Vrije Universiteit Brussel

FACULTEIT WETENSCHAPPEN EN BIO-INGENIEURSWETENSCHAPPEN
VAKGROEP COMPUTERWETENSCHAPPEN

Het flower-menu als alternatief voor een klassiek hiërarchisch menu

Masterproef ingediend in gedeeltelijke vervulling van de eisen voor het behalen van de graad Master of Science in de toegepaste informatica

Stijn Dejongh

Promotor: Prof. Dr. Olga De Troyer

Academiejaar 2014-2015

Samenvatting

Deze thesis gaat verder op een stage uitgevoerd bij de Belgische Geïntegreerde Politie. Een gespecialiseerde politiedienst (DRI) ervaart problemen met de interface van het bestaande softwareprogramma *PaCoS*. Men zoekt naar alternatieven op korte termijn. Op basis van een probleemanalyse en de aangeboden suggesties die te realiseren zijn binnen een beperkt tijdsbestek, zoeken we in deze scriptie verder een antwoord op de vraag:
"Hoe kunnen we zowel nieuwe gebruikers als ervaren gebruikers een object laten selecteren uit een uitgebreide bestaande hiërarchie van objecten op een goed bruikbare wijze?"

We overlopen de mogelijke alternatieven aan de hand van een literatuurstudie. Uit deze mogelijkheden en rekening houdende met enkele voorwaarden, die voortkomen uit het vooronderzoek bij de politiedienst, kiezen we voor de oplossing van het "flower-menu" en implementeren we dat. Onze toepassing maakt voornamelijk gebruik van een vast toestel, hoewel er rekening wordt gehouden met de mogelijkheid tot uitbreiding naar mobiele toestellen. We evalueren onze oplossing door het uitvoeren van een gebruikerstest met tien personen. Uit dit onderzoek blijkt dat onze oplossing sneller presteert dan het referentiemenu, wanneer het gebruikt werd in de expert-mode en evenwaardige snelheden haalt bij gebruik in novice-mode. Een gebruikersbevraging toont wel aan dat de gebruikers een voorkeur hebben voor het klassieke menu. We besluiten dat het flower-menu een waardig alternatief is voor het klassiek hiërarchisch menu en dat de voorkeur van de gebruikers vooral veroorzaakt wordt door hun uitgebreide ervaring met het klassieke menu.

Gezien het kleine aantal respondenten (10) in de testfase, suggereren we een ruimer onderzoek om onze bevindingen te controleren. De problemen gedetecteerd bij de herkenning van bepaalde markeringen vragen ook verdere uitwerking. We stellen ook voor om onze oplossing op mobiele toestellen te implementeren, aangezien de vraag hiernaar van de gebruikersgroep vrij groot is. Uitvoerigere testen met dezelfde testpersonen kunnen bovendien interessante bevindingen opleveren met betrekking tot het leerproces en de evolutie van hun voorkeur.

Sleutelwoorden:

hiërarchische menu's, marking-menu's, flower-menu's, *usability*, interfaceontwerp.

Declaration of Originality

I hereby declare that this thesis was entirely my own work and that any additional sources of information have been duly cited. I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this thesis has not been submitted for a higher degree to any other University or Institution.

Voorwoord

Met het schrijven van dit voorwoord hoop ik de laatste hand te leggen aan mijn thesis en per extensie ook aan mijn studententijd. Het is een bewogen periode geweest, maar ik mag wel stellen dat de voorbije jaren de mooiste van mijn leven waren.

Ik wil in het bijzonder professor De Troyer bedanken voor haar begeleiding bij het schrijven van deze thesis. Ze was steeds bereid om me met woord en daad bij te staan. Zonder haar zou het niet mogelijk geweest zijn deze masterproef tot een goed einde te brengen.

Mijn dank gaat ook uit naar de Vrije Universiteit Brussel en in het bijzonder naar de faculteit Wetenschappen, omdat ze mij de kans hebben gegeven om mijn studies af te ronden, ondanks de problemen die ik in mijn vorige studie was tegengekomen.

Ik apprecieer ook heel erg het professionalisme van het team van de Federale Politie waar ik mijn stage kon volbrengen en waardoor ik inspiratie kon opdoen voor mijn onderzoek.

De proefpersonen bedank ik voor hun geduld en voor hun bereidheid om de geregistreerde testgegevens te laten exploiteren.

Mijn ouders, broer en vriendin wil ik graag bedanken voor hun morele steun en wijze raad. Het is aangenaam om te weten dat ik steeds op jullie kan rekenen. Ook mijn vrienden en vriendinnen verdienen dank. Hierbij mijn excuses voor de ellenlange tirades over mijn werk. Het begon jullie vast te vervelen. Ik wil jullie ook bedanken voor de nodige afleiding. De boog kan immers niet steeds gespannen staan.

Mijn broer Sander wil ik bedanken voor de hulp met de statistische verwerking van de data. Ik wil ook alle *proofreaders* bedanken. Zonder jullie stond mijn thesis waarschijnlijk nog steeds vol met storende fouten.

Hartelijk bedankt allemaal!

Stijn Dejongh

Etterbeek, 11 augustus 2015

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

1

Inleiding en probleemstelling

Het onderwerp van het onderzoek uitgevoerd in het kader van deze masterproef is tot stand gekomen na de uitvoering van een stage bij de dienst DRI (Directorate of information) van de Belgische Geïntegreerde Politie, waarbij we een *usability*-analyse maakten van de *PaCos*-software. *PaCos* is een web-gebaseerde software die inbeslagnemingen beheert en opvolgt. De software moet gebruikt kunnen worden voor het aanmaken, beheren en opzoeken van inbeslagnemingen, alsook het opvolgen van de locaties van de in beslag genomen goederen. Zo houdt het systeem de huidige locatie en transfertgeschiedenis van elk in beslag genomen voorwerp nauwkeurig bij. We voerden een analyse uit van de bruikbaarheid ("*usability*") van de software en maakten een verslag van de opgemerkte problemen, waarbij steeds een advies ter verbetering werd meegegeven.

