2 Output handling 42
11. Opstellen testplan, uitvoeren tests en het documenteren van de resultaten 46
12. De productevaluatie 50
12.1 De producten 50
12.2 Evaluatie 50
12.2.1 De kwaliteit 50
12.2.2 Het nut 51
12.3 Verder gebruik 51
13. De procesevaluatie 53
14. Beroepstaken evaluatie 54
14.1 Beroepstaken zoals aangegeven in het afstudeerplan 54
14.2 Evaluatie uitgevoerde beroepstaken 54

Inleiding

Achtergrond

Liones ontwikkelt een browser based What You See Is What You Get(WYSIWYG) tekst editor genaamd FontoXML. Op de achtergrond van deze WYSIWYG tekst editor wordt de tekst omgezet naar Extensible Markup Language(XML) en opgeslagen. Deze aanpak zorgt er voor dat auteurs zonder technische voorkennis XML kunnen schrijven. Omdat FontoXML elke bewerking ook moet toepassen op het onderliggende XML document, een concept bekend als real-time validatie, is het nodig om alle invoer geheel af te vangen via JavaScript. Hierdoor maakt FontoXML geen gebruik van de in de browser ingebouwde invoervelden. Browsers zijn echter wel vooral ingericht voor het gebruik van deze ingebouwde invoervelden. Daarom hebben zij bij het ontwikkelen van FontoXML zich voornamelijk gericht op de klassieke invoer mogelijkheden: een westers toetsenbord en muis bij het gebruik van FontoXML op een PC. Zij willen dit graag uitbreiden met ondersteuning voor meerdere invoer mogelijkheden en ondersteuning voor het gebruik van bijvoorbeeld tablets en smartphones.

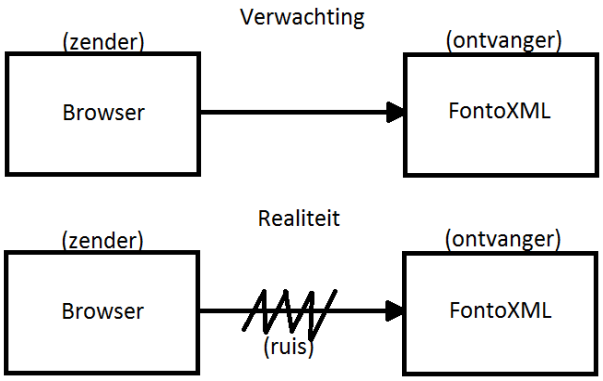
Probleem en uitvoering

Om FontoXML aantrekkelijker te maken voor een breder publiek moest er onderzoek uitgevoerd worden naar de gebruikersbehoefte voor ondersteuning van andere invoerapparaten. Hierbij moest gedacht worden aan een touchscreen op een tablet, maar ook verschillende toetsenborden of inputapparaten voor mensen met een beperking. Hierbij waren zij vooral geïnteresseerd in onscreen toetsenborden met autocomplete/autocorrect, het automatisch aanvullen of corrigeren van een getypt woord, op mobiele platforms. Zoals bijvoorbeeld Android, iOS en Input Method Editors (IME's), een methode voor het invoeren van bijvoorbeeld Japans of Chinees op een PC.

Het probleem dat hierbij naar voren komt is dat niet iedere browser en operating system combinatie de input aanleveren zoals je dat zou verwachten. Zo verwachten we bijvoorbeeld bij het aanslaan van een letter op een toetsenbord vier events, namelijk: een keydown event, een keypress event, een input event en een keyup event. Wanneer je dit doet bij het gebruik van een toetsenbord met een QWERTY layout en bij het gebruik van bijvoorbeeld Chrome op Windows is dit ook zo, maar wanneer je dit doet bij het gebruik van Chrome op Android is dit niet zo. Dit heeft te maken met de volgende factoren: Gebruikte invoer mogelijkheid, operating system en browser. De standaarden definiëren namelijk op meerdere niveaus bepaalde events, dingen die kunnen gebeuren met HTML elementen, maar in vrijwel geen enkele implementatie leveren deze in alle situaties het complete plaatje.

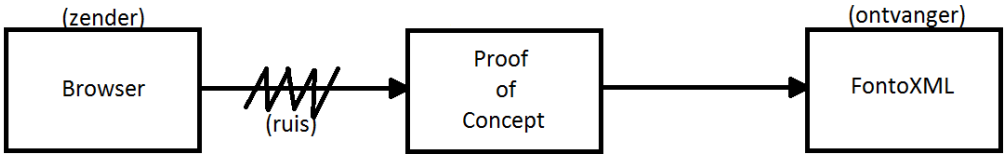
Dit kan je het beste vergelijken als het houden van een gesprek, hierbij is de browser de zender en is FontoXML de ontvanger. Zoals in elk normaal gesprek kan er ruis ontstaan, in het geval van het gesprek tussen de browser en FontoXML wordt deze ruis, waarmee het niet verzenden van bepaalde events wordt bedoeld, bijvoorbeeld veroorzaakt door de browser en welk operating system er gebruikt wordt (fig I.1).

FIG I.1



Om FontoXML geschikt te maken voor deze omgevingen en inputapparaten is er onderzoek uitgevoerd naar welke input methoden ondersteund moeten gaan worden door FontoXML. Gebaseerd op de resultaten van het onderzoek is een proof of concept gemaakt voor het robuust detecteren van invoer bij het gebruik van deze input methoden. Dit proof of concept zou je kunnen zien als de vertaler binnen het gesprek. Dit zorgt ervoor dat de zender zichzelf kan uiten zoals dit al gebeurde, maar dat de ontvanger wel ontvangt wat hij verwachtte (fig I.2). Het is hierbij belangrijk dat het proof of concept deze “vertaling” uitvoert zonder dat de gebruiker van FontoXML merkt dat dit gebeurt. Visuele vertraging is hierbij een belangrijk aspect. Hoe dit getoetst is wordt besproken in de hoofdstukken betreffende de ontwikkeling van het proof of concept.

FIG I.2



Opbouw

In het eerste deel van dit verslag(hoofdstuk 1 t/m 3) omschrijf ik het bedrijf, de concrete werkzaamheden zoals deze stonden aangegeven in het afstudeerplan en het tot stand komen van het plan van aanpak.
In het tweede deel van dit verslag(hoofdstuk 4 t/m 11) ga ik dieper in op het uitgevoerde onderzoek, het uitbrengen van een advies gebaseerd op het onderzoek, de gemaakte sprints en het testen.
In het derde en laatste deel van dit verslag(hoofdstuk 12 t/m 14) evalueer ik over het product, het ontwikkelproces en de beroepstaken.

1. Bedrijf / Organisatie

In dit hoofdstuk wordt het bedrijf waar de afstudeeropdracht is uitgevoerd besproken. Hierbij worden alleen de aspecten besproken die van belang zijn voor het uitgevoerde project.

1.1 Bedrijf / Organisatie

Liones is een full-service internetbureau waar ruim 30 professionals werken voor uiteenlopende klanten die zij helpen met hun content strategie. Ze werken met de nieuwste technieken en best-practices in combinatie met de nieuwste ontwikkelmethodes. Ze ontwikkelen projecten in de programmeertalen JavaScript en .NET (onder andere op basis van Lynkx, een in-house ontwikkeld framework). Zij doen dit op basis van de Scrum methode. Ze zijn een dynamisch bureau waar professionaliteit, persoonlijke ontwikkeling en collegialiteit belangrijke waarden zijn. Liones ontwikkelt een zogeheten What You See Is What You Get (WYSIWYG) tekst editor genaamd FontoXML. Op de achtergrond van deze WYSIWYG tekst editor wordt de tekst omgezet naar XML en opgeslagen.

1.2 Producten/diensten

FontoXML is een web based WYSIWYG tekst editor met als focus het gebruiksvriendelijk aanpassen van valide XML.

1.3 Omvang

Het team dat momenteel aan FontoXML werkt telt zo'n 10 medewerkers van Liones. De twee belangrijke personen uit dit team in betrekking tot het afstudeerproject zijn:

- Bert Willems - Product owner
- Stef Busking - Bedrijfsmentor

1.4 Klanten

- Wolters Kluwer
- RILM
- IAEA
- Index.fi

2. Concrete werkzaamheden

In dit hoofdstuk worden de concrete werkzaamheden omschreven zoals deze waren aangegeven in het afstudeerplan.

2.1 Werkzaamheden

- Plan van aanpak:
  Als eerste zal een plan van aanpak opgesteld worden. Hierin zal duidelijk gedefinieerd worden welke activiteiten zullen plaats vinden en welke producten dit zal opleveren.
- Onderzoek:
  Vervolgens zal begonnen worden met het onderzoek naar de verschillende methoden van input en de afhandeling daarvan binnen verschillende browsers en platforms. Ook zal onderzoek gedaan worden naar de bestaande code binnen FontoXML.
- Requirements definiëren:
  Er zal een lijst van requirements gemaakt worden waaraan het op te leveren proof of concept zal voldoen. Deze requirements zullen opgesteld worden aan de hand van de resultaten van het onderzoek en de wensen vanuit Liones.
- Ontwerpen proof of concept:
  Aan de hand van de resultaten van het onderzoek zal een ontwerp gemaakt worden voor een proof of concept dat om kan gaan met de verschillende soorten input binnen de verschillende browsers en platforms.
- Ontwikkelen proof of concept:
  Het realiseren van het ontwerp in code.
- Testen proof of concept:
  Het opstellen en uitvoeren van een testplan voor het proof of concept en het verwerken van de resultaten in een testrapport.

3. Het tot stand komen van het plan van aanpak

Het was van belang om een plan van aanpak op te stellen om zo een overzicht te krijgen van hoe het project doorlopen zou worden. Wat ik hierbij gelijk merkte was dat het moeilijk was om een keuze te maken betreffende ontwikkelmethode voor het proof of concept. Dit had twee redenen ten eerste was het op dat moment nog onduidelijk wat het proof of concept precies zou moeten bevatten, dit zou namelijk voortkomen uit het onderzoek dat later uitgevoerd is. Ten tweede wou ik een ontwikkelmethode kiezen die goed zou aansluiten bij het project en het bedrijf. Het oorspronkelijke idee hierbij was om de Scrum methode[1] te gebruiken.

Deze methode wordt ook binnen Liones gebruikt, maar door feedback tijdens het bespreken van mijn afstudeerplan op school kwam omhoog dat het toepassen van Scrum moeilijk tot onmogelijk zou zijn omdat ik het project alleen zou uitvoeren. Dit heeft er toe geleid dat ik eerst een dag lang onderzocht heb of er geen mogelijkheid bestond om Scrum met een projectgroep grootte van één persoon uit te voeren. De conclusie die ik trok is dat dit niet mogelijk was. Dit aangezien veel van de definiërende activiteiten van Scrum, zoals de daily stand-up en retrospectives, niet alleen uitgevoerd kunnen worden. Hierna ging ik op zoek naar een andere methode om te gebruiken. Ik kwam toen uit op de Watervalmethode, dit sloot qua fasering mooi aan op hoe ik het project zou gaan doorlopen. De Watervalmethode werkt namelijk met de volgende fasering:

- Definitiestudie/analyse
  In deze fase zou ik mijn onderzoek uitvoeren.
- Basisontwerp
  Na mijn onderzoek zou ik het basisontwerp opstellen.
- Technisch ontwerp/detailontwerp
  Voor het bouwen zou ik uit het basisontwerp een technisch ontwerp maken.
- Bouw
  In deze fase zou ik het proof of concept bouwen.
- Testen
  In deze fase zou ik het proof of concept dan testen en de resultaten hiervan vast leggen.
- Integratie, beheer en onderhoud
  Deze fase zou uitgevoerd kunnen worden indien er nog tijd over was. Ik zou dan het te bouwen proof of concept integreren in FontoXML.

Echter naarmate ik vordering maakte in mijn onderzoek merkte ik dat het moeilijk zou gaan worden om de Watervalmethode fasering aan te houden. Bij gebruik van de Watervalmethode kan er namelijk niet gestart worden met een fase tenzij de voorgaande fase afgesloten is. Dit zou moeilijk worden bij het uitvoeren van de fase waarin het technische ontwerp opgesteld moet worden. Wat ik namelijk voorspelde is dat er aan de start van deze fase nog niet genoeg inzicht zou zijn om een compleet ontwerp op te stellen. De details die hierin verwerkt moeten worden zouden beter bekend worden tijdens het ontwikkelen van het proof of concept en aan de hand hiervan zou het ontwerp aangepast gaan worden. Dit zorgde ervoor dat ik toch weer op zoek ging naar een alternatief. Na wat meer gesprekken te hebben gehad met medestudenten die

[1]Scrum guide: https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide.pdf#zoom=100

ook bezig waren met afstuderen bij Liones (of al klaar waren met afstuderen) kwam ik erachter dat zij hun project hadden uitgevoerd door elementen van Scrum te gebruiken ondanks een projectgroep grootte van één persoon. Dit deden zij door de aspecten van Scrum, die je niet alleen kan uitvoeren, mee te draaien met het projectteam waar hun project (het meeste) bij aansloot.

Zodoende koos ik ervoor om ook mijn planning en plan van aanpak aan te passen om zo toch gebruik te kunnen gaan maken van elementen van de Scrum methode. Voornamelijk de daily stand up, retrospectives, backlogs en korte sprints. Dit hield in dat in de initiële planning (fig 3.1) er geen fasering of iteraties duidelijk zijn voor het bouwen van het proof of concept, deze worden namelijk pas bepaald tijdens het opstellen van elke sprint (de Scrum versie van fasering). Scrum is alleen gebruikt voor het bouwen van het proof of concept. Verder bevat de planning de andere benodigde activiteiten zoals het onderzoek, opstellen van requirements en het testen van het proof of concept.

FIG 3.1

PRODUCT	UITVOERDATUM	DUUR
Plan van aanpak	10-02-2014 t/m 12-02-2014	16 uur (2 dagen)
Onderzoeksrapport - Vooronderzoek - Literatuuronderzoek - Verwerken bevindingen - Conclusie	12-02-2014 t/m 10-03-2014	140 uur (17.5 dagen) - 16 uur - 58 uur - 56 uur - 10 uur
Adviesrapport	11-03-2014	8 uur (1 dag)
Requirements - Requirements uit het onderzoek opstellen - Requirements uit de wensen van Liones opstellen	12-03-2014	8 uur (1 dag) - 4 uur - 4 uur
Ontwerp - Ontwerpen proof of concept - Ontwerpen integratie met FontoXML	13-03-2014 t/m 19-03-2014	40 uur (5 dagen) - 30 uur - 10 uur
Proof of concept	19-03-2014 t/m 14-05-2014	320 uur (8 weken)
Testplan	15-05-2014	8 uur (1 dag)
Testrapport	16-05-2014 t/m 19-05-2014	16 uur (2 dagen)

4. Het onderzoek

4.1 Het doel van het onderzoek

Dit project is begonnen met een onderzoek. Dit onderzoek had meerdere doelen. Onderzoek naar welke manieren van input er allemaal bestaan. Naar hoe andere platformen, waaronder concurrenten, hier dan mee om gaan en ook welke van deze manieren theoretisch gebruikt zouden worden door toekomstige eindgebruikers van FontoXML. Stuk voor stuk allemaal belangrijke onderdelen die gingen bepalen wat het proof of concept precies zou gaan bevatten en voor het beantwoorden van de hoofdvraag van het onderzoek. Het doel van het onderzoek was dan ook om deze vragen te beantwoorden en zo de scope van de rest van het project te bepalen.

4.2 Het definiëren van de hoofdvraag en deelvragen

Aan het begin van een onderzoek bepaal je wat er precies onderzocht gaat worden. De zogeheten hoofdvraag en deelvragen. Hierbij is de hoofdvraag de vraag waarover je uiteindelijk een conclusie wilt hebben en zijn de deelvragen vragen waarvan de gezamenlijke conclusies ervoor zorgen dat de hoofdvraag beantwoord kan worden.

FIG 4.1

Het onderzoek is uitgevoerd met gebruik van het Big 6 stappenplan[2] (fig 4.1).Dit stappenplan bestaat uit de volgende stappen:

1. Task definition
    - Het definiëren van het probleem
    - Het identificeren van de informatie requirements van het probleem
2. Information seeking strategies
    - Bepalen van het bereik van mogelijke bronnen
    - Evalueren van de verschillende mogelijke bronnen om prioriteiten te bepalen

[2] Officiële Big6 website: http://big6.com/

3. Location and access
    - Het vinden van bronnen
    - Het vinden van informatie binnen de bronnen
4. Use of information
    - Lees / hoor / bekijk de informatie uit de bron
    - Informatie uit de bron halen
5. Synthesis
    - Het organiseren van de informatie uit meerdere bronnen
    - Het maken van een product betreffende deze informatie
6. Evaluation
    - Het beoordelen van dit product
    - Het beoordelen van het proces

Het bepalen van de hoofdvraag was al grotendeels bepaald door de specificatie van de afstudeeropdracht. Deze specificeerde namelijk dat Liones FontoXML beschikbaar wil maken voor gebruikers die liever op een ander apparaat willen werken of teksten in bijvoorbeeld een taal zoals Chinees willen schrijven. Dit wilden ze doen door betere ondersteuning te bieden voor vreemde invoerapparaten. De hoofdvraag werd dan ook:

“Welke input methoden op welke devices en platforms moeten toegevoegd worden aan FontoXML?”.

De volgende stap was het definiëren van deelvragen die de benodigde hulp zouden bieden voor het beantwoorden van de hoofdvraag. Om de deelvragen te bepalen moet er goed gekeken worden uit welke onderdelen de hoofdvraag eigenlijk bestaat. Bij deze hoofdvraag is dat “Welke input methoden op welke devices en platforms” om dit deel van de vraag te beantwoorden moest ik inventariseren welke input methoden er eigenlijk allemaal bestonden en waar deze nou precies verschillend zijn. De deelvraag die hierbij ontstond is: *“Wat zijn de verschillen tussen alle input methoden op de verschillende operating systems, browsers en devices?”*.

Een andere deelvraag was: *“Hoe gaan code editors om met het verwerken van input methoden die gebruik maken van compositions, IME’s of een andere schrijfrichting?”* deze deelvraag was niet afgeleid vanuit de hoofdvraag maar een vraag vanuit Liones die wel aansloot bij mijn onderzoek. Compositions worden gebruikt bij talen zoals Japans en Chinees, waarbij het woord eerst fonetisch getypt wordt om daarna samengevoegd te worden tot het Japanse of Chinese karakter. Deze karakters kunnen dan wederom ook weer samengevoegd worden om een nieuw karakter te vormen. Bijvoorbeeld Japans maakt hier gebruik van:
De karakters voor O(お) en Ni(に) worden samengevoegd om ONi(鬼) te vormen.

Nadat ik onderzocht zou hebben welke input methoden er nou allemaal bestaan zal er onderzocht worden welke nou echt gebruikt zouden worden door FontoXML eindgebruikers. Hieruit ontstond de deelvraag *"Welke van de input methoden zullen door de eindgebruikers van FontoXML gebruikt worden?"*. Ook wou Liones dat ik zou kijken hoe de concurrenten reageren op deze selectie van inputmethoden, dit leverde de volgende deelvraag op *"Welke input methoden ondersteunen de concurrenten?"*.

Als extra zou de deelvraag *"Hoe gaan open source browsers om met het verwerken van input methoden die gebruik maken van compositions?"* meegenomen worden in het onderzoek. Deze deelvraag zou niet nodig zijn voor het beantwoorden van de hoofdvraag, maar zou wel belangrijke informatie opleveren voor het ontwikkelen van het proof of concept. Omdat deze browsers open source zijn is het mogelijk om de source code door te kunnen nemen. Dit zou inzicht kunnen bieden hoe deze browsers de events aangeleverd door het operating system vertalen naar JavaScript events.

Dit waren dus de deelvragen waarmee het onderzoek gestart zou worden:

- *"Wat zijn de verschillen tussen alle input methoden op de verschillende operating systems, browsers en devices?"*
- *"Hoe gaan code editors om met het verwerken van input methoden die gebruik maken van compositions, IME's of een andere schrijfrichting?"*
- *"Welke van de input methoden zullen door de eindgebruikers van FontoXML gebruikt worden?"*
- *"Welke input methoden ondersteunen de concurrenten?"*
- *"Hoe gaan open source browsers om met het verwerken van input methoden die gebruik maken van compositions?"*

4.3 Vooronderzoek

Voordat er met het onderzoek zelf begonnen kon worden is er vooronderzoek gedaan naar of het op te lossen probleem niet al eerder is opgelost. Er is dus vooronderzoek gedaan of er niet al een oplossing bestond voor het robuust afhandelen van input geleverd door verschillende devices met verschillende operating systems en browsers. Het resultaat hiervan is dat daar geen oplossingen voor bestonden. Dit was ook het verwachtte resultaat, aangezien er aangegeven was dat vanuit Liones hier ook al naar gezocht was. De volgende stap was om zogeheten literatuuronderzoek uit te voeren. Dit gebeurd door middel van het opzoeken van informatiebronnen die gebruikt kunnen worden voor het beantwoorden van de betreffende onderzoeksvragen. Een van de methoden die hierbij is toegepast is de Snowball methode. Hierbij wordt er gebruik gemaakt van het doorzoeken van informatiebronnen waarnaar gerefereerd wordt in eerder verzamelde informatiebronnen. Dit heeft geleid tot een uiteindelijke lijst met meer dan 40 informatiebronnen.

4.4 Het onderzoek

4.4.1 “Wat zijn de verschillen tussen alle input methoden op de verschillende operating systems, browsers en devices?”

Toen er genoeg informatiebronnen verzameld waren om de hoofdvraag en deelvragen mee te kunnen beantwoorden kon het daadwerkelijke onderzoek beginnen. Om de deelvraag “Wat zijn de verschillen tussen alle input methoden op de verschillende operating systems, browsers en devices?” te kunnen beantwoorden was er zowel theoretisch onderzoek nodig (het doorlezen van de informatiebronnen) als een experiment. Namelijk het testen van verschillende input methoden. Dit wordt gedaan op basis van een door Stef Busking, een van de FontoXML developers bij Liones, geschreven stuk test code. Het testen vindt met behulp van deze tool plaats op combinaties van verschillende operating systems en browsers. De tool die hiervoor gebruikt wordt is aangepast, hoe dit gedaan is en waarom wordt later in dit hoofdstuk omschreven. Voor het theoretische deel is er stapsgewijs gewerkt. De eerste stap die hierbij gemaakt is was het vast leggen van welke verschillende input methoden er nu allemaal zijn. Dit is gedaan door het doornemen van de informatiebronnen en het vastleggen van de inputmethoden in een tabel (fig 4.2).

FIG 4.2

Type	**Subtypes**	**Omschrijving**
Keyboards	Computer(Physical) keyboard, Software keyboard, Chorded keyboard / Keyer, Lighted Program Function Keyboard(LPFK)	Een keyboard is een human interface device (HID) gerepresenteerd door een layout van knoppen. Elke knop, of toets, kan gebruikt worden voor het invoeren van een linguïstische karakter of het aanroepen van een computer functie.
Pointing devices	Mouse, Mini-mouse, Trackball, Pointing stick, Finger tracking, Stylus, Touchpad, Touchscreen	Een pointing device is elk HID dat de gebruiker de mogelijkheid geeft om ruimtelijke data door te spelen naar de computer.
Audio input devices	Microphone	Audio input devices worden gebruikt om geluid te ontvangen of om geluid te maken.

Om de scope van de rest van het onderzoek te kunnen beperken was het van belang om hierna vast te leggen wat de subtypes precies waren van deze inputmethoden (fig 4.3, fig 4.4, fig 4.5) en wat de beschikbaarheid was per operating system. De operating systems die hierbij bekeken waren zijn: Windows, Mac OS, Windows Phone, iOS, Android en BlackBerry OS. Gebaseerd hierop kon er al een beslissing gemaakt worden betreffende welke subtypes niet meegenomen zouden worden in de rest van het onderzoek.

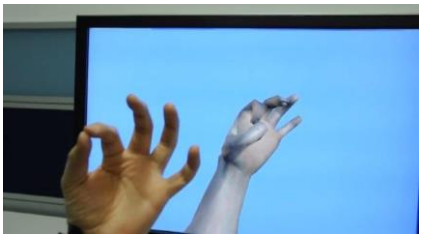
FIG 4.3

Keyboards

Subtype	**Omschrijving**
Computer (Physical) keyboard 	Een keyboard in de computer wereld is een device in de style van een typemachine. Het maakt gebruik van een opstelling van knoppen, of toetsen, die gebruikt worden als hendels of elektronische schakelaars. Wordt gebruikt als de main vorm van input voor computers. Is beschikbaar in vele verschillende varianten geschikt voor verschillende regionen in de wereld.
Software keyboard 	Een software keyboard is een software matige vorm van een keyboard. In plaats van het fysieke apparaat wordt het keyboard op het beeld getoond en de toetsen kunnen door middel van de muis of touch aangeslagen worden.
Chorded keyboard / Keyer 	Een chorded keyboard is een input device dat de gebruiker in staat stelt om karakters in te voeren door middel van het tegelijk indrukken van verschillende toetsen, zoals men bijvoorbeeld een akkoord aanslaat op een piano.
Lighted Program Function Keyboard(LPFK) 	Een LPFK is een keyboard met daarop meerdere toetsen, deze toetsen zijn gebonden aan functies aangegeven in software die dit support. Een populaire vorm hiervan is niet een pure LPFK maar een combinatie tussen een LPFK en een normaal keyboard.

FIG 4.4

Pointing devices

Subtype	**Omschrijving**
Mouse / Mini-mouse / Trackball  	Een mouse is een klein handheld device dat bewogen wordt over een horizontale oppervlakte. Een traditionele mouse maakte gebruik van een bal en twee schachten aan de onderkant om verplaatsing te meten. Tegenwoordig wordt dit bepaald door middel van een zichtbaar of infrarood licht aan de onderkant van de mouse. Een mini-mouse is simpelweg een kleinere versie van een normale mouse. Een trackbal is een omgekeerd versie van een traditionele mouse. Hierbij zit de bal niet aan de onderkant maar aan de bovenkant van het device en wordt de positie bepaald niet door het verplaatsen van het device zelf, maar door het rollen met de bal.
Pointing stick 	Een pointing stick is een device bestaande uit een pressure-sensitive knobbel dat gebruikt kan worden als een joystick. Het kan gevonden worden in bepaalde laptoppen tussen de ‘G’, ‘H’ en ‘B’ toetsen.
Finger tracking 	Een finger tracking device volgt de beweging van vingers in een 3D ruimte, zonder dat de vingers hierbij contact hebben met het scherm of een ander device.
Stylus 	Een stylus is een klein pen-vormig instrument dat gebruikt kan worden om input te leveren aan devices met een touchscreen. Dit kan verschillen van simpele handelingen zoals het “klikken” tot het schrijven van tekst in combinatie met handschrift herkenningssoftware. Een stylus kan ook een of meerdere knoppen hebben waarmee

	de functionaliteit lichtelijk veranderd, bijvoorbeeld om een screenshot te maken van een deel van het scherm (aangegeven met de stylus).
Touchpad 	Een touchpad is een plat oppervlakte dat vinger contact kan detecteren. Hiermee kan positie bepaald worden. Touchpads vinden we meestal terug in laptoppen. In sommige gevallen kunnen er “gestures” gebruikt worden.
Touchscreen 	Een touchscreen is een device embedded in een andere device zoals een TV, monitor, laptop, telefoon etc. Dit zorgt ervoor dat er contact met het scherm geregistreerd kan worden inclusief “gestures”.

FIG 4.5

Audio input devices

Subtype	**Omschrijving**
Microphone 	Een microfoon is een elektromechanisch device dat geluid omzet in een elektrisch signaal.

Het eerste wat opvalt bij deze subtypes is dat sommige hiervan erg “exotisch”, of niet meer relevant, zijn. Zo zien we bijvoorbeeld bij keyboards de chorded keyboard, een methode van input waar momenteel weinig commercieel beschikbare versies van bestaan. Bij pointing devices zien we de pointing stick, niet zozeer moeilijk verkrijgbaar, maar zeker ook niet een device dat hedendaags nog veel gebruikt wordt. Tijdens een van de Scrum stand-ups is dan ook in overleg met Bert Willems, de product owner en een van de FontoXML developers bij Liones, besloten om de scope van het onderzoek te beperken. De volgende devices zouden worden meegenomen in de rest van het onderzoek: Physical keyboard, Software keyboard, Mouse / Mini-mouse / Trackball, Stylus, Touchpad, Touchscreen.

Na deze selectie was het van belang om de populariteit te bekijken van de browsers per operating system. De splitsing per operating system is belangrijk omdat de implementaties vaak sterke afwijkingen vertonen tussen de verschillende operating systems. Dit zou helpen bij het maken van de selectie voor welke browsers getest zouden worden bij het experiment. Met als doel om een zo groot mogelijk percentage af te testen. Deze informatie was te vinden in drie van de informatiebronnen die van te voren verzameld waren. Op deze sites[3] werden statistieken bijgehouden over de populariteit van de browsers per operating system. Deze zijn vervolgens vastgelegd in een matrix (fig 4.6) voor het onderzoek. In deze matrix wordt aangegeven of een browser beschikbaar is voor dat operating system (een groene tint voor wel beschikbaar en rood voor niet beschikbaar) en worden de browser gerangschikt op populariteit op het betreffende operating system (aangegeven door zowel de tint groen als een nummering).

FIG 4.6

	Chrome	Firefox	Internet Explorer	Safari	Opera	UC Browser	Android browser	BlackBerry OS Browser
Windows	#1	#2	#3	#4	#5			
Mac OS	#1	#2		#3	#4			
Windows Phone	#3		#1			#2		
iOS	#4			#1	#2	#3		
Android	#4	#5			#2	#3	#1	
BlackBerry OS *	#5	#6			#3	#4	#2	#1

*BlackBerry OS is hierbij uniek omdat er de mogelijkheid bestaat om Android applicaties te installeren. De ranking bij BlackBerry OS is dan ook hierop gebaseerd met als enige uitzondering de BlackBerry OS browser.

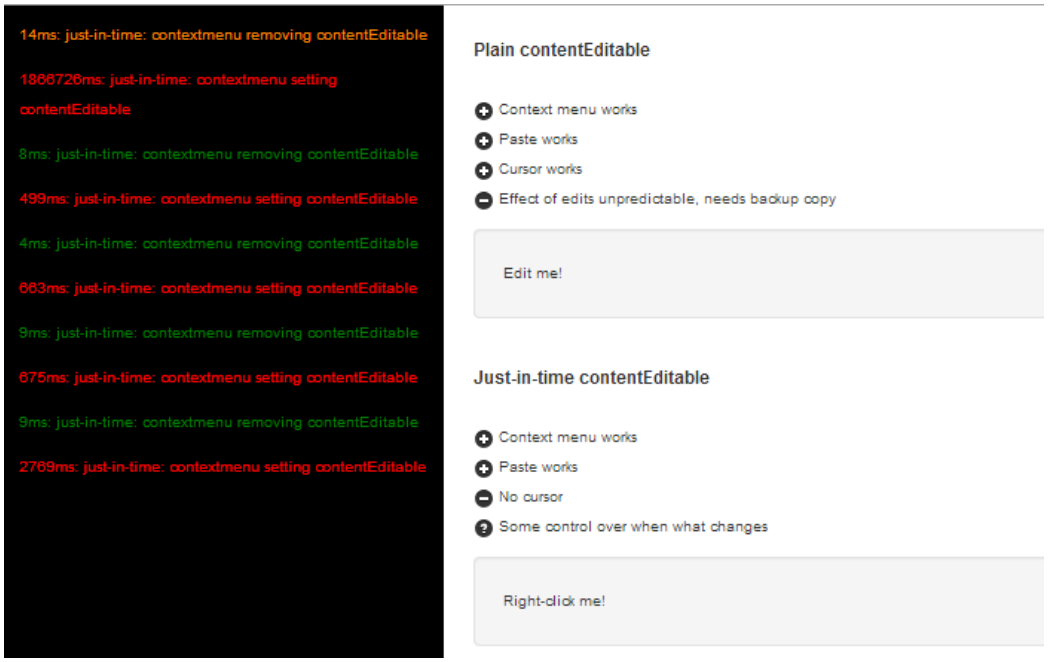
Toen dit vastgelegd was zijn er nog wat aanpassingen gemaakt aan de tool voor het testen. Belangrijk hierbij was dat het duidelijk zou zijn hoeveel tijd er tussen de binnenkomende events (fig 4.7) zit. Tijdens de ontwikkeling van het proof of concept wordt dit dan ook bijgehouden. Met als doel om bij te houden dat het proof of concept niet teveel vertraging zou opleveren. Dit is gedaan door bij elk binnenkomend event de tijd op te slaan en daar de tijd van het voorgaande event af te trekken. Wat er hierdoor overblijft, is het aantal milliseconden tussen de twee events.

De events worden geschreven naar een “log” scherm binnen de tool. Hierin worden de events gerangschikt op snel (tot 12 milliseconden), langzamer (tot 16 milliseconden) en langzaam (alles boven 16 milliseconden). Deze waardes zijn gebaseerd op het laagst aantal milliseconden

[3] Website met statistieken betreffende browser gebruik op Windows: http://download.cnet.com/windows/web-browsers/ , Website met statistieken betreffende browser gebruik op Mac OS: http://download.cnet.com/mac/web-browsers/ , Website met statistieken betreffende browser gebruik op mobile devices: http://www.netmarketshare.com/browser-market-share.aspx?qprid=0&qpcustomd=1

waarbij het menselijk mogelijk is om vertraging op te merken[4]. In realiteit ligt dit voor meeste mensen dichter bij de 100 milliseconden. Nadat dit gedaan was zijn de verschillende browser en operating system combinaties getest en de resultaten hiervan vast gelegd.

FIG 4.7



De resultaten van deze tests gaven aan waar de verschillen tussen de operating system en browser combinaties lagen en welke events eventueel niet ondersteund worden of anders worden aangeleverd dan verwacht. De conclusie die uiteindelijk getrokken kon worden was dat het grootste verschil zat tussen het gebruik van een desktop operating system of een mobile operating system. Voornamelijk bij de verwachte output van events en wat er daadwerkelijk werd aangeleverd als output.

4.4.2 “Welke input methoden ondersteunen de concurrenten?”

Voor het beantwoorden van deze deelvraag is er een experiment uitgevoerd waarbij een selectie van de gevonden input methoden in de voorgaande deelvraag getest wordt in een selectie van concurrenten. De selectie van input methode is gebaseerd op de aangegeven prioriteit vanuit Liones, de waarschijnlijkheid van gebruik en of een methode niet simpelweg een alternatieve manier is voor de hoofdmethode van input. Dit leverde de volgende lijst op:

- Physical keyboard
- Muis
- Software keyboard
- Stylus
- Touchscreen

[4] Topic op stack overflow waar een antwoord te vinden is betreffende de menselijk merkbare vertraging: http://stackoverflow.com/questions/6880856/whats-the-minimum-lag-detectable-by-a-human

De selectie concurrenten was aangegeven vanuit Liones zelf en was als volgt:

- Xeditor[5]
  'Met Xeditor, de web based XML-Editor, kunnen auteurs gestructureerde documenten in XML data format creëren, door middel van een media neutrale data opslag. De gebruiksvriendelijke front-end leidt auteurs intuïtief via de gedefinieerde documentstructuur (XSD / DTD). De real-time validatie functie voorkomt onjuiste bewerkingen.'
- Google Docs[6]
  "Google Docs is een online dienst van Google. Met deze dienst kunnen opgemaakte documenten, spreadsheets en presentaties aangemaakt worden, die online worden opgeslagen."
- Google Script[7]
  Google Script is een online dienst van Google. Met deze dienst kunnen er projecten gebouwd worden in Google Apps Script. Deze worden dan opgeslagen op Google Drive of uitgebracht in de Chrome Web Store.
- WebODF[8]
  'WebODF is een JavaScript library die het mogelijk maakt om gemakkelijk Open Document Format (ODF) ondersteuning toe te voegen aan je website en mobile en desktop applicaties.'
- Ace[9]
  'Ace is een integreerbare code editor geschreven in JavaScript. Het heeft dezelfde features als bijvoorbeeld Sublime Text, Vim en TextMate.'
- CodeMirror[10]
  'CodeMirror is een veelzijdige text editor geïmplementeerd in JavaScript voor de browser.'

Voor de testen met een fysiek toetsenbord (physical keyboard) is er ook gebruik gemaakt van verschillende talen:

- US_English
  De standaard taal van het operating system waarop gewerkt werd. Maakt geen gebruik van speciale invoer mogelijkheden en gebruikt het meest gebruikte layout voor keyboards.
- Japanese
  Maakt gebruik van compositions en een IME.
- Chinese(Simplified)
  Maakt gebruik van compositions en de Pinyin IME.
- Arabic
  Arabic maakt gebruik van een schrijfrichting van rechts naar links en hierbij moet de caret aan de linkerkant getoond worden en moet de schrijfrichting ook correct aangepast

[5] Officiële Xeditor website: http://www.xeditor.com/
[6] Officiële Google Docs website: http://docs.google.com/
[7] Officiële Google Script website: http://www.google.com/script/start/
[8] Officiële WebODF website: http://www.webodf.org/
[9] Officiële Ace website: http://ace.c9.io/
[10] Officiële Code Mirror website: http://codemirror.net/

worden.

Het experiment is uitgevoerd op de volgende operating systems:

- Windows
- Mac OS
- iOS
- Android

De keuze om Windows Phone en BlackBerry OS niet mee te nemen in het experiment is gedaan om de scope te beperken. Deze twee operating systems vielen af vanwege het lage marktaandeel. Deze scope beperking is ook doorgetrokken voor de rest van het onderzoek.

Tijdens het experiment zijn voor elke input methode per operating system alle concurrenten afgegaan. Zo is bijvoorbeeld eerst op Windows elke concurrent getest hoe deze reageerde op input geleverd door een fysiek toetsenbord. Hierbij zijn ook de verschillende talen getest. Daarna is hetzelfde proces doorlopen voor de andere operating systems voordat er gewisseld werd naar een andere input methode. Dit zorgde er voor dat er zo accuraat mogelijk vergelijkingen gemaakt konden worden.

De resultaten voor dit experiment waren gevarieerd. Zo was het duidelijk dat uit alle concurrenten Google Docs het beste reageerde op de verschillende input methoden, ongeacht het gebruikte operating system. In tegenstelling tot WebODF dat in bepaalde operating system en browser combinaties compleet niet ingeladen kon worden. De resultaten van de concurrent die het meeste overeenkomt met FontoXML waren het meest interessant. Zo was het duidelijk dat hier nog problemen waren met het verwerken van talen die gebruik maken van compositions. Dit leverde namelijk onwenselijke aanpassingen in het onderliggende XML op. Deze resultaten verschilde ook weer per gebruikte browser en operating system.