Het onderzoek in deze masterproef gaat verder in op één van de belangrijkste tekortkomingen, nl. het maken van selecties binnen het voorwerpstypemenu. Gebruikers moeten de in beslag genomen voorwerpen aan de corresponderende inbeslagneming kunnen toevoegen. Het spreekt voor zich dat er een enorme verscheidenheid aan voorwerpen bestaat. Dit heeft als gevolg dat er specifieke formulieren bestaan voor bepaalde voorwerpscategorieën. De gebruiker moet dus eerst een voorwerpstype kiezen, alvorens hij naar een gespecialiseerd formulier geleid wordt. Dit gebeurt nu door middel van een

hiërarchisch menu. Dit menu is te zien in afbeelding 1.1. Voorwerpen zijn onderverdeeld in een aantal door de politie vooraf vastgestelde categorieën en subcategorieën die samen een hiërarchie vormen. Voor het selecteren van minder gebruikte voorwerpstypes moet gebruikers navigeren via de menu-optie "*ander*".

Figuur 1.1: Het huidige menu voor de selectie van een voorwerpstype binnen *PaCoS*.

Uit gesprekken met de mensen van de Federale Politie en uit de uitgevoerde analyse bleek dat er een probleem is met de bruikbaarheid van het menu. Het menu is erg groot, maar bevat toch meerdere categorieën genaamd "*Ander*". Deze zijn niet transparant voor de gebruiker, aangezien hij geen manier heeft om te weten welke voorwerpstypen er onder deze "*ander*"-categorieën schuilen. Daarnaast is het menu ongeschikt voor een mobiele versie van de applicatie die men in de toekomst beoogt.

In dit werkstuk gaan we op zoek naar een oplossing voor het bovengenoemde *usability*-probleem, m.a.w. we willen een aantoonbaar betere (bruikbaardere) manier om een selectie te maken uit een uitgebreide hiërarchie. Hierbij beogen we een betere *usability* voor zowel *expert users*, als *novice users*. Onze aangereikte oplossing is rechtstreeks van toepassing op de *PaCoS*-software, maar omdat dit document ook een overzicht biedt van bestaande soorten hiërarchische menu's en hun eigenschappen, kan ons werkstuk ook gebruikt worden als leidraad bij het kiezen van een gepast menu in gelijkaardige situaties.

In de volgende secties overlopen we onze onderzoeksvraag, randvoorwaarden voor de oplossing, de onderzoeksmethode en geven we een korte beschrijving van de structuur van de masterproef.

1.1 Onderzoeksvraag

Zoals hierboven vermeld, zoeken we een alternatief voor het bestaande voorwerpstype-selectiemenu binnen de *PaCoS*-software. We zoeken een oplossing die enerzijds afgestemd is op nieuwe gebruikers (*novice users*) en anderzijds geoptimaliseerd kan worden voor ervaren gebruikers (*expert users*). Dit is een zogenoemd praktisch probleem (Wieringa, 2009). Om een praktisch probleem op te lossen moeten we vaak één of meerdere kennisvragen oplossen (Wieringa, 2009). In ons geval is de belangrijkste kennisvraag die beantwoord moet worden de volgende:

"Hoe kunnen we zowel nieuwe gebruikers als ervaren gebruikers een object laten selecteren uit een uitgebreide bestaande hiërarchie van objecten op een goed bruikbare wijze?"

Met een goed bruikbare wijze, refereren we naar het begrip *"usability"* gedefinieerd in HCI.

Definitie 1 ***Usability*** *is een* quality attribute *dat inschat hoe makkelijk te gebruiken gebruikersinterfaces zijn. Het woord "*usability*" refereert ook naar methoden voor het verbeteren van het gebruiksgemak tijdens het ontwikkelingsproces. Usability wordt gekenmerkt door vijf kwaliteitscomponenten:*

- Leercurve*: Hoe eenvoudig is het voor gebruikers om standaardtaken uit de voeren wanneer ze de eerste keer met het ontwerp in aanraking komen?*
- Efficiëntie*: Hoe snel kunnen gebruikers hun taken uitvoeren zodra ze bekend zijn met het ontwerp?*
- Onthoudbaarheid*: Hoe makkelijk worden gebruikers opnieuw vertrouwd met het ontwerp nadat ze dit een tijd lang niet gebruikt hebben?*

> - Fouten*: Hoe veel fouten maken de gebruikers, hoe ernstig zijn deze fouten en hoe makkelijk kunnen gebruikers de fouten verbeteren?*
> - Tevredenheid*: Hoe aangenaam is het om met het ontwerp te werken?*
>
> *(Nielsen, 2003)*

Aangezien de Belgische Geïntegreerde Politie van plan is om de *PaCoS*-software ook op mobiele toestellen aan te bieden, houden we hier in ons onderzoek eveneens rekening mee. We houden voor ogen dat onze oplossing ook bruikbaar moet zijn op toestellen met een beperkte schermoppervlakte.

1.2 Randvoorwaarden voor de oplossing

In eerste instantie willen we een oplossing die bruikbaar is binnen de *PaCoS*-applicatie. Daarom moeten we rekening houden met enkele beperkingen en vereisten: de huidige informaticastructuur, de gebruikers en enkele andere specifieke beperkingen. We bespreken deze in de volgende secties.

1.2.1 Systeemvereisten

De voorgestelde oplossing moet integreerbaar zijn in de bestaande applicatie. Dit wil zeggen dat we rekening moeten houden met het feit dat we ontwikkelen voor een web-gebaseerde toepassing. Bij voorkeur doen we geen of weinig beroep op *server-side*-applicaties die momenteel nog niet aangeboden worden, zodat er geen aanpassingen hoeven te gebeuren aan de onderliggende informatica-infrastructuur.

Men wil het bestaande systeem ook uitbrengen voor mobiele toestellen. Het is echter nog niet helemaal duidelijk of de bestaande interface gebruikt zal worden (via de webbrowser van het mobiele toestel) of dat er een *stand alone*-applicatie ontwikkeld zal worden. We moeten er dus alvast rekening mee houden dat de interface mogelijk op kleinere schermen weergegeven zal moeten worden. Efficiënt omspringen met de beschikbare schermruimte is dus zeker aan te raden. Bij de geplande uitbreiding naar mobiele platformen zal het dus waarschijnlijk nodig zijn om meerdere browsers en platformen te

ondersteunen. We houden onze voorgestelde oplossing liefst zo compact en duidelijk mogelijk.

1.2.2 Gebruikers

Om tot een geschikte oplossing te komen is het belangrijk om rekening te houden met de verschillende gebruikers die met de oplossing zullen moeten werken. Uit de eerder uitgevoerde analyse (zie bijlage) halen we drie gebruikersklassen, namelijk: *de federale politieagent*, *de lokale politieagent*, en *de externe partner*. Elk van deze gebruikertypes heeft zijn eigen karakteristieke eigenschappen, die weergegeven worden in de tabellen 1.1, 1.2 en 1.3.

In user-interface-design is het algemeen bekend dat men bij het zoeken naar een oplossing voor een user-interface-probleem ook rekening moet houden met de vertrouwdheid die de gebruikers hebben met hun taak en het systeem (Stone, Jarrett, Woodroffe & Minocha, 2005). Dit wil zeggen dat we de gebruikers kunnen verdelen in twee groepen. Enerzijds hebben we gebruikers die goed bekend zijn met de taak en de verschillende objecttypes die door het systeem aangeboden worden (hier voorwerpstypen); dit zijn de zogenaamde *expert users*. Anderzijds hebben we de gebruikers die minder bekend zijn met de opties van het systeem en meer begeleiding nodig hebben bij het selecteren van het juiste objecttype. Deze groep noemen we *novice users*.

Expert users zoeken vooral naar efficiëntie in het gebruik van de software. Zij weten precies wat ze willen bekomen en willen hun doel zo snel mogelijk bereiken. De oplossing moet dus de mogelijkheid bieden om snel en efficiënt een objecttype te selecteren.

De *novice users* hebben meer begeleiding nodig. We denken bij deze categorie gebruikers vooral aan occasionele gebruikers die niet volledig bekend zijn met de classificatiehiërarchie die de politie gebruikt en dus meer moeten zoeken naar het type dat correspondeert met het voorwerp dat zij voorhanden hebben. Dit type gebruiker heeft meer sturing nodig.

We kunnen stellen dat de gebruikersklasse *"Federale politieagent"* overeenkomt met de groep *expert users*. De klassen *"Lokale politieagent"* en *"Externe partner"* corresponderen met de groep *novice users*.

Gebruikersklasse: **Federale politieagent**	
Type gebruiker:	Directe gebruiker
Bekendheid met systeem:	Geen
Bekendheid met taak:	Expert
Gebruiksfrequentie:	Dagelijks
Vrijwillig/verplicht gebruik:	Verplicht gebruik
Ervaring met computers:	Matig tot goed
Andere gebruikte systemen:	Bekend met andere software uitgebracht door de Federale Politie
Opleiding:	Hoger opgeleid
Motivatie:	Beroepshalve (verplicht gebruik)
Aantal gebruikers:	Onbekend
Training:	Beperkte introductie tot systeem

Tabel 1.1: Overzicht van de gebruikersklasse *Federale politieagent*

Gebruikersklasse: **Lokale politieagent**	
Type gebruiker:	Directe gebruiker
Bekendheid met systeem:	Geen
Bekendheid met taak:	Omvangrijk
Gebruiksfrequentie:	Dagelijks
Vrijwillig/verplicht gebruik:	Verplicht gebruik
Ervaring met computers:	Beperkt
Andere gebruikte systemen:	Onbekend
Opleiding:	Minstens middelbare school
Motivatie:	Beroepshalve (verplicht gebruik)
Aantal gebruikers:	Onbekend
Training:	Beperkt

Tabel 1.2: Overzicht van de gebruikersklasse *Lokale politieagent*

Gebruikersklasse: **Externe partner**	
Type gebruiker:	Directe gebruiker
Bekendheid met systeem:	Geen
Bekendheid met taak:	Matig
Gebruiksfrequentie:	Wekelijks
Vrijwillig/verplicht gebruik:	Verplicht gebruik
Ervaring met computers:	Onbekend
Andere gebruikte systemen:	Onbekend
Opleiding:	Minstens middelbare school
Motivatie:	Samenwerking met de politie in het kader van inbeslagnemingen door het verwerken, onderzoeken of opslaan van de in beslag genomen goederen.
Aantal gebruikers:	Onbekend
Training:	Beperkt

Tabel 1.3: Overzicht van de gebruikersklasse *Externe partner*

1.2.3 Bestaande hiërarchie

De bestaande hiërarchie voor het classificeren van voorwerpen moet gerespecteerd worden. Dit komt enerzijds doordat een deel van de gebruikte structuur bij wet bepaald is (de categoriën van het laagste niveau en een deel van de bovenliggende categorieën). Anderzijds worden de bestaande classificaties al jaren gebruikt door de Geïntegreerde Politie en zijn er al verscheidene interne statistische studies uitgevoerd die gebruik maken van deze structuur. Zo werd er bijvoorbeeld gekeken wat het aandeel van de categorie *Vuurwapens* is tegenover het totale aandeel inbeslagnemingen.

Het opstellen en gebruiken van een nieuwe classificatiestructuur zou inhouden dat men de oude cijfergegevens niet meer kan vergelijken met de nieuwe. Een andere optie is dat men de nieuwe structuur enkel voor de weergave op de *front-end* gebruikt en achter de schermen blijft werken met de oude hiërarchie. Dit zou tot gevolg hebben dat het personeel dat reeds bekend is met de oude classificatie deze kennis niet kan benutten bij het gebruik van de nieuwe classificatiestructuur. Erger nog: het omgooien van de bestaande structuur zou de leercurve van het gebruik van de nieuwe oplossing mogelijk nog steiler maken.
Geen van beide gevolgen zijn wenselijk en we nemen ons voor om bij het opstellen van de uiteindelijke oplossingen geen eigen voorwerpscategorieën aan te maken.