De conclusie die gemaakt kon worden aan het einde van het experiment was dat, ondanks er bij meeste operating system en concurrent combinaties wel een probleem zit, er voor elke input methode wel een combinatie bestond waarbij de input methode correct werkte. Het belang van deze conclusie was dat het aantoonde dat het mogelijk zou moeten zijn om voor het proof of concept oplossingen te kunnen maken voor deze input methoden.

4.4.3 “Welke van de input methoden zullen door de eindgebruikers van FontoXML gebruikt worden?”

Het beantwoorden van deze deelvraag is gedaan door het bekijken van de gemaakte persona (profielen van niet bestaande mensen die representatief zijn voor een archetype van gebruiker) en het bespreken van waarschijnlijkheid van gebruik samen met Bert Willems. Normaliter zou een deelvraag zoals deze beantwoordt worden door het uitvoeren van kwantitatief onderzoek, namelijk het enquêteren van huidige gebruikers, maar omdat FontoXML momenteel nog niet in gebruik is was daar geen mogelijkheid toe. Na het bestuderen van de persona en het overleg was de conclusie dat de eindgebruikers naast hun computers eventueel tablets zouden gebruiken voor FontoXML met de daarbij behorende input methoden en dat eindgebruikers uit regionen van de wereld waar gebruik wordt gemaakt van aparte input methoden zoals IME’s gebruik zouden willen maken van FontoXML.

4.4.4 “Hoe gaan code editors om met het verwerken van input methoden die gebruik maken van compositions, IME’s of een andere schrijfrichting?”

Deze deelvraag lijkt in eerste instantie niet echt een relatie te hebben met de hoofdvraag, maar als we kijken naar waar compositions gebruikt worden (namelijk door de IME’s) en dat IME’s een manier van input leveren is. Vinden we toch de relatie tot de hoofdvraag. Het belang om dit te toetsen in code editors was om te ondervinden hoe zij dit oploste in het visuele gebied. Dit biedt dan inzicht in wat de gebruikers van deze code editors wellicht belangrijk vonden. Het experiment dat uitgevoerd is voor het antwoord hierop bestond uit een selectie maken van code editors, dit is gedaan door de vier meest gebruikte code editors binnen Liones (Sublime Text 3, PHPStorm, Notepad++, Microsoft Visual Studio) te gebruiken, en hier invoer in te testen in het Japans (fig 4.8), Chinees (fig 4.9) en Arabisch (fig 4.10). Voor deze testen is alleen de invoertaal aangepast en niet de taal van het operating system zelf. Dit houdt in dat alle menu items en dergelijke nog in het Engels zijn.

FIG 4.8

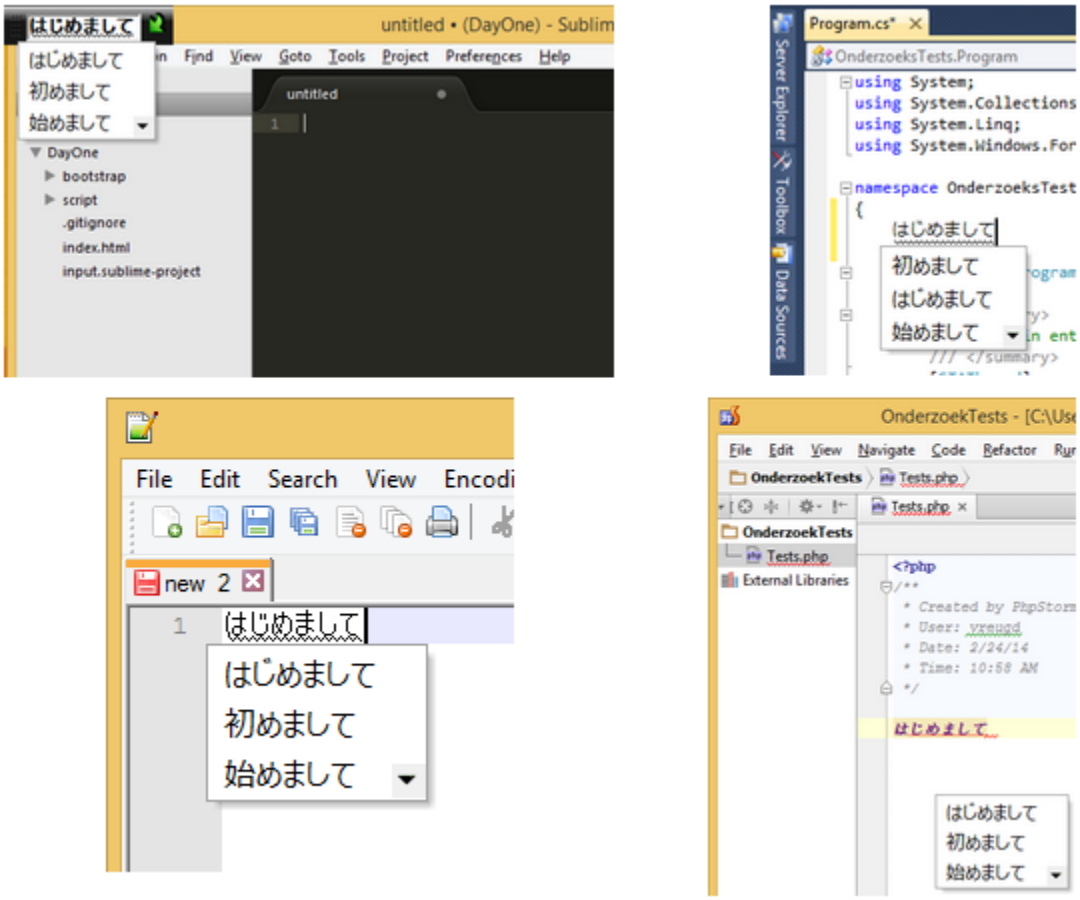
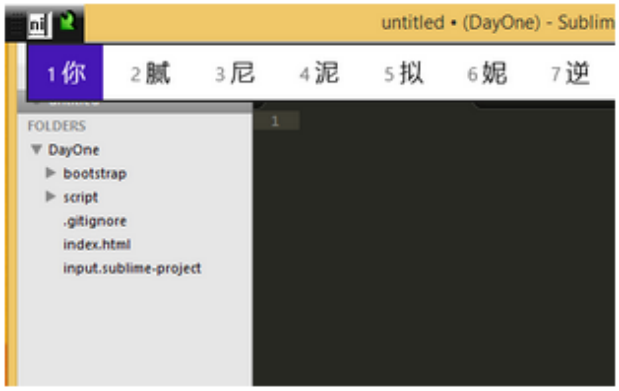
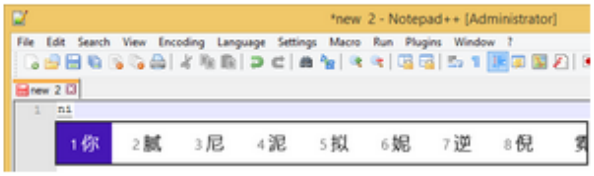
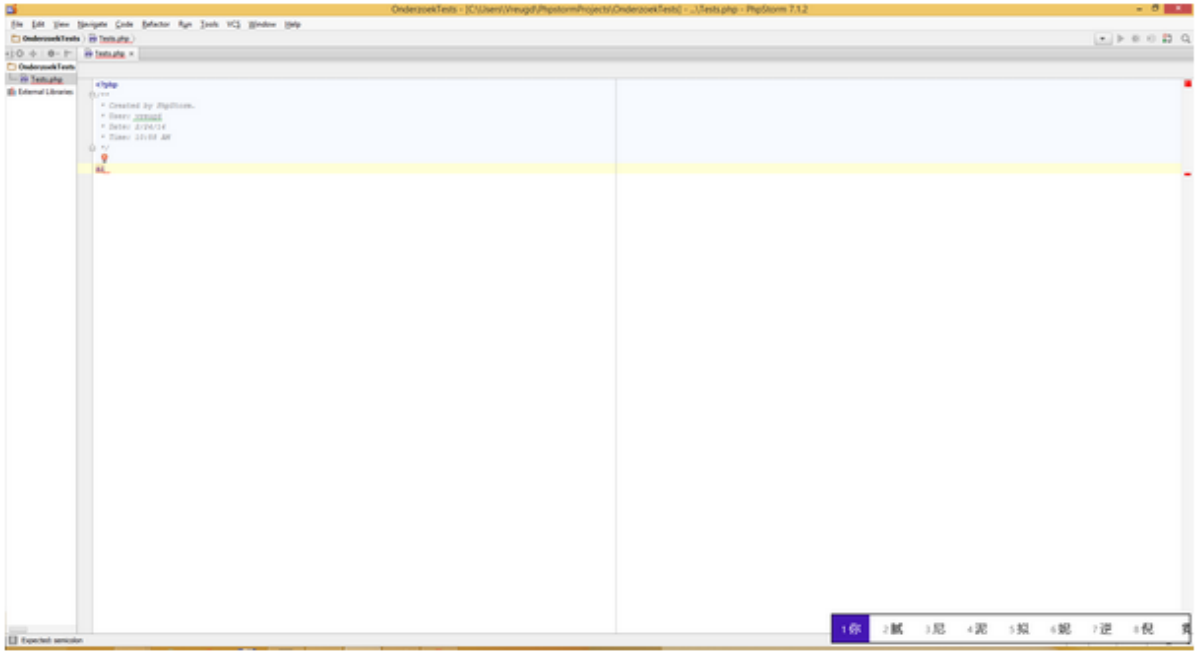


FIG 4.9


untitled • (DayOne) - Sublim
1 你 2 膩 3 尼 4 泥 5 拟 6 妮 7 逆
FOLDERS
DayOne
bootstrap
script
.gitignore
index.html
input.sublime-project


*new 2 - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
new 2
ni
1 你 2 膩 3 尼 4 泥 5 拟 6 妮 7 逆 8 倪



/

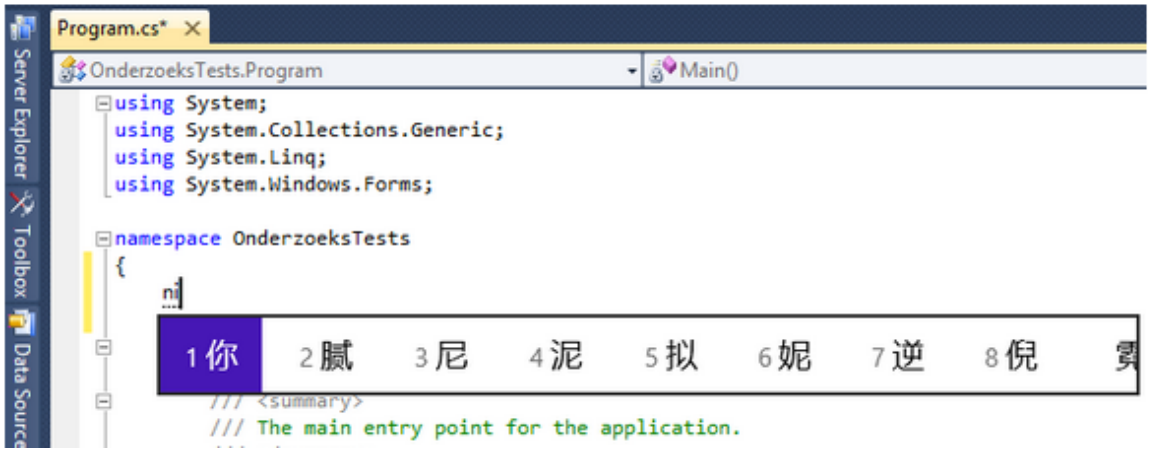
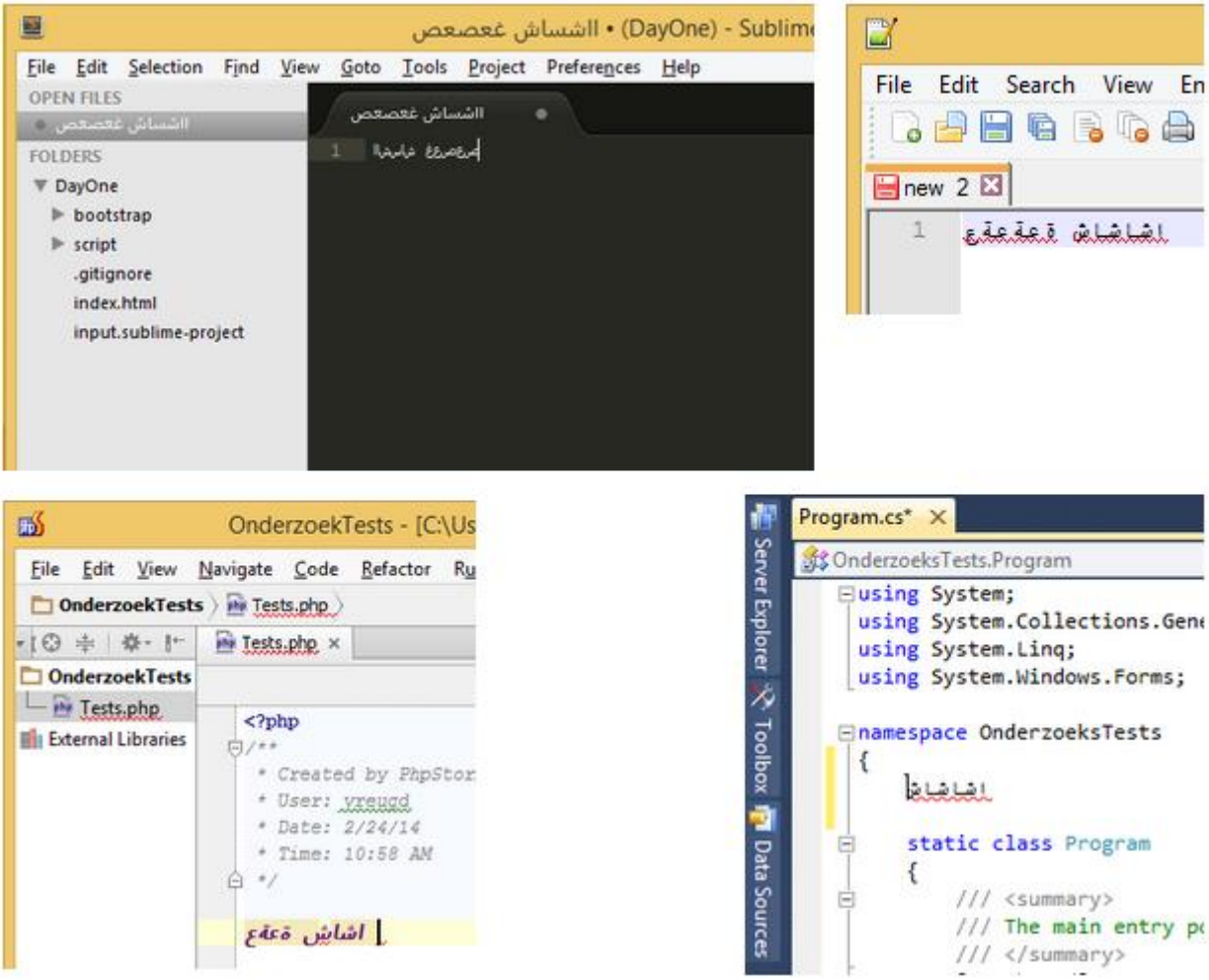

Program.cs*
OnderzoeksTests.Program
Main()
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;
namespace OnderzoeksTests
{
ni
1 你 2 膩 3 尼 4 泥 5 拟 6 妮 7 逆 8 倪
/// <summary>
/// The main entry point for the application.
Server Explorer
Toolbox
Data Sourc

FIG 4.10



Uit de resultaten kan geconcludeerd worden dat voor de talen die gebruik maken van compositions en IME's (Japans en Chinees) is nagedacht over het plaatsen van de IME zelf. Zo zien we dat Sublime Text 3 en PHPStorm deze geheel verplaatsen naar boven of onder in het scherm, op deze manier komen ze niet over onder- en/of bovenliggende tekst en blijft het overzicht bewaard. Voor Arabisch zien we dat alleen Microsoft Visual Studio het plaatsen van de caret, de streep die aangeeft waar getypt gaat worden, correct afhandelde. Het belang van deze resultaten voor de hoofdvraag is dat het aangeeft dat er elegante oplossingen zijn voor het gebruik van een IME. Waarbij het overzicht voor de rest van de content nog belangrijk is, zoals dit het geval is bij FontoXML, en dat het moeilijk is om het gebruik van een taal met een andere schrijfrichting correct te verwerken. Dit zou meetellen bij het maken van de conclusie op de hoofdvraag.

4.4.5 “Hoe gaan open source browsers om met het verwerken van input methoden die gebruik maken van compositions?”

Nadat deze deelvragen beantwoord waren werd het duidelijk dat ondanks de eerdere aangebrachte scope beperkingen het niet mogelijk zou worden om binnen de vooraf geplande tijd ook de deelvraag “Hoe gaan open source browsers om met het verwerken van input methoden die gebruik maken van compositions?”. Zoals aangegeven had deze deelvraag ook geen deel in het beantwoorden van de hoofdvraag. Deze deelvraag was puur bedoeld als hulpmiddel voor meer inzicht voor het te ontwikkelen proof of concept, hierdoor was het een logische stap om deze deelvraag niet te beantwoorden.

4.4.6 Eindconclusie

Voor het trekken van een conclusie betreffende de hoofdvraag van het onderzoek “Welke input methoden op welke devices en platforms moeten toegevoegd worden aan FontoXML?” is er gekeken naar de conclusies van de deelvragen. Deze leiden tot de volgende conclusie op de hoofdvraag. Om een breder publiek aan te kunnen spreken met FontoXML adviseer ik om ondersteuning te gaan aanbieden voor IME gebruikende talen en talen met een schrijfrichting van rechts naar links. Ook moet algemene ondersteuning aangeboden gaan worden voor de volgende combinaties:

- Android - Android browser
- Android - Chrome
- iOS - Safari
- iOS - Chrome

Deze combinaties zijn geselecteerd op de conclusies van de verschillende deelvragen en hoe goed deze presteerde bij het gebruik van Xeditor. De concurrent die het meest te vergelijken is met FontoXML. Ook zijn de meest populaire browsers uitgekozen voor beide tablet operating systems, deze presteerde misschien op sommige vlakken minder dan andere browsers maar mogen vanwege hun populariteit niet missen.

5. Opstellen adviesrapport

Om het resultaat van het onderzoek om te zetten in een advies voor Liones is er een adviesrapport opgesteld. In dit rapport staat een beknopte versie van het onderzoek met daarbij hoe dit uitgevoerd is en bevat ook het uiteindelijke advies gebaseerd hierop. De reden hiervoor is dat het belangrijk is om de resultaten van het onderzoek en het uiteindelijke advies in een kleiner format aan te bieden zodat men bij Liones een goed, maar snel overzicht kunnen krijgen over wat er precies is gedaan aan onderzoek en wat dit heeft opgeleverd zonder het uitgebreide onderzoeksrapport te hoeven doornemen.

Om dit onderdeel van het project af te sluiten is het advies overgedragen aan Bert Willems en Stef Busking. Dit gebeurde in de vorm van een adviesgesprek. Hierin zijn de resultaten van het onderzoek besproken en is het advies overgedragen. Hierna is besproken wat dit zou betekenen voor de uiteindelijke scope van het proof of concept. Dit wordt besproken in het volgende hoofdstuk.

6. Opstellen van requirements

6.1 Het doel

Het doel van het opstellen van requirements is om duidelijkheid te creëren betreffende wat het proof of concept precies moet bevatten. Het kan gezien worden als de punten waaraan het proof of concept gemeten wordt en waarmee bepaald wordt of het proof of concept voltooid is. Ook is het van belang voor het creëren van een product backlog zoals deze gebruikt wordt bij Scrum.

6.2 Activiteiten

Direct na het overdragen van het advies is er een lijst met requirements opgesteld. Dit is gedaan gebaseerd op het advies zelf en de verdere wensen vanuit Liones. Tijdens het opstellen van de requirements waren er discussies tussen de twee stakeholder, Stef Busking en Bert Willems, over zaken zoals in wat voor vorm het proof of concept zou komen en of het niet slim zou zijn om al direct wat scope beperking toe te passen door bijvoorbeeld niet twee browsers voor Android en iOS te ondersteunen maar er maar een te ondersteunen. Ik heb hierin een adviserende rol gespeeld om dit tot een set van requirements te kunnen omvormen.

Na dit gesprek is de volgende lijst van requirements opgesteld:
Het proof of concept moet:

- Ontwikkeld worden in de vorm van een JavaScript library voor een goede aansluiting op FontoXML.
- Event handling bevatten voor:
    - Standaard input(westers toetsenbord)
    - IME's
    - Compositions
    - Auto-completes / auto-corrects.
- Event handling bevatten voor het gebruik van Chrome op iOS en Android.
- Bij het afvangen van input alles bewaren en ook in de originele volgorde.
- Robuust en uitbreidbaar zijn.
- Hot-key handling moet vrij blijven zodat FontoXML deze door middel van aparte modules kan afhandelen. Speciale toetsen zoals backspace vallen ook onder hot-key handling.
- Als output een event stream aanbieden. Deze stream mag de DOM niet aanpassen zodat FontoXML de bewerking eerst kan valideren op de XML.
- Alleen actief zijn wanneer nodig, wanneer dit niet het geval is moeten er geen onnodige controles uitgevoerd worden zodat de library geen ongewenste invloed uitoefent.
- Een manier van monitoring aanbieden. Hierin moet duidelijk zijn welke strategie er toegepast wordt en waarom, zodat hier eventuele afwijkingen uit afgelezen kunnen worden.

Uit deze requirements is een product backlog opgesteld. Deze bevat de volgende drie categorieën: user stories, epics en prototypes. Een user story is een taak die nieuwe functionaliteit representeert, een epic is een taak die te groot is om binnen een sprint te kunnen

en een prototype is een taak die nodig is voor het maken van een beslissing maar waarvan het niet vast staat of de functionaliteit waarde zal hebben. Deze product backlog zag er als volgt uit:

- USER STORIES:
    - Basis structuur van de library aanbrengen
    - Event handling: westers toetsenbord
    - Output handling: westers toetsenbord
    - Event handling: IME's
    - Output handling: IME's
    - Event handling: compositions en auto-completes
    - Output handling: compositions en auto-completes
    - Event handling: Chrome op iOS
    - Output handling: Chrome op iOS
    - Event handling: Chrome op Android
    - Output handling: Chrome op Android
    - Sortering event handling
    - Hot-key handling
    - Monitoring
- EPICS:
    - Robuustheid en uitbreidbaarheid
    - Output stream restricties
    - Bepaling wanneer de library actief is
- PROTOTYPES:
    - Maak een basis JavaScript library

Er is hier gekozen om geen gebruik te maken van de standaard notatie van user stories waarbij elke user story geformuleerd wordt vanuit het oog van de gebruiker. Bijvoorbeeld: "Als gebruiker wil ik het proof of concept als library gebruiken". De reden hiervoor is omdat het voor het proof of concept nog niet duidelijk was hoe de gebruiker dit zou willen, dit zou tijdens de ontwikkeling ontdekt worden.

7. Opstellen initieel ontwerp

7.1 Het doel

Het doel van het opstellen van initieel ontwerp is, om voordat er met het bouwen begonnen wordt, al op een abstracte manier na te denken over wat voor problemen je gaat tegen komen. Door dit van te voren al in een ontwerp gemaakt met behulp van UML (Unified Modeling Language) op te lossen moet dit tijd winst opleveren tijdens het bouwen.

7.2 Activiteiten

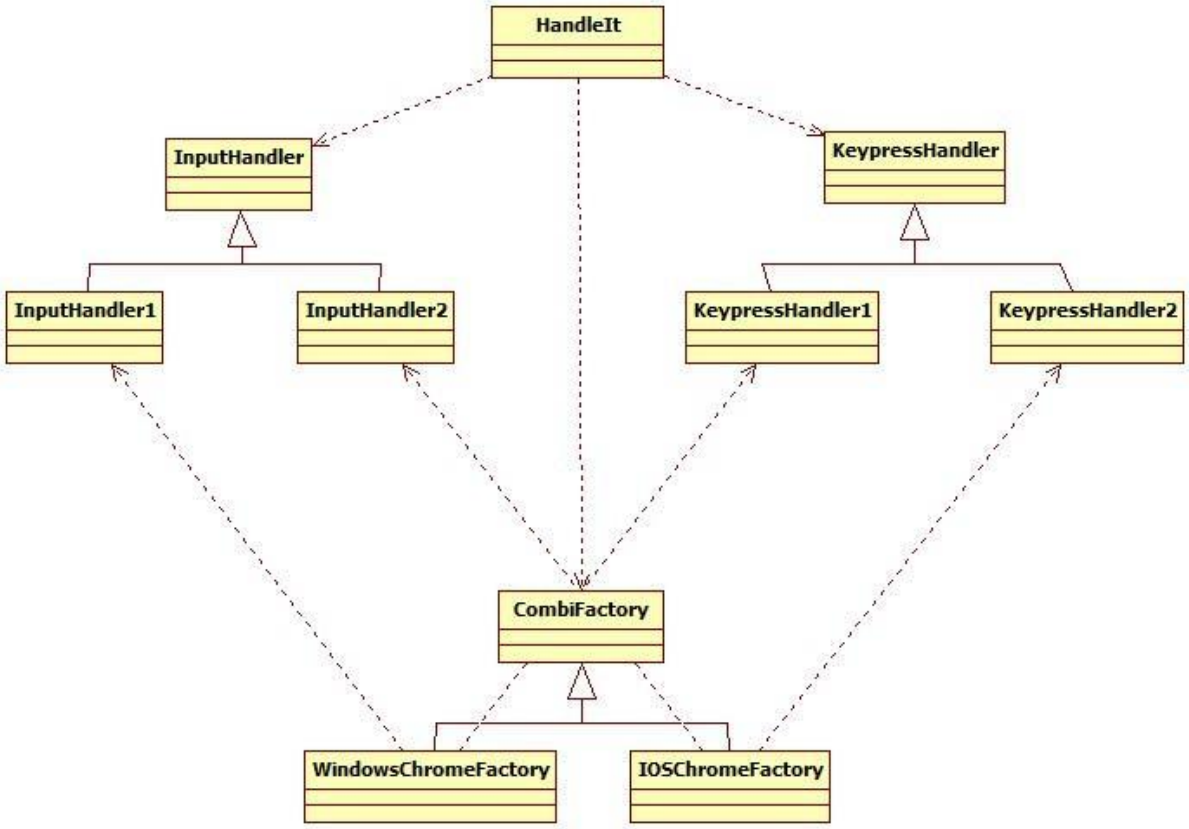
Voor het opstellen van het ontwerp moet er eerst nagedacht worden over de precieze "problemen" binnen het te ontwikkelen proof of concept. Deze analyse begon voor het proof of concept met vast leggen waar er precies verwachte verschillen zouden zijn in de afhandeling van de aangeleverde input events. De verwachting hierbij was dat dit bij de combinatie van operating system en browser zou liggen. Om dit in het ontwerp te kunnen verwerken is uitgezocht of er design patterns zijn waarmee dit opgelost kan worden. Een design pattern is een generiek opgezette softwarestructuur die een bepaald veelvoorkomend type software-ontwerpprobleem oplost. Dit moet gezien worden als een soort sjabloon waarmee het probleem kan worden aangepakt. Ook is er rekening gehouden met het feit dat het proof of concept in de vorm van een library moest zijn. *'Een library in informatica is een verzameling van implementaties van gedrag, geschreven in termen van een taal, met een goed gedefinieerde interface waarmee dit gedrag wordt aangeroepen.'*[11]

Een van de problemen die het design pattern zou moeten oplossen. Was dat er altijd gebruik gemaakt moest kunnen worden van dezelfde generieke aanspreekmethode zonder dat daarbij bekend was wat de exacte invulling was. Dit zou ervoor zorgen dat de invulling van deze methoden dynamisch gevuld zouden kunnen worden gebaseerd op operating system en browser. Het invullen van methoden gebaseerd op operating system is geen onbekend probleem binnen het ontwikkelen van software. De verwachting was dat er weinig hergebruik van code zou zijn en dat er vele unieke implementaties van dezelfde concepten nodig waren. Een design waar aan gedacht werd was het Abstract Factory pattern. Dit design pattern zorgt ervoor dat de software een generieke interface kan aanspreken en dat door middel van de Abstract Factory de invulling hiervoor wordt geregeld.

Tijdens het opstellen van het ontwerp werd het duidelijk dat het ontwerp niet afgemaakt kon worden zonder te beginnen met het ontwikkelen van het proof of concept. Zoals te zien is in het ontwerp (fig 7.1) is het niet duidelijke welke unieke implementaties er precies nodig zijn. Tijdens het ontwikkelen zou namelijk duidelijk worden welke unieke implementaties nou allemaal nodig zouden zijn en waar de verwerking misschien niet zo veel zou verschillen als van te voren gedacht. Omdat er gebruik gemaakt wordt van elementen van Scrum is dit niet erg. In Scrum wordt er namelijk gehanteerd dat er alleen het nodige ontworpen wordt.

[11] Bron van definitie library: http://en.wikipedia.org/wiki/Library_(computing)

FIG 7.1


HandleIt
InputHandler
KeypressHandler
InputHandler1
InputHandler2
KeypressHandler1
KeypressHandler2
CombiFactory
WindowsChromeFactory
IOSChromeFactory

8. Sprint 1: Basics en westers toetsenbord

8.1 Het doel

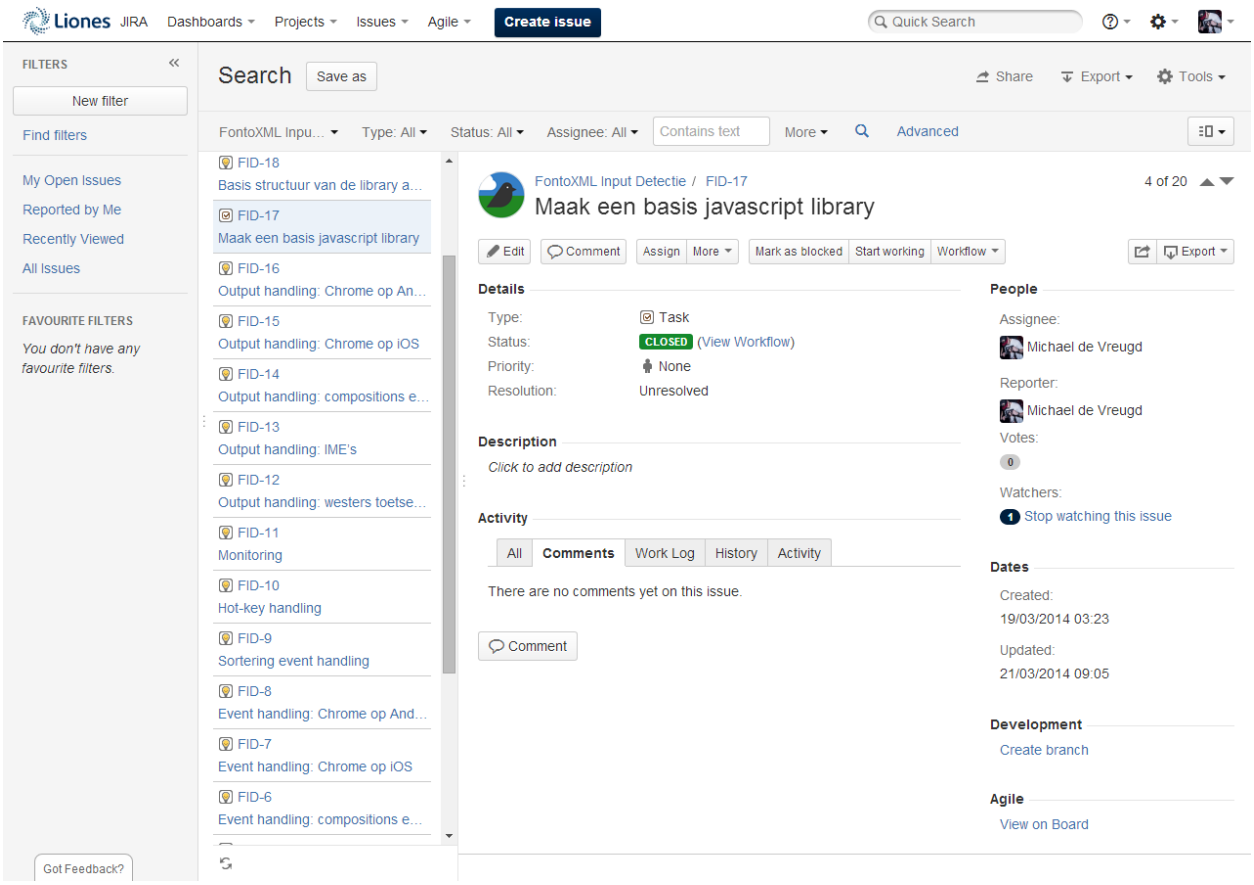
Het doel van deze eerste sprint was om de basis op te bouwen voor het proof of concept. Een proof of concept is een bewijs dat bepaalde van te voren bedachte concepten mogelijk zijn om te realiseren. De basis die gebouwd is in deze sprint kan gezien worden als de fundering, als dit niet sterk genoeg zou zijn zou de rest kunnen instorten. Wat dit in hield was dat de basis structuur voor de library aangebracht moest worden, er een manier moest zijn om bij te houden wat voor keuzes gemaakt worden voor de invulling van de methoden en dat de afhandeling van een westers toetsenbord met daarbij gebruik van een taal die gebruik maakt van het Latijnse alfabet. De keuze hiervoor was omdat hierbij de minste afwijkingen verwacht werden. Dit zou zorgen voor een basis waaraan de meer complexe afhandelingen toegevoegd zouden worden.

8.2 Activiteiten

Het eerste wat gedaan is, voordat er begonnen kon worden met de ontwikkeling, was uitzoeken hoe er precies een library gemaakt kan worden met gebruik van JavaScript. Omdat er nog geen ervaring was met het opstellen hiervan is er begonnen met een simpele basis, namelijk het volgen van een online tutorial. Dit zou een basis opleveren voor het proof of concept en tegelijkertijd zou er geleerd worden waarom een library zo is opgesteld en hoe dit dan precies werkt. In de tutorial wordt namelijk stap voor stap uitgelegd wat er gedaan moet worden en waarom dit gedaan wordt. Nadat de tutorial was doorgelopen is er ook nog gekeken naar de source code van jQuery. De keuze hiervoor was vanwege het feit dat jQuery een vergelijkbaar doel heeft, maar dan op DOM niveau. Namelijk de abstractie van browserverschillen en het aanbieden van een nette API.

Nadat dit gedaan was kon er echt begonnen worden met het werk voor de eerste sprint. Voor het bijhouden van het werk binnen een sprint wordt er gebruik gemaakt van een zogeheten sprint backlog. Deze sprint backlog wordt opgesteld door items uit de product backlog te nemen die relevant zijn voor de uit te voeren sprint en deze te bundelen tot een sprint backlog. Om deze zaken bij te houden wordt er bij Liones gebruik gemaakt van Jira(fig 8.1) een online tool gemaakt voor het bijhouden van de status van projecten en die hierbij ruimte biedt voor het specifiek bijhouden van Scrum projecten.

FIG 8.1



De sprint backlog bevatte de volgende user stories:

- Basis structuur van de library aanbrengen
- Event handling: westers toetsenbord
- Output handling: westers toetsenbord
- Sortering event handling
- Monitoring

De eerste user story die hierbij werd aangepakt was: “Basis structuur van de library aanbrengen”. Door het doorlopen van de tutorial voor het maken van een JavaScript library was hiervan al een groot deel gedaan. In ieder geval hoe er gecommuniceerd zou worden met de library vanuit de code waarin de library gebruikt zal worden.

Voor deze user story was het dan ook voornamelijk van belang dat er gerealiseerd werd hoe het opgestelde ontwerp met het Abstract Factory pattern erin verwerkt zou worden. Binnen dit design pattern wordt er gebruik gemaakt van overerving. Overerving is een concept binnen programmeren dat betekent dat een klasse eigenschappen, afhankelijkheden, functies en procedures erft van een superklasse. Terwijl dit binnen veel talen gedaan kan worden door dit simpel aan te geven bij de klasse zelf “Class x overerft van class y”, is dit binnen JavaScript niet

mogelijk. JavaScript kent hierbij een ander concept genaamd prototype inheritance. Feitelijk gezien werkt dit hetzelfde, maar hoe dit in de code opgesteld wordt is geheel anders. Omdat er geen ervaring was met deze manier van overerving koste het afronden van deze user story meer tijd dan verwacht.

Aan het einde kon het idee uitgevoerd worden waarbij de library altijd op dezelfde manier aangesproken kon worden zonder daar zelf aanlevering bij te doen welke operating system en browser er gebruikt worden. Hierbij werden dan de unieke implementaties voor alle event afhandelingen ingevuld gebaseerd op het operating system en browser. Dit sloot aan bij het oorspronkelijke idee waarin verwacht werd dat er veel unieke implementaties nodig zouden zijn.

Nadat de basis structuur aangebracht was, was het belangrijk om bij te kunnen houden wat de library precies voor keuzes maakte zonder daarvoor extra handelingen uit te moeten voeren. De user story die hier relevant aan was is: “Monitoring”. Deze user story hield in dat het mogelijk moest zijn om bij te houden wat er precies gebeurde met de library. Om dit te realiseren is er gebruik gemaakt van de mogelijkheid van JavaScript om naar een console in de browser te kunnen schrijven, een techniek die veel gebruikt wordt in de ontwikkeling van JavaScript applicaties. Hierbij wordt bij elke keuze gemaakt binnen de library, zoals bijvoorbeeld de specifieke implementatie voor de afhandeling van een event type, geschreven naar de console.