1.3 Onderzoekmethode

Onze onderzoekmethode is gebaseerd op de *Design Science*-onderzoeksmethodologie (Peffers, Tuunanen, Rothenberger & Chatterjee, 2007).
Deze onderzoeksmethode richt zich op de ontwikkeling van een artefact als oplossing voor een bepaald praktisch probleem, wat inderdaad ons doel is. Na het bestuderen van het probleem, volgt er in de methodologie dan ook een fase waarin een oplossing wordt voorgesteld die vervolgens wordt ontwikkeld en geëvalueerd, waarna conclusies worden geformuleerd. Dit proces is iteratief zodat de oplossing kan worden bijgestuurd indien de resultaten niet bevredigend zijn.

Om tot een oplossing te komen maken we eerst een overzicht van de mogelijke alternatieven. Aangezien de Belgische Geïntegreerde Politie reeds gekozen had voor een hiërarchisch menu voor het selecteren van voorwerpstypes, respecteren we hun keuze. We houden ons bij het zoeken naar een oplossing aan het gebruik van het concept *menu*. Binnen de oplossing van menu's, bekijken we eerst de bestaande alternatieven. Hierbij worden de positieve en negatieve aspecten van elk menutype besproken. Deze analyse wordt beschreven in hoofdstuk 2. Op basis hiervan selecteren we een menutype dat we verder uitwerken en aanpassen aan de doelstellingen. Deze oplossing wordt beschreven in hoofdstuk 3. Vervolgens implementeren we het voorgestelde menu. De implementatie is beschreven in hoofdstuk 4. Deze implementatie wordt daarna geëvalueerd via de vergelijking met een referentiemenu aan de hand van gebruikerstests. De evaluatie van de oplossing is beschreven in hoofdstuk 5. Ten slotte worden conclusies geformuleerd in hoofdstuk 6.

1.4 Structuur van de masterproef

Deze masterproef bestaat uit een aantal delen. In het eerste deel voeren we een literatuuronderzoek betreffende verschillende types van hiërarchische menu's. Deze studie is te vinden in hoofdstuk 2. In een volgend deel werken we een oplossing uit. De gemaakte overwegingen en het daaruit voorkomende model worden toegelicht in hoofdstuk 3. Het daarbij aansluitend implementatieverslag bevindt zich in hoofdstuk 4. Tot slot analyseren we deze oplossing in hoofdstuk 5 en formuleren we onze conclusies in hoofdstuk 6.

2

Menu's voor grote hiërarchieën

Wanneer we een grote hoeveelheid aan mogelijke opties hebben, kunnen we die op verschillende manieren weergeven. Ofwel presenteren we de gebruiker een hiërarchie waarbinnen hij zijn keuzes steeds verfijnt tot hij uiteindelijk zijn gezocht alternatief heeft gevonden, of we tonen onmiddellijk alle mogelijke keuzes. Dit laatste is schijnbaar onmogelijk te doen wanneer we werken met uitgebreide hiërarchieën waarbij er veel mogelijke keuzes zijn. De beschikbare schermgrootte is immers beperkt. Het probleem van het navigeren binnen een uitgebreid datanet wordt al langer onderzocht binnen het gebied van de *graphical visualisation*.

Aangezien we een oplossing zoeken voor een bestaande applicatie waarbinnen geopteerd werd voor een hiërarchisch menu, zullen we eerst onderzoeken of er een beter bruikbaar alternatief menutype bestaat voor ons doel. In dit hoofdstuk bespreken we dan ook, op basis van een literatuurstudie, bestaande menustijlen die de gebruiker toelaten om een keuze te maken uit een grote hoeveelheid opties. Daarna gaan we in op de onderliggende eigenschappen van verschillende menutypes. Tot slot geven we, in sectie 2.3, een overzicht van de besproken menutypes en eigenschappen.

2.1 Soorten menu's

We bespreken in de volgende secties verschillende menutypes die in de literatuur voorgesteld worden in de context van het selecteren van een optie uit een uitgebreide lijst van opties: *Fisheye-menu's*, *Split-menu's*, *Pie-menu's* en *Marking-menu's*. Deze menu's worden vaak vergeleken met het klassiek hiërarchisch menu.

Definitie 2 *Een* hiërarchisch menu *is een menu dat een boomstructuur volgt en bestaat uit verschillende niveaus. Per niveau heeft de gebruiker de keuze uit een aantal opties. Deze keuze leidt dan weer tot een set nieuwe keuzes in het volgende niveau. Zo komt de gebruiker geleidelijk tot een eindknoop, de uiteindelijke keuze. Een hiërarchisch menu heeft twee bepalende karakteristieken: zijn* ***diepte*** *en zijn* ***breedte****. De diepte van het menu is het aantal niveaus in de boomstructuur. De breedte is het aantal keuzes dat er per niveau wordt aangeboden.*

2.1.1 Fisheye-menu's

Spence bespreekt in 1982, in zijn paper over een geoptimaliseerde professionele werkomgeving, een techniek die een oplossing kan bieden voor het probleem van de beperkte schermgrootte (Spence & Apperley, 1982). Hij bespreekt het inzoomen op het actieve focuspunt met behoud van het overzicht op de globale werkomgeving. Later onderzocht men deze *fisheye view* als alternatief voor de visualisatie van allerhande data. Schaffer oppert een *fisheye view* als oplossing voor het navigeren binnen een geclusterd netwerk van knopen (Schaffer et al., 1996). Masui gebruikt de techniek binnen zijn *LensBar* systeem (Masui, 1998) voor het doorlopen van een lange lijst van data. Bederson (2000) past met zijn *fisheye-menu's* als eerste de *fisheye view* toe op de problematiek van het weergeven van vele keuzemogelijkheden binnen een menu.