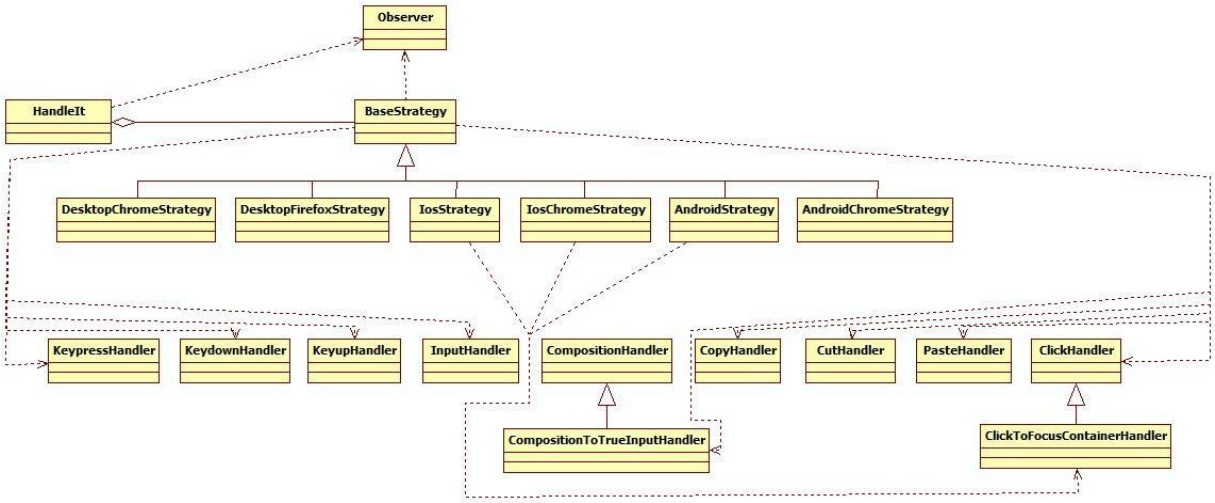
Met de basis structuur en monitoring op hun plek kon er begonnen worden met het verwerken van de eerste specificatie. De afhandeling van events en output voor een westers toetsenbord in combinatie met een taal die gebruik maakt van het Latijnse alfabet. Om dit te realiseren zijn de volgende stappen uitgevoerd. Voor elk van de input events die aangeleverd kunnen worden door een westers toetsenbord is er een basis afhandelingsmethode gemaakt. Binnen deze methode wordt de event opgevangen en wordt deze geschreven naar de console.

Nadat dit gebouwd was werd het duidelijk dat dit niet genoeg controle mogelijkheid zou opleveren voor verdere ontwikkeling en gebruik van de library. Vragen zoals “Maar wat als ik nou maar een beperkte selectie van events wilt ontvangen?” en “Hoe kan ik opvragen/bijhouden welke events er allemaal binnen gekomen zijn en naar buiten gestuurd zijn?” moeten in de toekomst worden opgelost. Hierdoor kwam het besluit om in de library een centraal punt aan te brengen waar alles wordt bijgehouden en waar het mogelijk zou zijn om zelf logica aan te leveren.

Dit centrale punt, een klasse genaamd “Observer”, is opgezet volgens het Singleton design pattern. Het Singleton design pattern zorgt ervoor dat het aantal objecten dat van de klasse kunnen worden aangemaakt beperkt zijn tot één. Dit zorgt ervoor dat de bijgehouden lijsten van binnenkomende en uitgaande events maar binnen één instantie zitten en dus opgevraagd kunnen worden. Deze klasse ging in een later stadium een nog veel belangrijkere rol spelen zoals omschreven zal worden in de komende twee hoofdstukken betreffende de twee volgende sprints.

Op de laatste dag van de sprint is er een code review geweest samen met een collega hierbij werd duidelijk dat de toepassing van het Abstract Factory pattern niet de beste oplossing was voor het probleem. De keuze was om dit om te zetten naar het gebruik van alleen het Strategy pattern. Een design pattern dat ook door het Abstract Factory pattern gebruikt wordt. Het Strategy pattern definieert een familie van algoritmes, kapselt elk algoritme en maakt deze wisselbaar binnen de familie. Dit zou er voor zorgen dat het DRY (Don't Repeat Yourself, een principe binnen software ontwikkeling die ervoor moet zorgen dat er zo min mogelijk herhaling is) principe beter toegepast kon worden. Omdat er niet meer gewerkt wordt met een interface waarin geen logica staat, maar alle basis algoritmes in een basis strategie te vinden zijn en alleen overschreven hoeven worden door een algoritme uit hun familie daar waar nodig. Het strategy pattern is eerst doorgevoerd in het ontwerp en daarna in de code. In het ontwerp (fig 8.2) is te zien dat er nu, in plaats van unieke implementaties per combinatie, vaker gebruik wordt gemaakt van dezelfde implementaties van event afhandeling. Omdat deze implementaties gebruik maken van dezelfde interface maakt het voor de rest van de library niet uit welke er gebruikt wordt. Hierbij is ook wat meer abstractie toegepast zoals het terug brengen van specifieke strategieën voor Mac OS en Windows naar een strategie voor "desktop operating systems".

FIG 8.2



Aan het einde van de sprint stond de user story "Sortering event handling" nog open. Omdat er binnen Scrum wordt gehanteerd dat de sprint wordt afgesloten ongeacht of alle user stories zijn afgehandeld is deze user story dan ook weer terug gezet op de product backlog. Aan het einde van elke sprint wordt er een retrospective gehouden. Hierin wordt besproken wat er goed ging en welke lessen er geleerd zijn. Hieruit werd duidelijk dat alle zaken redelijk goed waren verlopen en dat als les meegenomen kon worden dat er beter gepland kon worden welke issues er precies in de sprint zouden passen.

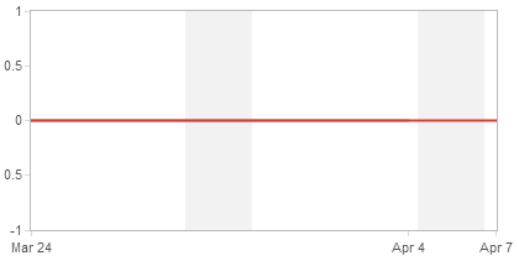
Ook kon er aan het einde van de sprint een burn down chart gemaakt worden met behulp van Jira (fig 8.3). In deze burn down chart zien we helaas een platte lijn. Dit komt omdat er voor

deze sprint geen storypoints toegekend waren en er geen optie is binnen Jira om dit achteraf alsnog toe te kennen om hier dan wel een verloop in te kunnen zien. Hier is rekening mee gehouden tijdens de overige sprints.

FIG 8.3

Sprint Report Basics & westers toetsenbord

Closed Sprint 24/Mar/14 8:30 AM - 07/Apr/14 8:49 AM Linked pages View Basics & westers toetsenbord in Issue Navigator



* Issue added to sprint after start time

Completed Issues View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (0)
FID-4 *	Event handling: westers toetsenbord	Story	None	CLOSED	-
FID-11 *	Monitoring	Story	None	CLOSED	-
FID-12 *	Output handling: westers toetsenbord	Story	None	CLOSED	-
FID-18 *	Basis structuur van de library aanbrengen	Story	None	CLOSED	-

Issues Not Completed View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (0)
FID-9 *	Sortering event handling	Story	None	IN PROGRESS	-

Als laatste stap is er aan het einde van de sprint weer een test gedaan met behulp van de in hoofdstuk 4 beschreven test tool. Deze test gaf aan dat de ontwikkelingen tot dus ver geen merkbare vertraging met zich mee brachten.

9. Sprint 2: Mobile devices en IME's

9.1 Het doel

Het doel voor de tweede sprint van het ontwikkelen van het proof of concept was om de basis die gemaakt was tijdens de eerste sprint uit te breiden met specificaties voor het gebruik van mobile devices en dan specifiek mobile devices met Android of iOS en IME's. Uit het onderzoek was gebleken dat de grootste verschillen tussen verwachte en daadwerkelijk ontvangen events namelijk bij deze twee groepen van input ligt. Liones had ook aangegeven dat hier voornamelijk hun focus ligt. Door de afhandeling hiervoor in de tweede en middelste sprint te plannen was er ruimte om hier eventueel user stories van over te houden en nog af te kunnen halen in de volgende sprint.

9.2 Activiteiten

Het eerste wat gedaan is, voordat er een selectie van user stories gemaakt kon worden, was het toekennen van story points aan de user stories. De telling die hierbij is toegekend is: één story point staat gelijk aan twee uur werk. Dit was om te voorkomen dat er bij het maken van een burn down chart aan het einde van de sprint er weer alleen een platte lijn zichtbaar zou zijn zoals bij de voorgaande sprint. Ook was er tijdens de voorgaande wekelijkse retrospective met Bert Willems en Stef Busking naar voren gekomen dat het verstandig zou zijn om een eigen implementatie te maken van de Event klasse zoals deze bekend is binnen JavaScript. Dit zou ervoor zorgen dat wanneer er in de toekomst veranderingen zouden zijn aan de klasse die effect hebben op hoe de library hier gebruik van maakt. Deze op een centrale plek verwerkt moeten worden binnen de library en de rest van de library nog steeds gebruik kan maken van de Event klasse zoals dat voorheen gebeurde. Ook biedt dit mogelijkheden betreffende het normaliseren van implementaties van attributen en methoden die niet beschikbaar zijn binnen elke browser. Voor het implementeren van deze verandering is er dus een user story aangemaakt zodat deze meegenomen kon worden met deze sprint.

De sprint backlog bevatte de volgende user stories:

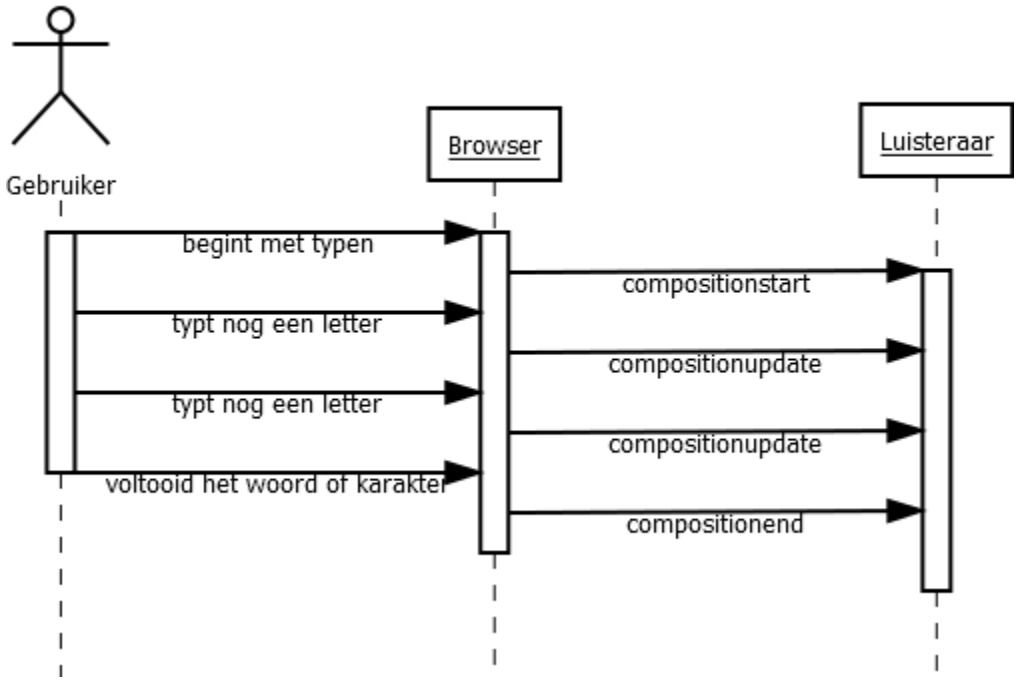
- Event handling: IME's
- Event handling: compositions en auto-completes
- Event handling: Chrome op iOS
- Event handling: Chrome op Android
- Decorate Event for future changes

9.2.1 Realisatie van event handling voor IME's

De eerste user story die hierbij afgehandeld is was event handling voor IME's. De keuze hiervoor was dat de verwachting was dat hier ideeën en concepten uit zouden komen die hergebruikt konden worden voor het afhandelen van events op Android en iOS. Het eerste idee wat hiervoor naar voren kwam was om een zogeheten "input trap" te implementeren. Wat dit inhoudt is dat de gebruiker niet meer in het oorspronkelijke element aan het typen is, maar dat de invoer "gevangen" wordt in een nieuw ingevoegd element. De gebruiker merkt hier zelf dan niets van. Hier kan dan gemakkelijk de input uit worden gelezen. Bij het gebruik van een IME is er een stroom van events die altijd hetzelfde is. Zodra er begonnen wordt met typen komt er

een “compositionstart” event door, daarna wordt voor elke toets aanslag een “compositionupdate” event doorgegeven en wordt er geëindigd met een “compositionend” event wanneer het gewenste karakter is ingevoerd (fig 9.1).

FIG 9.1



Om de input trap te realiseren wordt zodra er een compositionstart event binnenkomt een nieuw element in de html geplaatst. Hierna wordt de caret verplaatst naar dit element en typt de gebruiker dus in het element. Zodra er dan een compositionend event binnenkomt wordt de ingevoerde tekst opgehaald uit het nieuw geplaatste element. Daarna wordt dit element weer verwijderd en wordt de ingevoerde tekst geplaatst op de oorspronkelijke locatie van de caret. De keuze hiervoor komt voort uit het feit dat een van de doelen van de library is om alleen de daadwerkelijk ingevoerde waardes door te laten. Dit houdt in dat alle events die voor het compositionend event vallen niet van belang zijn. Bij het compositionend event weten we dat de gebruiker klaar is met het invoeren van bijvoorbeeld een woord in het Japans. Deze kan dan uitgelezen worden en door de library doorgegeven worden in de output. Na het bespreken van deze aanpak met Stef Busking en Bert Willems werd het duidelijk dat het niet wenselijk is om een nieuw element in de content zelf te plaatsen. Dit zou in het geval van FontoXML voor problemen kunnen zorgen. Dit stoort namelijk met het automatisch bijwerken van de HTML wanneer de XML verandert.

Het idee dat hierop volgde was dan ook om wel gebruik te maken van een input trap, maar het element dat hiervoor gebruikt wordt buiten de originele elementen te plaatsen. Dit element bevindt zich dan nog wel in het body element, maar zit dan helemaal onderaan. Dit element zou dan alleen visueel getoond worden op de oorspronkelijk plaats van de caret. Hierdoor zou de illusie ontstaan voor de gebruiker dat er nog altijd in het oorspronkelijke element getypt wordt.

Alhoewel dit zo best simpel klinkt zit hier toch een redelijke complexiteit aan. Het berekenen van de locatie van de caret om zo daar het element te plaatsen bleek afhankelijk van zeer veel factoren zoals de plaatsing van het element van oorsprong binnen de gehele pagina, de hoeveelheid al aanwezige karakters in het element van oorsprong en bijvoorbeeld spaties en enters hierin. Dit is dan ook uiteindelijk niet gelukt. Wel kwam naar voren dat de afhandeling van IME events veel makkelijker gedaan kon worden. Het compositionend event bleek namelijk al de uiteindelijk ingevoerde waarde in zich te hebben. Doordat er al begonnen was met te complex nadenken over het probleem was deze simpele oplossing geheel onopgemerkt gebleven. De uiteindelijke oplossing was dan ook om uit het compositionend event de waarde te halen en alleen deze door te geven in de output. Toen deze methode getest werd in de Chrome browser op Android bleek dat deze zonder aanpassing ook werkte. Chrome op Android maakt namelijk ook gebruik van de drie composition events, ongeacht in welke taal er getypt wordt. Dit was een belangrijke vondst omdat bij het gebruik van Chrome op Android geen normale keypress events doorkomen.

9.2.2 Complexiteit van event handling op iOS en afhandeling voor autocorrect

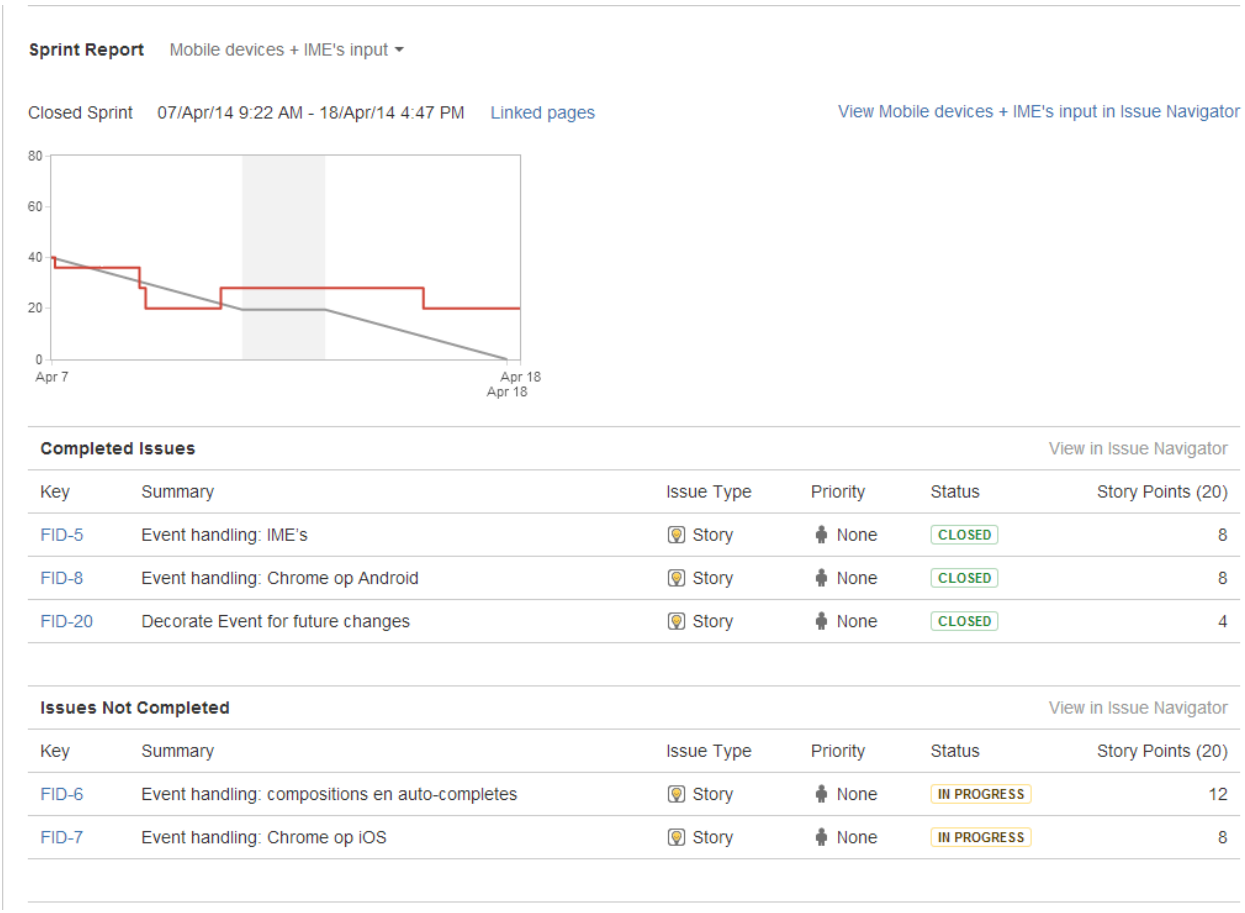
Na dit directe succes met Chrome op Android was er de hoop dat deze oplossing ook in ieder geval deels zou werken voor Chrome op iOS. Helaas was dit niet zo. Omdat Chrome op iOS niet zozeer echt de Chrome browser is maar eerder de Safari browser met een Chrome skin reageerde deze dan ook grotendeels zoals je zou verwachten van de Safari browser. Dit houdt in dat in tegenstelling tot Chrome op Android er wel keypress events doorkomen in plaats van de drie composition events. Hierdoor was het afhandelen in de basis eenvoudiger. Echter voor het automatisch corrigeren van woorden wordt helemaal geen melding gedaan in de vorm van een event. Dit houdt in dat wanneer je puur alleen naar de keypress events zou kijken je niet het correct beeld door krijgt van wat er nou daadwerkelijk ingevoerd is. Zo zal er wanneer je bijvoorbeeld in het Nederlands “halo” typt in plaats van “hallo” vier keypress events doorkomen. Namelijk één per letter, maar omdat er door iOS een automatische correctie is uitgevoerd naar “hallo” komt dit niet overeen met de uiteindelijk geleverde input.

Om automatisch gecorrigeerde woorden af te kunnen handelen was dus een andere aanpak nodig. Het eerste idee wat hiervoor naar voren kwam was om de complexe oplossing die oorspronkelijk bedacht was voor het afhandelen van de input van IME’s aan te passen zodat het herbruikbaar zou zijn voor dit probleem. Wat hiervoor aangepast moest worden was het moment wanneer de inputtrap geplaatst zou worden, dit moest nu op het keydown event in plaats van het compositionstart event, en een oplossing voor het probleem van het tonen van de inputtrap op de plaats van de caret. Het idee was hierbij om in plaats van de container visueel op de plaats van de caret te laten zien deze onzichtbaar te laten zijn. Dan zou hier daadwerkelijk in getypt worden, maar door tegelijk ook de getypte karakters op de oorspronkelijke locatie van de caret te plaatsen zou de illusie ontstaan dat daar getypt wordt. Nadat voor deze methode een prototype gemaakt was dat werkte was duidelijk dat dit geen optie was als oplossing. Dit zorgde voor visuele problemen zoals het raar zien verspringen van de caret en het langzaam of niet correct tonen van de getypte letters. Ook bleek dat deze methode compleet niet werkte op iOS. De caret weigerde zich te verplaatsen en uiteindelijk kon er compleet niet meer getypt worden. Na wat gerichte zoekopdrachten op dit specifieke probleem was het al snel duidelijk waarom dit niet werkte voor iOS. iOS zorgt er namelijk voor

dat de caret niet verplaatst mag worden[12]. De logica hierachter zou zijn dat er veel websites zijn die bij binnenkomst de caret direct in een zoekveld plaatsen. Als iOS dit toestaat dan zou het veroorzaken dat wanneer de gebruiker navigeert naar een website waar dit gebeurd direct het onscreen toetsenbord getoond zou worden. Dit is volgens Apple niet wenselijk.

Uiteindelijk werd deze sprint afgesloten met nog twee user stories niet opgelost, namelijk: Event handling: Chrome op iOS en Event handling: compositions en auto-completes. Deze user stories zouden dan ook meegenomen moeten worden in de volgende sprint. In de burn down chart(fig 9.2) voor deze sprint is goed te zien hoe na het bespreken van de in eerste instantie toegepaste oplossing voor IME's de user story weer geopend is en opnieuw is afgehandeld.

FIG 9.2



Sprint Report Mobile devices + IME's input

Closed Sprint 07/Apr/14 9:22 AM - 18/Apr/14 4:47 PM Linked pages View Mobile devices + IME's input in Issue Navigator

Completed Issues View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (20)
FID-5	Event handling: IME's	Story	None	CLOSED	8
FID-8	Event handling: Chrome op Android	Story	None	CLOSED	8
FID-20	Decorate Event for future changes	Story	None	CLOSED	4

Issues Not Completed View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (20)
FID-6	Event handling: compositions en auto-completes	Story	None	IN PROGRESS	12
FID-7	Event handling: Chrome op iOS	Story	None	IN PROGRESS	8

Als laatste stap is er aan het einde van de sprint wederom een test gedaan met behulp van de in hoofdstuk 4 beschreven test tool. Deze test gaf aan dat ondanks het steeds complexer worden van de library er tot dus ver geen merkbare vertraging waren.

[12] Thread op het officiële apple forum waarin wordt uitgelegd dat dit niet kan en waarom dit niet kan: https://discussions.apple.com/thread/4710766?tstart=0

10. Sprint 3: iOS, output en intelligentie

10.1 Het doel

Het doel van deze sprint was om zo snel mogelijk de afhandeling voor events op iOS en autocomplete en autocorrect afhandeling af te maken. Zodat de overige user stories van de product backlog ook nog verwerkt konden worden. Deze user stories betroffen de afhandeling van output door de library en het toevoegen van patroon herkenning. Deze patroon herkenning zou ervoor moeten zorgen dat de library niet meer vast zou zitten aan instellingen voor een specifieke browser en operating system combinatie. In plaats hiervan zou een gebruiker eerst gevraagd worden om een paar handelingen uit te voeren waaraan herkent kan worden welke manier van event afhandeling toegepast moet worden.

10.2 Activiteiten

Voordat er met deze sprint begonnen kon worden was het verstandig om de toegekende story points omhoog te brengen naar het punt waar één story point gelijk staat aan één uur werk. Dit zou er voor zorgen dat de burn down chart een meer accurate representatie zijn van de gespendeerde tijd. Hierna is er weer een sprint backlog opgesteld. Voor deze sprint waren dat alle overige user stories van de product backlog.

De sprint backlog bevatte de volgende user stories:

- Event handling: compositions en auto-completes
- Event handling: Chrome op iOS
- Sortering event handling
- Hot-key handling
- Output handling: IME's
- Output handling: compositions en auto-completes
- Output handling: Chrome op iOS
- Output handling: Chrome op Android
- Suggestie algoritme bij geen combination factory optie

10.2.1 Vervolg iOS en autocorrect

De eerste twee user stories waaraan gewerkt werd waren de twee user stories die nog open stonden aan het inde van de voorgaande sprint. Namelijk het afhandelen van events op Chrome op iOS en afhandelen van compositions en auto-completes. Zoals aangegeven in het laatste hoofdstuk was hierbij een nieuwe complexiteit omhoog gekomen waarbij op iOS de caret niet verplaatst mag worden. Dit hield in dat voorgaande oplossingen niet hergebruikt konden worden voor dit probleem. Dit zorgde er voor dat er naar wat meer creatieve oplossingen gezocht werd. Zo was de eerst volgende oplossing die uitgeprobeerd werd één waarbij in plaats van de caret te verplaatsen naar een nieuw element, een "optische illusie" implementatie. Het element waar de caret in zit leeg te maken, de inhoud die hier in zat te verplaatsen naar een nieuw element en deze op de oorspronkelijke plek te plaatsen met de juiste grootte en style. Door dan bij het aanslaan van een spatie of enter de inhoud van het element uit te lezen en dit dan terug te plaatsen in de oorspronkelijke inhoud zou het eventueel gecorrigeerde woord doorgegeven kunnen worden. Nadat dit gedaan was wordt de content weer in het geheel terug geplaatst in

het oorspronkelijke element en wordt de oorspronkelijke style en grootte hier weer op toegepast.

Om dit te realiseren kwam de behoefte omhoog om te kunnen bepalen wanneer functies wel of niet uitgevoerd werden door de Observer klasse. Om dit te realiseren zijn er aanpassingen gemaakt aan deze klasse. Hier kunnen nu functies toegevoegd worden en deze kunnen dan op non-actief gezet worden wanneer nodig. Dit zou er voor zorgen dat deze zo lang ze niet uitgevoerd moeten worden ook niet uitgevoerd worden. Dan wanneer deze functies weer nodig zijn kunnen ze weer op actief gesteld worden.

Alhoewel het wel lukte om een prototype te maken van de “optische illusie” implementatie kon er niet verzekerd worden dat deze oplossing altijd zou werken. Omdat deze oplossing gebruik maakt van het verplaatsen van elementen zou het voor kunnen komen dat dit de layout van een website/web applicatie kapot kan maken. Dus ondanks dat deze oplossing werkend was voor het proof of concept is er besloten om deze oplossing niet te gebruiken. Het werk dat hiervoor gedaan was bracht wel de inspiratie voor een andere oplossing. Voor deze oplossing wordt op het moment van een click event (in het geval van iOS is dit wanneer het scherm aangeraakt wordt) binnen een tekst de huidige inhoud van het element opgesplitst in wat er voor de caret zit en wat er na de caret zit. Dit wordt hierna opgeslagen in een onzichtbaar element. Dan wanneer de gebruiker een spatie of enter aanslaat wordt de inhoud van het element waarin getypt wordt opgehaald. Hier wordt dan de oorspronkelijke inhoud, die voor het typen was opgeslagen, vanaf gehaald. Wat hierdoor overblijft, is het zojuist getypte woord(of letter), ook als deze automatisch gecorrigeerd is. Na deze oplossing uitgeprobeerd te hebben op iOS bleek dat deze werkte. Wel was het nog nodig om de acties niet alleen uit te voeren bij het invoeren van een spatie en een enter, maar ook als de gebruiker buiten het tekstveld klikt of op een ander element klikt. Deze acties zorgen er namelijk ook voor dat wat tot dan toe getypt is eventueel automatisch gecorrigeerd wordt.

Het vinden van een goed toepasbare oplossing voor het afhandelen van events op iOS en het afhandelen van automatische correcties had helaas veel tijd gekost. Het was hierdoor duidelijk dat het niet meer zou lukken om alle user stories die nog open stonden voor de sprint nog af te handelen. Het was dus noodzakelijk om een beslissing te maken welke van de user stories niet afgehandeld zouden worden. De factor waar hierbij rekening gehouden moet worden is: “Is dit van essentie voor het afleveren van een goed werkend proof of concept?”. Wat hierbij opvalt, is dat een groot deel van de overgebleven user stories onder output afhandeling vallen. De enige twee die hier niet onder vallen waren: “Hot-key handling” en “Suggestie algoritme bij geen combination factory optie”. Hoe de library omgaat met output is zeker een essentie voor een goed werkend proof of concept, want ook al wordt er nog zoveel gedaan met de binnenkomende events als dit niet correct uit de library gehaald kan worden heeft dit geen waarde. Deze user stories zouden dus sowieso afgehandeld moeten worden. Door de toevoeging van deze user stories was er realistisch gezien geen tijd om de “Suggestie algoritme bij geen combination factory optie” user story nog af te maken. De “Hot-key handling” user story was nog een mogelijkheid als de output afhandeling snel genoeg gedaan zou kunnen worden.

10.2.2 Output handling

Ondanks dat voor de afhandeling voor de output veel verschillende user stories waren aangemaakt kon dit meer gezien worden als één user story. Namelijk hoe de library omgaat met de output. Een eis vanuit Liones was dat de library een output stream moest aanbieden waar gemakkelijk toegang tot is. Een output stream is waar een programma, applicatie of een library de uitgaande data in plaatst. Deze is dan aanspreekbaar om gebruik van te maken. Om dit te realiseren worden zoals eerder aangegeven bij de Observer klasse alle binnenkomende en uitgaande events geregistreerd en vast gehouden. Om hier dan een output stream van te maken wordt bij elke registratie van een uitgaand event gecontroleerd of er een functie is geregistreerd om uitgevoerd te worden voor dat event type en deze wordt dan uitgevoerd. Deze functies kunnen aangeleverd worden door de gebruiker.

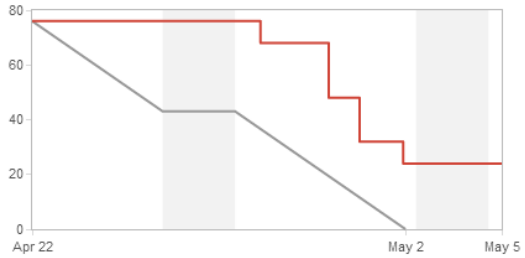
Voor de user story voor het afhandelen van hot-keys, toetsen(combinaties) zoals backspace en ctrl+B, moest gezorgd worden dat deze niet in de output stream terecht komen. De reden hiervoor is het feit dat deze zogeheten hot-keys in het geval van FontoXML een impact kunnen hebben op hoe de XML er uit ziet en wat voor regels hierbij komen kijken. Deze hot-keys worden dan ook door een aparte module afgehandeld. Door dit aan te geven in de Observer klasse worden deze dan ook niet aan de output stream toegevoegd.

Aan het einde van deze sprint stond de “Suggestie algoritme bij geen combination factory optie” user story nog open. Omdat dit de laatste sprint was zal deze user story dan ook niet meer afgehandeld worden tijdens het afstudeerproject. Wel is het de bedoeling dat de library uiteindelijk op de manier bedoelt in de user story gaat werken. Als we kijken naar de burn down chart (fig 10.1) aan het einde van deze sprint kunnen we zien dat het uitwerken van de iOS afhandeling en composition en auto-complete afhandeling veel meer tijd in beslag hebben genomen dan oorspronkelijk bedacht.

FIG 10.1

Sprint Report iOS, output and intelligence

Closed Sprint 22/Apr/14 9:30 AM - 05/May/14 8:46 AM Linked pages View iOS, output and intelligence in Issue Navigator



Completed Issues View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (52)
FID-6	Event handling: compositions en auto-completes	Story	None	CLOSED	12
FID-7	Event handling: Chrome op iOS	Story	None	CLOSED	8
FID-9	Sortering event handling	Story	None	CLOSED	8
FID-10	Hot-key handling	Story	None	CLOSED	8
FID-13	Output handling: IME's	Story	None	CLOSED	4
FID-14	Output handling: compositions en auto-completes	Story	None	CLOSED	4
FID-15	Output handling: Chrome op iOS	Story	None	CLOSED	4
FID-16	Output handling: Chrome op Android	Story	None	CLOSED	4

Issues Not Completed View in Issue Navigator

Key	Summary	Issue Type	Priority	Status	Story Points (24)
FID-19	Suggestie algoritme bij geen combination factory optie	Story	None	OPEN	24

Als laatste stap zijn er aan het einde van deze sprint nog ontwerpen gemaakt. In het klassendiagram (fig 10.2) wordt duidelijk gemaakt wat de uiteindelijke structuur is van de library met de daarbij behorende functions. In het sequentiediagram (fig 10.3) wordt duidelijk gemaakt hoe de library bepaald welke strategieën gekozen worden.

FIG 10.2

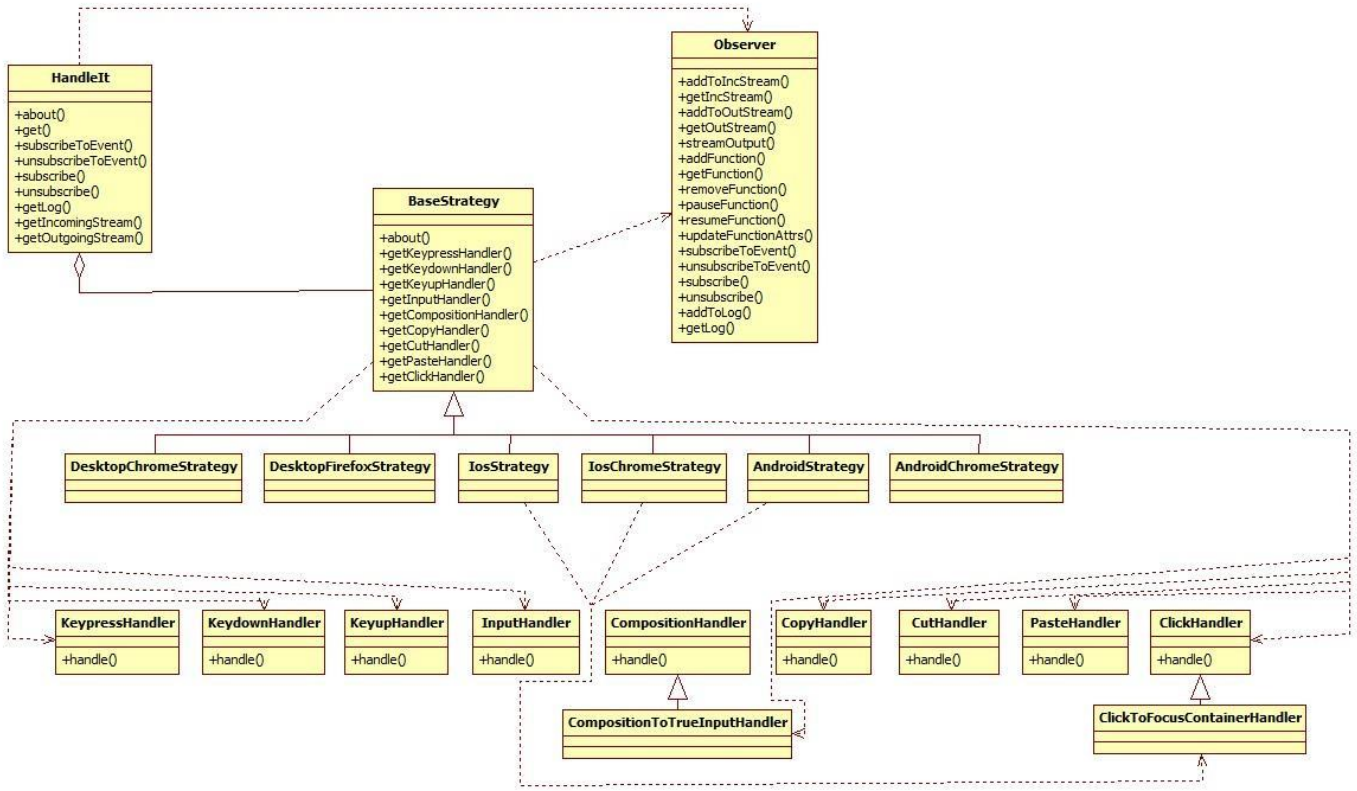
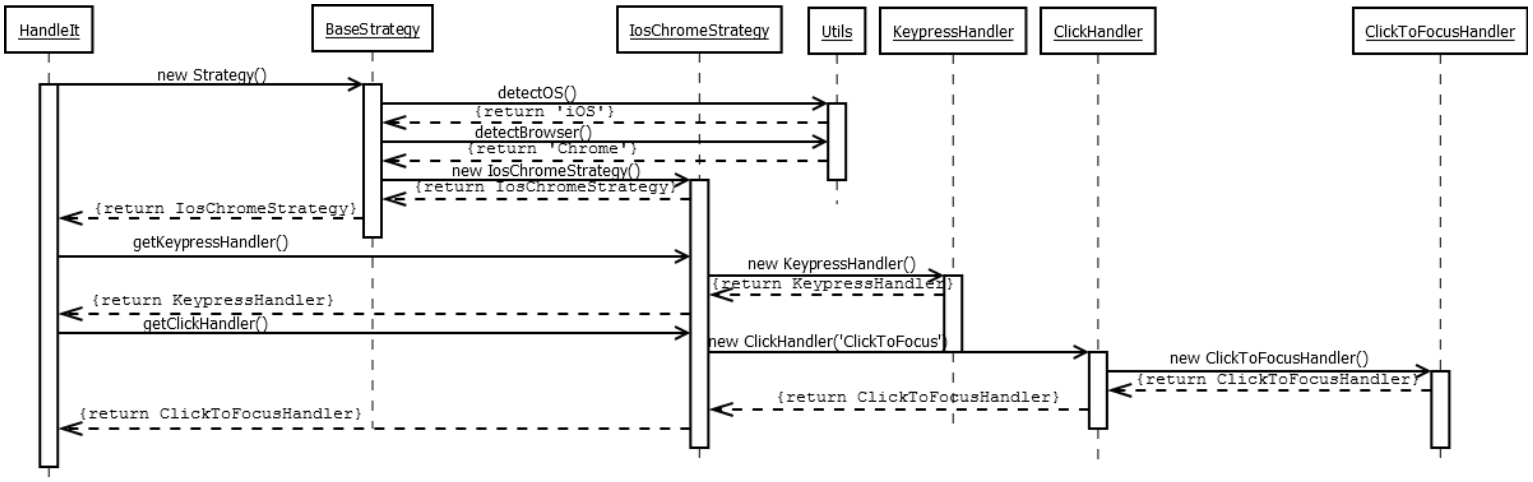


FIG 10.3

Voor dit sequence diagram is er aangenomen dat met iOS en Chrome wordt gewerkt. Tevens is er als voorbeeld alleen getoond hoe een standaard implementatie gevuld wordt en een unieke implementatie.