Fisheye-menu's bieden de gebruiker alle mogelijke opties aan in een eenvoudige, lange lijst. Om deze lijst integraal te kunnen weergeven worden de keuzemogelijkheden echter erg verkleind weergegeven. Een fisheye-menu schaalt dynamisch de grootte van het lettertype van de menu-items, zodat de lijst het volledige scherm kan innemen. Enkel de elementen die zich rond de cursor bevinden, worden uitvergroot weergegeven. Het uitvergrote gebied

wordt de *context* genoemd. Met deze methode behoudt de gebruiker een overzicht van alle opties en kunnen die toch leesbaar gemaakt worden. De ontwerper van het menu kan de focusgrootte instellen en zo bepalen wat het verschil in weergave is tussen de uitvergrote en de kleine elementen (Bederson, 2000).

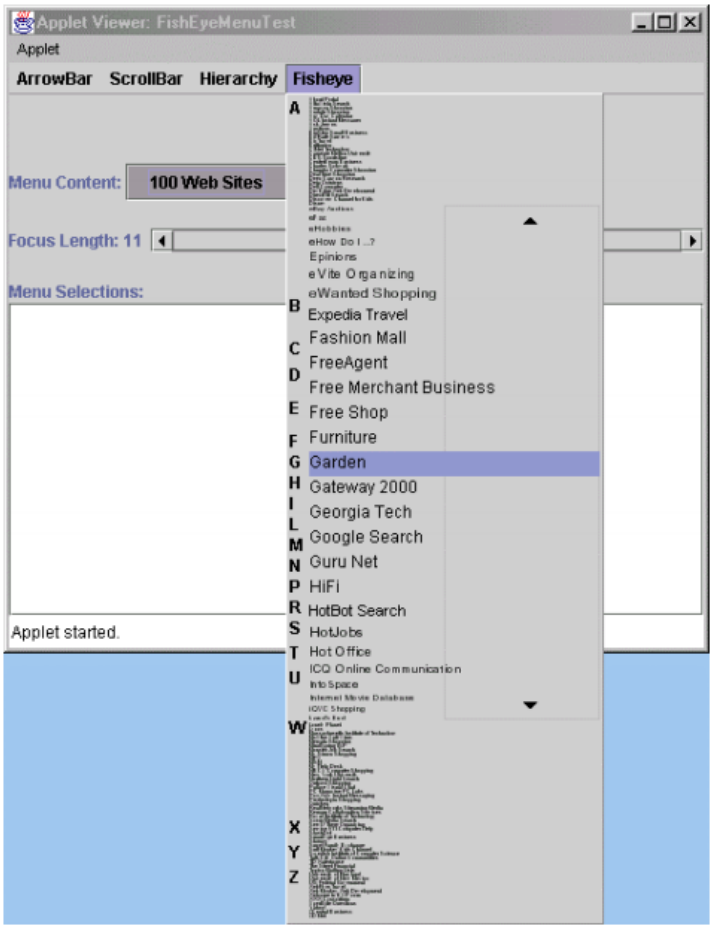
Bederson (2000) geeft een oplossing voor het voorkomen van onleesbare menu-items, wat vaak zal voorkomen door de gebruikte verkleining van het lettertype. De lijst van keuzemogelijkheden moet voorafgegaan worden door een alfabetische index, die niet mee geschaald wordt met de focus van het fisheye-menu. Op deze manier heeft de gebruiker altijd wel een idee over de plaats van het gezochte item. Een ander probleem is dat de gebruiker, wanneer hij een bepaald item uitvergroot heeft, de cursor slechts verticaal hoeft te bewegen over de afstand van de minimale lettergrootte om te focussen op het volgende item. Dit constant verspringen van context zou de navigatie erg moeilijk maken. Daarom hebben de onderzoekers voorzien in de mogelijkheid om een context vast te zetten door de cursor naar de rechterkant van het menu te bewegen, de zogenaamde *focus-lock mode*. Een voorbeeld van de werking van het fisheye-menu is te zien in afbeelding 2.1.

Bederson (2000) probeerde het nut van zijn menu te verifiëren aan de hand van gebruikerstesten met tien vrijwilligers. De vrijwilligers bestonden uit zowel programmeurs (*expert users*) als niet-programmeurs (*novice users*). De uitgevoerde test was echter volledig kwalitatief. Geen enkele kwantitatieve parameter werd bijgehouden omdat men de gebruikers enkel vroeg naar hun voorkeur uit vier menutypes:

- een **menu met pijliconen** onder- en bovenaan om op en neer te scrollen
- een **menu met scrollbars**
- een **hiërarchisch menu**
- het **fisheye-menu**

Het resultaat van de analyse was dat de gebruikers gemiddeld gezien hiërarchische menu's prefereerden boven fisheye-menu's. Wanneer de resultaten opgedeeld werden per gebruikerstype bleken de *expert users* het fisheye-menu lichtjes beter te bevinden dan het hiërarchisch menu. De *novice users* vonden het hiërarchisch menu dan weer beduidend beter. Uit de testen bleek ook dat de gebruikers, zonder enige uitleg, moeite hadden om de werking van het

fisheye-menu te begrijpen (Bederson, 2000). Deze resultaten zijn echter geheel kwalitatief en bieden dus geen informatie over efficiëntie en effectiviteit.



Figuur 2.1: De werking van Bederson's fisheye-menu (overgenomen uit (Bederson, 2000)).

Hornbæk werkte verder op Bederson's werk. Hij voerde een empirisch experiment uit met 12 testgebruikers om de *usability* van fisheye-menu's te kunnen bepalen (Hornbæk & Hertzum, 2007). De testgebruikers kregen vier verschillende menutypes aangeboden waarin ze een selectietaak moesten uitvoeren. Er werden data bijgehouden over de volgende aspecten: nauwkeurigheid, selectietijd, tevredenheid van de gebruikers en oogbewegingen. De gebruikers moesten zowel gekende als ongekende taken uitvoeren. De *gekende taken* waren taken waarbij de gebruikers wisten waar een element zich in het menu bevond. Bij *ongekende taken* moesten ze zelf op zoek naar de locatie

van het item. De gebruikers hadden allemaal reeds ervaring met computers en specifiek met hiërarchische menu's. Men probeerde in het experiment het effect van de verschillende ontwerpkeuzes die gemaakt kunnen worden bij het implementeren van een fisheye-menu te bepalen. Zo werkte men met vier menu's:

- het **fisheye-menu** zoals beschreven door Bederson (Bederson, 2000)
- een **overview-menu** dat een alfabetische index heeft en in zijn detailgebied de menu-items weergeeft die overeenkomen met de index waar de cursor actueel boven staat
- een **multi-focus-menu** dat werkt zoals het fisheye-menu, maar belangrijke items buiten het focusgebied ook in een groter lettertype weergeeft. Dit menu heeft geen alfabetische index naast de menu-items. Door elk eerste item van een nieuwe letter als belangrijk te markeren, is er wel een alfabetisch overzicht binnen het menu.
- een **hiërarchisch menu**

De experimenten toonden aan dat voor gekende taken de gebruikers het minste fouten maakten met het hiërarchisch menu. Voor niet-gekende taken waren er tussen de verschillende menu's geen significante verschillen in gemaakte fouten. Inzake selectietijd voor gekende taken presteerden de gebruikers significant beter met hiërarchische menu's: ze navigeerden 55% sneller dan met de andere menu's. Het gebruik van het fisheye-menu bleek snellere tijden op te leveren dan wanneer gebruikers met het overview-menu werkten. Voor ongekende taken waren er ook in selectietijd geen significante verschillen. Gemiddeld gezien hadden de gebruikers geen uitgesproken voorkeur voor één van de menutypes. Het hiërarchisch menu werd hoger gewaardeerd dan het fisheye-menu. De onderzoekers merkten tijdens de testen op dat de *focus-lock*-methode verwarrend overkwam voor de gebruikers. Naast deze experimenten bespreekt Hornbæk ook eventuele verbeteringen aan het fisheye-menu (Hornbæk & Hertzum, 2007), zoals het weergeven van meer informatie in de context. Hornbæk verwacht echter niet dat deze aanpassingen de problemen van het fisheye-menu zouden wegwerken (Hornbæk & Hertzum, 2007).

Bederson (2000) haalt in zijn paper aan dat menu's steeds vaker lange lijsten met opties worden, zeker als ze gebruikt worden voor het selecteren van data in plaats van het selecteren van een uit te voeren actie. Hij stelt dat hiërarchieën het grote nadeel hebben dat nieuwe gebruikers onwetend zijn over de plaats van het gewenste item in de algemene structuur. Hierdoor moeten

ze de hele lijst van opties afgaan om dit specifieke item te vinden. Deze stelling wordt echter niet verder onderbouwd. Verder bespreekt hij de mogelijkheid tot sneltoetsen via het toetsenbord die sommige menu's aanbieden, maar hij haalt aan dat die weinig of niet gebruikt worden. Fisheye-menu's zouden het meeste voordeel bieden aan gebruikers die uitsluitend met de muis werken (Bederson, 2000).

De *fisheye view* heeft een bepaald potentieel in het interface-ontwerp voor gebruik op kleinere schermen zoals smartphones en tablets (Gutwin & Fedak, 2004; Karlson, Bederson & SanGiovanni, 2005).

Uit het vergelijkende literatuuronderzoek kunnen we besluiten dat het grote voordeel van een fisheye-menu is dat het een vaste portie van het scherm inneemt en de inhoud dynamisch schaalt om hierbinnen te passen. Tegelijkertijd is dit ook het grootste nadeel van dit menutype. Daar waar Bederson (2000) beweert dat hiërarchische menu's onhandig zijn omdat de gebruiker veel tijd verliest door onwetendheid over de categorie waarbinnen de gewenste keuze zich bevindt, heeft een fisheye-menu het nadeel dat de exacte benaming van het item gekend moet zijn om het terug te vinden in de lijst. Een fisheye-menu zou wel een overzicht bieden van het volledige scala aan mogelijkheden, maar in de praktijk zullen de meeste items onleesbaar worden naarmate het aantal opties groter wordt. De gebruiker moet dan nog steeds de hele lijst doorlopen alvorens de gewenste keuze te kunnen maken. Daarnaast gaf de gebruikerstest van Bederson aan dat de meeste gebruikers hiërarchische menu's prefereren (Bederson, 2000). Fisheye-menu's zijn een interessant en verfrissend concept maar hun praktisch nut is nog niet bewezen, aangezien hiërarchische menu's steeds significant beter bleken te presteren (Hornbæk & Hertzum, 2007). Hoewel het concept fisheye-menu's in eerste instantie niet erg geslaagd lijkt te zijn, blijkt het onderliggende concept (de *fisheye view*) wel erg nuttig in situaties waar gebruikers met een klein scherm moeten werken (Gutwin & Fedak, 2004; Karlson et al., 2005).

2.1.2 Split-menu's

Smith raadt ontwerpers aan om bij het ordenen van data veelgebruikte items bovenaan in een weergave te tonen (Smith & Mosier, 1986). Sears past deze aanbeveling toe op het ontwerp van menu's en komt zo tot een aanpak met zijn *Split-menu's* (Sears & Shneiderman, 1994).

Split-menu's zijn menu's die bestaan uit twee partities: in de bovenste partitie worden de items getoond die vaak geselecteerd worden, terwijl de onderste partitie gebruikt wordt voor de minder frequent gebruikte links. De items binnen deze partities moeten geordend worden zoals men zou doen indien men geen gebruik zou maken van split-menu's, de *traditionele ordening* (Sears & Shneiderman, 1994). Split-menu's dienen dus vooral als versnellingsmechanisme bovenop een bestaand geordend menu en niet zozeer als alleenstaand paradigma. Een voorbeeld van een split-menu voor het selecteren van een lettertype is te zien in afbeelding 2.2.