11. Opstellen testplan, uitvoeren tests en het documenteren van de resultaten

Nadat alle sprints voltooid waren was het van belang om tests op te stellen die de werking van het proof of concept in zijn huidige vorm moeten waarborgen. Dit brengt voor de library die getest is wat problemen met zich mee. Zo is de library bijvoorbeeld geheel afhankelijk van daadwerkelijke invoer geleverd bij het gebruik van een browser met een specifieke invoermethode.

Deze fase van het project begon met het opstellen van een testplan. Dit plan moet aangeven wat er wel en niet getest gaan worden van het proof of concept, wat hierbij de aanpak is, wat de uiteindelijke producten zijn uit het test proces, eventuele randvoorwaarden en nog vele andere aspecten. Om dit gestroomlijnd te documenteren in een testplan is er gebruik gemaakt van de IEEE 829-2008 structuur[13], ook bekend als "829 Standard for Software Test Documentation". Dit zorgt er voor dat er gestructureerd nagedacht wordt over het hoe en wat van het testen van het proof of concept. Zo is er eerst nagedacht over welke onderdelen van de library vanuit het perspectief van de gebruiker getest moeten worden. Voor dit project is dat als volgt geformuleerd:

- Afhandeling input bij het gebruik van Engels en Japans en een westers toetsenbord op de volgende browser en operating system combinaties:
    - Windows / Chrome
    - Windows / Firefox
    - Mac OS / Chrome
    - Mac OS / Firefox
    - Android / Chrome
    - Android / Android browser
    - iOS / Chrome
    - iOS / Safari

Dit dekt de werking van de library vanuit het perspectief van de gebruiker en de aspecten die verwerkt zijn in het proof of concept.

Hierna is bedacht hoe dit dan getest zou worden Wat de aanpak zou zijn. Voor het testen van de aspecten zoals hier boven geformuleerd zouden er acceptatie tests uitgevoerd worden. Het doel hiervan is om te toetsen of de library werkt zoals verwacht gebaseerd op de requirements. Wat hiervoor nodig is voor dit project is een zelf gebouwde interface waarin de gebruiker gevraagd wordt specifieke input te leveren. Dit zou dan door de library correct doorgegeven moeten worden in de output stream. Ook is de source code nodig zodat dit door middel van code reviews getoetst kan worden of dit voldoet aan de requirements. Het was tevens van belang om unit testen op te stellen, deze zouden dan direct kunnen dienen als regressie tests

[13] Wikipedia met uitleg wat deze standard inhoudt: http://en.wikipedia.org/wiki/IEEE_829, Officiele pagina voor de standaard: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4578383&url=http%3A%2F%2Fieeexplore.ieee.org%2Fstamp%2Fstamp.jsp%3Ftp%3D%26arnumber%3D4578383

zodat wanneer er in de toekomst verder gewerkt zal worden aan de library de interne werking intact blijft.

Voor de unit tests moest de interne werking van alle losse eenheden van de library getest worden, maar ook de gehele werking de library van input tot output. Het zou hiervoor het mooiste zijn als je dit geautomatiseerd kan testen. Zeker aangezien het goed mogelijk is dat in de toekomst de library nog veel uitgebreider zal worden in de selectie van browser en operating system combinaties. Het opstellen van de unit tests begon dan ook met de zoektocht naar een mogelijkheid om hiervoor tests op te stellen en geautomatiseerd te laten uitvoeren. Hiervoor is eerst gekeken naar wat de frameworks, die gebruikt worden voor het testen van JavaScript, hiervoor kunnen betekenen. Zoals bij test frameworks en libraries voor andere talen is hiervoor wel de optie om met gemak geautomatiseerd te testen en om meerdere browsers tegelijk te testen. Wat hierbij niet mogelijk was, en wat wel nodig was om de daadwerkelijke werking van het proof of concept te testen, is om echte fysiek geleverde input te testen. Dat is eigenlijk ook heel logisch, want het moet echt *fysiek* geleverde input zijn. Wat test frameworks hier eventueel alleen konden doen was input *simuleren*. Uiteindelijk is dan ook de keuze gemaakt om gebruik te maken van de test frameworks die gebruikt worden in het FontoXML project, zodat dit goed aansluit hierop. De libraries en frameworks die in FontoXML voor testen gebruikt worden zijn:

- Testem[14]
  'Test'Em Scripts! Een test runner die JavaScript unit testen leuk maakt.'
- Mocha[15]
  'Mocha is een feature-rich JavaScript test framework draaiend op node.js en de browser en maakt asynchroon testen simpel en leuk.'
- Chai[16]
  'Chai is een BDD / TDD assertion library voor node.js en de browser dat heerlijk samen met elk JavaScript testing framework gebruikt kan worden.'
- Sinon[17]
  'Standalone test spies, stubs en mocks voor JavaScript. Geen afhankelijkheden en werkt met elk unit testing framework.'
- RequireJS[18]
  'RequireJS is een JavaScript file en module loader.'

Na de rest van het testplan in te vullen met zaken zoals verantwoordelijkheden en planning moesten de tests uitgevoerd worden. Voor de unit tests moest hiervoor de tests geschreven worden en alle test frameworks ingesteld worden. Met het gebruik van het Testem framework is tegelijk alle tests draaien in verschillende browsers gemakkelijk. Waar wel aan gedacht moet worden hierbij is dat er voor deze operating system en browser combinaties andere invullingen van situaties ontstaan in de library. Er worden dus verschillende tests uitgevoerd, die wel dezelfde situaties testen, maar met verschillende invulling gebaseerd op operating system en

[14] Officiële GitHub pagina van Testem: https://github.com/airportyh/testem/
[15] Officiële Mocha pagina: http://visionmedia.github.io/mocha/
[16] Officiële Chai pagina: http://chaijs.com/
[17] Officiële Sinon pagina: http://sinonjs.org/
[18] Officiële RequireJS pagina: http://requirejs.org/

browsers. Testem wordt gebruikt door middel van terminal of de command prompt en biedt hierbij een interface (fig 11.1). Ook wordt er door gebruik van Mocha en Chai een web based interface (fig 11.2) aangeboden waarin aangegeven wordt of tests slagen of falen.

FIG 11.1

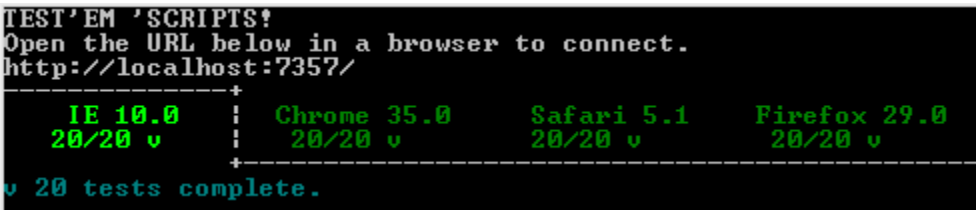
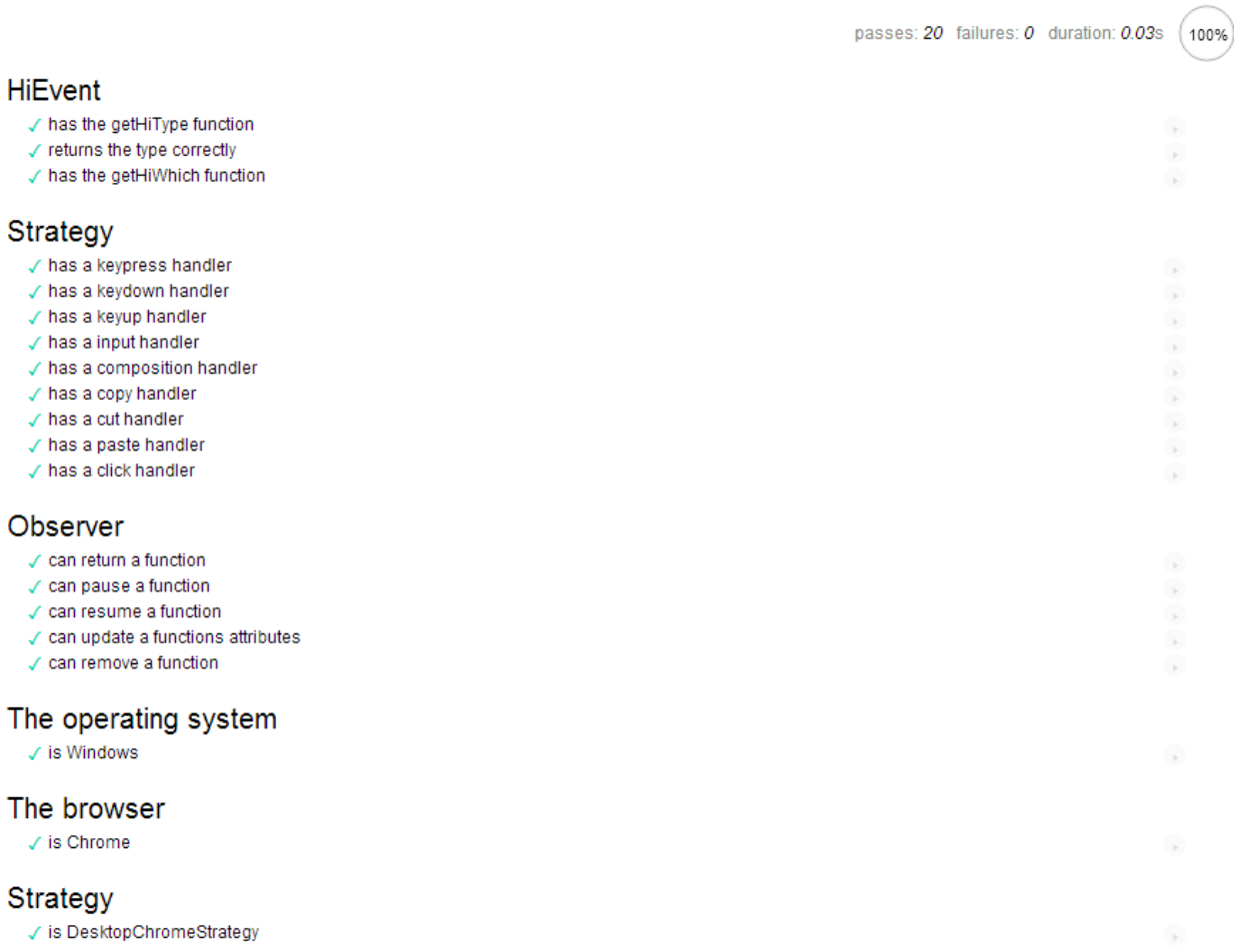


FIG 11.2



Omdat Testem werkt met het gebruik van een port op de localhost kan er ook met gemak getest worden op devices die werken met Android of iOS. Dit wordt gedaan door binnen hetzelfde netwerk connectie te maken met het IP-adres waarop Testem draait en deze te openen op de aangegeven port.

Door het gemak waarmee deze tests uitgevoerd kunnen worden werken deze perfect als regressietests. Elke keer wanneer de library veranderingen ondergaat zullen deze tests uitgevoerd worden voordat de veranderingen commit mogen worden. Ook zullen er in de toekomst meer tests bijgeschreven kunnen worden die dan ook automatisch uitgevoerd worden bij het uitvoeren van Testem.

De gebruikers acceptatie tests zijn voor het proof of concept op een redelijk officieuze manier uitgevoerd. Hierbij is het normaal het idee om de gebruiker de library te laten gebruiken en of dit dan overeenkomt met de requirements zoals deze gedefinieerd waren. Omdat er voor dit project geen tijd meer was om deze interface te bouwen is er gekozen om demo's te geven met de interface die er voor de ontwikkeling is gebruikt. Dit is gaandeweg de ontwikkeling van het proof of concept gedaan en aan de hand van de feedback hierop zijn er waar nodig aanpassingen gemaakt. Wanneer er in de toekomst gewerkt zal worden naar een release kandidaat van de library zal dit op een betere manier moeten gebeuren.

12. De productevaluatie

In dit hoofdstuk zal ik evalueren over de uiteindelijk opgeleverde producten. Dit zal ik doen door eerst een kleine omschrijving te geven van de opgeleverde producten. Daarna zal ik evalueren over de kwaliteit en het nut van de opgeleverde producten. Als laatste zal ik omschrijven hoe de opgeleverde producten verder in gebruik genomen moeten worden door Liones.

12.1 De producten

Dit project kende twee grote producten waarover echt geëvalueerd kan worden. Namelijk het onderzoeksrapport en het proof of concept.

Het onderzoeksrapport beantwoordt de hoofdvraag "*Welke input methoden op welke devices en platforms moeten toegevoegd worden aan FontoXML?*". Dit wordt gedaan door eerst antwoord te geven op de volgende deelvragen:

- Wat zijn de verschillen tussen alle input methoden op de verschillende operating systems, browsers en devices?
- Welke input methoden ondersteunen de concurrenten?
- Welke van de input methoden zullen door de eindgebruikers van FontoXML gebruikt worden?
- Hoe gaan code editors om met het verwerken van input methoden die gebruik maken van compositions, IME's of een andere schrijfrichting?

Het antwoord geven op deze vragen wordt gedaan door een mix van uitzoekwerk, onderzoek en het uitvoeren van experimenten.

Het proof of concept is een JavaScript library voor het omvormen van input op allerlei apparaten naar één algemeen event type. Wat dit inhoudt is dat ongeacht welke combinatie van operating system en browser er gebruik gemaakt wordt er altijd hetzelfde type event doorgegeven wordt uiteindelijk. De library doet dit momenteel voor de volgende operating system en browser combinaties:

- Windows / Chrome
- Windows / Firefox
- Mac OS / Chrome
- Mac OS / Firefox
- Android / Chrome
- Android / Android browser
- iOS / Chrome
- iOS / Safari

12.2 Evaluatie

12.2.1 De kwaliteit

Als we naar het onderzoeksrapport kijken dan vind ik de kwaliteit variëren tussen erg goed en matig. Om dit te beoordelen moeten we kijken vanuit twee verschillende perspectieven. Het onderzoek zelf bevat namelijk veel uitzoek werk en zou daarom misschien niet voldoen aan de criteria voor een goed onderzoek. Maar als er gekeken wordt naar welke informatie dit heeft

opgeleverd voor het project en voor Liones dan is dit erg goed uitgevoerd. Het rapport bevat de informatie die nodig was om een advies uit te kunnen brengen naar Liones toe over welke input methoden ondersteund moeten gaan worden. Het biedt ook de aanknopingspunten om in de toekomst nog verder te gaan met het onderzoek zelf, waarbij de vragen en delen van de beantwoording hiervan hergebruikt kunnen worden. Dus vanuit dat perspectief wil ik me ook positief uitlaten over dit product. Over het geheel zou ik, als ik een cijfer zou moeten geven, dit product tussen een 6,5 en 7,5 geven. Ondanks dat er niet echt valt te spreken van een volledig valide onderzoek is het uiteindelijke onderzoeksrapport wel dat nodig was vanuit het bedrijf en voor het project. Het biedt concrete aanknopingspunten om het proof of concept naar een volwaardige implementatie om te zetten. Dit was belangrijker voor de product owner dan het feitelijk af zijn van de library.

Voor de kwaliteit van het proof of concept moeten we kijken naar verschillende onderdelen. Namelijk de functionele kant en de net zo belangrijke niet functionele kant. Aan de functionele kant vind ik dat het zeer goed in elkaar zit. Het proof of concept kan de daadwerkelijke input door geven ook in de speciale gevallen zoals bij het automatisch corrigeren van een woord op iOS en bij het gebruik van IME's. Het proof of concept doet op dit vlak dus wat het zou moeten doen. Het bewijst dat het mogelijk is om dit soort unieke gevallen te stroomlijnen naar een robuuste universele vorm. Aan de niet functionele kant moeten we voornamelijk kijken naar hoe de library structureel in elkaar zit. De input komt binnen en wordt opgeslagen. Deze wordt dan omgevormd en als output weer doorgegeven. Deze omvorming wordt door aparte klassen uitgevoerd en het is dan ook mogelijk om hier klassen aan toe te voegen, dit door middel van het slim toepassen van design patterns. Ook is het mogelijk om zelf te bepalen wat de library precies moet doen met de output door het zelf aan kunnen leveren van een function die wordt uitgevoerd moet worden met het output event als parameter. Over het geheel is in mijn mening het proof of concept dan ook van hoge kwaliteit en zou ik, als ik een cijfer zou moeten geven, het proof of concept tussen een 8 en een 9 geven.

12.2.2 Het nut

Het nut van het onderzoeksrapport was duidelijk aanwezig omdat het de nodige informatie gaf voor het kunnen bepalen van de requirements en de bijbehorende prioriteiten. Dit was dus ook de benodigde informatie voor het kunnen ontwikkelen van het proof of concept. Zoals eerder aangegeven biedt het de benodigde concrete aanknopingspunten om het proof of concept naar een volwaardige implementatie om te kunnen zetten.

Het nut van het proof of concept is duidelijk aanwezig. Er zijn tijdens het onderzoek en de ontwikkeling van het proof of concept situaties naar voren gekomen waarbij het niet makkelijk is om direct de echte invoer van de gebruiker uit te lezen. Het nut van de library is dan ook om dit wel makkelijk te maken en dat doet het in de huidige proof of concept vorm al.

12.3 Verder gebruik

Om het proof of concept verder in gebruik te kunnen gaan nemen moet er gewerkt worden naar een zogeheten "release candidate". Wat hiervoor nog zal gebeuren is de omschakeling van het detecteren van het operating system en de browser naar het detecteren van invoer patronen en hierbij dan een invulling van afhandeling te geven. Belangrijkste reden hiervoor is het verhogen

van de robuustheid ten opzichte van toekomstige versies en andere nog onbekende combinaties. Ook moet de library nog verder ontwikkeld worden zodat er rekening gehouden kan worden met integratie in systemen waarin bijvoorbeeld veel met de selectie gedaan wordt. Gebaseerd hierop zouden eventueel wat alternatieve methoden voor het afvangen van de input geïmplementeerd moeten worden. Het idee is om zodra er een release candidate is deze uit te brengen als een open source project. Ook zal er door mij verder gewerkt gaan worden om deze library toe te passen in het FontoXML project.

13. De procesevaluatie

Ik kan met een goed gevoel terug kijken naar dit project. Alhoewel niet alles is verlopen zoals ik had gehoopt zijn er toch in mijn mening positieve resultaten geboekt. Ik heb mijzelf wegwijs gemaakt in het programmeren in JavaScript. Ook heb ik ondanks dat ik alleen dit project heb uitgevoerd toch veel geleerd over het werken met Scrum elementen en dit doen in een groot team.

Het alleen werken aan een project is in mijn mening goed verlopen. Ik heb met een enkele uitzondering hier en daar niet langer vast gezeten dan nodig. Wanneer ik het nodig achtte heb ik hulp gevraagd aan collega's. Ook ben ik tevreden over mijn aanpak van het ontwikkelen van het proof of concept. Het eerst op een ruwe manier neerzetten van een werkende techniek om deze dan later te verfijnen beviel mij zeer en was ook in het voordeel van het project. Minder goed ging het met het vooraf beperken van de scope van het onderzoek, de planning van het algehele project en de planning van de sprints. Zo heb ik problemen gehad met het goed kunnen plannen van het project door niet direct genoeg duidelijkheid te hebben over welke methode ik nou zou gebruiken. Ook heb ik tijdens het plannen van de sprints tot twee maal toe onderschat hoe lang een bepaalde user story precies zou gaan duren. Voor het onderzoek is er tijdens het uitvoeren pas scope beperkingen uitgevoerd, dit heeft mij waarschijnlijk wel tijd gekost.

Als ik dit project opnieuw zou moeten uitvoeren zou ik er direct voor kiezen om Scrum elementen te gebruiken en dan ook het gehele project in sprints te plannen, in plaats van alleen de ontwikkeling. Ik zou vooraf beter opstellen wat de beperkingen van de scope van het onderzoek zijn en ik zou mezelf meer tijd geven voor een beter inzicht in hoe veel tijd user stories zouden gaan innemen.

14. Beroepstaken evaluatie

Alle beroepstaken zijn zelfstandig uitgevoerd, het niveau is dus ook vanuit de zelfstandig categorie bepaald. Zoals dit is aangegeven in: Beroepstaken opleiding informatica versie 1.1.

14.1 Beroepstaken zoals aangegeven in het afstudeerplan

1.4 Uitvoeren analyse door definitie van requirements niveau 3
Er zullen door Liones requirements aangeleverd worden. Verder zal uit het onderzoek de overige requirements gehaald worden.

3.2 Ontwerpen systeemdeel niveau 3
Het proof of concept zal met input van meerdere invoerapparaten om kunnen gaan binnen verschillende browsers en platforms. Dit zal uiteindelijk opgenomen worden in FontoXML dat aan verschillende klanten verkocht zal gaan worden.

3.3 Bouwen applicatie niveau 4
Bij het ontwikkelen van het proof of concept zal rekening gehouden worden met de bestaande code binnen FontoXML. Ook zal dit proof of concept later opgenomen worden als onderdeel van FontoXML.

3.5 Uitvoeren van en rapporteren over het testproces niveau 3
Het proof of concept zal worden getest en deze resultaten zullen worden vastgelegd in een testrapport.

14.2 Evaluatie uitgevoerde beroepstaken

Wanneer er wordt aangegeven dat een deel van de beroepstaak niet van toepassing (n.v.t.) is betekent dit dat deze buiten het vooraf afgesproken niveau valt.

1.4 Uitvoeren analyse door definitie van requirements niveau 3
Vooraf afgesproken niveau: Lastig

Te behalen criteria:

1. De juiste requirements zijn opgesteld, zowel functioneel als niet-functioneel.
    a. De requirements zijn volledig en dekkend.
    b. De requirements zijn ondubbelzinnig en begrijpbaar voor de stakeholders.
    c. De requirements zijn consistent.
2. De requirements zijn onafhankelijk geformuleerd.
3. De requirements zijn traceerbaar.
4. De requirements zijn geprioriteerd en gevalideerd door de stakeholders.

Behaald:

1. De opgestelde requirements zijn goed gekeurd door de stakeholders.
2. De requirements zijn niet afhankelijk van elkaar, bevatten geen verwijzingen naar elkaar

en konden apart afgehandeld worden.
3. De requirements waren in de vorm van user stories te vinden in Jira. Hierin was duidelijk wat de status was van de requirement en wanneer deze afgehandeld is.
4. Tijdens het opstellen van de requirements en gedurende het ontwikkel proces van het proof of concept zijn de requirements geprioriteerd door middel van het aanbrengen van de sprints. Hierbij is gehanteerd dat de sprints gevuld zijn op basis van de prioritering. Per sprint is de prioritering opnieuw bepaald.

3.2 Ontwerpen systeemdeel niveau 3
Vooraf afgesproken niveau: Lastig

Te behalen criteria:
1. De juiste ontwerpstrategie is gekozen.
    a. Er is gebruik gemaakt van de juiste ontwerptechniek.
    b. De ontwerptechniek is op de juiste wijze toegepast.
    c. Er is gebruik gemaakt van een relevante tool.
2. Voor elk systeemdeel zijn de juiste onderliggende structuur en gedrag beschreven.
    a. Structuur is in overeenstemming met de functionele eisen.
    b. Gedrag is in overeenstemming met de functionele eisen.
    c. Er is rekening gehouden met de kwaliteitsattributen: zie ISO 9126. (N.v.t.)
    d. Er is rekening gehouden met beveiligingseisen. (N.v.t.)
    e. Er is gebruik gemaakt van de juiste design patterns. (N.v.t.)
    f. Er is rekening gehouden met abstractie, decompositie en modularisatie.
    g. Er is, indien relevant, gebruik gemaakt van standaard algoritmiek.

Behaald:
1. De eerste keuze voor het Abstract Factory pattern bleek in een later stadium van het ontwikkelen van het proof of concept een vergissing. Door voortschrijdend inzicht is er in een later stadium van ontwikkeling gekozen voor het Strategy pattern en is deze correct toegepast. Voor het ontwerpen is er gebruik gemaakt van StarUML.
2. In de uiteindelijke ontwerpen, gemaakt aan het einde van sprint 3, worden de gehele structuur omschreven met daarbij het gedrag. Hierin wordt duidelijk hoe het proof of concept als library gebruikt kan worden. Hoe deze omgaat met de input en output streams en hoe er omgegaan wordt met hot-key handling.

3.3 Bouwen applicatie niveau 4
Vooraf afgesproken niveau: Complex

Te behalen criteria:
1. Het (detail)ontwerp is op de juiste wijze verfijnd en/of getransformeerd.
    a. Er is rekening gehouden met hergebruik.
    b. Er is rekening gehouden met testbaarheid.
    c. Er is rekening gehouden met wijzigbaarheid.
2. Het systeemdeel is op de juiste wijze gebouwd en werkend opgeleverd.

    a. Er is op de juiste wijze gebruik gemaakt van de mogelijkheden van de gebruikte programmeertaal.
    b. Er is gebruik gemaakt van programmeerstandaarden.
    c. Er is op de juiste manier gebruik gemaakt van een eventueel framework.
    d. Er is gebruik gemaakt van een ontwikkelomgeving met testomgeving en versiebeheertool.
3. De applicatie is op de juiste wijze samengesteld en werkend opgeleverd.

Behaald:

1. Omdat het proof of concept in de vorm van een library is kan deze optimaal hergebruikt worden. Daar waar mogelijk is er rekening gehouden met testbaarheid de library. Zo is deze modulair opgesteld. Hierdoor is het mogelijk om correct unit tests uit te voeren. De units zijn namelijk los van de logica te testen. De library is opgesteld zodat het mogelijk is om wijzigingen aan te brengen binnen de unieke implementaties en units of nieuwe implementaties toe te voegen. Alle unieke implementaties zijn los van elkaar opgesteld. Hierdoor is het mogelijk om aanpassingen aan te brengen in de invulling hiervan zonder daarbij andere delen de library te raken.
2. Er is gebruik gemaakt van het prototype inheritance principe van JavaScript. Verder is er rekening gehouden met programmeerstandaarden en concepten zoals DRY (Don't Repeat Yourself). Er is gebruik gemaakt van het RequireJS framework. Dit is een framework voor het laden van files en modules. Vergelijkbaar met de functionaliteit van import in Java. Omdat het een proof of concept betreft was het niet nodig om verschillende omgevingen te gebruiken. Wel is er gebruik gemaakt van Git als versiebeheertool.
3. Aan het einde van elke sprint is er een werkende versie van het proof of concept opgeleverd.

3.5 Uitvoeren van en rapporteren over het testproces niveau 3

Vooraf afgesproken niveau: Lastig

Te behalen criteria:

1. De testbasis is op de juiste wijze geselecteerd.
2. Het logisch testontwerp is op de juiste wijze m.b.v. een testontwerptechniek opgesteld.
3. Het fysiek testontwerp is op de juiste wijze m.b.v. logisch testontwerp opgesteld.
4. Testdraaiboek (met testscripts) is op de juiste wijze samengesteld.
5. Er is op de juiste wijze gebruik gemaakt van tooling en testframework. (N.v.t.)
6. De testen zijn volgens de planning uitgevoerd.
7. Er is een juiste testrapportage opgesteld.
8. Er is op grond van de testrapportages de juiste conclusie getrokken. (N.v.t.)

Behaald:

1. De testbasis op geselecteerd op basis van de structuur van het proof of concept en de opgestelde requirements.
2. Er is gebruik gemaakt van een beslissingstabel voor de verschillende event afhandeling

implementaties in de library. Ook is er gebruik gemaakt van error guessing.

3. De beslissingstabel is omgezet naar alle fysieke invullingen daarvan. Error guessing is gebruikt bij het opstellen van de unit tests.
4. Omdat het een proof of concept betreft is er geen uitgebreid testdraaiboek aanwezig. Wel is in de test documentatie omschreven hoe en wanneer de aanwezige testscripts uitgevoerd moeten worden.
5. (N.v.t)
6. De testen zijn uitgevoerd zoals oorspronkelijk in het testplan aangegeven.
7. Er is een testplan opgesteld volgens de IEEE-829 2008 structuur. En een testrapport met behulp van een TMap Next template. Deze definiëren structuren waarin een brede selectie van aspecten binnen het testen wordt aangegeven. Door hieruit een keuze te maken van wat ik nodig achtte voor dit project is de in mijn mening juiste testrapportage opgesteld.
8. (N.v.t)

Afstudeerplan

Informatie afstudeerder en gastbedrijf (*structuur niet wijzigen***)**

Afstudeerblok: 2014-1.1 (start uiterlijk 10 februari 2014)
Startdatum uitvoering afstudeeropdracht: 10 februari 2014
Inleverdatum afstudeerdossier volgens jaarrooster: 6 juni 2014

Studentnummer: 08090769
Achternaam: dhr de Vreugd
Voorletters: M.C.
Roepnaam: Michael
Adres: Asstraat 10
Postcode: 2516KA
Woonplaats: Den Haag
Telefoonnummer: 0638664674
Mobiel nummer: 0638664674
Privé emailadres: mcdevreugd@gmail.com

Opleiding: Informatica
Locatie: Den Haag
Variant: voltijd

Naam studieloopbaanbegeleider: J.C.M. Wieman
Naam begeleidend examinator: E.M. van Doorn
Naam tweede examinator: G.M. Tuk

Naam bedrijf: Liones
Afdeling bedrijf: R&D
Bezoekadres bedrijf: Polakweg 7
Postcode bezoekadres: 2288 GG
Postbusnummer:
Postcode postbusnummer:
Plaats: Rijswijk (ZH)
Telefoon bedrijf: 070 319 19 23
Telefax bedrijf: 070 319 38 25
Internetsite bedrijf: www.liones.nl

Achternaam opdrachtgever: dhr Wijma
Voorletters opdrachtgever: G.T.
Titulatuur opdrachtgever:
Functie opdrachtgever: Software Architect
Doorkiesnummer opdrachtgever:
Email opdrachtgever: wijma@liones.nl

Achternaam bedrijfsmentor: dhr Willems
Voorletters bedrijfsmentor: B.
Titulatuur bedrijfsmentor:

Functie bedrijfsmentor: Product Architect
Doorkiesnummer bedrijfsmentor:
Email bedrijfsmentor: bert.willems@liones.nl

NB: bedrijfsmentor mag dezelfde zijn als de opdrachtgever

Doorkiesnummer afstudeerder:
Functie afstudeerder (deeltijd/duaal):

Titel afstudeeropdracht: Ontwikkeling universele input detectie en verwerking voor FontoXML bij Liones.

Opdrachtomschrijving

1. Bedrijf

Liones is een full-service internetbureau waar ruim 30 professionals werken voor uiteenlopende klanten die zij helpen met hun content strategie. Ze werken met de nieuwste technieken en best-practices in combinatie met de nieuwste ontwikkelmethodes. Ze ontwikkelen projecten in de programmeertalen PHP (uitsluitend open-source) en .NET (op basis van Lynkx, een in-house ontwikkeld framework) en doen dit op basis van de SCRUM methode. Ze zijn een dynamisch bureau waar professionaliteit, persoonlijke ontwikkeling en collegialiteit belangrijke waarden zijn. Liones ontwikkelt momenteel een zogeheten What You See Is What You Get (WYSIWYG) tekst editor genaamd FontoXML. Op de achtergrond van deze WYSIWYG tekst editor wordt de tekst omgezet naar XML en opgeslagen. Het bedrijf Liones bestaat veertien jaar. De afstudeeropdracht zal worden uitgevoerd bij de afdeling Research & Development.

2. Probleemstelling

Liones ontwikkelt de FontoXML tekst editor. Bij het ontwikkelen van deze WYSIWYG tekst editor hebben zij zich voornamelijk gericht op de klassieke invoer mogelijkheden: een westers toetsenbord en muis. Om FontoXML aantrekkelijker te maken voor een breder publiek willen zij onderzoek uitvoeren naar de gebruikersbehoefte voor ondersteuning voor andere invoerapparaten en vervolgens ondersteuning aanbieden. Hierbij moet gedacht worden aan touchscreens op tablets, maar ook verschillende toetsenborden of inputapparaten voor mensen met een beperking. Hierbij zijn zij vooral geïnteresseerd in onscreen toetsenborden met autocomplete/autocorrect (mobiele platforms) en Input Method Editors (IME's). Om FontoXML geschikt te maken voor deze omgevingen en inputapparaten willen zij onderzoek uitvoeren naar de verschillende vormen van input en zal een proof of concept gemaakt worden voor het robuust detecteren van invoer, in allerlei vormen, op allerlei browsers en platforms.

3. Doelstelling van de afstudeeropdracht

Voor deze afstudeeropdracht zal onderzoek uitgevoerd worden naar het antwoord op de vraag welke methoden van invoer mogelijk zijn en/of de eindgebruikers van FontoXML deze vormen

van input ook daadwerkelijk zullen gebruiken. Na het onderzoek zal een proof of concept gemaakt worden waarin het robuust de methoden van invoer kan detecteren, in allerlei vormen, op allerlei browsers en platforms.

4. Resultaat

Het opgeleverde onderzoek zal inzicht geven in het antwoord op de vraag waarom invoer methoden, browsers en platforms wel of niet opgenomen zullen worden in het proof of concept. Het proof of concept zal als basis fungeren om de functionaliteit te implementeren in FontoXML. Dit zal de eindgebruikers in staat stellen om FontoXML te gebruiken op de manier die zij willen, met de invoer apparaten die zij willen.

5. Uit te voeren werkzaamheden, inclusief een globale fasering, mijlpalen en bijbehorende activiteiten

De uit te voeren werkzaamheden zullen volgens de SCRUM methode worden uitgevoerd.

Plan van aanpak: Als eerste zal een plan van aanpak opgesteld worden. Hierin zal duidelijk gedefinieerd worden welke activiteiten zullen plaats vinden en welke producten dit zal opleveren. (3 dagen)

Onderzoek: Vervolgens zal begonnen worden met het onderzoek naar de verschillende methoden van input en de afhandeling daarvan binnen verschillende browsers en platforms. Ook zal onderzoek gedaan worden naar de bestaande code binnen FontoXML. (8 dagen)

Requirements definiëren: Er zal een lijst van requirements gemaakt worden waaraan het op te leveren proof of concept zal voldoen. Deze requirements zullen opgesteld worden aan de hand van de resultaten van het onderzoek en de wensen vanuit Liones. (3 dagen)

Ontwerpen proof of concept: Aan de hand van de resultaten van het onderzoek zal een ontwerp gemaakt worden voor een proof of concept dat om kan gaan met de verschillende soorten input binnen de verschillende browsers en platforms. (10 dagen)

Ontwikkelen proof of concept: Het realiseren van het ontwerp in code. (58 dagen)

Testen proof of concept: Het opstellen en uitvoeren van een testplan voor het proof of concept en het verwerken van de resultaten in een testrapport.(3 dagen)

6. Op te leveren (tussen)producten

Producten voortkomend uit punt 5:

- Plan van aanpak
- Requirements
- Onderzoeksrapport
- Ontwerp

- Proof of concept
- Testplan
- Testrapport

7. Te demonstreren competenties en wijze waarop

1.4 Uitvoeren analyse door definitie van requirements niveau 3
Er zullen door Liones requirements aangeleverd worden. Verder zal uit het onderzoek de overige requirements gehaald worden.

3.2 Ontwerpen systeemdeel niveau 3
Het proof of concept zal met input van meerdere invoerapparaten om kunnen gaan binnen verschillende browsers en platforms. Dit zal uiteindelijk opgenomen worden in FontoXML dat aan verschillende klanten verkocht zal gaan worden.

3.3 Bouwen applicatie niveau 4
Bij het ontwikkelen van het proof of concept zal rekening gehouden worden met de bestaande code binnen FontoXML. Ook zal dit proof of concept later opgenomen worden als onderdeel van FontoXML.

3.5 Uitvoeren van en rapporteren over het testproces niveau 3
Het proof of concept zal worden getest en deze resultaten zullen worden vastgelegd in een testrapport.

Plan van Aanpak

Project:	**Input detectie**
Date:	13-03-2014
Owner:	Bert Willems
Client:	Liones
Version:	v1.3
Project members:	Michael de Vreugd

Inhoudsopgave

1. Bedrijf en opdrachtsomschrijving 64
1.1 Bedrijf 64
1.2 Probleemstelling 64
1.3 Doelstelling 64
2. Scope 65
2.1 Project scope 65
2.2 Product scope 65
3. Afbakening opdracht 66
4. Risicofactoren 66
5. Projectorganisatie 67
5.1 Projectleden: 67
5.2 Productowner: 67
5.3 FontoXML team: 68
5.4 Verslaglegging: 68
6. Planning 69
6.1. Detailplanning 70
7. Beschrijving mijlpaalproducten 71

1. Bedrijf en opdrachtsomschrijving

1.1 Bedrijf

Liones is een full-service internetbureau waar ruim 30 professionals werken voor uiteenlopende klanten die zij helpen met hun content strategie. Ze werken met de nieuwste technieken en best-practices in combinatie met de nieuwste ontwikkelmethodes. Ze ontwikkelen projecten in de programmeertalen PHP (uitsluitend open-source) en .NET (op basis van Lynkx, een in-house ontwikkeld framework) en doen dit op basis van de SCRUM methode. Ze zijn een dynamisch bureau waar professionaliteit, persoonlijke ontwikkeling en collegialiteit belangrijke waarden zijn. Liones ontwikkelt momenteel een zogeheten What You See Is What You Get (WYSIWYG) tekst editor genaamd FontoXML. Op de achtergrond van deze WYSIWYG tekst editor wordt de tekst omgezet naar XML en opgeslagen. Het bedrijf Liones bestaat veertien jaar.

1.2 Probleemstelling

Liones ontwikkelt de FontoXML tekst editor. Bij het ontwikkelen van deze WYSIWYG tekst editor hebben zij zich voornamelijk gericht op de klassieke invoer mogelijkheden: een westers toetsenbord en muis. Om FontoXML aantrekkelijker te maken voor een breder publiek willen zij onderzoek uitvoeren naar de gebruikersbehoefte voor ondersteuning voor andere invoerapparaten en vervolgens ondersteuning aanbieden. Hierbij moet gedacht worden aan touchscreens op tablets, maar ook verschillende toetsenborden of inputapparaten voor mensen met een beperking. Hierbij zijn zij vooral geïnteresseerd in onscreen toetsenborden met autocomplete/autocorrect (mobiele platforms) en Input Method Editors (IME's). Om FontoXML geschikt te maken voor deze omgevingen en inputapparaten willen zij onderzoek uitvoeren naar de verschillende vormen van input en zal een proof of concept gemaakt worden voor het robuust detecteren van invoer, in allerlei vormen, op allerlei browsers en platforms.