Sears argumenteert dat het gebruik van split-menu's voordelig is wanneer een kleine deelverzameling aan selecties vaak voorkomt (Sears & Shneiderman, 1994). Dit is vaak het geval bij commando's op de computer. Verder nuanceert hij zijn stelling door te vermelden dat dit voordeel afhankelijk is van de plaatsing van het menu-item in alternatieve ontwerpen. Zo zal er een minimale snelheidswinst tegenover een ander menu zijn wanneer de veel gekozen items daar ook bovenaan staan.

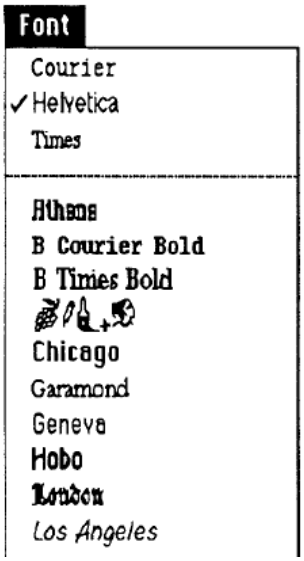
Sears (1994) voerde een *usability test* uit om het nut van de split-menu's te evalueren. Deze test verliep in twee fasen. In de eerste fase werden op twee fysieke sites data over de gemaakte menuselecties binnen een bepaald programma verzameld. Deze data werden dan, na filtering, gebruikt om split-menu's op te stellen voor dit programma. In de tweede fase werden de nieuwe menu's geïmplementeerd op de sites en bleef men data over het gebruik verzamelen. Na deze studie werden de gebruikers gevraagd welk menu hun voorkeur droeg. Uit de analyse van de data bleek dat het split-menu gemiddeld tot snellere selectietijden leidde dan zijn tegenhanger (een alfabetisch geordend menu). Uit de bevraging bleek ook dat gebruikers liever met het split-menu werkten dan met het traditionele menu (Sears & Shneiderman, 1994).

Hiernaast voerde Sears (1994) ook een gecontroleerde gebruikerstest uit. Hierbij kregen 38 testpersonen, die goed bekend waren met computergebruik, 3 verschillende menu's aangeboden waarbinnen ze een bepaald item moesten

vinden. De gebruikte menu's waren:

- een **alfabetisch geordend menu**
- een **menu geordend op selectie-frequentie**
- het **split-menu**

Om het eventuele relatieve voordeel van de split-menu's tegenover de locatie van de items binnen het alfabetisch menu te bepalen, werden er drie versies van de test met het alfabetisch menu opgesteld: de frequent geselecteerde items werden bovenaan, in het midden en onderaan de lijst gekozen. Opnieuw werd aan de gebruikers na de test gevraagd om de menu's te rangschikken volgens hun persoonlijke voorkeur. Na analyse bleek dat de gebruikers het liefst werkten met split-menu's, gevolgd door alfabetische menu's. Menu's geordend op frequentie werden het minst gesmaakt. Wellicht door de schijnbaar willekeurige ordening die hierbinnen bestaat. Split-menu's bleken zoals verwacht significant sneller dan alfabetische menu's, wanneer de meest geselecteerde items onderaan dit menu stonden. Over het algemeen vond men echter geen significant verschil in snelheid tussen het split-menu en de alfabetische menu's. Men vond ook geen significante verschillen inzake de gemaakte fouten (Sears & Shneiderman, 1994).



Figuur 2.2: De werking van Sears' split-menu (overgenomen uit (Sears & Shneiderman, 1994)).

We besluiten dat in bepaalde gevallen het gebruik van split-menu's erg nuttig blijkt. Gezien de vrij lage implementatiekosten en de hoge appreciatie door de gebruikers, moeten ze zeker overwogen worden. Anderzijds kan men stellen dat *expert users* exact weten waar ze een item moeten vinden binnen een menu. Het voordeel van een split-menu tegenover een ander menu komt neer op het feit dat er minder schermafstand afgelegd hoeft te worden en mogelijk minder *clicks* gemaakt. Sears merkt ook op dat split-menu's erg gebruikerafhankelijk zijn (Sears & Shneiderman, 1994). Hij raadt hun gebruik aan wanneer het mogelijk is om elke gebruiker een individueel (gepersonaliseerd) menu te geven.

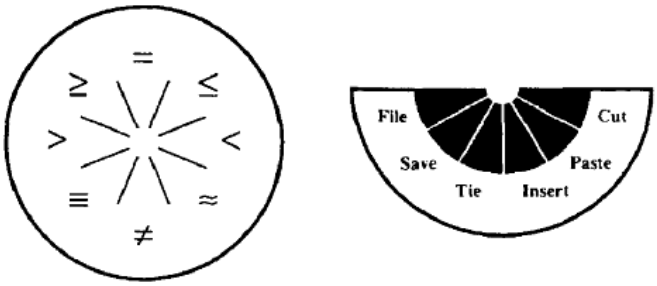
Split-menu's worden verder aangehaald in de secties 2.2.1.1 en 2.2.1.3.