1.3 Doelstelling

Voor dit project zal onderzoek uitgevoerd worden naar het antwoord op de vraag welke methoden van invoer mogelijk zijn en/of de eindgebruikers van FontoXML deze vormen van input ook daadwerkelijk zullen gebruiken. Na het onderzoek zal een proof of concept gemaakt worden waarin het robuust de methoden van invoer kan detecteren, in allerlei vormen, op allerlei browsers en platforms.

2. Scope

2.1 Project scope

Om dit project succesvol af te kunnen sluiten moeten de volgende activiteiten uitgevoerd worden:

ACTIVITEIT	SPECIFICATIE
Uitvoeren onderzoek	- Het definiëren van een hoofdvraag en deelvragen. - Het uitvoeren van vooronderzoek en het vergaren van bronnen. - Het uitvoeren van literatuuronderzoek voor het beantwoorden van de deelvragen. - Het verwerken van de bevindingen van het literatuuronderzoek. - Het uitvoeren van experimenten voor het beantwoorden van de deelvragen. - Een conclusie trekken over de bevindingen om zo de hoofdvraag te beantwoorden.
Advies uitbrengen	- Het schrijven van een adviesrapport met daarin een beknopte omschrijving van het uitgevoerde onderzoek en de resultaten. - Gebaseerd op de resultaten van het onderzoek een advies uitbrengen aan Liones betreffende welke input methoden op welke platformen en devices toegevoegd moeten worden aan FontoXML.
Opstellen requirements	- Gebaseerd op de resultaten van het adviesgesprek de requirements opstellen. Dit zal dus zowel de wensen vanuit Liones bevatten als de requirements gebaseerd op het onderzoek.
Ontwerpen proof of concept	- Het door middel van verschillende diagrammen in kaart brengen hoe het proof of concept gebouwd zal gaan worden. - Het door middel van verschillende diagrammen in kaart brengen hoe het te bouwen proof of concept in FontoXML opgenomen zou kunnen worden.
Bouwen proof of concept	- Het ontwikkelen van een proof of concept gebaseerd op de opgestelde ontwerpen.
Testen proof of concept	- Het opstellen van een testplan en het uitvoeren hiervan.

2.2 Product scope

De product scope zal bepaald worden door het uit te voeren onderzoek.

3. Afbakening opdracht

- Het proof of concept zal geen andere functionaliteit bevatten dan het afvangen van input.
- Onderzoek naar gebruikersbehoeften zal alleen uitgevoerd worden indien er een mogelijkheid bestaat tot het enquêteren van de eindgebruikers. Indien dit niet mogelijk is zal er een conclusie getrokken worden gebaseerd op gemaakte persona profielen. Indien dit ook niet mogelijk is zal het gebaseerd worden op wat de concurrenten aanbieden en de wensen van Liones.

4. Risicofactoren

-**Het onderzoek kan niet afgerond worden binnen de geplande tijd vanwege de complexiteit van het onderwerp.**
In het geval dat dit voorkomt moet er verder gegaan worden met de tot dat punt vergaarde informatie. Door een goed geprioriteerde structuur te gebruiken bij het onderzoek moeten de belangrijkste punten dan al onderzocht zijn.

- Uit het onderzoek blijkt dat een te brede selectie van platformen en invoermogelijkheden verwerkt moeten worden in het proof of concept.
Wanneer dit voorkomt zal er gezamenlijk met Bert Willems en Stef Busking een sub selectie gemaakt moeten worden op basis van de resultaten van het onderzoek.

-Het blijkt onmogelijk voor een invoermethode om dit te verwerken (binnen het vastgestelde aantal milliseconden waarin dit moet gebeuren).
Indien dit voorkomt moet deze methode buiten de scope gezet worden tot een later punt zodat dit niet de voortgang van het project beïnvloed.

5. Projectorganisatie

Werken aan het project gebeurd op het kantoor van Liones. Het project zal uitgevoerd worden vanaf 10-02-2014 (kalenderweek 7) en 17 weken duren.

Documenten worden aangemaakt op Google Drive en gedeeld met het DITA team.

Voor dit project is een verdeling van functies niet van toepassing. Alle uitvoerende taken zullen door Michael de Vreugd uitgevoerd worden.

5.1 Projectleden:

NAAM	06-NUMMER	EMAIL
Michael de Vreugd	06 38 66 46 74	michael.de.vreugd@liones.nl

5.2 Productowner:

NAAM	06-NUMMER	EMAIL
Bert Willemse	-	bert.willemse@liones.nl

5.3 FontoXML team:

NAAM	06-NUMMER	EMAIL
Jan Benedictus	-	jan.benedictus@liones.nl
Bert Willems	-	bert.willems@liones.nl
Stef Busking	-	stef.busking@liones.nl
Vincent Smedinga	-	vincent.smedinga@liones.nl
Thomas Brekelmans	-	thomas.brekelmans@liones.nl
Marijn van Butselaar	-	marijn.van.butselaar@liones.nl
Wybe Minnebo	-	wybe.minnebo@liones.nl
Youri Bosselaar	-	youri.bosselaar@liones.nl

5.4 Verslaglegging:

Tijdens het afstuderen wordt aan de Haagse Hogeschool verslag gelegd door middel van het bedrijfsbezoek, voortgangsverslag en het afstudeerdossier.

Gedurende het onderzoek zal er wekelijks een moment ingepland worden met Bert Willemse en Stef Busking om verslag uit te brengen over de uitgevoerde werkzaamheden en de uit te voeren werkzaamheden. Op deze manier is er de mogelijkheid tot het geven van feedback en sturing binnen het onderzoek. Tijdens het verrichten van de werkzaamheden voor het proof of concept zal er een dagelijkse stand-up meeting gehouden worden volgens de SCRUM methode.

6. Planning

De onderstaande planning zal worden gebruikt voor dit project.

PRODUCT	KALENDERWEEK
Plan van aanpak	7
Onderzoeksrapport	7-11
Adviesrapport	11
Requirements	11
Ontwerp	11-12
Proof of concept	12-19
Testplan	20
Testrapport	20

6.1. Detailplanning

In de onderstaande planning zijn 4 uren weg gelaten die gebruikt zijn voor het gereed maken van de werkplek op de startdag.
Onderstaande detailplanning laat de onderdelen per product zien en de daarbij ingeplande duur.

PRODUCT	UITVOERDATUM	DUUR
Plan van aanpak	10-02-2014 t/m 12-02-2014	16 uur (2 dagen)
Onderzoeksrapport - Vooronderzoek - Literatuuronderzoek - Verwerken bevindingen - Conclusie	12-02-2014 t/m 10-03-2014	140 uur (17.5 dagen) - 16 uur - 58 uur - 56 uur - 10 uur
Adviesrapport	11-03-2014	8 uur (1 dag)
Requirements - Requirements uit het onderzoek opstellen - Requirements uit de wensen van Liones opstellen	12-03-2014	8 uur (1 dag) - 4 uur - 4 uur
Ontwerp - Ontwerpen proof of concept - Ontwerpen integratie met FontoXML	13-03-2014 t/m 19-03-2014	40 uur (5 dagen) - 30 uur - 10 uur
Proof of concept	19-03-2014 t/m 14-05-2014	320 uur (8 weken)
Testplan	15-05-2014	8 uur (1 dag)
Testrapport	16-05-2014 t/m 19-05-2014	16 uur (2 dagen)

7. Beschrijving mijlpaalproducten

1. **Plan van aanpak**

In het plan van aanpak zal de opdracht worden omschreven en ook afgebakend. Dit zal ervoor zorgen dat de opdrachtgever en de project groep een duidelijk beeld hebben van de opdracht.

2. **Onderzoeksrapport**

In het onderzoeksrapport zal gedefinieerd worden wat de onderzoeksvragen zijn en zullen de antwoorden op deze vragen verwerkt worden. Aan het einde van het rapport zal er een conclusie getrokken worden over de antwoorden op de onderzoeksvragen.

3. **Adviesrapport**

Een rapport met hierin een advies betreffende welke invoer methoden op welke platforms opgenomen moet worden in het proof of concept. Dit advies wordt gebaseerd op de resultaten uit het onderzoeksrapport.

4. **Requirements**

Een lijst van requirements waaraan het proof of concept moet voldoen. Deze requirements zullen opgesteld worden aan de hand van de resultaten van het onderzoek en de wensen vanuit Liones.

5. **Ontwerp**

Het ontwerp door middel van diagrammen voor het te ontwikkelen proof of concept.

6. **Proof of concept**

Een op zichzelf staand proof of concept dat om kan gaan met de verschillende soorten input binnen verschillende browsers en platforms.

7. **Testplan**

Het testplan zal de te gebruiken test methoden omschrijven en voor welke onderdelen gebruikt zullen worden. Ook zal het de opzet voor deze testen bevatten.

8. **Testrapport**

Het testrapport omschrijft het proces van de testen en de resultaten van de testen zoals deze te vinden zijn in het testplan.

Onderzoeksrapport

Project:	**Input detectie**
Date:	12-03-2014
Owner:	Bert Willems
Client:	Liones
Version:	v1.0
Project members:	Michael de Vreugd

Inhoudsopgave

1. Inleiding 74
1.1 Onderzoekskader 74
1.1.1 Verantwoording 74
1.2 Hoofdvraag 75
1.3 Doelstelling 75
1.4 Deelvragen 75
2. Theoretisch Kader 76
2.1 Wat is een What You See Is What You Get(WYSIWYG) editor? 76
2.2 Wat zijn input methoden? 76
2.3 Wat zijn Input Method Editors(IME's)? 76
2.4 Wat zijn compositions? 76
3. Concept 77
3.1 Wat zijn de verschillen tussen alle input methoden op de verschillende operating systems, browsers en devices? 77
3.2 Welke input methoden ondersteunen de concurrenten? 98
3.3 Welke van de input methoden zullen door de eindgebruikers van FontoXML gebruikt worden? 108
3.4 Hoe gaan open source browsers om met het verwerken van input methoden die gebruik maken van compositions? 111
3.5 Hoe gaan code editors om met het verwerken van input methoden die gebruik maken van compositions, IME's of een andere schrijfrichting? 112
3.5.1 Sublime Text 2 112
3.5.2 PHPStorm 116
3.5.3 Notepad++ 119
3.5.4 Microsoft Visual Studio 120
4. Conclusie 123
Bibliografie* 124

1. Inleiding

1.1 Onderzoekskader

FontoXML is een What You See Is What You Get (WYSIWYG) editor. Bij het ontwikkelen van FontoXML heeft Liones zich voornamelijk gericht op de klassieke invoer mogelijkheden, een westers toetsenbord en muis.

Om FontoXML aantrekkelijker te maken voor een breder publiek moet er onderzocht worden welke andere invoer mogelijkheden er nu eigenlijk allemaal zijn, waarin zij verschillen en hoe de concurrenten hier mee omgaan.

De interesse van Liones ligt hierbij voornamelijk bij onscreen toetsenborden met autocomplete/autocorrect(mobiele platforms) en Input Method Editors (IME's). Uiteindelijk moet er bekeken worden welke van deze invoer mogelijkheden de eindgebruikers van FontoXML zouden gebruiken.

1.1.1 Verantwoording

De hoofdvraag is ontstaan door de wens van Liones om FontoXML uit te breiden met meerdere invoer mogelijkheden. Zij hebben zelf al deskresearch op het internet gedaan naar de mogelijkheden hiervoor en hebben ondervonden dat dit niet zo makkelijk is als het simpel toevoegen van het afvangen van input events.

Uit hun deskresearch bleek dat hierbij wat technische problemen om de hoek komen kijken. Toetsenborden met een autocomplete/autocorrect functionaliteit passen soms direct de onderliggende HTML aan, zonder daarbij de verwachte input events aan te leveren. Hiernaast gaan veel browsers vreemd om met keypress events, vooral op mobiele platforms. Chrome op Android stuurt bijvoorbeeld wel keypress events, maar zonder de keycode. Andere browsers verzuimen ze helemaal te sturen.

Ook bleek uit hun deskresearch dat het verwerken van input door middel van Input Method Editors voor vreemd gedrag kan zorgen. Hierbij moet vooral gedacht worden aan talen die gebruik maken van zogeheten compositions. Dit wordt bijvoorbeeld bij Japans gebruikt, hierin worden eerst verschillende karakters getypt en getoond om daarna samengevoegd te worden tot één karakter.

1.2 Hoofdvraag

Welke input methoden op welke devices en platforms moeten toegevoegd worden aan FontoXML?

1.3 Doelstelling

Het doel van dit onderzoek is het uitkiezen van input methoden die toegevoegd moeten worden aan FontoXML en op wat voor manier dit eventueel zou kunnen. Na dit onderzoek zal de conclusie gebruikt worden bij het ontwerpen en bouwen van een proof of concept voor het robuust detecteren van deze input methoden.

1.4 Deelvragen

Om een duidelijk beeld te krijgen van hoe er precies een antwoord gegeven kan worden op de hoofdvraag zijn de volgende deelvragen opgesteld:

- Wat zijn de verschillen tussen alle input methoden op de verschillende operating systems, browsers en devices?
- Welke input methoden ondersteunen de concurrenten?
- Welke van de input methoden zullen door de eindgebruikers van FontoXML gebruikt worden?
- Hoe gaan open source browsers om met het verwerken van input methoden die gebruik maken van compositions?
- Hoe gaan code editors om met het verwerken van input methoden die gebruik maken van compositions, IME's of een andere schrijfrichting?

Door de eerste 3 deelvragen te beantwoorden zal er een gestructureerd en onderbouwd antwoord gegeven kunnen worden op de hoofdvraag. De overige deelvragen zijn bedoelt als onderzoek naar hoe de input methoden verwerkt kunnen worden in een proof of concept.

2. Theoretisch Kader

2.1 Wat is een What You See Is What You Get(WYSIWYG) editor?

Een What You See Is What You Get(WYSIWYG) editor is een systeem waarin content(zowel tekst als grafisch) getoond wordt op het scherm zoals het uiteindelijke product er ook uit komt te zien tijdens het aanpassen.*[01]*

2.2 Wat zijn input methoden?

Onder input methoden wordt verstaan manieren om data en controle signalen te leveren aan een information processing system. Dit onderzoek is voornamelijk, maar niet exclusief, gericht op toetsenborden. Dit kunnen dus fysieke toetsenborden zijn, maar ook bijvoorbeeld digitale toetsenborden zoals deze te vinden zijn op smartphones of IME's.*[02]*

2.3 Wat zijn Input Method Editors(IME's)?

Een Input Method Editor is een operating system component of programma dat elk soort data, zoals toetsenbord aanslagen of muis bewegingen, als input kan registreren. Op deze manier kunnen gebruikers karakters en symbolen gebruiken die normaal niet te vinden zijn op hun input devices.*[03]*

2.4 Wat zijn compositions?

Een composition is een samenstelling van karakters. Meerdere karakters die worden samengevoegd tot één karakter. Bijvoorbeeld Japans maakt hier gebruik van:
De karakters voor O(お) en Ni(に) worden samengevoegd om ONi(鬼) te vormen.*[04]*

3. Concept

3.1 Wat zijn de verschillen tussen alle input methoden op de verschillende operating systems, browsers en devices?

Als eerste moet er uitgezocht worden welke input methoden bestaan. Hierbij is alleen onderzoek gedaan naar voor FontoXML relevante input types. Deskresearch heeft tot de volgende lijst van types en sub-types geleid.

Input methoden *[02]*

Type	**Sub-types**	**Omschrijving**
Keyboards	Computer(Physical) keyboard, Software keyboard, Chorded keyboard / Keyer, Lighted Program Function Keyboard(LPFK)	Een keyboard is een human interface device(HID) gerepresenteerd door een layout van knoppen. Elke knop, of toets, kan gebruikt worden voor het invoeren van een linguïstische karakter of het aanroepen van een computer functie.
Pointing devices	Mouse, Mini-mouse, Trackball, Pointing stick, Finger tracking, Stylus, Touchpad, Touchscreen	Een pointing device is elk HID dat de gebruiker de mogelijkheid geeft om ruimtelijke data door te spelen naar de computer.
Audio input devices	Microphone	Audio input devices worden gebruikt om geluid te ontvangen of om geluid te maken.

Keyboards *[05]*

Sub-type	**Omschrijving**
Computer (Physical) keyboard	Een keyboard in de computer wereld is een device in de style van een typemachine. Het maakt gebruik van een opstelling van knoppen, of toetsen, die gebruikt worden als hendels of elektronische schakelaars. Wordt gebruikt als de main vorm van input voor computers. Is beschikbaar in vele verschillende varianten geschikt voor verschillende regionen in de wereld.
Software keyboard	Een software keyboard is een software matige vorm van een keyboard. In plaats van het fysieke apperaat wordt het keyboard op het beeld getoond en de toetsen kunnen door middel van de muis of touch aangeslagen worden.
Chorded keyboard / Keyer	Een chorded keyboard is een input device dat de gebruiker in staat stelt om karakters in te voeren door middel van het tegelijk indrukken van verschillende toetsen, zoals men bijvoorbeeld een

	akkoord aanslaat op een piano.
Lighted Program Function Keyboard(LPFK)	Een LPFK is een keyboard met daarop meerdere toetsen, deze toetsen zijn gebonden aan functies aangegeven in software die dit support. Een populaire vorm hiervan is niet een pure LPFK maar een combinatie tussen een LPFK en een normaal keyboard.

Pointing devices *[06]*

Sub-type	**Omschrijving**
Mouse / Mini-mouse / Trackball	Een mouse is een klein handheld device dat bewogen wordt over een horizontale oppervlakte. Een traditionele mouse maakte gebruik van een bal en twee schachten aan de onderkant om verplaatsing te meten. Tegenwoordig wordt dit bepaald door middel van een zichtbaar of infrarood licht aan de onderkant van de mouse. Een mini-mouse is simpelweg een kleinere versie van een normale mouse. Een trackbal is een omgekeerd versie van een traditionele mouse. Hierbij zit de bal niet aan de onderkant maar aan de bovenkant van het device en wordt de positie bepaald niet door het verplaatsen van het device zelf, maar door het rollen met de bal.
Pointing stick	Een pointing stick is een device bestaande uit een pressure-sensitive knobbel dat gebruikt kan worden als een joystick. Het kan gevonden worden in bepaalde laptops tussen de ‘G’, ‘H’ en ‘B’ toetsen.
Finger tracking	Een finger tracking device volgt de beweging van vingers in een 3D ruimte, zonder dat de vingers hierbij contact hebben met het scherm of een ander device.
Stylus	Een stylus is een klein pen-vormig instrument dat gebruikt kan worden om input te leveren aan devices met een touchscreen. Een stylus kan ook een of meerdere knoppen hebben waarmee de functionaliteit lichtelijk veranderd, bijvoorbeeld om een screenshot te maken van een deel van het scherm(aangegeven met de stylus).
Touchpad	Een touchpad is een plat oppervlakte dat vinger contact kan detecteren. Hiermee kan positie bepaald worden. Touchpads vinden we meestal terug in laptops. In sommige gevallen kunnen er

	“gestures” gebruikt worden.
Touchscreen	Een touchscreen is een device embedded in een andere device zoals een TV, monitor, laptop, telefoon etc. Dit zorgt ervoor dat er contact met het scherm geregistreerd kan worden inclusief “gestures”.

Audio input devices *[02]*

Sub-type	**Omschrijving**
Microphone	Een microfoon is een elektromechanisch device dat geluid omzet in een electrisch signaal.

Geconcludeerd kan worden dat onder de hoofdtypes het grootste verschil zit. Bij de sub-types is de manier van invoeren wel verschillend, maar het uiteindelijke resultaat blijft hetzelfde. Een ander groot verschil zit in de verschillende beschikbare toetsenborden toegespitst op regionen en/of talen.

Hierna moest de beschikbare input methoden per operating system in kaart brengen. Dit is gedaan door middel van een matrix. In het geval van Chorded keyboard / Keyer en LPFK kan het hardware matig of software matig zijn.

Keyboards *[07]*

	Physical keyboard	**Software keyboard**	**Chorded keyboard / Keyer**	**LPFK**
Windows				
Mac OS				
Windows Phone				
iOS				
Android				
BlackBerry OS				

Pointing Devices *[07]*

	Mouse / Mini-mouse / Trackball	Pointing stick	Finger tracking	Stylus	Touchpad	Touchscreen
Windows						
Mac OS						
Windows Phone						
iOS						
Android						
BlackBerry OS						

*Oranje in dit schema houdt in dat deze manier van input alleen in speciale hardware gevallen van belang is, zoals bijvoorbeeld All-in-one touchscreen pc's die een touchscreen hebben en gebruik kunnen maken van een stylus.

Voor audio devices is het maken van een matrix niet nodig. Alle platformen ondersteunen een microfoon of bevatten een microfoon in de hardware. Ook beschikken alle platformen over software voor speech-to-text. In het geval van de mobile operating systems is dit onderdeel van het operating system. Bij Windows en Mac OS moet hiervoor software aangeschaft worden.

Hierna moest in kaart brengen wat de meest populaire browsers zijn per device en OS. En moet uitgezocht worden wat de features voor deze browsers zijn per device en OS betreffende input.

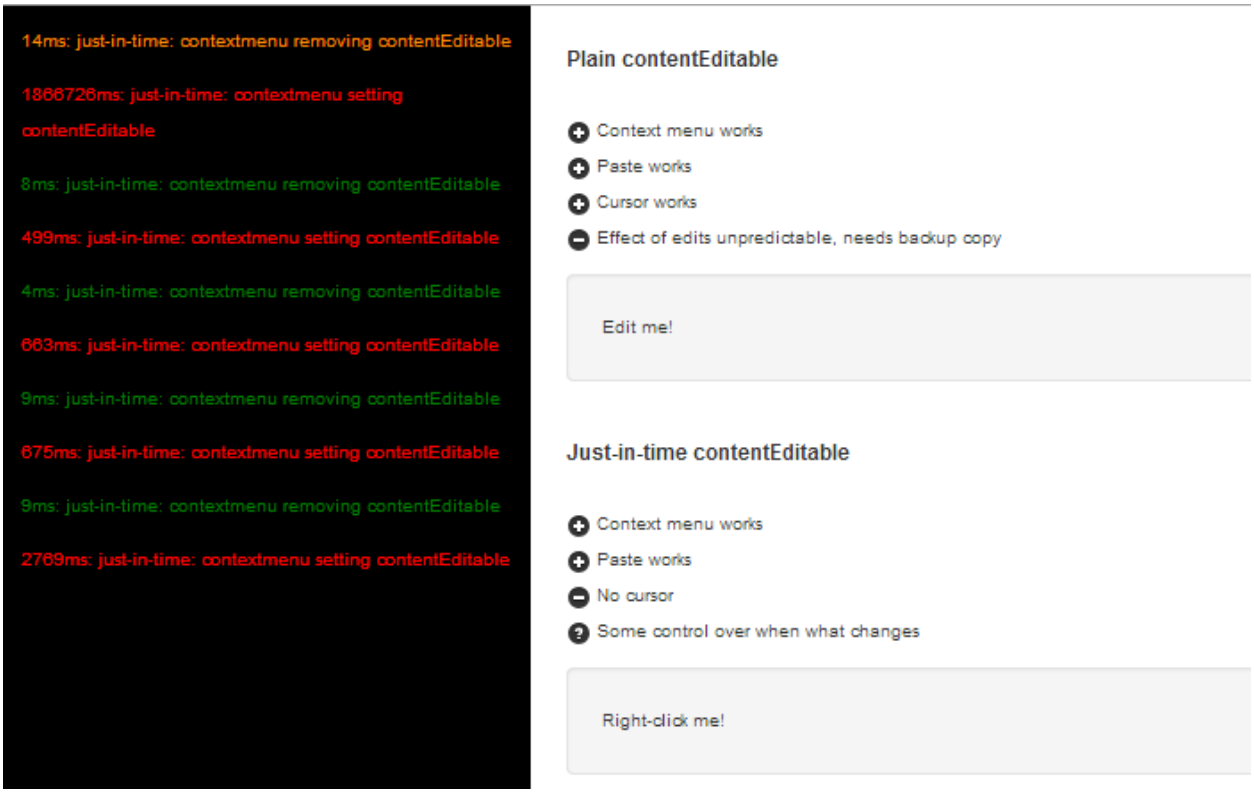
De volgende matrix is gebruikt om dit in kaart te brengen. In de matrix is er voor de smartphone en smartphone/tablet device categorieën één set van statistieken gebruikt.*[08][09][10]*

	Chrome	Firefox	Internet Explorer	Safari	Opera	UC Browser	Android browser	BlackBerry OS Browser
Windows	#1	#2	#3	#4	#5			
Mac OS	#1	#2		#3	#4			
Windows Phone	#3		#1			#2		
iOS	#4			#1	#2	#3		
Android	#4	#5			#2	#3	#1	
BlackBerry OS *	#5	#6			#3	#4	#2	#1

*BlackBerry OS is hierbij uniek omdat er de mogelijkheid bestaat om Android applicaties te installeren. De ranking bij BlackBerry OS is dan ook hierop gebaseerd met als enige uitzondering de BlackBerry OS browser. *[11]*

Om de uiteindelijk te detecteren input af te vangen is er gebruik gemaakt van een stuk test code geschreven door Stef Busking en licht aangepast door Michael de Vreugd. Dit stuk code verschaft een simpele interface met daarin zeven textarea's waarin verschillende acties getest kunnen worden. Wanneer er een actie uitgevoerd wordt op een van deze textarea's wordt er een stuk log getoond met daarin de afgevangen actie en de hoeveelheid milliseconden tussen de verschillende stappen van verwerking voor deze actie. Deze zijn vervolgens colour-coded gebaseerd op de volgende stappen:

- Groen: Alles lager dan 12ms
- Oranje: Alles tussen de 16ms en 12ms
- Rood: Alles hoger dan 16ms



Om de scope van het onderzoek te beperken zijn alleen unieke input methoden getest, dit houdt in dat input methoden zoals bijvoorbeeld een software keyboard op Windows en Mac OS niet getest zijn.

Input methoden aangegeven in het oranje zijn op het moment van het onderzoek nog niet uitgevoerd voor beperking van de scope. De selectie is gemaakt na een gesprek met Bert Willems.

De log output gaat van nieuw naar oud, hierin zal de eerste output altijd als rood gemarkeerd worden omdat dit een nieuw ingevoerd event is. Het belangrijke is de verwerkingstijd van de opvolgende events.

Windows - Chrome

	Chrome
Physical keyboard	Plain contentEditable: 2ms: editable: input "" 1ms: editable: keypress 113 "q" 597ms: editable: keydown 81 "Q" Events: 1ms: Simulating insert: "ɐ" 2ms: events keypress 97 "a" 342ms: events keydown 65 DOM Mutation events / observers: 3032ms: mutation observer found [--- "", +++ "q"] Input redirection: 6ms: input-redirection: keydown got input: "" 4093ms: input-redirection: keydown 81 focusing inputRedirectionTarget Event.clipboardData paste capture: 1ms: Simulating insert: "◖ ⅃ꓤ⅄Ↄ ⊥" 1ms: clipboard-data-paste: text/html "<html> <body> <!--StartFragment--><span style="color: rgb(51, 51, 51); font-family: 'Helvetica Neue', Helvetica, Arial, sans-serif; font-size: 10px; font-style: normal; font-variant: normal; font-weight: normal; letter-spacing: normal; line-height: 20px; orphans: auto; text-align: left; text-indent: 0px; text-transform: none; white-space: normal; widows: auto; word-spacing: 0px; -webkit-text-stroke-width: 0px; background-color: rgb(255, 255, 255); display: inline !important; float: none;">Direct<!--EndFragment--> </body> </html>" 1ms: clipboard-data-paste: text/plain "Direct" 3984ms: clipboard-data-paste: paste text/plain, text/html Event.clipboardData copy override: 3244ms: clipboard-data-copy: copy setting clipboard content
Chorded keyboard / Keyer	
LPFK	
Mouse / Mini-mouse / Trackball	Just-in-time contentEditable: 10ms: just-in-time: contextmenu removing contentEditable 7122ms: just-in-time: contextmenu setting contentEditable

Windows - Firefox

	Firefox
Physical keyboard	Plain contentEditable: 4ms: editable: input "" 4ms: editable: keypress 113 "q" 870ms: editable: keydown 81 "Q"

	Events: 3ms: Simulating insert: "ɐ" 6ms: events keypress 97 "a" 84208ms: events keydown 65 DOM Mutation events / observers: 429ms: mutation observer found [--- "", +++ "q"] Input redirection: 1ms: Simulating insert: "ɐ" 6ms: input-redirection: keydown got input: "a" 3365ms: input-redirection: keydown 65 focusing inputRedirectionTarget Event.clipboardData paste capture: Geen reactie Event.clipboardData copy override: 3448ms: clipboard-data-copy: copy setting clipboard content
Chorded keyboard / Keyer	
LPFK	
Mouse / Mini-mouse / Trackball	Just-in-time contentEditable: 10ms: just-in-time: contextmenu removing contentEditable 1128ms: just-in-time: contextmenu setting contentEditable

Windows - Internet Explorer

	Internet Explorer
Physical keyboard	Plain contentEditable: 2ms: editable: keypress 113 "q" 478ms: editable: keydown 81 "Q" Events: 1ms: Simulating insert: "ɐ" 1ms: events keypress 97 "a" 1053ms: events keydown 65 DOM Mutation events / observers: 480ms: mutation observer found [--- "", +++ "q"] Input redirection: 1ms: Simulating insert: "ɐ" 7ms: input-redirection: keydown got input: "a"

	71768ms: input-redirection: keydown 65 focusing inputRedirectionTarget Event.clipboardData paste capture: 375ms: clipboard-data-paste: paste not supported [object DragEvent] Event.clipboardData copy override: 480ms: clipboard-data-copy: copy not supported [object DragEvent]
Chorded keyboard / Keyer	
LPFK	
Mouse / Mini-mouse / Trackball	Just-in-time contentEditable: 189ms: just-in-time: contextmenu removing contentEditable 1296ms: just-in-time: contextmenu setting contentEditable

Windows - Safari

	Safari
Physical keyboard	Plain contentEditable: 2ms: editable: input "" 1ms: editable: keypress 113 "q" 501ms: editable: keydown 81 "Q" Events: 1ms: Simulating insert: "ɐ" 2ms: events keypress 97 "a" 726ms: events keydown 65 DOM Mutation events / observers: 992ms: mutation DOMCharacterDataModified [--- "", +++ "q"] Input redirection: 2ms: Simulating insert: "ɐ" 7ms: input-redirection: keydown got input: "a" 2638ms: input-redirection: keydown 65 focusing inputRedirectionTarget Event.clipboardData paste capture: 1ms: Simulating insert: "◖ıʁƎↃ ⊥" 1ms: clipboard-data-paste: text/plain "Direct" 1ms: clipboard-data-paste: Text "Direct" 2909ms: clipboard-data-paste: paste Text, text/plain Event.clipboardData copy override: 576ms: clipboard-data-copy: copy setting clipboard content
Chorded keyboard / Keyer	
LPFK	

Mouse / Mini-mouse / Trackball	Just-in-time contentEditable: 11ms: just-in-time: contextmenu removing contentEditable 1538ms: just-in-time: contextmenu setting contentEditable

Windows - Opera

	Opera
Physical keyboard	Plain contentEditable: 2ms: editable: input "" 1ms: editable: keypress 113 "q" 525ms: editable: keydown 81 "Q" Events: 1ms: Simulating insert: "ɐ" 1ms: events keypress 97 "a" 647ms: events keydown 65 DOM Mutation events / observers: 600ms: mutation observer found [--- "", +++ "q"] Input redirection: 6ms: input-redirection: keydown got input: "" 834ms: input-redirection: keydown 65 focusing inputRedirectionTarget Event.clipboardData paste capture: 1ms: Simulating insert: "◖ ıʁƎↃ ⊥" 1ms: clipboard-data-paste: text/html "<html> <body> <!--StartFragment--><span style="color: rgb(51, 51, 51); font-family: 'Helvetica Neue', Helvetica, Arial, sans-serif; font-size: 10px; font-style: normal; font-variant: normal; font-weight: normal; letter-spacing: normal; line-height: 20px; orphans: auto; text-align: left; text-indent: 0px; text-transform: none; white-space: normal; widows: auto; word-spacing: 0px; -webkit-text-stroke-width: 0px; background-color: rgb(255, 255, 255); display: inline !important; float: none;">Direct<!--EndFragment--> </body> </html>" 1ms: clipboard-data-paste: text/plain "Direct" 4320ms: clipboard-data-paste: paste text/plain, text/html Event.clipboardData copy override: 527ms: clipboard-data-copy: copy setting clipboard content
Chorded keyboard / Keyer	
LPFK	
Mouse / Mini-mouse / Trackball	Just-in-time contentEditable: 7ms: just-in-time: contextmenu removing contentEditable 666ms: just-in-time: contextmenu setting contentEditable

Mac OS - Chrome

	Chrome

Physical keyboard	Plain contentEditable: 2ms: editable: input "" 2ms: editable: keypress 113 "q" 620ms: editable: keydown 81 "Q" Events: 1ms: Simulating insert: "ɐ" 2ms: events keypress 97 "a" 766ms: events keydown 65 DOM Mutation events / observers: 0ms: mutation observer found [--- "", +++ "q"] 18710ms: mutation observer found [--- " ", +++ "q"] Input redirection: 1ms: Simulating insert: "b" 14ms: input-redirection: keydown got input: "q" 1177ms: input-redirection: keydown 81 focusing inputRedirectionTarget Event.clipboardData paste capture: 2ms: Simulating insert: "◖ ɹəƎɔ ⊥" 1ms: clipboard-data-paste: text/html "<meta charset='utf-8'><span style="color: rgb(51, 51, 51); font-family: 'Helvetica Neue', Helvetica, Arial, sans-serif; font-size: 10px; font-style: normal; font-variant: normal; font-weight: normal; letter-spacing: normal; line-height: 20px; orphans: auto; text-align: left; text-indent: 0px; text-transform: none; white-space: normal; widows: auto; word-spacing: 0px; -webkit-text-stroke-width: 0px; background-color: rgb(255, 255, 255); display: inline !important; float: none;">Direct" 4ms: clipboard-data-paste: text/plain "Direct" 24124ms: clipboard-data-paste: paste text/plain, text/html Event.clipboardData copy override: 337ms: clipboard-data-copy: copy setting clipboard content
Chorded keyboard / Keyer	
LPFK	
Mouse / Mini-mouse / Trackball	Just-in-time contentEditable: 10ms: just-in-time: contextmenu removing contentEditable 3748ms: just-in-time: contextmenu setting contentEditable

Mac OS - Firefox

	Firefox
Physical	Plain contentEditable:

keyboard	2ms: editable: input "" 2ms: editable: keypress 113 "q" 522ms: editable: keydown 81 "Q" Events: 1ms: Simulating insert: "ɐ" 3ms: events keypress 97 "a" 492ms: events keydown 65 DOM Mutation events / observers: 584ms: mutation observer found [--- "", +++ "q"] Input redirection: 1ms: Simulating insert: "b" 4ms: input-redirection: keydown got input: "q" 803ms: input-redirection: keydown 81 focusing inputRedirectionTarget Event.clipboardData paste capture: Geen reactie. Event.clipboardData copy override: 53099ms: clipboard-data-copy: copy setting clipboard content
Chorded keyboard / Keyer	
LPFK	
Mouse / Mini-mouse / Trackball	Just-in-time contentEditable: 9ms: just-in-time: contextmenu removing contentEditable 918ms: just-in-time: contextmenu setting contentEditable

Mac OS - Safari

	Safari
Physical keyboard	Plain contentEditable: 2ms: editable: input "" 2ms: editable: keypress 113 "q" 452ms: editable: keydown 81 "Q" Events: 0ms: Simulating insert: "ɐ" 1ms: events keypress 97 "a" 655ms: events keydown 65

	DOM Mutation events / observers: 632ms: mutation observer found [--- "", +++ "q"] Input redirection: 1ms: Simulating insert: "b" 11ms: input-redirection: keydown got input: "q" 644ms: input-redirection: keydown 81 focusing inputRedirectionTarget Event.clipboardData paste capture: 1ms: Simulating insert: "◖ ıʁƎↃ ⊥" 1ms: clipboard-data-paste: CorePasteboardFlavorType 0x7374796C "" 2ms: clipboard-data-paste: dyn.ah62d4rv4gk81g7d3ru "L &&&&&&" 0ms: clipboard-data-paste: CorePasteboardFlavorType 0x54455854 "" 1ms: clipboard-data-paste: com.apple.traditional-mac-plain-text "" 0ms: clipboard-data-paste: CorePasteboardFlavorType 0x7573746C "" 1ms: clipboard-data-paste: dyn.ah62d4rv4gk81n65yru "" 0ms: clipboard-data-paste: CorePasteboardFlavorType 0x75743136 "" 4ms: clipboard-data-paste: public.utf16-external-plain-text "" 1ms: clipboard-data-paste: text/plain "Direct" Event.clipboardData copy override: 51910ms: clipboard-data-copy: copy setting clipboard content
Chorded keyboard / Keyer	
LPFK	
Mouse / Mini-mouse / Trackball	Just-in-time contentEditable: 6ms: just-in-time: contextmenu removing contentEditable 1820ms: just-in-time: contextmenu setting contentEditable

Mac OS - Opera

	Opera
Physical keyboard	Plain contentEditable: 2ms: editable: input "" 1ms: editable: keypress 113 "q" 421ms: editable: keydown 81 "Q" Events:

	1ms: Simulating insert: "ɐ" 2ms: events keypress 97 "a" 406ms: events keydown 65 DOM Mutation events / observers: 520ms: mutation observer found [--- "", +++ "q"] Input redirection: 1ms: Simulating insert: "b" 4ms: input-redirection: keydown got input: "q" 587ms: input-redirection: keydown 81 focusing inputRedirectionTarget Event.clipboardData paste capture: 2ms: Simulating insert: "◖ıʁƎↃ ⊥" 1ms: clipboard-data-paste: text/html "<meta charset='utf-8'><span style="color: rgb(51, 51, 51); font-family: 'Helvetica Neue', Helvetica, Arial, sans-serif; font-size: 10px; font-style: normal; font-variant: normal; font-weight: normal; letter-spacing: normal; line-height: 20px; orphans: auto; text-align: left; text-indent: 0px; text-transform: none; white-space: normal; widows: auto; word-spacing: 0px; -webkit-text-stroke-width: 0px; background-color: rgb(255, 255, 255); display: inline !important; float: none;">Direct" 1ms: clipboard-data-paste: text/plain "Direct" 14636ms: clipboard-data-paste: paste text/plain, text/html Event.clipboardData copy override: 215334ms: clipboard-data-copy: copy setting clipboard content
Chorded keyboard / Keyer	
LPFK	
Mouse / Mini-mouse / Trackball	Just-in-time contentEditable: 12ms: just-in-time: contextmenu removing contentEditable 1969ms: just-in-time: contextmenu setting contentEditable

Windows Phone

Om de scope van het project te beperken worden voor versie 1.0 van het onderzoek geen experimenten uitgevoerd voor Windows Phone.

iOS - Safari

	Safari
Physical keyboard	
Software keyboard	Plain contentEditable: 13ms: editable: input “”

	2ms: editable: keypress 113 “q” 264ms: editable: keydown 81 “Q” Events: 6ms: Simulating insert “ɐ” 10ms: events keypress 97 “a” 152180ms: events keydown 65 DOM Mutation events / observers: 8ms: mutation observer found [---””, +++”q”] 100310ms: mutation observer found [---””, +++”q”] Input redirection: 39ms: input-redirection: keydown got input: “” 260919ms: input-redirection: keydown 65 focusing inputRedirectionTarget Event.clipboardData paste capture: 6ms: Simulating insert: “◖ ⅃ꓤƎↃ ⊥” 3ms: clipboard-data-paste: public.rtf “{\rtf1\ansi\ansicpg1252 {\fonttbl\f0\fnil\fcharset0 HelveticaNeue;}{\colortbl;\red255\green255\blue255;\red51\green51\blue51;}\deftab720 \pard\pardeftab720\sl400\partightenfactor0\f0\fs20 \cf2 \expnd0\expndtw0\kerning0\outl0\strokewidth0 \strokec2 Direct}” 3ms: clipboard-data-paste: com.apple.rtfd “” 10ms: clipboard-data-paste: text/plain “Direct” 8ms: clipboard-data-paste: text/uri-list “” 4ms clipboard-data-paste: com.compuserve.gif “” 4ms: clipboard-data-paste: public.jpeg “” 6ms: clipboard-data-paste: public.tiff “” 7ms: clipboard-data-paste: public.png “” 13ms: clipboard-data-paste: apple web archive pasteboard type “” Event.clipboardData copy override: 485756ms: clipboard-data-copy: copy setting clipboard content
Chorded keyboard / Keyer	
Stylus	Just-in-time contentEditable:
Touchpad	
Touchscreen	Just-in-time contentEditable:

iOS - Opera*

	Opera
Physical keyboard	
Software keyboard	Plain contentEditable: Events: DOM Mutation events / observers: Input redirection: Event.clipboardData paste capture: Event.clipboardData copy override:
Chorded keyboard / Keyer	
Stylus	Just-in-time contentEditable:
Touchpad	
Touchscreen	Just-in-time contentEditable:

*De Opera browser weigerde op iOS te navigeren naar de testcode op mijn pc, dit kon dus niet getest worden.

iOS - UC Browser

	UC Browser
Physical keyboard	
Software keyboard	Plain contentEditable: 13ms: editable: input “” 2ms: editable: keypress 113 “q” 634ms: editable keydown 81 “Q” Events: 2ms: Simulating insert: “ɐ” 5ms: events keypress 97 “a” 167158ms: events keydown 65 DOM Mutation events / observers: 234ms: mutation observer found [---””, +++”q”] Input redirection: 29ms: input-redirection: keydown got input: “” 155728ms: input-redirection: keydown 65 focusing inputRedirectionTarget Event.clipboardData paste capture: 3ms: Simulating insert: “◖ ıʁƎↃ ⊥” 2ms: clipboard-data-paste: public.rtf “”

	2ms: clipboard-data-paste: com.apple.rtfd “” 2ms: clipboard-data-paste: text/plain “Direct” 2ms: clipboard-data-paste: text/uri-list “” 2ms clipboard-data-paste: com.compuserve.gif “” 2ms: clipboard-data-paste: public.jpeg “” 2ms: clipboard-data-paste: public.tiff “” 2ms: clipboard-data-paste: public.png “” 4ms: clipboard-data-paste: apple web archive pasteboard type “” Event.clipboardData copy override: 183463ms: clipboard-data-copy: copy setting clipboard content
Chorded keyboard / Keyer	
Stylus	Just-in-time contentEditable:
Touchpad	
Touchscreen	Just-in-time contentEditable:

iOS - Chrome

	Chrome
Physical keyboard	
Software keyboard	Plain contentEditable: 13ms: editable: input “” 2ms: editable: keypress 113 “q” 304ms: editable keydown 81 “Q” Events: 2ms: Simulating insert: “ɐ” 4ms: events keypress 97 “a” 41840ms: events keydown 65 DOM Mutation events / observers: 384ms: mutation observer found [---””, +++”q”] Input redirection: 3ms: Simulating insert: "ɐ" 102ms: input-redirection: keydown got input: "a" 132638ms: input-redirection: keydown 65 focusing inputRedirectionTarget

	Event.clipboardData paste capture: 3ms: Simulating insert: “◖ ıʁƎↃ ⊥” 2ms: clipboard-data-paste: public.rtf “{\rtf1\ansi\ansicpg1252 {\fonttbl\f0\fnil\fcharset0 HelveticaNeue;}{\colortbl;\red255\green255\blue255;\red51\green51\blue51;}\deftab720 \pard\pardeftab720\sl400\partightenfactor0\f0\fs20 \cf2 \expnd0\expndtw0\kerning0\outl0\strokewidth0 \strokec2 Direct}” 2ms: clipboard-data-paste: com.apple.rtfd “” 2ms: clipboard-data-paste: text/plain “Direct” 2ms: clipboard-data-paste: text/uri-list “” 2ms clipboard-data-paste: com.compuserve.gif “” 2ms: clipboard-data-paste: public.jpeg “” 3ms: clipboard-data-paste: public.tiff “” 4ms: clipboard-data-paste: public.png “” 4ms: clipboard-data-paste: apple web archive pasteboard type “” Event.clipboardData copy override: 124338ms: clipboard-data-copy: copy setting clipboard content
Chorded keyboard / Keyer	
Stylus	Just-in-time contentEditable:
Touchpad	
Touchscreen	Just-in-time contentEditable:

Android - Android Browser

	Android Browser
Physical keyboard	
Software keyboard	Plain contentEditable: 11ms: editable: input “” 7ms: editable: keypress 113 “q” 709ms: editable: keydown 81 “Q” Events: 10ms: Simulating insert: “ɐ” 13ms: events keypress 97 “a” 454ms: events keydown 65

	DOM Mutation events / observers: 813ms: mutation DOMCharacterDataModified [---"", +++"q"] Input redirection: 7ms: Simulating insert: "b" 14ms: input-redirection: keydown got input: "q" 2155ms: input-redirection: keydown 81 focusing inputRedirectionTarget Event.clipboardData paste capture: Geen reactie Event.clipboardData copy override: 141548ms: clipboard-data-copy: copy setting clipboard content
Chorded keyboard / Keyer	
Mouse / Mini-mouse / Trackball	
Stylus	Just-in-time contentEditable:
Touchpad	
Touchscreen	Just-in-time contentEditable:

Android - Opera

	Opera
Physical keyboard	
Software keyboard	Plain contentEditable: 16ms: editable: input "" 9ms: editable: compositionupdate 0 "" 521ms: editable: keydown 0 "" Events: 14ms: events keydown 0 15ms: events input 2983ms: events compositionupdate DOM Mutation events / observers: 5ms: mutation observer found [---"", +++"qqqqq"] 132ms: mutation observer found [---"", +++"q"] Input redirection:

	17ms: input-redirection: keydown got input: “” 681ms: input-redirection: keydown 0 focusing inputRedirectionTarget Event.clipboardData paste capture: Kan het plakken menu niet getoond krijgen. Event.clipboardData copy override: Geen reactie
Chorded keyboard / Keyer	
Mouse / Mini-mouse / Trackball	
Stylus	Just-in-time contentEditable:
Touchpad	
Touchscreen	Just-in-time contentEditable:

Android - UC Browser

	UC Browser
Physical keyboard	
Software keyboard	Plain contentEditable: 20ms: editable: keydown 81 “Q” 11ms: editable: input “” 1242ms: editable: compositionupdate 0 “” Events: 10ms: events keydown 65 15ms: events input 652ms: events compositionupdate DOM Mutation events / observers: 6ms: mutation DOMCharacterDataModified [---””, +++”qqqqq”] 15660ms: mutation DOMCharacterDataModified [---”qqqqq”, +++””] Input redirection: 14ms: input-redirection: keydown got input:”” 42ms: input-redirection: keydown 81 focusing inputRedirectionTarget 25ms: input-redirection: keydown got input:”” 18454ms: input-redirection: keydown 81 focusing inputRedirectionTarget

	Event.clipboardData paste capture: Geen reactie Event.clipboardData copy override: Geen reactie
Chorded keyboard / Keyer	
Mouse / Mini-mouse / Trackball	
Stylus	Just-in-time contentEditable:
Touchpad	
Touchscreen	Just-in-time contentEditable:

Android - Chrome

	Chrome
Physical keyboard	
Software keyboard	Plain contentEditable: 14ms: editable: keydown 0 “” 14ms: editable: input “” 396ms: editable: compositionupdate 0 “” Events: 18ms: events keydown 0 14ms: events input 1278ms: events compositionupdate DOM Mutation events / observers: 4ms: mutation observer found [---””, +++”qqqqq”] 759ms: mutation observer found [---””, +++”q”] Input redirection: 19ms: input-redirection: keydown got input: “” 211ms: input-redirection: keydown 0 focusing inputRedirectionTarget Event.clipboardData paste capture: 3809ms: clipboard-data-pase: paste Event.clipboardData copy override: 49781ms: clipboard-data-copy: copy setting clipboard content

Chorded keyboard / Keyer	
Mouse / Mini-mouse / Trackball	
Stylus	Just-in-time contentEditable:
Touchpad	
Touchscreen	Just-in-time contentEditable:

Android - Firefox

	Firefox
Physical keyboard	
Software keyboard	Plain contentEditable: 16ms: editable: input “” 21ms: editable: compositionend 0 “” 2590ms: editable: compositionstart 0 “” Events: 21ms: events input 34ms: events compositionend 3410ms: events compositionstart DOM Mutation events / observers: 15ms: mutation observer found [---””,+++”q”] 4412ms: mutation observer found [---””,+++””] Input redirection: 11ms: Simulating insert: “b” 41ms: input-redirection: keydown got input:”q” 13299ms: input-redirection: keydown 81 focusing inputRedirectionTarget Event.clipboardData paste capture: Kan het plakken menu niet getoond krijgen Event.clipboardData copy override: Geen reactie
Chorded keyboard / Keyer	
Mouse / Mini-	

mouse / Trackball	
Stylus	Just-in-time contentEditable:
Touchpad	
Touchscreen	Just-in-time contentEditable:

BlackBerry OS
Om de scope van het onderzoek te beperken is ervoor gekozen om geen experimenten op het BlackBerry OS platform uit te voeren. Dit vanwege het snel dalende marktaandeel van BlackBerry en het feit dat de nieuwste versies van BlackBerry OS gebruik kunnen maken van Android applicaties.

Geconcludeerd kan worden dat het verschil op operating system niveau voornamelijk tussen mobile en pc operating systems ligt, waarbij er ook weer een groot verschil is in welke browser gebruikt wordt. Algemene reactie tijden voor mobile operating systems zijn langzamer dan de operating systems voor pc's. Over het geheel genomen is Mac OS het snelste met de afhandeling van de events. Ook zien we verschil tussen het wel of niet kunnen afvangen van events per operating system of de manier waarop de events worden afgehandeld.

3.2 Welke input methoden ondersteunen de concurrenten?

Om deze deelvraag te beantwoorden zijn een selectie van de bij deelvraag 1 gedefinieerde operating system en input methoden gecombineerd en getest bij de concurrenten. De selectie is gemaakt na een gesprek met Bert Willems en gebaseerd op de besproken prioriteit, waarschijnlijkheid van gebruik en of een methode niet simpel weg een alternatieve manier is voor de hoofdmethode.

De concurrenten die hierbij bekeken zijn:

- Xeditor *[12]*
- Google Docs *[13]*
- Google Script *[14]*
- WebODF *[15]*
- Ace *[16]*
- Code Mirror *[17]*

Voor de input method "Physical keyboard" op Windows en Mac OS is gebruik gemaakt van de volgende talen:

English (United States)	Windows display language: Enabled Keyboard layout: US Date, time, and number formatting	- US_English: Maakt gebruik van de meest gebruikte QWERTY layout
日本語	Windows display language: Available Input method: Microsoft IME	- Japanese: Maakt gebruik van compositions en een IME.
中文(中华人民共和国)	Windows display language: Available Input method: Microsoft Pinyin	- Chinese(Simplified): Maakt gebruik van de Pinyin IME.
العربية (الإمارات العربية المتحدة)	Windows display language: Available Keyboard layout: Arabic (101)	- Arabic: Maakt gebruik van een rechts naar links schrijfrichting.

Voor Japanese zijn de volgende zinnen/woorden getest:

Japans	**Engels**	**Composition**
はじめ　まして / HaJiMe MaShiTe	Nice to meet you	Nee
鬼 / ONi	Demon / Ghost	Ja

Voor de testen is de Chrome browser gebruikt, tenzij een concurrent niet correct werkt in de combinatie van platform en Chrome. In dat geval is in de bevindingen aangegeven in welke browser het getest is en indien nodig in welke browser welke problemen voorkomen.

Windows

Concurrent	**Input method**	**Bevindingen**
Xeditor	Physical keyboard	Chrome: ● English: Werkt naar behoren. ● Japanese: Bij het invoeren van Japanese binnen een nieuw <doc_p> element wordt het nieuwe <doc_p> element weggehaald en worden de ingevoerde tekens bij het daarop volgende <doc_p> element toegevoegd. ● Chinese: Bij het invoeren van Simplified Chinese verdwijnt de Pinyin IME zodra er begonnen wordt met typen. Dit resulteert soms in het verschijnen van het fonetisch getypte en soms alleen in het verdwijnen van het nieuw aangemaakte <doc_p> element. ● Arabic: Tekent bij het gebruik van Arabic de cursor niet op de juiste plaats. Opera:

		● English: Werkt naar behoren. ● Japanese: Bij het invoeren van Japanese binnen een nieuw <doc_p> element wordt het nieuwe <doc_p> element weggehaald en worden de ingevoerde tekens bij het daarop volgende <doc_p> element toegevoegd of de ingevoerde tekens verdwijnen zo snel als dat ze verschijnen. ● Chinese: Bij het invoeren van Simplified Chinese verdwijnt de Pinyin IME zodra er begonnen wordt met typen. Dit resulteert soms in het verschijnen van het fonetisch getypte en soms alleen in het verdwijnen van het nieuw aangemaakte <doc_p> element. ● Arabic: Tekent bij het gebruik van Arabic de cursor niet op de juiste plaats. Firefox: ● English: Werkt naar behoren. ● Japanese: Werkt naar behoren. ● Chinese: Werkt naar behoren. ● Arabic: Toont na het gebruik van een spatie de cursor aan de verkeerde kant, maar werkt verder naar behoren. Safari: ● English: Werkt naar behoren. ● Japanese: Werkt naar behoren. ● Chinese: Werkt naar behoren. ● Arabic: Tekent bij het gebruik van Arabic de cursor niet op de juiste plaats.
	Mouse / Mini-mouse / Trackball	Werkt naar behoren, inclusief right-click context menu.
Google Docs	Physical keyboard	Werkt naar behoren, inclusief de veel gebruikte shortcuts.
	Mouse / Mini-mouse / Trackball	Werkt naar behoren, inclusief right-click context menu.
Google Script	Physical keyboard	Tekent bij het gebruik van Arabic de cursor niet op de juiste plaats. Werkt verder naar behoren, inclusief de veel gebruikte shortcuts.
	Mouse / Mini-mouse / Trackball	Werkt naar behoren, inclusief right-click context menu.
WebODF	Physical keyboard	Tekent bij het gebruik van Arabic de cursor niet op de juiste plaats. Werkt verder naar behoren, inclusief de veel gebruikte shortcuts.

	Mouse / Mini-mouse / Trackball	Werkt naar behoren, exclusief right-click context menu.
Ace	Physical keyboard	Tekent bij het gebruik van Arabic de cursor niet op de juiste plaats. Werkt verder naar behoren, inclusief de veel gebruikte shortcuts. Tekent voor IME's en compositions een eigen IME.
	Mouse / Mini-mouse / Trackball	Werkt naar behoren, inclusief right-click context menu.
Code Mirror	Physical keyboard	Tekent tijdens het typen bij Arabic de cursor wel op de juiste plaats, maar plaatst deze na een enter weer aan de verkeerde kant. Werkt verder naar behoren, inclusief de veel gebruikte shortcuts.
	Mouse / Mini-mouse / Trackball	Werkt naar behoren, inclusief right-click context menu.

Windows Phone

Om de scope van het project te beperken worden voor versie 1.0 van het onderzoek geen experimenten uitgevoerd voor Windows Phone.

Mac OS

Concurent	**Input method**	**Bevindingen**
Xeditor	Physical keyboard	Chrome: Bij het gebruik van het invoeren van een ‘ gevolgd door een e kan dit als eerste karakter van een leeg element niet worden samengevoegd maar voor alle navolgende elementen wel. ● Japanese: Bij het invoeren van Japanese binnen een nieuw <doc_p> element wordt het nieuwe <doc_p> element weggehaald en worden de ingevoerde tekens bij het daarop volgende <doc_p> element toegevoegd. ● Chinese: Bij het invoeren van Simplified Chinese verdwijnt de Pinyin IME zodra er begonnen wordt met typen. Dit resulteert soms in het verschijnen van het fonetisch getypte en soms alleen in het verdwijnen van het nieuw aangemaakte <doc_p> element. Safari: Bij Safari werkt het invoeren van een ‘ gevolgd door een e niet voor het samenvoegen van deze tot een karakter. Bij het gebruik van Japanese veroorzaakt dit raar gedrag bij het maken van een composition. Algemeen: ● Arabic: Cursor wordt aan de verkeerde kant getekend.

	Mouse / Mini-mouse / Trackball	Werkt naar behoren, inclusief right-click context menu.
Google Docs	Physical keyboard	Werkt naar behoren.
	Mouse / Mini-mouse / Trackball	Werkt naar behoren, inclusief right-click context menu.
Google Script	Physical keyboard	Werkt naar behoren. Tekent bij Arabic de cursor aan de verkeerde kant.
	Mouse / Mini-mouse / Trackball	Werkt naar behoren, inclusief right-click context menu.
WebODF	Physical keyboard	Werkt naar behoren. Tekent bij Arabic de cursor aan de verkeerde kant. Kan geen ‘ en e samenvoegen. Bij Japanese wordt het fonetisch ingetypte niet getoond vanwege de input-trap techniek. Hetzelfde geldt bij Chinese.
	Mouse / Mini-mouse / Trackball	Werkt naar behoren, inclusief right-click context menu.
Ace	Physical keyboard	Tekent bij het gebruik van Arabic de cursor niet op de juiste plaats. Werkt verder naar behoren, inclusief de veel gebruikte shortcuts. Tekent voor IME’s en compositions een eigen IME.
	Mouse / Mini-mouse / Trackball	Werkt naar behoren, inclusief right-click context menu.
Code Mirror	Physical keyboard	Werkt naar behoren. Tekent bij Arabic de cursor aan de verkeerde kant.
	Mouse / Mini-mouse / Trackball	Werkt naar behoren, inclusief right-click context menu.

iOS(Test device: Generation 4 iPad, iOS 7)

Concurent	**Input method**	**Bevindingen**
Xeditor	Physical keyboard	
	Software keyboard	Safari: Lichte lag tijdens het typen. Vertoont wat raar gedraag met het backspacen van Japans, zoals het opnieuw verschijnen van weggehaalde woorden.

		Opera: Laad de demo niet correct in en kan niet getest worden. UC Browser: Tijdens het typen verspringt bij elke aanslag het scherm een stuk naar beneden, hierdoor kan niet gezien worden of er zichtbaar problemen zijn. Chrome: Lichte lag tijdens het typen, pagina duurde erg lang om in te laden.
	Stylus	
	Touchpad	
	Touchscreen	Werkt naar behoren.
Google Docs	Physical keyboard	
	Software keyboard	Safari: Werkt naar behoren Opera: Geen optie om documenten te kunnen bewerken UC Browser: Duidelijk zichtbare lag Chrome: Werkt naar behoren
	Stylus	
	Touchpad	
	Touchscreen	Werkt naar behoren.
Google Script	Physical keyboard	
	Software keyboard	Safari: Zeer lichte lag, maakt geen gebruik van autocorrect/autocomplete. Werkt verder naar behoren. Opera: Wordt niet ondersteund. UC Browser: Lichte input lag, maakt geen gebruik van autocorrect/autocomplete. Werkt verder naar behoren. Chrome: Duidelijke input lag, maakt geen gebruik van autocorrect/autocomplete. Werkt verder naar behoren.
	Stylus	

	Touchpad	
	Touchscreen	UC Browser: Tekent een eigen cursor en laat de OS bepaalde cursor zien. Overige: Werkt naar behoren.
WebODF	Physical keyboard	
	Software keyboard	ALGEMEEN: Support geen continues press voor de backspace button. Safari: Werkt naar behoren. Opera: Laadt niet en kan dus niet getest worden. UC Browser: Werkt naar behoren Chrome: Werkt naar behoren
	Stylus	
	Touchpad	
	Touchscreen	Tekent een eigen cursor alsmede het laten zien van de OS bepaalde cursor. Werkt verder naar behoren.
Ace	Physical keyboard	
	Software keyboard	Safari: Kan geen composition karakters verwerken, gebruikt geen autocorrect/autocomplete. Werkt verder naar behoren. Opera: In het editable veld drukken opent niet het software toetsenbord, dit kan dus ook niet getest worden. UC Browser: Tijdens het typen verspringt bij elke aanslag het scherm een stuk naar beneden, hierdoor kan niet gezien worden of er zichtbaar problemen zijn. Chrome: Kan geen composition karakters verwerken, gebruikt geen autocorrect/autocomplete. Werkt verder naar behoren.
	Stylus	
	Touchpad	
	Touchscreen	Werkt naar behoren.

Code Mirror	Physical keyboard	
	Software keyboard	Safari: Gebruikt geen autocorrect/autocomplete. Werkt verder naar behoren. Opera: In het editable veld drukken opent niet het software toetsenbord, dit kan dus ook niet getest worden. UC Browser: Tijdens het typen verspringt bij elke aanslag het scherm een stuk naar beneden, hierdoor kan niet gezien worden of er zichtbaar problemen zijn. Chrome: Gebruikt geen autocorrect/autocomplete. Werkt verder naar behoren.
	Stylus	
	Touchpad	
	Touchscreen	Werkt naar behoren.

Android (Test device: Samsung Galaxy Note II, Android 4.3)

Concurent	**Input method**	**Bevindingen**
Xeditor	Physical keyboard	
	Software keyboard	Android browser: Werkt naar behoren inclusief autocomplete. Chrome: Werkt naar behoren inclusief autocomplete. Firefox: Duidelijke lag bij snel typen. Laat bij het invoeren van Japanese soms fonetische karakters staan. Werkt verder naar behoren. Opera: Laat bij het invoeren van Japanese soms fonetische karakters staan. Werkt verder naar behoren.
	Mouse / Mini-mouse / Trackball	
	Stylus	Werkt naar behoren.
	Touchpad	
	Touchscreen	Werkt wat moeizaam, maar dit zou ook aan de site waarin de demo embedded is kunnen liggen.
Google Docs	Physical keyboard	
	Software keyboard	Werkt naar behoren inclusief autocomplete.

	Mouse / Mini-mouse / Trackball	
	Stylus	Werkt naar behoren.
	Touchpad	
	Touchscreen	Werkt naar behoren.
Google Script	Physical keyboard	
	Software keyboard	Werkt naar behoren, inclusief autocomplete.
	Mouse / Mini-mouse / Trackball	
	Stylus	Werkt naar behoren.
	Touchpad	
	Touchscreen	- Het verschuiven van de cursor werkt niet.
WebODF	Physical keyboard	
	Software keyboard	Werkt niet correct in de volgende vier browsers: - Android browser - Chrome - Firefox - Opera Problemen zijn: - Woorden komen incorrect door of met teveel letters. - Backspace werkt niet. - Composition talen werken niet (compositions kunnen niet gemaakt worden).
	Mouse / Mini-mouse / Trackball	
	Stylus	Werkt naar behoren.
	Touchpad	
	Touchscreen	- Het verplaatsen van de cursor werkt niet. - Reageert zeer schokkerig.
Ace	Physical keyboard	

	Software keyboard	Werkt niet correct in de volgende vier browsers: - Android browser - Chrome - Firefox - Opera Problemen zijn: - Woorden komen incorrect door of met teveel letters. - Backspace werkt niet. - Composition talen werken niet (compositions kunnen niet gemaakt worden).
	Mouse / Mini-mouse / Trackball	
	Stylus	Werkt naar behoren.
	Touchpad	
	Touchscreen	- Het verschuiven van de cursor werkt niet.
Code Mirror	Physical keyboard	
	Software keyboard	Werkt naar behoren, inclusief autocomplete.
	Mouse / Mini-mouse / Trackball	
	Stylus	Werkt naar behoren.
	Touchpad	
	Touchscreen	- Het verschuiven van de cursor werkt niet.

<u>BlackBerry OS</u>
Om de scope van het onderzoek te beperken is ervoor gekozen om geen experimenten op het BlackBerry OS platform uit te voeren. Dit vanwege de snel dalende marktaandeel van BlackBerry en het feit dat de nieuwste versies van BlackBerry OS gebruik maken van Android applicaties.

Geconcludeerd kan worden dat ondanks er bij meeste operating system en concurrent combinaties wel een probleem zit, er ook voor elke input methode wel een combinatie is die werkt. Het meest voorkomende probleem is het tekenen van de cursor aan de correcte kan bij het gebruik van Arabic. Voor het ontwikkelen van het proof of concept moet dus ook hiernaar gekeken worden.

3.3 Welke van de input methoden zullen door de eindgebruikers van FontoXML gebruikt worden?

Voor het beantwoorden van deze vraag zou er normaliter onderzoek gedaan moeten worden door middel van bijvoorbeeld het enquêteren van de eindgebruikers of het gebruik van een andere methode waarbij de eindgebruikers betrokken zouden zijn. Omdat FontoXML nog in ontwikkeling is zijn er momenteel nog geen bekende eindgebruikers. Voor het beantwoorden van deze vraag is er dus gebruik gemaakt van andere middelen.

Voor het ontwikkelen van de user experience twee persona aangemaakt.*[18][19]* Gebaseerd hierop zal een conclusie getrokken worden over hoe zij gebruik zouden willen maken van FontoXML. Ook is er een gesprek geweest met Bert Willemse waarin het gebruik van devices in combinaties met FontoXML besproken is.

Uit een gesprek met Bert Willemse is gebleken dat het voor de scope van het project slim is om de volgende vormen tot een later tijdstip opzij te zetten:

- BlackBerry OS vanwege het sterk afnemende en kleine marktaandeel van dit operating system.
- Exotisch weinig voorkomende input methoden zoals finger tracking, chorded keyboards en LPFK's

Ook bleek uit het gesprek dat het wel slim is om niet westerse input zoals Japans, Chinees etc. te gaan ondersteunen, alhoewel dit niet uit de huidig aanwezig persona zal blijken.

De persona zijn als volgt:

THE PERSON	THE JOB
* Intelligent	* Occasional writer of single topics
* Passion for engineering	* Content creation, metadata creation
* Perfectionist	* With the company for 17 years
* Feels he knows better	* Subject matter expert

MOTIVATION	FRUSTRATION
* Structuring helps him in ordering his product knowledge. * Wants his products to be used properly. Documentation helps people do that.	* The time it takes to write his documentation * Importing data from other systems is way too complicated * “Useless” fields in templates * Software “just never works”

FONTO OFFERS JORGEN
* Template-oriented writing * Distraction-free editor: no unnecessary options * Structuring guidance by explaining template-fields

THE PERSON	THE JOB
* Educated * Passion for writing * Tidy * Enthousiastic * Perfectionist	* Professional writer of narrative content * Content creation, metadata creation, structuring, adding semantics * Feels responsible for the end product

MOTIVATION	FRUSTRATION
* Wants to understand structure * Affection with the end product * Sharing her knowledge in a valuable way	* Losing control of her project * Not enough time to work as carefully as she would like * Inconsistency in editor behaviour * The meaning of “topic types”

FONTO OFFERS OLIVIA
* Freedom in writing and managing her topics * A useful amount of structuring options * Motivation by visualising the end product * Guidance by suggesting useful improvements and wizards * An intuitive way to re-use content

Om een conclusie te trekken over de persona is voornamelijk gekeken naar wat FontoXML in de huidige staat biedt voor deze personen. Welke methoden kunnen wij hun aanbieden zonder deze punten aan te passen / moeilijker te maken of om deze punten te verrijken?

Het belangrijkste dat we voor Jorgen kunnen we lezen is dat FontoXML hem een “Distraction-free editor” aanbiedt met “no unnecessary options”. De verwachting hierbij is dat uit de onderzochte methoden hij geneigd zal zijn om tablets te gebruiken die een groot scherm hebben, en opties bieden om het zo dicht mogelijk bij de standaard versie van FontoXML te kunnen houden. Smartphones zouden te onoverzichtelijk worden en zouden afleiden met hoe deze gebruikt moeten worden.

Voor Olivia kunnen we lezen dat FontoXML haar “freedom in writing and managing her topics” aanbiedt en “a useful amount of structuring options”. Om deze opties te kunnen vallen ook voor haar smartphones af. Deze zouden met het kleinere scherm simpel weg niet genoeg opties kunnen aanbieden.

Opvallend hierbij is dat bij het maken van de persona nog niet is gedacht aan bijvoorbeeld buitenlandse gebruikers. Dit waarschijnlijk omdat hier momenteel ook nog geen ondersteuning voor is.

Geconcludeerd kan worden dat de eindgebruikers naast hun computers eventueel tablets zouden gebruiken voor FontoXML. Ook kunnen we concluderen dat eindgebruikers uit regionen van de wereld waar gebruik wordt gemaakt van aparte input methoden zoals IME’s gebruik zouden willen maken van FontoXML.

3.4 Hoe gaan open source browsers om met het verwerken van input methoden die gebruik maken van compositions?

Voor het beantwoorden van deze vraag is de source van deze browsers gedownload en doorgezocht naar de manier van afhandeling voor de vooraf gedefineerde belangrijke input method/platform combinaties.

Deze deelvraag is niet beantwoord voor versie 1.0 van dit onderzoek.

3.5 Hoe gaan code editors om met het verwerken van input methoden die gebruik maken van compositions, IME's of een andere schrijfrichting?

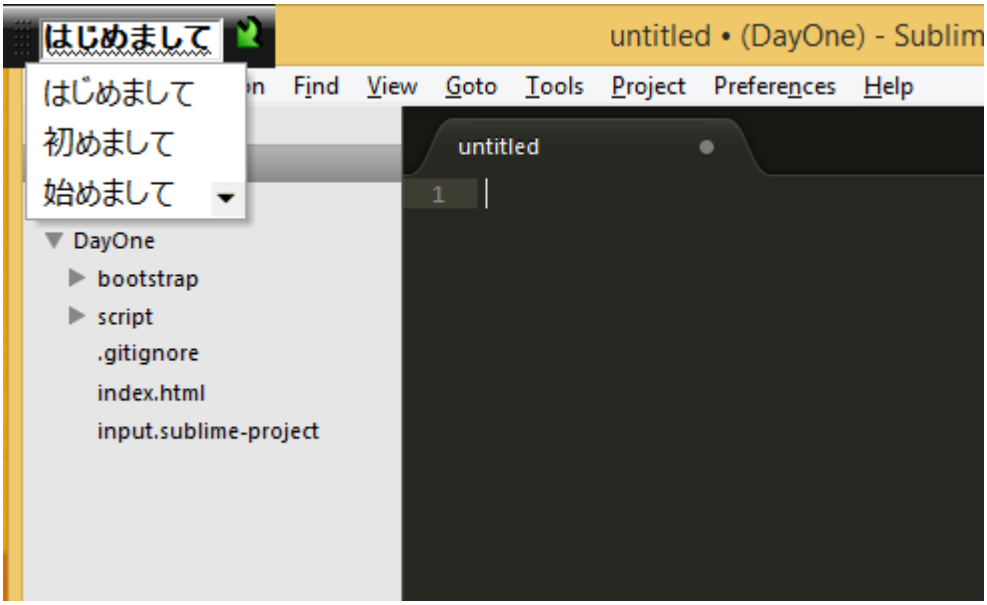
Voor het beantwoorden van deze vraag zijn vier code editors uitgeprobeerd met het invoeren van Japanese, Simplified Chinese en Arabic.

De code editors die getest zijn:

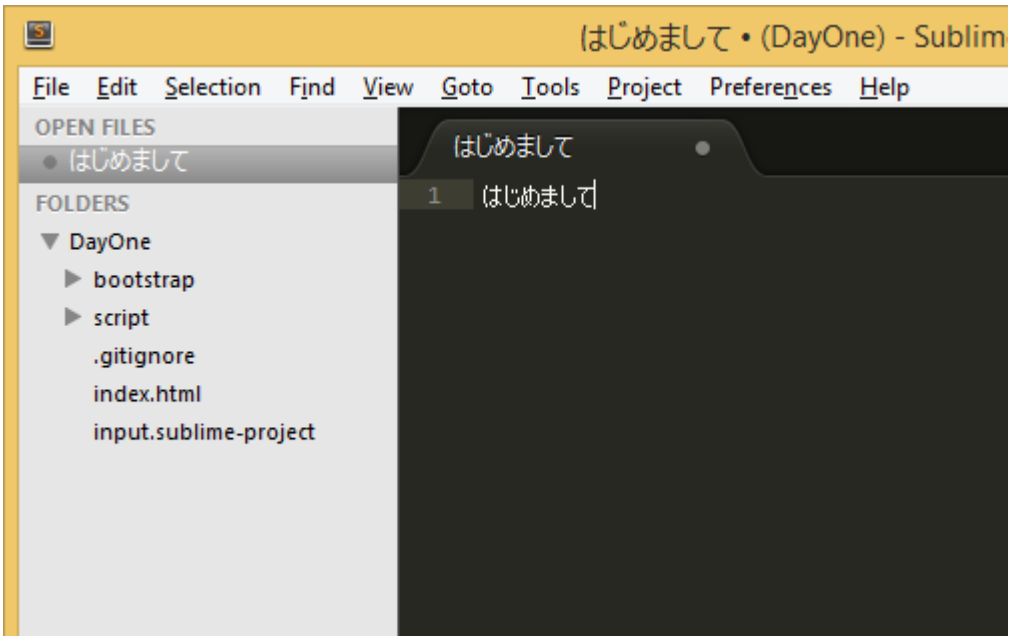
- Sublime Text 2
- PHPStorm
- Notepad++
- Microsoft Visual Studio

3.5.1 Sublime Text 2

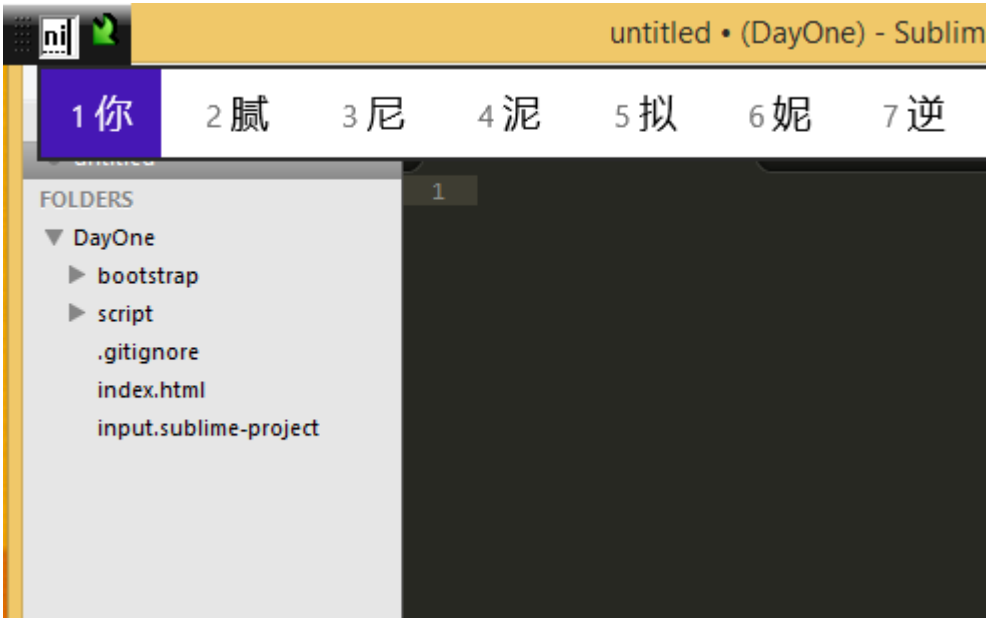
Bij het invoeren van Japanese zien we dat, in plaats van direct de input in de main view te plaatsen, Sublime Text 2 een aparte view opent waarin de input terecht komt en er gebruik gemaakt wordt van de IME. Hierin zijn ook gelijk de composition opties te zien en selecteren.



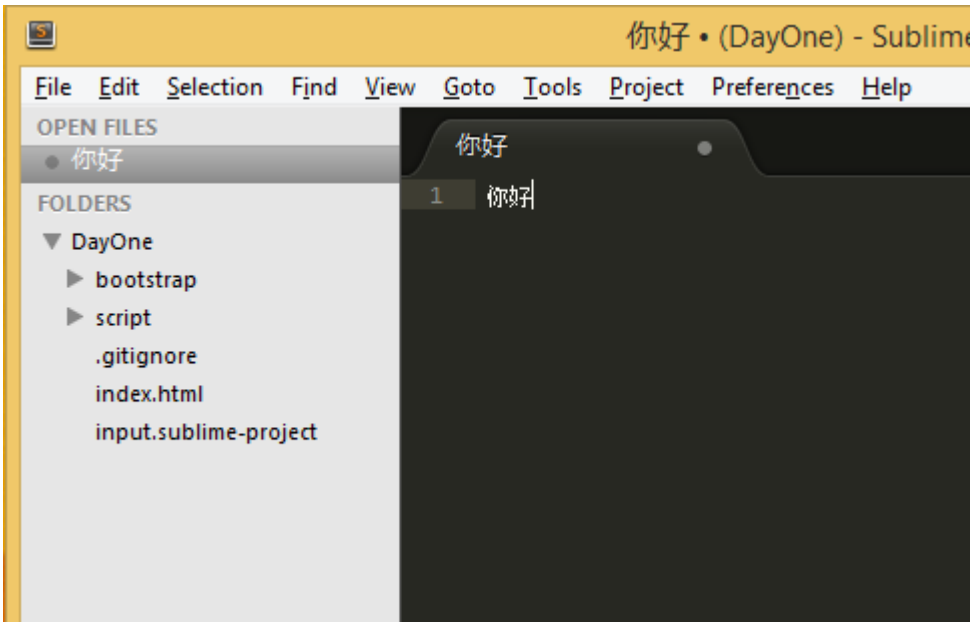
Pas wanneer er op enter gedrukt wordt, wordt de (composition) input in de main view geplaatst.



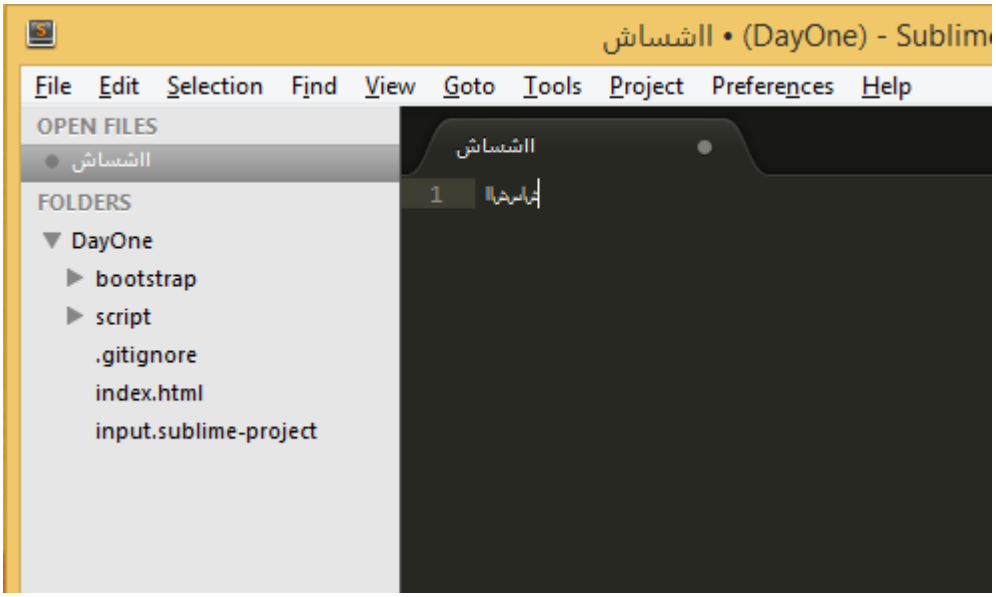
Ook bij Simplified Chinese zien we dat, in plaats van de input direct in de main view te verwerken, Sublime Text 2 een aparte view opent voor het gebruik van de Pinyin IME.



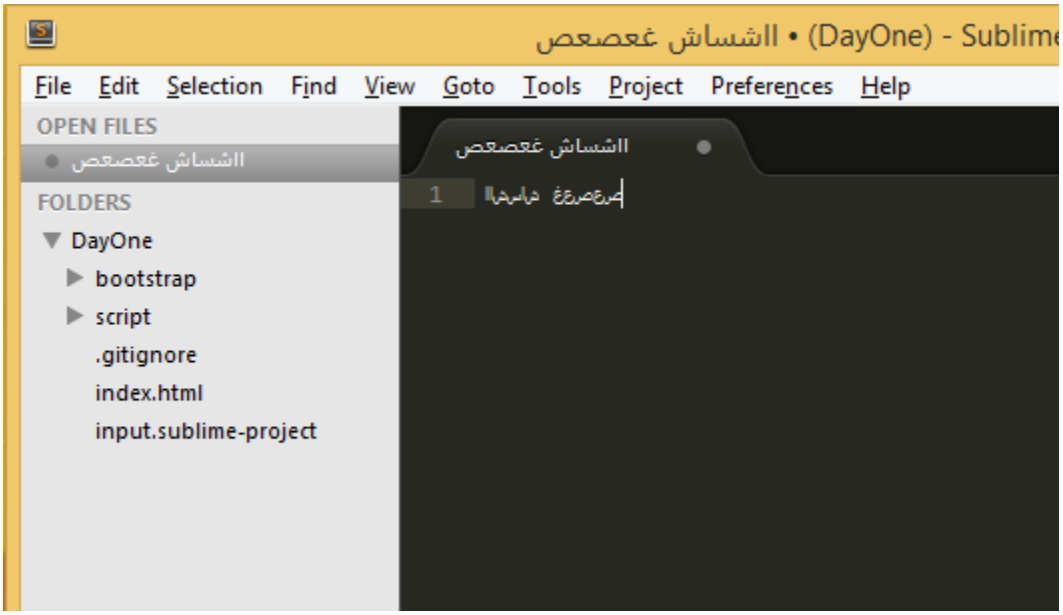
Pas wanneer er op enter gedrukt wordt, wordt de input in de main view geplaatst.



Bij het invoeren van Arabic wordt de input direct in de main view geplaatst. Wel wordt de cursor aan de verkeerde kant getekend. Dit zou links moeten zijn in het geval van Arabic.

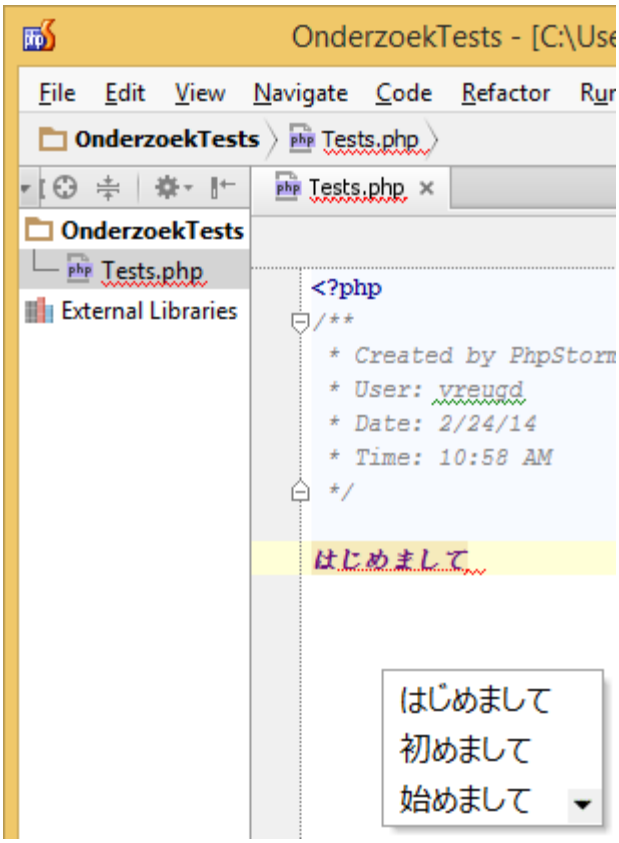


Wat we ook zien bij Sublime Text 2 is dat bij het gebruik van Arabic de invoer directie niet correct wordt aangepast. Bij Arabic moet dit van rechts naar links zijn, maar in Sublime Text 2 blijft dit van links naar rechts.

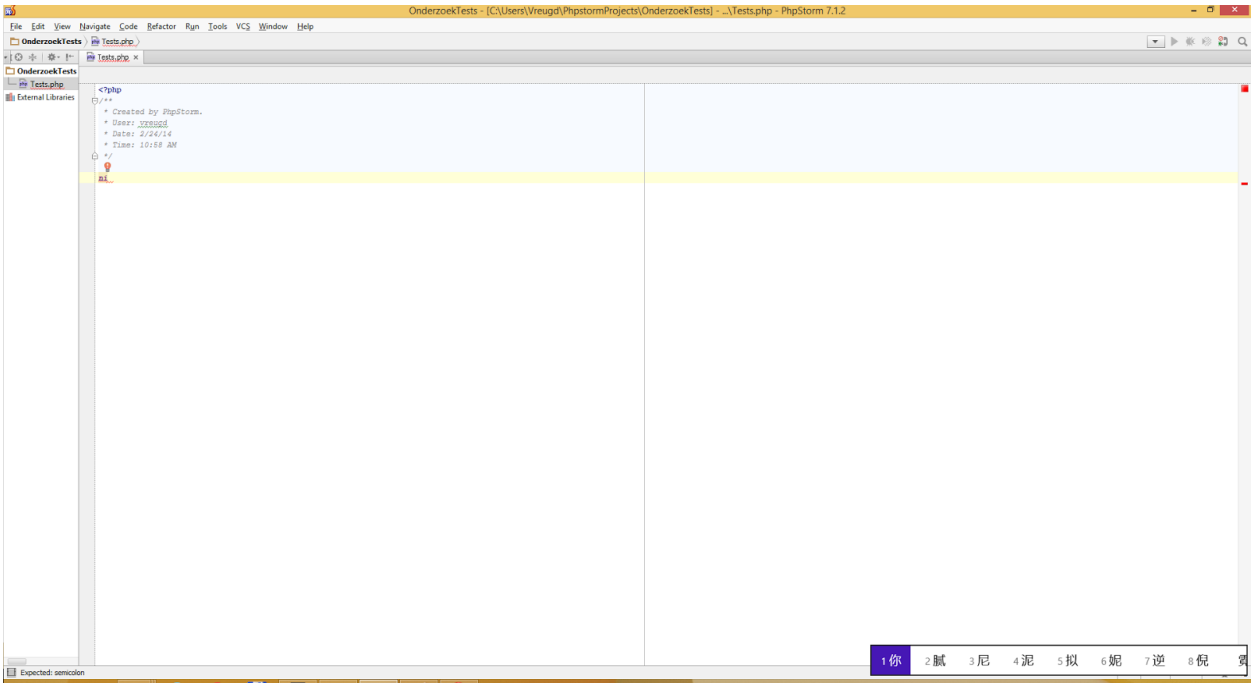


3.5.2 PHPStorm

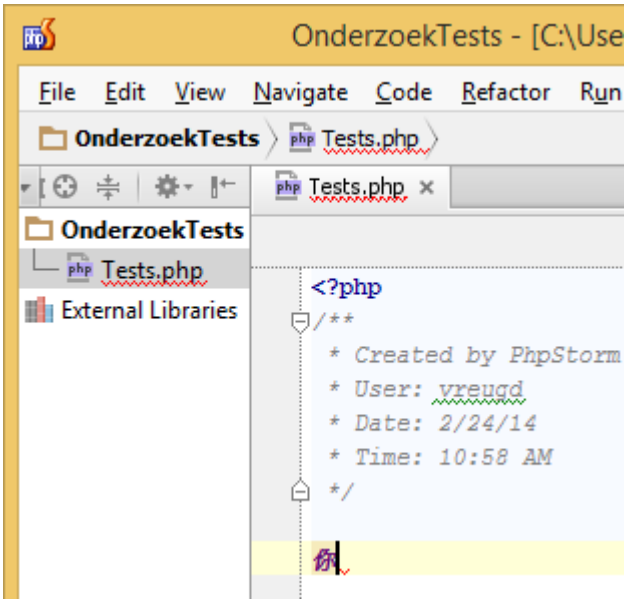
Bij het invoeren van Japanese wordt de input direct in de main view geplaatst, wat we hierbij zien is dat er voor gekozen is om het IME scherm 4 regels lager te plaatsen zodat een overview over de onderstaande regels bewaart blijft.



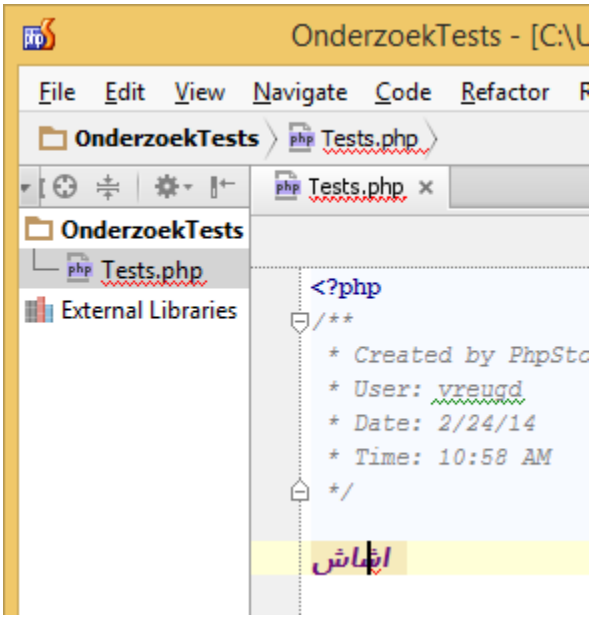
Bij het invoeren van Simplified Chinese zien we dat het fonetisch getypte als eerste verschijnt in de main view. De Pinyin IME wordt rechts onder in het scherm getoont.



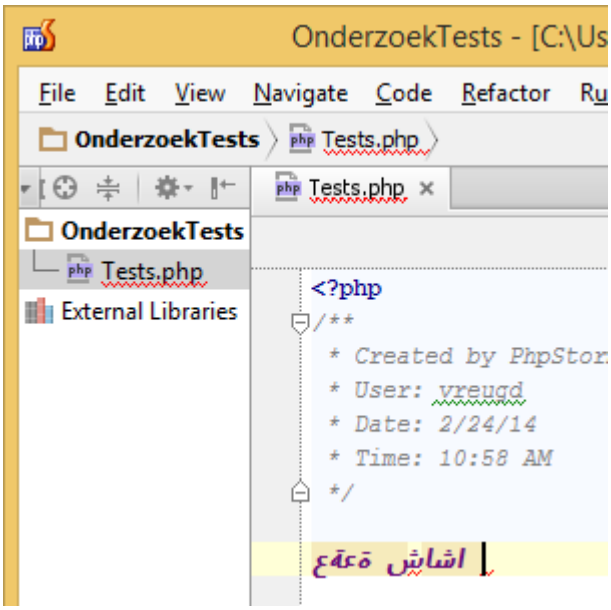
Na het maken van een selectie uit de Pinyin IME wordt het Simplified Chinese in de main view geplaatst als vervanger voor het fonetisch getypte.



Bij het invoeren van Arabic wordt de input direct in de main view geplaatst. En we zien dat de cursor niet op de juiste plek geplaatst wordt.

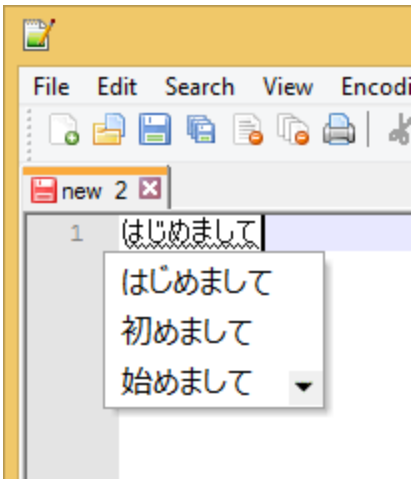


Wat we ook zien bij het invoeren van Arabic input is dat tijdens het typen het getypte wel aan de rechterkant van de lijn verschijnt, maar na het gebruik van een enter of spatie aan de linkerkant van de lijn geplaatst wordt.

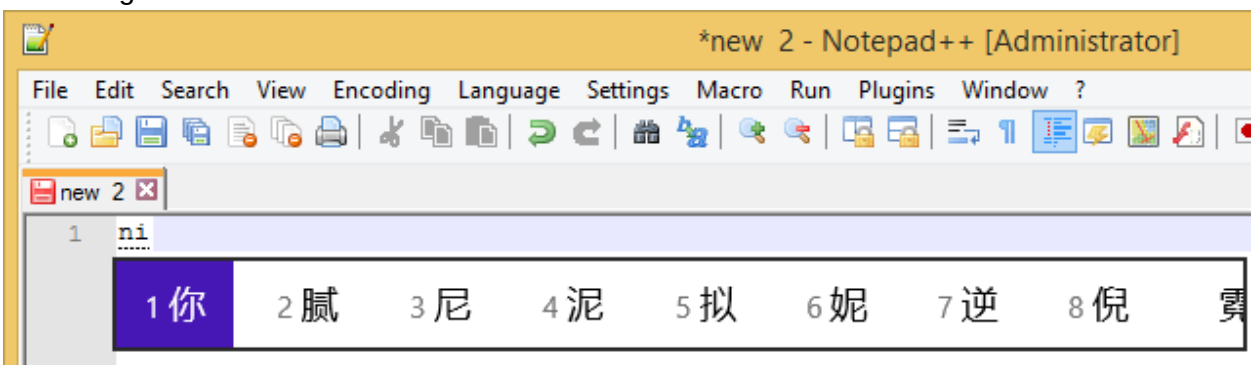


3.5.3 Notepad++

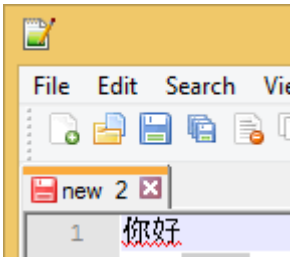
Bij het invoeren van Japanese zien we dat de input direct in de main view geplaatst wordt, ook zien we dat het IME direct onder de regel geplaatst wordt zoals dit ook normaliter bij het typen van Japanese gebeurd.



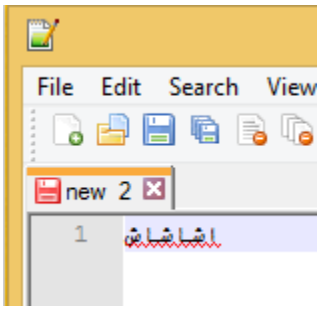
Bij het invoeren van Simplified Chinese wordt eerst het fonetisch getypte in de main view geplaatst met daaronder het Pinyin IME zoals dit ook normaliter bij het typen van Simplified Chinese gebeurd.



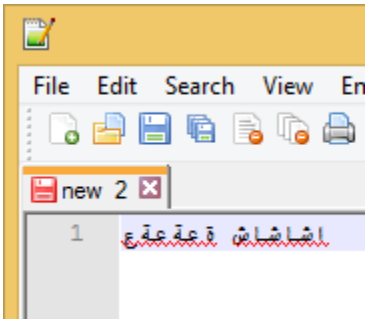
Na het selecteren uit het Pinyin IME menu wordt de fonetische tekst vervangen door Simplified Chinese.



Bij het invoeren van Arabic wordt de input direct in de main view geplaatst.

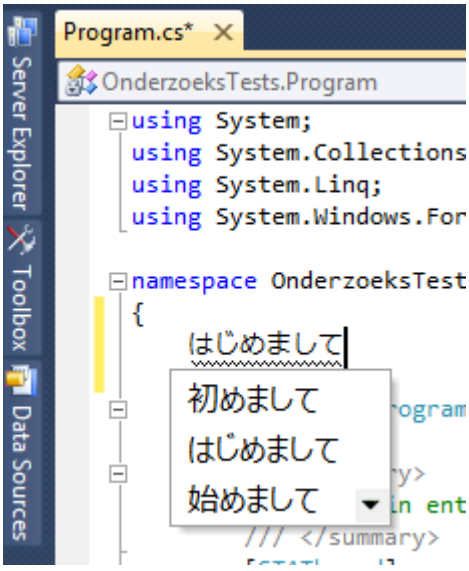


Wat we wel zien bij het invoeren van Arabic in Notepad++ is dat de tekst wel correct van rechts naar links getypt kan worden, maar de cursor rechts in plaats van links getoond wordt.

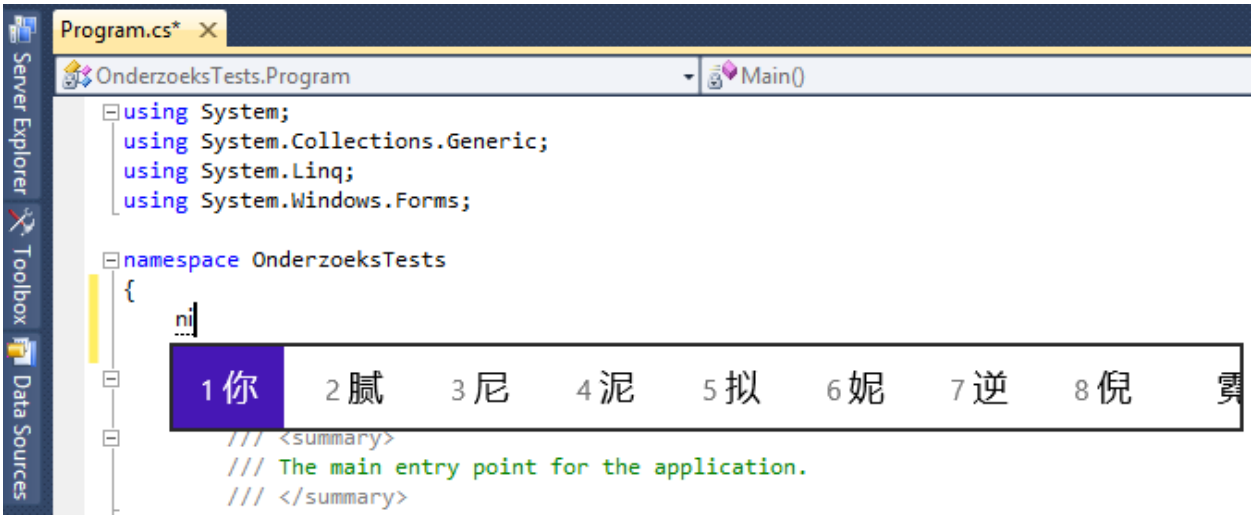


3.5.4 Microsoft Visual Studio

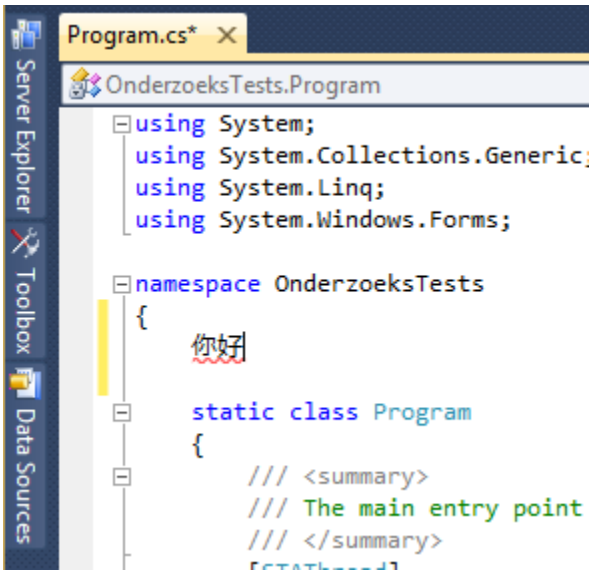
Bij het invoeren van Japanese zien we dat de input direct in de main view geplaatst wordt, ook zien we dat het IME direct onder de regel geplaatst wordt zoals dit ook normaliter bij het typen van Japanese gebeurd.



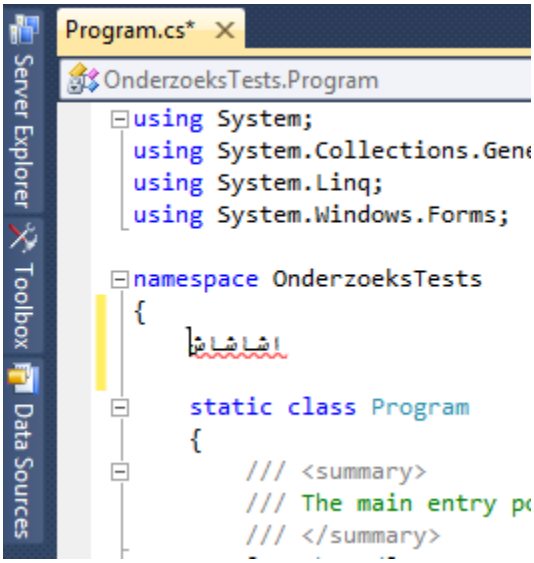
Bij het invoeren van Simplified Chinese wordt eerst het fonetisch getypte in de main view geplaatst met daaronder het Pinyin IME zoals dit ook normaliter bij het typen van Simplified Chinese gebeurd.



Na het selecteren uit het Pinyin IME menu wordt de fonetische tekst vervangen door Simplified Chinese.



Bij het invoeren van Arabic wordt de input direct in de main view geplaatst ook wordt de schrijfrichting aangepast naar van rechts naar links en wordt de cursor op de juiste plaats getekend.



4. Conclusie

Als eindconclusie op de vraag “Welke input methoden op welke devices en platforms moeten toegevoegd worden aan FontoXML?” is het volgende geconcludeerd:

Om een breder publiek aan te kunnen spreken met FontoXML is het verstandig om ondersteuning te gaan aanbieden voor IME gebruikende talen en talen met een schrijfrichting van rechts naar links. Ook moet algemene ondersteuning in aangeboden gaan worden voor de volgende combinaties:

- Android - Android browser
- Android - Chrome
- iOS - Safari
- iOS - Chrome

Deze combinaties zijn geselecteerd op de conclusies van de verschillende deelvragen en hoe goed deze presteerde bij het gebruik van Xeditor. De concurrent die het meest te vergelijken is met FontoXML. Ook zijn de meest populaire browsers uitgekozen voor beide tablet operating systems, deze presteerde misschien op sommige vlakken minder dan andere browsers maar mogen vanwege hun populariteit niet missen.

Bibliografie*

[01] (Wikipedia) http://en.wikipedia.org/wiki/WYSIWYG
[02] (Wikipedia) http://en.wikipedia.org/wiki/Input_device
[03] (Wikipedia) http://en.wikipedia.org/wiki/Input_method_editor
[04] (Wikipedia) http://en.wikipedia.org/wiki/Japanese_input_methods
[05] (Wikipedia) http://en.wikipedia.org/wiki/Computer_keyboard
[06] (Wikipedia) http://en.wikipedia.org/wiki/Pointing_device
[07] (Search engines e.d.) Het bepalen van de beschikbaarheid van input devices per operating system is gedaan door simpel search engine werk en is onder voorbehoud. Deze resultaten zijn onder voorbehoud aangezien het altijd mogelijk is dat input methoden alsnog toegevoegd/ondersteund gaan worden.
[08] (Statistics) http://download.cnet.com/windows/web-browsers/
[09] (Statistics) http://download.cnet.com/mac/web-browsers/
[10] (Statistics) http://www.netmarketshare.com/browser-market-share.aspx?qprid=0&qpcustomd=1
[11] (News) http://androidworld.nl/nieuws/blackberry-10-gaat-android-apps-ondersteunen/
[12] (Demo) http://www.xeditor.com/portal/xeditor/en/xeditor-live-demo-751
[13] (Web application) https://drive.google.com/
[14] (Web application) http://www.google.com/script/start/
[15] (Demo) http://www.webodf.org/demo/
[16] (Demo) http://ace.c9.io/
[17] (Demo) http://codemirror.net/
[18] (Intern document) https://docs.google.com/a/liones.nl/file/d/0B78Tdxw8Fa55RnhneEUwcDROQnc/edit
[19] (Intern document) https://docs.google.com/a/liones.nl/file/d/0B78Tdxw8Fa55aXFncEtYLTZOUFU/edit

* Bronnen van het type (Statistics) worden real-time berekent en zijn dus gebruikt zoals ze aangetroffen zijn ten tijden van het onderzoek.

Adviesrapport

Project:	**Input detectie**
Date:	13-03-2014
Owner:	Bert Willems
Client:	Liones
Version:	v1.0
Project members:	Michael de Vreugd

Inhoudsopgave

Inleiding 127

1 Probleemstelling en hoofdvraag 128

1.1 Probleemstelling 128

1.2 Hoofdvraag 128

2 Methode en werkwijze 129

3 Resultaten 130

3.1 Wat zijn de verschillen tussen alle input methoden op de verschillende operating systems, browsers en devices? 130

3.2 Welke input methoden ondersteunen de concurrenten? 130

3.3 Welke van de input methoden zullen door de eindgebruikers van FontoXML gebruikt worden? 131

4 Conclusies en aanbevelingen 132

Inleiding

Dit rapport is geschreven met als doelstelling om een beknopte versie van het uitgevoerde onderzoek te geven en uiteindelijk advies uit te brengen betreffende de kwestie welke input methoden op welke platforms toegevoegd moeten worden aan FontoXML.

Om tot een conclusie te kunnen komen voor deze kwestie was het uitvoeren van een onderzoek nodig. Het onderzoek is uitgevoerd vanaf 13-02-2014 t/m 12-03-2014. Het onderzoek is uitgevoerd door Michael de Vreugd met hulp van Thomas Brekelmans voor de experiment gedeeltes op Mac OS. In het onderzoek is onderzocht wat de verschillen zijn tussen de beschikbare input methoden op de verschillende operating systems, browsers en devices, wat de concurrenten ondersteunen betreffende input methoden en welke input methoden door de eindgebruikers van FontoXML gebruikt zouden worden. Waar ook naar gekeken is hoe code editors omgaan met het verwerken van input methoden die gebruik maken van compositions, IME's of een andere schrijfrichting. Wat voor de eerste versie van het onderzoek niet meegenomen is om de scope te beperken zijn de volgende punten:

- Experimenten met een chorded keyboard en een LPFK.
- Experimenten met het Windows Phone operating system.
- Experimenten met het BlackBerry OS operating system.
- Het uitzoeken van hoe open source browsers omgaan met het verwerken van input methoden die gebruik maken van compositions.

Het is mogelijk dat deze punten op een later tijdstip alsnog verwerkt worden in het onderzoek indien nodig.

Gebaseerd op dit advies zal er een proof of concept ontwikkeld worden waarin de geselecteerde input methoden afgevangen worden.

1 Probleemstelling en hoofdvraag

In dit hoofdstuk wordt de probleemstelling nogmaals vertelt en de hoofdvraag waarover het advies gaat.

1.1 Probleemstelling

Liones ontwikkelt de FontoXML tekst editor. Bij het ontwikkelen van deze WYSIWYG tekst editor hebben zij zich voornamelijk gericht op de klassieke invoer mogelijkheden: een westers toetsenbord en muis. Om FontoXML aantrekkelijker te maken voor een breder publiek willen zij onderzoek uitvoeren naar de gebruikersbehoefte voor ondersteuning voor andere invoerapparaten en vervolgens ondersteuning aanbieden. Hierbij moet gedacht worden aan touchscreens op tablets, maar ook verschillende toetsenborden of inputapparaten voor mensen met een beperking. Hierbij zijn zij vooral geïnteresseerd in onscreen toetsenborden met autocomplete/autocorrect (mobiele platforms) en Input Method Editors (IME's). Om FontoXML geschikt te maken voor deze omgevingen en inputapparaten willen zij onderzoek uitvoeren naar de verschillende vormen van input en zal een proof of concept gemaakt worden voor het robuust detecteren van invoer, in allerlei vormen, op allerlei browsers en platforms.

1.2 Hoofdvraag

Welke input methoden op welke devices en platforms moeten toegevoegd worden aan FontoXML?

2 Methode en werkwijze

In dit hoofdstuk wordt uitgelegd welke methoden en werkwijzen er gebruikt zijn voor het onderzoek.

De volgende methoden zijn toegepast voor het onderzoek:

- Big6 ™:
  Een zes stappen model wat als doel heeft te helpen bij het oplossen van problemen en het maken van beslissingen met behulp van informatie.
- Snowball:
  De Snowball methode is een manier van het uitvoeren en gebruiken van literatuuronderzoek. Bij deze methode wordt een literatuurlijst bijgehouden van alle gevonden bronnen van informatie. Vervolgens wordt er van deze bronnen bekeken of de hoofdbron of eventueel bron verwijzingen binnen de bron ook nuttige informatie opleveren voor het onderzoek.

Voor het uitvoeren van het onderzoek zelf is er eerst begonnen met vooronderzoek naar de hoofdvraag. Omdat de hoofdvraag zo specifiek is toegespitst op het FontoXML product kwam hier geen resultaat uit. Hierna is wat licht vooronderzoek uitgevoerd naar van te voren aangegeven problemen met het detecteren van input op bepaalde operating system en browser combinaties, dit leverde bronnen op die in de toekomst gebruikt konden worden voor het onderzoek en eventueel voor het ontwikkelen van het proof of concept. Vervolgens is er literatuuronderzoek uitgevoerd dit leverde de benodigde informatie en bronnen op die nodig waren om te beginnen met het beantwoorden van de deelvragen. Tijdens het uitvoeren van het onderzoek zijn er ook verschillende experimenten uitgevoerd met het afvangen van input. Dit alles bij elkaar heeft ervoor gezorgd dat de deelvragen en uiteindelijk de hoofdvraag beantwoordt konden worden.

3 Resultaten

In dit hoofdstuk wordt per deelvraag uit het onderzoek kort uitgelegd wat er precies uitgevoerd is voor dat deel van het onderzoek en wat de conclusie was voor de betreffende deelvraag. Voor de volledige uitwerking kan het onderzoeksrapport gelezen worden.

3.1 Wat zijn de verschillen tussen alle input methoden op de verschillende operating systems, browsers en devices?

Voor het beantwoorden van deze deelvraag is er als eerste onderzoek gedaan naar wat voor verschillende input methoden er bestaan. Deze zijn toen in een matrix gezet en verschaft van een uitleg per hoofdtype en subtype. Wat daaruit geconcludeerd kon worden was dat onder de hoofdtypes het grootste verschil zit. Bij de sub-types is de manier van invoeren wel verschillend, maar het uiteindelijke resultaat blijft hetzelfde. Een ander groot verschil zit in de verschillende beschikbare toetsenborden toegespitst op regionen en/of talen.

Hierna is in twee matrices vastgelegd wat de beschikbaarheid van de verschillende input methoden is per operating system en is onderzoek gedaan naar de beschikbare browsers per operating system en met name de meest populaire browsers. Deze zijn vastgelegd in matrices. Toen dit gedaan was zijn er experimenten uitgevoerd betreffende het afvangen van de input en de snelheid waarmee dit gebeurd. Wat daar uit geconcludeerd kon worden was dat het verschil op operating system niveau voornamelijk tussen mobile en pc operating systems ligt, waarbij er ook weer een groot verschil is in welke browser gebruikt wordt. Algemene reactie tijden voor mobile operating systems zijn langzamer dan de operating systems voor pc's. Over het geheel genomen is Mac OS het snelste met de afhandeling van de events. Ook zien we verschil tussen het wel of niet kunnen afvangen van events per operating system of de manier waarop de events worden afgehandeld.

3.2 Welke input methoden ondersteunen de concurrenten?

Voor het beantwoorden van deze deelvraag zijn er experimenten uitgevoerd bij de volgende concurrenten:

- Xeditor
- Google Docs
- Google Script
- WebODF
- Ace
- Code Mirror

Bij het uitvoeren van experimenten zijn voor het testen van een toetsenbord de volgende talen getest:

- US_English
- Japanese
- Chinese(Simplified)
- Arabic

Wat hieruit geconcludeerd kon worden was dat ondanks er bij meeste operating system en concurrent combinaties wel een probleem zit, er ook voor elke input methode wel een combinatie is die werkt. Het meest voorkomende probleem is het tekenen van de cursor aan de

correcte kan bij het gebruik van Arabic. Voor het ontwikkelen van het proof of concept moet dus ook hiernaar gekeken worden.

3.3 Welke van de input methoden zullen door de eindgebruikers van FontoXML gebruikt worden?

Om deze deelvraag te kunnen beantwoorden is er gekeken naar de twee persona die aangemaakt zijn voor het user experience team en naar de uitkomst van een gesprek met Bert Willemse. Wat hieruit geconcludeerd kon worden was dat de eindgebruikers naast hun computers eventueel tablets zouden gebruiken voor FontoXML. Ook kunnen we concluderen dat eindgebruikers uit regionen van de wereld waar gebruik wordt gemaakt van aparte input methoden zoals IME's gebruik zouden willen maken van FontoXML

4 Conclusies en aanbevelingen

Als eindconclusie op de vraag “Welke input methoden op welke devices en platforms moeten toegevoegd worden aan FontoXML?” is het volgende geconcludeerd:

Om een breder publiek aan te kunnen spreken met FontoXML is het verstandig om ondersteuning te gaan aanbieden voor IME gebruikende talen en talen met een schrijfrichting van rechts naar links. Ook moet algemene ondersteuning in aangeboden gaan worden voor de volgende combinaties:

- Android - Android browser
- Android - Chrome
- iOS - Safari
- iOS - Chrome

Deze combinaties zijn geselecteerd op de conclusies van de verschillende deelvragen en hoe goed deze presteerde bij het gebruik van Xeditor. De concurrent die het meest te vergelijken is met FontoXML. Ook zijn de meest populaire browsers uitgekozen voor beide tablet operating systems, deze presteerde misschien op sommige vlakken minder dan andere browsers maar mogen vanwege hun populariteit niet missen.

Requirements en product backlog

Project:	**Input detectie**
Date:	18-03-2014
Owner:	Bert Willems
Client:	Liones
Version:	v1.1
Project members:	Michael de Vreugd

Inleiding

In dit document zijn de requirements opgesteld waaraan het proof of concept voor het “Input detectie” project moet voldoen. Deze requirements zijn opgesteld aan de hand van de resultaten van het onderzoeksrapport en de wensen vanuit Liones. Voor dit project zijn alle requirements van het functional type. Uit deze lijst is een product backlog opgesteld conform de SCRUM methode. De items in deze lijst zijn onderverdeeld in de volgende vijf categorieën: user stories, , bugs, chores, epics en prototypes. Een user story is een taak die nieuwe functionaliteit representeert, een bug is een taak die werk aan een defect representeert, chore is een taak die gedaan moet worden maar geen business value heeft, een epic is een taak die te groot is om binnen een sprint te kunnen en een prototype is een taak die nodig is voor het maken van een beslissing maar waarvan het niet vast staat of de functionaliteit waarde zal hebben. De product log kan en zal nog groeien naarmate het project vordert.

Requirements

Het proof of concept moet:

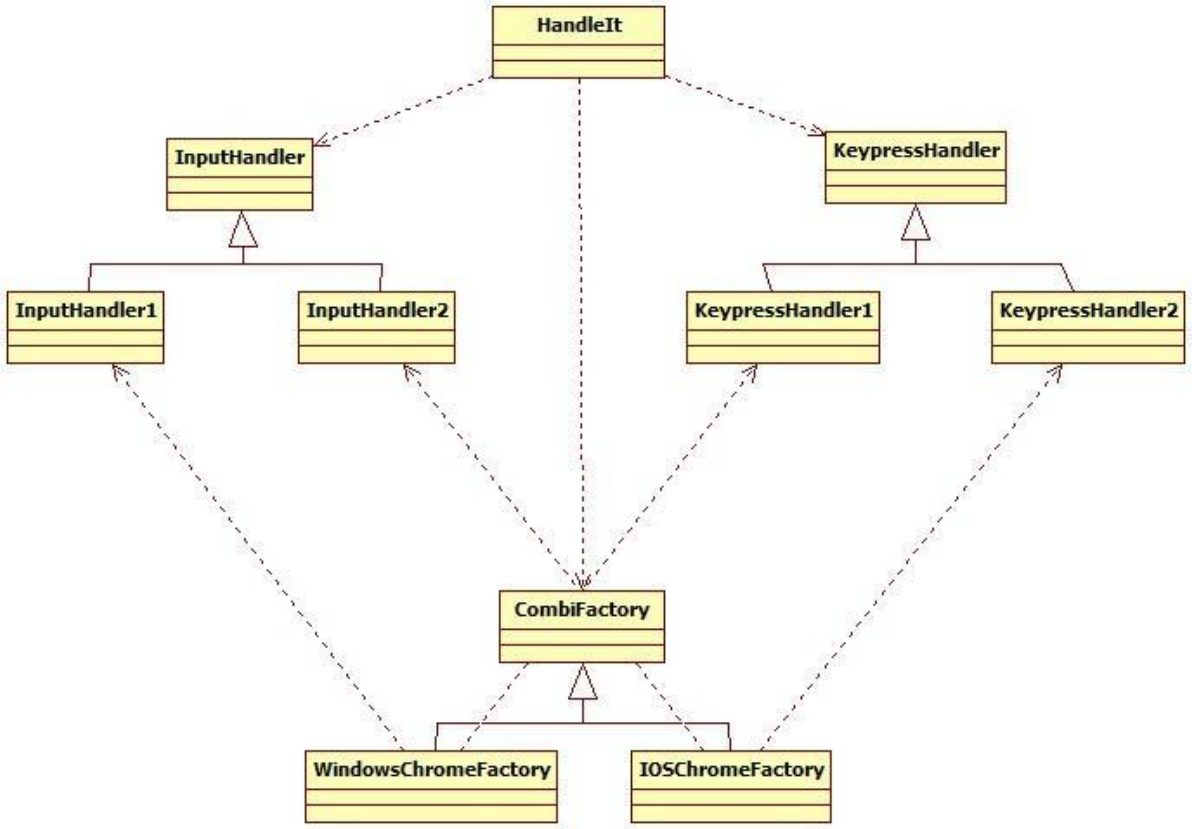
- Ontwikkeld worden in de vorm van een JavaScript library voor een goede aansluiting op FontoXML.
- Event handling bevatten voor:
    - Standaard input(westers toetsenbord)
    - IME's
    - Compositions
    - Auto-completes / auto-corrects.
- Event handling bevatten voor het gebruik van Chrome op iOS en Android.
- Bij het afvangen van input alles bewaren en ook in de originele volgorde.
- Robuust en uitbreidbaar zijn.
- Hot-key handling moet vrij blijven zodat FontoXML deze door middel van aparte modules kan afhandelen. Speciale toetsen zoals backspace vallen ook onder hot-key handling.
- Als output een event stream aanbieden. Deze stream mag de DOM niet aanpassen zodat FontoXML de bewerking eerst kan valideren op de XML.
- Alleen actief zijn wanneer nodig, wanneer dit niet het geval is moeten er geen onnodige controles uitgevoerd worden zodat de library geen ongewenste invloed uitoefent.
- Een manier van monitoring aanbieden. Hierin moet duidelijk zijn welke strategie er toegepast wordt en waarom, zodat hier eventuele afwijkingen uit afgelezen kunnen worden.

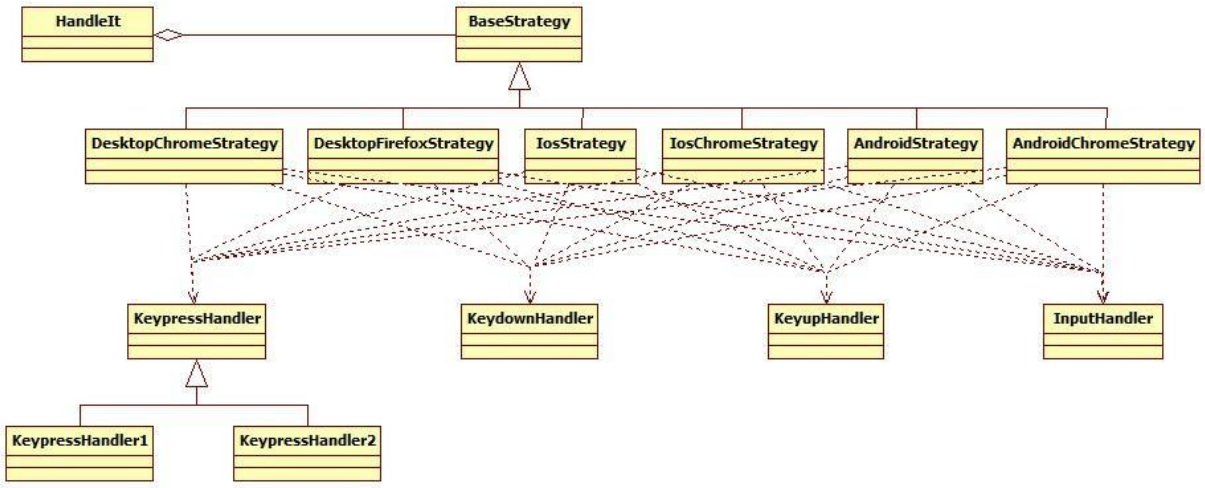
Product backlog

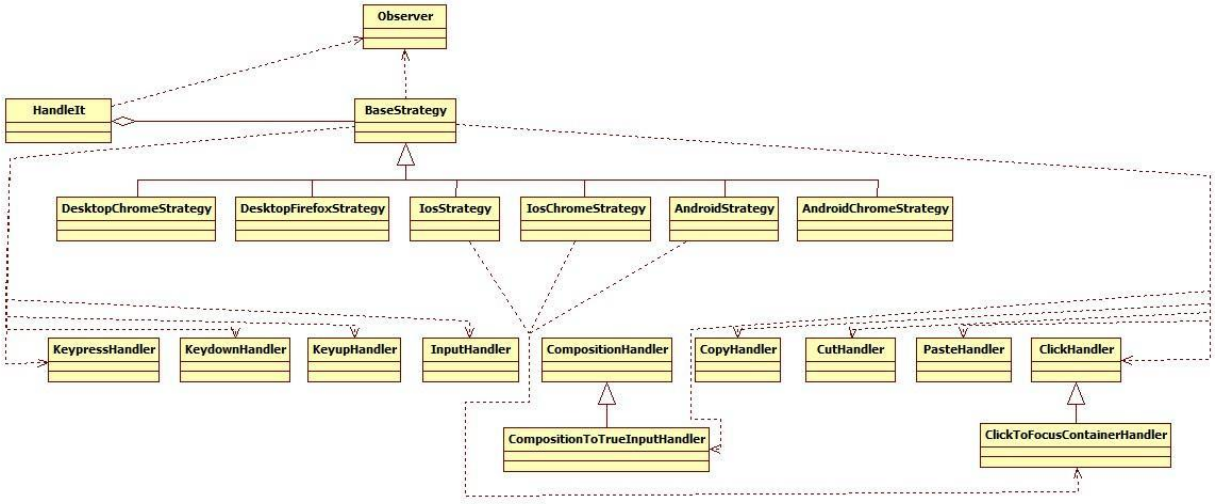
- USER STORIES:
    - Basis structuur van de library aanbrengen
    - Event handling: westers toetsenbord
    - Output handling: westers toetsenbord
    - Event handling: IME's
    - Output handling: IME's
    - Event handling: compositions en auto-completes
    - Output handling: compositions en auto-completes
    - Event handling: Chrome op iOS
    - Output handling: Chrome op iOS
    - Event handling: Chrome op Android
    - Output handling: Chrome op Android
    - Sortering event handling
    - Hot-key handling
    - Monitoring
- BUGS:
    -
- CHORES:
    -
- EPICS:
    - Robuustheid en uitbreidbaarheid
    - Output stream restricties
    - Bepaling wanneer de library actief is
- PROTOTYPES:
    - Maak een basis javascript library

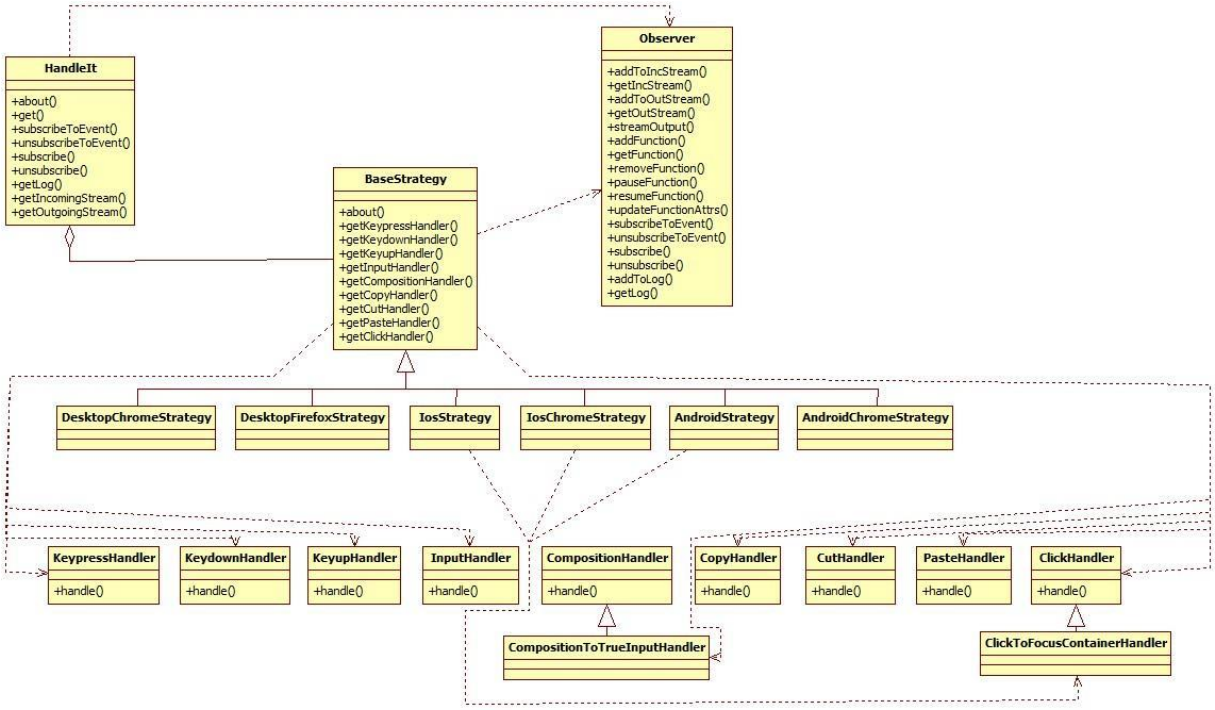
Ontwerpen

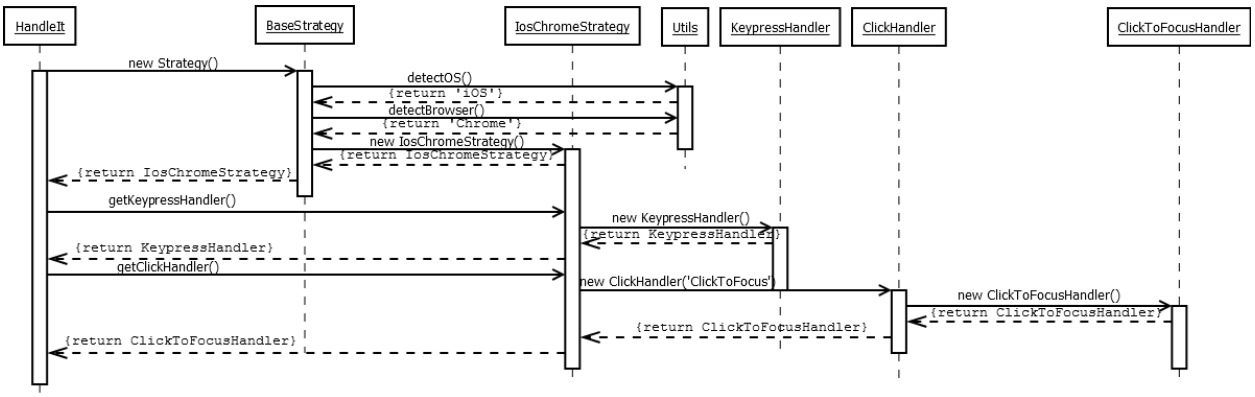
Project:	**Input detectie**
Date:	03-06-2014
Owner:	Bert Willems
Client:	Liones
Version:	v1.0
Project members:	Michael de Vreugd


HandleIt
InputHandler
KeypressHandler
InputHandler1
InputHandler2
KeypressHandler1
KeypressHandler2
CombiFactory
WindowsChromeFactory
IOSChromeFactory


HandleIt
BaseStrategy
DesktopChromeStrategy
DesktopFirefoxStrategy
IosStrategy
IosChromeStrategy
AndroidStrategy
AndroidChromeStrategy
KeypressHandler
KeydownHandler
KeyupHandler
InputHandler
KeypressHandler1
KeypressHandler2


Observer
HandleIt
BaseStrategy
DesktopChromeStrategy
DesktopFirefoxStrategy
IosStrategy
IosChromeStrategy
AndroidStrategy
AndroidChromeStrategy
KeypressHandler
KeydownHandler
KeyupHandler
InputHandler
CompositionHandler
CopyHandler
CutHandler
PasteHandler
ClickHandler
CompositionToTrueInputHandler
ClickToFocusContainerHandler


HandleIt
+about()
+get()
+subscribeToEvent()
+unsubscribeToEvent()
+subscribe()
+unsubscribe()
+getLog()
+getIncomingStream()
+getOutgoingStream()
Observer
+addToIncStream()
+getIncStream()
+addToOutStream()
+getOutStream()
+streamOutput()
+addFunction()
+getFunction()
+removeFunction()
+pauseFunction()
+resumeFunction()
+updateFunctionAttrs()
+subscribeToEvent()
+unsubscribeToEvent()
+subscribe()
+unsubscribe()
+addToLog()
+getLog()
BaseStrategy
+about()
+getKeypressHandler()
+getKeydownHandler()
+getKeyupHandler()
+getInputHandler()
+getCompositionHandler()
+getCopyHandler()
+getCutHandler()
+getPasteHandler()
+getClickHandler()
DesktopChromeStrategy
DesktopFirefoxStrategy
IosStrategy
IosChromeStrategy
AndroidStrategy
AndroidChromeStrategy
KeypressHandler
+handle()
KeydownHandler
+handle()
KeyupHandler
+handle()
InputHandler
+handle()
CompositionHandler
+handle()
CopyHandler
+handle()
CutHandler
+handle()
PasteHandler
+handle()
ClickHandler
+handle()
CompositionToTrueInputHandler
ClickToFocusContainerHandler


HandleIt
BaseStrategy
IosChromeStrategy
Utils
KeypressHandler
ClickHandler
ClickToFocusHandler
new Strategy()
detectOS()
{return 'iOS'}
detectBrowser()
{return 'Chrome'}
new IosChromeStrategy()
{return IosChromeStrategy}
{return IosChromeStrategy}
getKeypressHandler()
new KeypressHandler()
{return KeypressHandler}
{return KeypressHandler}
getClickHandler()
new ClickHandler('ClickToFocus')
new ClickToFocusHandler()
{return ClickToFocusHandler}
{return ClickToFocusHandler}
{return ClickToFocusHandler}

Testplan

Project:	**Input detectie**
Date:	27-05-2014
Owner:	Bert Willems
Client:	Liones
Version:	v1.0
Project members:	Michael de Vreugd

Inhoudsopgave

Inleiding 143
1. Onderdelen die getest worden. 144
2. Onderdelen die niet getest worden.....144
3. Aanpak.....144
4. Testproducten 144
5. Testgevallen.....145
6. Exacte criteria 145
7. Randvoorwaarden.....145
8. Verantwoordelijkheden.....145
9. Personeel en opleidingsbehoefte 146
10. Planning.....146
11. Risico’s 146
12. Goedkeuringen 146

Inleiding

Dit rapport is geschreven met als doelstelling om inzicht te bieden in hoe de “HandleIt” library getest wordt. Dit rapport is opgesteld volgens de IEEE 829-2008 structuur, ook bekend als de "829 Standard for Software Test Documentation".

1. Onderdelen die getest worden.

- Afhandeling input bij het gebruik van Engels en Japans en een westers toetsenbord op de volgende browser en operating system combinaties:
    - Windows / Chrome
    - Windows / Firefox
    - Mac OS / Chrome
    - Mac OS / Firefox
    - Android / Chrome
    - Android / Android browser
    - iOS / Chrome
    - iOS / Safari

2. Onderdelen die niet getest worden.

Vanuit het perspectief van de gebruiker zal alles worden getest.

3. Aanpak

Voor het testen van de library worden verschillende aanpakken gebruikt. Deze aanpakken dienen verschillende doeleindes.

Aanpak	**Doel**	**Benodigdheden**	**Toegepast op**
Acceptatie tests	Toetsen of de library werkt zoals verwacht gebaseerd op de requirements.	Een zelf gebouwde interface waarin de gebruiker gevraagd wordt specifieke invoer te leveren en dit dan door de library correct doorgegeven moet worden. De source code	De onderdelen zoals aangegeven bij hoofdstuk 1.
Unit testen	Toetsen of de individuele onderdelen binnen de library werken naar behoren.	Een combinatie van de volgende Javascript frameworks en libraries: - Testem - Mocha - Chai - Sinon-chai - RequireJS	De losse onderdelen van de library die als een unit beschouwd kunnen worden.

4. Testproducten

- Test plan

- Test script
- Test rapport

5. Testgevallen

Acceptatie tests:

- De library moet event handling bevatten voor standaard input(westers toetsenbord), IME's, compositions en auto-completes.
- De library moet event handling bevatten voor het gebruik van Chrome op iOS en Android.
- De library moet bij het afvangen van input alles bewaren en ook in de correcte volgorde.
- De library moet robuust en uitbreidbaar zijn.
- Hot-key handling moet vrij blijven, speciale toetsen zoals backspace vallen ook onder hot-key handling.
- De library moet als output een event stream aanbieden, deze stream mag de DOM niet aanpassen.
- De library moet alleen actief zijn wanneer nodig, wanneer dit niet het geval is moeten er geen onnodige controles uitgevoerd worden.
- De library moet een manier van monitoring aanbieden, hierin moet duidelijk zijn welke strategie er toegepast wordt en waarom.

Unit tests:

- Wordt het Event object op de juiste manier gedecorate?
- Wordt Strategy correct gevuld met alle benodigde Event handlers?
- Kan Observer correct omgaan met het toevoegen, pauzeren, hervatten en verwijderen van functies?
- Kan Observer correct ouput leveren door middel van een zelf aangeleverde callback?
- Kan de library correct het gebruikte operating system en browser detecteren?
- Kan de library gebaseerd op deze detectie de Strategy correct vullen?

6. Exacte criteria

- Voor de acceptatie tests is het resultaat positief wanneer er aan alle aangegeven requirements voldaan wordt.
- Voor de unit tests is het resultaat positief wanneer alle unit tests slagen.

7. Randvoorwaarden

Voor het kunnen uitvoeren van de testen is het volgende nodig:

- Een computer/laptop met Windows
- Een computer/laptop met Mac OS
- Een iPad of iPhone
- Een tablet/smartphone met Android

8. Verantwoordelijkheden

Acceptatie tests dienen uitgevoerd te worden door de product owner.

Unit tests moeten uitgevoerd worden wanneer er aanpassingen gemaakt worden die leiden tot een nieuwe release. Deze tests worden dan uitgevoerd door de developer verantwoordelijk voor de betreffende release.
In het geval van een pull request via Git is degene die deze request goed keurt verantwoordelijk voor het uitvoeren van de unit tests.

9. Personeel en opleidingsbehoefte

Basis kennis van het gebruik van de benodigde frameworks en libraries zoals aangegeven bij hoofdstuk 3 zijn een vereiste voor het uitvoeren van de unit tests.

10. Planning

Alle tests worden minimaal uitgevoerd bij het opleveren van een nieuwe release van de library. In het geval van de unit tests worden deze ook uitgevoerd bij elke grote verandering in de library als regressie tests.

11. Risico's

- Beschikbaarheid van benodigde apparatuur is te laag.
    - Aanvraag doen voor een vergroting van het aanbod van deze apparatuur, of het beter coördineren van de beschikbaarheid hiervan.

12. Goedkeuringen

De product owner is in samenwerking met de developer(s) verantwoordelijk voor het goedkeuren van de resultaten en het uitbrengen van de release.

Testrapport

Project:	**Input detectie**
Date:	27-05-2014
Owner:	Bert Willems
Client:	Liones
Version:	v0.1
Project members:	Michael de Vreugd

Inleiding

Dit document rapporteert over de resultaten van de in het testplan gespecificeerde tests voor het Input Detectie project. Dit document is gebaseerd op de TMap Next template die beschikbaar is via http://www.tmap.net. Hierbij zijn elementen die niet van toepassing zijn op dit moment weg gelaten uit het document.

Dit is een groei document en zal bijgehouden worden gedurende de ontwikkeling van de HandleIt library.

Version informatie

Versie	Datum	Opmerkingen	Auteur
v0.1	27-05-2014	Eerste versie	Michael de Vreugd

Distributielijst

Naam	Bedrijf/Functie
Bert Willems	Liones / Product Architect

Inhoudsopgave

Inleiding 148
1 Management Summary 151
1.1 Project assignment and development 151
1.2 Relation with other projects and documentation 151
1.3 Status table 151
1.4 Release advice 151
1.5 Transfer 151
1.6 Recommendations 151
1.7 Releasement of test team 151
2 Evaluation of the test object 152
2.1 Defects 152
2.1.1 Status per severity 152
2.2 Status of test goals 152
3 Product risks and strategy redress 153
3.1 Discrepancy and redress of test plan 153
3.2 Consequences for the test process 153
3.2.1 General 153
3.2.2 Test goals 153
4 Release advice 154
4.1 Advice 154
4.2 Results 154
4.3 Risk analysis 155
5 Transfer 156
6 Recommendations for future tests 156

1 Management Summary

1.1 Project assignment and development

Dit document betreft de resultaten en omliggende factoren betreffende het testen van de HandleIt library. Deze library is ontwikkeld om als module opgenomen te worden in het FontoXML project.

1.2 Relation with other projects and documentation

Het HandleIt project heeft een los verband met het FontoXML project. Dit houdt in dat het ontwikkeld is voor gebruik in FontoXML, maar een op zichzelf staand element is. Het document waarin het plan voor de tests omschreven is, is beschikbaar in de Input Detectie map op de Liones Google Drive.

1.3 Status table

Component	Status	Remarks
Quality of the test object		
Product Risks		
Evaluation Process		

Status: Ver van af, In ontwikkeling, Klaar

1.4 Release advice

Nog niet van toepassing

1.5 Transfer

<< Summary of section [6] >>

1.6 Recommendations

<< Summary of section [7] >>

1.7 Releasement of test team

Nog niet van toepassing

2 Evaluation of the test object

2.1 Defects

2.1.1 Status per severity

	Open	Closed	Total period	Total Cumul.
Obstructing	**0**	**0**	**0**	**0**
Severe	**0**	**0**	**0**	**0**
Disruptive	**0**	**0**	**0**	**0**
Cosmetic	**0**	**0**	**0**	**0**
Total	0	0	0	0

2.2 Status of test goals

Test goal	Status / explanation
Gebruikers requirements ten opzichte van FontoXML zijn voldaan.	In progress
Alle interne units werken naar behoren.	In progress

3 Product risks and strategy redress

3.1 Discrepancy and redress of test plan

3.2 Consequences for the test process

3.2.1 General

3.2.2 Test goals

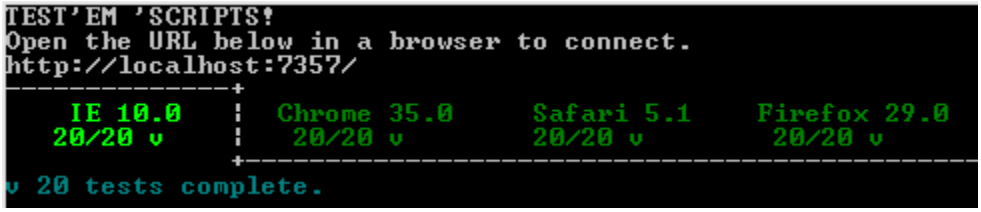
4 Release advice

Er zal een release advies gegeven worden zodra er een release candidate is. Op dit moment verkeert het project in een zogeheten “proof of concept” fase.

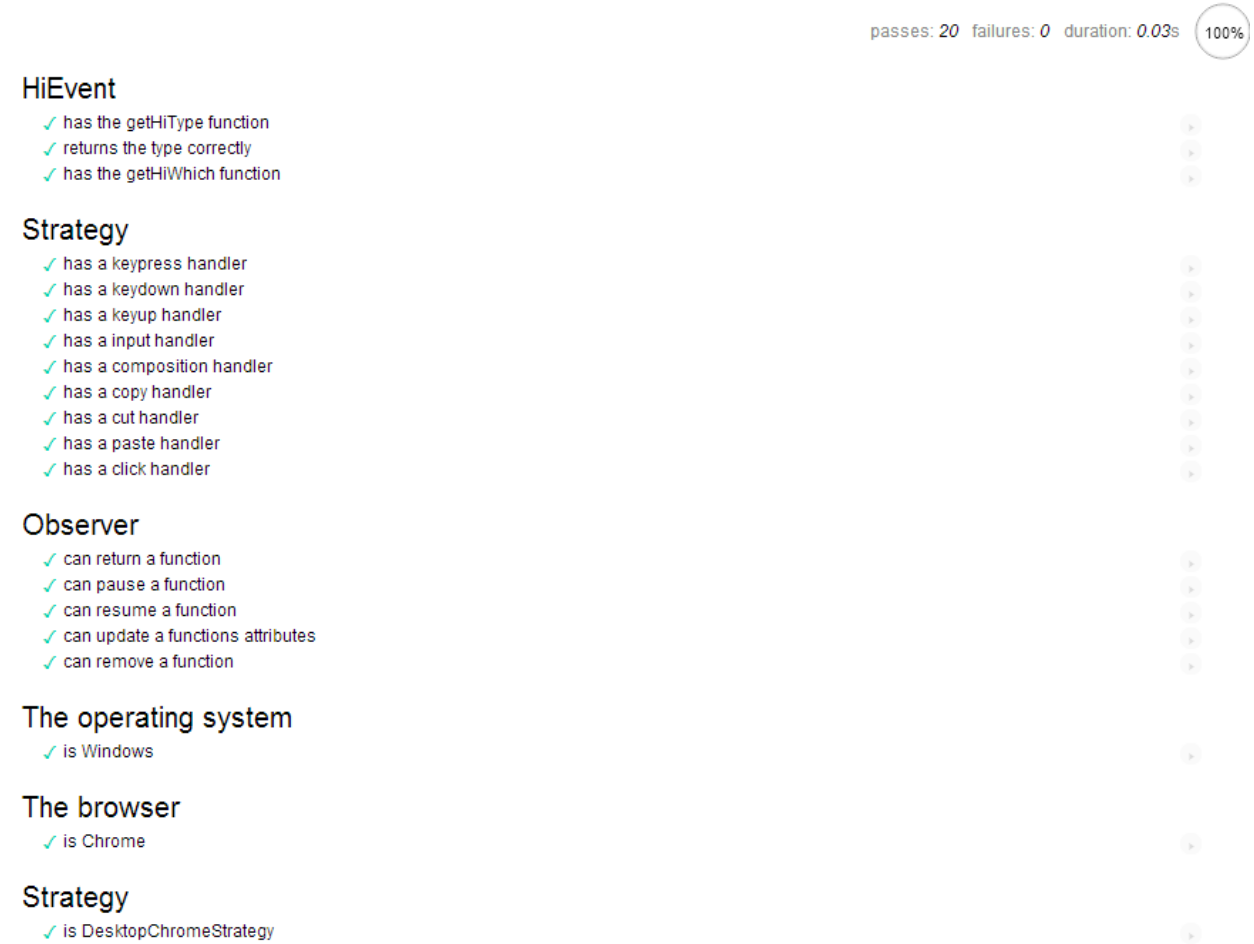
4.1 Advice

4.2 Results

Resultaten Unit tests zoals terug te vinden in de testem.js interface:



De resultaten zoals ze in de browser(Chrome in dit geval) interface te vinden zijn:



De acceptatie tests zijn uitgevoerd door middel van het uitvoeren van demo's voor Stef Busking en Bert Willems waarin aangetoond wordt dat de functionele requirements aanwezig zijn, naarmate er naar een release candidate gewerkt wordt zal dit op een meer officieuze manier gebeuren.
De acceptatie tests voor inhoudelijke zaken van de library zijn uitgevoerd door middel van code reviews.

4.3 Risk analysis

5 Transfer

Dit project zal als een open source library beschikbaar worden gesteld via GitHub. Voor het opnemen in FontoXML zullen hier eventuele aanpassingen aangebracht worden in samenwerking met de overige developers van FontoXML.

6 Recommendations for future tests

In dien het in de toekomst mogelijk wordt om correct en betrouwbaar gebruikers input te simuleren op de verschillende devices waarvoor HandleIt ondersteuning biedt. Dan dient deze methode toegepast te worden in de tests.

Bespreking concept	**Tussentijds assessment**	Eerste beoordeling

Formulier tussentijds assessment

Student: Michael de Vreugd **Studentnummer:** 08090769

Datum: 15 mei 2014 **eerste / tweede TTA:** eerste

Tijdens het tussentijds assessment is het volgende geconstateerd:		**ja**	**nee**
a	*Het voortgangsverslag is ontvangen*	X	
b	*Het afstudeerdossier is digitaal beschikbaar*		X
c	*Het afstudeerdossier is opgebouwd conform de richtlijnen*	X	
d	*Het goedgekeurde afstudeerplan is aanwezig*	X	
e	*Het plan van aanpak is aanwezig*	X	
f	*Reeds geleverd commentaar is aanwezig*	X	
g	*Het afstudeerdossier geeft voldoende inzicht in de stand van zaken*		X
h	*De afstudeeropdracht is tot nu toe naar behoren uitgevoerd*	X	X

Aanpak	**O**	**T**	**V**	**G**
Passend			X	
Theoretisch verantwoord		X		
Samenhang uitvoering beroepstaken			X	

Beroepstaken op afgesproken niveau uitgevoerd?		**O**	**T**	**V**	**G**
1	1.4 Uitvoeren analyse door definitie van requirements niveau 3		X		
2	3.2 Ontwerpen systeemdeel niveau 3	X			
3	3.3 Bouwen applicatie niveau 4		X		

4	3.5 Uitvoeren van en rapporteren over het testproces niveau 3	X			
5					
6					

Producten	**O**	**T**	**V**	**G**
Tussenproducten	onbeoordeelbaar			
Eindproducten		X		

Effectief communiceren	**O**	**T**	**V**	**G**
Binnen afstudeerbedrijf	ontbreekt			
Afstudeerdossier	ontbreekt			

Reflectie	**O**	**T**	**V**	**G**
Inzicht in eigen functioneren	ontbreekt			
Inzicht in eigen leerproces	ontbreekt			

<u>Toelichting per beoordelingscriterium</u>

Aanpak
De gevolgde aanpak op basis van eigen onderzoek en een Proof of Concept is op zichzelf goed. De gevolgde ontwikkelmethodiek wordt ten onrechte als Scrum aangeduid en mist de juiste theoretische onderbouwing.

Beroepstaken op afgesproken niveau uitgevoerd?
Ontwerp is alleen in de vorm van tekst aanwezig. De beroepstaak testen is totaal niet te beoordelen omdat in dit stadium van het verslag het hele hoofdstuk ontbreekt. Datzelfde geldt voor de eigen evaluatie van de beroepstaken.

Producten
De tussen- en eindproducten zijn onvoldoende beschreven om goed te kunnen worden beoordeeld.

Effectief communiceren
Het afstudeerverslag is op dit moment te onvolledig: het testen ontbreekt en de bijbehorende conclusies en evaluaties ook

Reflectie
Het hoofdstuk Reflectie is nog niet ingevuld.

Advies

X	Inleveren **(bindend advies)**
	Verlengen **(vrijblijvend advies)**
	Stoppen **(vrijblijvend advies)**

Besluit student

Aankruisen welke beslissing de student heeft genomen (alleen na vrijblijvend advies)

	Afstudeerdossier wordt op afgesproken datum ingeleverd	Inleverdatum:
	Afstudeerperiode wordt verlengd	Inleverdatum:
	Student stopt met afstudeeropdracht	

Naam begeleidend examinator: E.M. van Doorn

Naam tweede examinator: G.M. Tuk

Datum: 15 mei 2014

Dit formulier wordt door de tweede examinator digitaal ingevuld, waarna de begeleidend examinator het per email verstuurt naar de student met een cc naar de coördinator van ICT & Media @ Work (A.M.Schipper@hhs.nl). Het formulier dient door de student te worden opgenomen in het afstudeerdossier.

DE HAAGSE HOGESCHOOL

Academie voor ICT & Media

Den Haag
Delft
Zoetermeer

Evaluatieformulier afstuderen

In te vullen door opdrachtgever c.q. bedrijfsmentor(en)

Student:	Michael de Vreugd
Periode:	3
Bedrijf c.q. instelling:	Liones
Bedrijfsmentor:	Stef Busking
Plaats:	Rijswijk
Datum:	2 juni ’14

1. Heeft de student zich zelf snel en goed ingewerkt in het bedrijf en de uit te voeren afstudeeropdracht?

Ja

2. Hoe beoordeelt u de communicatieve vaardigheden van de student (in de samenwerking met collega’s, in contacten met de opdrachtgever, bij mondelinge presentaties, schriftelijke rapportages)?

Prima. Heldere presentaties en rapportages, duidelijke en precieze vragen op technisch vlak.

3. Hoe heeft de student tijdens het uitvoeren van de opdracht gefunctioneerd?

- **Qua verantwoordelijkheid** goed / ~~voldoende~~ / ~~matig~~ / ~~onvoldoende~~
- **Qua zelfstandigheid** goed / ~~voldoende~~ / ~~matig~~ / ~~onvoldoende~~
- **Qua planmatig werken** goed / ~~voldoende~~ / ~~matig~~ / ~~onvoldoende~~
- **Qua creativiteit** goed / ~~voldoende~~ / ~~matig~~ / ~~onvoldoende~~
- **Qua productiviteit** goed / ~~voldoende~~ / ~~matig~~ / ~~onvoldoende~~
- **Qua samenwerken met collega's** goed / ~~voldoende~~ / ~~matig~~ / ~~onvoldoende~~
- **Qua draagvlakontwikkeling** goed / ~~voldoende~~ / ~~matig~~ / ~~onvoldoende~~
- **Qua inspelen op bedrijfscultuur** goed / ~~voldoende~~ / ~~matig~~ / ~~onvoldoende~~
- **Qua rekening houden met de specifieke context van het bedrijf** goed / ~~voldoende~~ / ~~matig~~ / ~~onvoldoende~~
- **Qua het op gang brengen van de nodige veranderingen** goed / ~~voldoende~~ / ~~matig~~ / ~~onvoldoende~~

4. Hoe beoordeelt u de kennis en kunde van de student in verhouding tot wat u verwacht van een bijna afgestudeerde?

Prima

5. Hoe beoordeelt u de kwaliteit van de opgeleverde (tussen)producten?

Prima, steeds duidelijke voortgang te zien.

6. Bent u tevreden over het opgeleverde (eind)product?

- **In hoeverre heeft u gekregen wat is afgesproken?**

Geheel volgens afspraak. Michael heeft niet alleen een belangrijke bijdrage geleverd aan zowel onze kennis over invoerverwerking over een groot scala aan platforms, browsers en invoermethodes, maar ook een prima basis neergezet qua implementatie waarmee deze technieken op termijn in ons product geïntegreerd kunnen worden.

- **In hoeverre voldoet het (eind)product aan uw verwachtingen?**

Uitstekend. Er zijn zelfs oplossingen gevonden voor situaties die in onze eerdere ervaringen erg moeilijk bleken te onderscheppen, zoals ondersteuning voor autocorrect op iOS.

- **Wat is de bruikbaarheid en onderhoudbaarheid hiervan?**

Hoewel het doel van de opdracht meer een prototype was dan een voor productie geschikt eindproduct is het Michael toch gelukt om een robuuste basis neer te zetten. Ik denk dat de ontwikkelde code daarom prima om te zetten is naar een vorm die direct in FontoXML geïntegreerd kan worden.

- **Wat gebeurt er met het opgeleverde (eind)product?**

De ontwikkelde code zal de basis vormen voor de volgende iteratie van de bibliotheek voor invoerafhandeling in FontoXML. Vergeleken met de huidige implementatie biedt dit ons een uitstekende basis voor het omgaan met IME's en andere niet-westerse invoer, wat eerder niet mogelijk was. Bovendien is zowel de code als het onderzoek uitermate bruikbaar wanneer wij in de toekomst FontoXML geschikt willen maken voor het gebruik op mobiele apparaten, waar het correct afvangen van invoer tot voor dit onderzoek een van de belangrijkste blokkerende punten was.

- **Kunt u direct met het opgeleverde product aan de slag?**

Ja. De code en documentatie zijn van goede kwaliteit en hierdoor direct bruikbaar.

7. Zijn er nog aspecten voor u van belang die nog niet aan de orde zijn geweest?

Naast de kwaliteit van de opgeleverde producten heeft Michael vooral ook laten zien dat hij erg snel en zelfstanding met nieuwe materie aan de slag kan, maar ook snel en goed geformuleerde vragen kan stellen aan collega's.

8. Bent u bereid een volgende keer weer uw medewerking te verlenen aan het beschikbaar stellen van een afstudeerplaats (graag met toelichting)**?**

Ja. Binnen de ontwikkeling van FontoXML en de familie aan verwante producten komen continu interessante vraagstukken naar boven. De aard van FontoXML levert tegelijk zowel diepgaande technische uitdagingen als interessante onderzoeksvragen op het vlak van gebruikersinteractie. Onze ervaring heeft laten zien dat hier leuke puzzels tussen zitten waar studenten vaak een uitstekende bijdrage voor kunnen leveren, terwijl ze tegelijk kennis kunnen maken met het werken in een bedrijf als Liones.