2.1.3 Pie-menu's

Taartmenu's (Engels: *pie menus*) of **radiale menu's** worden beschreven in het werk van Wiseman. Daar bespreekt men als eerste het gebruik van een circulair selectiemenu (Wiseman, Lemke & Hiles, 1969). Een schets van dit soort menu's is te vinden in afbeelding 2.3. Callahan voerde een empirische studie uit naar de voordelen van pie-menu's ten opzichte van lineaire menu's, met menu's van breedte acht (Callahan et al., 1988). Zijn hypothese was dat het gebruik van pie-menu's sneller is en dat het soort items van belang is voor de ordening ervan. Hij testte zijn hypothesen aan de hand van een gebruikersstudie met 33 deelnemers. Deze gebruikers hadden allen weinig tot geen kennis van computersystemen. Tijdens het experiment werden het aantal gemaakte fouten en de selectiesnelheid bijgehouden. Uit het onderzoek bleek dat de gebruikers significant sneller werkten met de pie-menu's. Ze maakten ook significant minder fouten bij gebruik van de taartmenu's dan met de lineaire menu's. Het vermoeden dat het effect van de ordening van de items beïnvloed wordt door de eigenschappen van deze items, werd ook bevestigd. Callahan haalt aan dat slechts een beperkt aantal items in een pie-menu geplaatst kan worden voor dit onbruikbaar wordt (Callahan et al., 1988). Deze limiet wordt echter niet kwantitief bepaald.

Fitzmaurice breidde de werking van pie-menu's uit met zijn *tracking-menu* (Fitzmaurice, Khan, Pieké, Buxton & Kurtenbach, 2003). Een tracking-menu is een radiaal menu dat steeds onder de cursor blijft. Dit gebeurt door het menu te verplaatsen wanneer de gebruiker de muisaanwijzer aan de rand van het radiale menu plaatst. Men gebruikt de analogie van een potlood dat men binnen een rond deksel verplaatst. Wanneer men met het potlood tegen de rand van het deksel duwt, zal het deksel zich verplaatsen. Men voerde

een kwalitatieve gebruikerstest uit op zes gebruikers. Bij de gebruikerstest maakte men gebruik van een tekentablet en een digitale pen. De gebruikers waren artiesten die ervaring hadden met deze hardware. De gebruikers werkten met een standaard implementatie van het menu (via een *tool*-paneel in de software) en het tracking-menu. Tijdens de gebruikerstest werden de opmerkingen van de gebruikers genoteerd. Uit de resultaten van het experiment bleek dat gebruikers een voorkeur hadden voor het tracking-menu. Men vond het heen-en-weer gaan tussen het hoofdvenster en het *tool*-paneel erg vervelend (Fitzmaurice et al., 2003).



Figuur 2.3: Een schets van een pie-menu. (overgenomen uit (Callahan et al., 1988)).

We besluiten dat pie-menu's een interessant menutype zijn. Wanneer ze gebruikt worden met kleinere hiërarchieën leveren deze menu's een aanzienlijke snelheidswinst op. Het nadeel van pie-menu's is echter dat men slechts een beperkt aantal opties aan de gebruiker kan aanbieden alvorens de weergave problematisch wordt (Callahan et al., 1988). Dit soort menu is dus niet bruikbaar voor grote hiërarchieën. Tracking-menu's leveren een extra voordeel aan de gebruikers door te verzekeren dat het menu de muisaanwijzer blijft volgen. Op deze manier zijn de selectie-opties steeds makkelijk bereikbaar. Gebruikers bleken deze tracking-menu's te smaken (Fitzmaurice et al., 2003).

2.1.4 Marking-menu's

Kurtenbach stelt in zijn doctoraatsthesis dat een goede interface niet zozeer makkelijk te gebruiken moet zijn, maar wel vlot *novice* gebruikers in expert-gebruikers moet laten evolueren (G. P. Kurtenbach, 1993). Volgens Kurtenbach moet een menu drie componenten hebben om deze overgang makkelijk te maken:

- de *Novice Component*, waarbinnen er ruimte is voor verkenning, uitleg en leren.
- de *Transition Component*, waarbij de gebruiker expertgedrag kan oefenen maar toch kan terugkeren naar *novice*-functionaliteit wanneer het nodig is.
- de *Expert Component*, waarbij de gebruiker snelle, efficiënte handelingen uitvoert.

Kurtenbach (G. P. Kurtenbach, 1993) stelt dat zijn *marking-menu's* hieraan voldoen, aangezien die uit twee modi bestaan. De eerste is toegespitst op nieuwe gebruikers, terwijl de tweede voor experts bedoeld is. De *transition component* wordt ingevuld doordat de eerste modus de gebruikers het gebruik van de tweede modus aanleert en dat de gebruikers in *expert mode* steeds kunnen teruggrijpen naar de *novice mode* om hun geheugen op te frissen. De vergelijking tussen beide functionaliteiten is uitgebeeld in figuur 2.4.

Marking-menu's zijn menu's waarbinnen gebruikers een selectie kunnen maken door zowel een optie aan te klikken als door een *mark* (een teken) te maken die overeenkomt met de gewenste optie. Deze markering zal dan geïnterpreteerd worden door het systeem, waardoor de aangewezen optie geselecteerd zal worden.

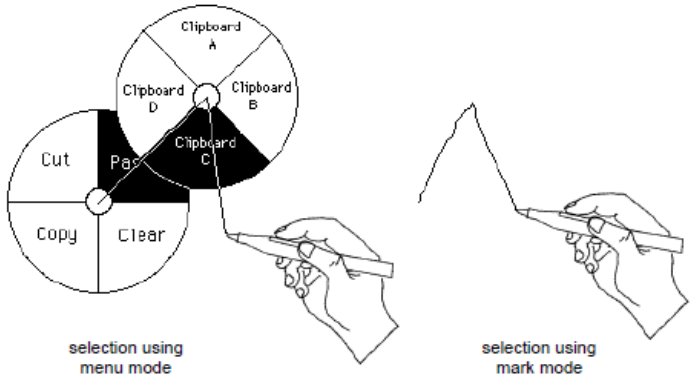
In de volgende secties gaan we achtereenvolgens over tot de bespreking van standaard marking-menu's en hiërarchische marking-menu's. In een volgende sectie bekijken we de voor- en nadelen van deze menu's. Tot slot hebben we het over de mogelijke verfijningen op de marking-menu's.

2.1.4.1 Standaard marking-menu's

Kurtenbach (1993) haalt aan dat er in het verleden al onderzoek gedaan werd naar het gebruik van *marks* (tekens) als invoergegevens. De moeilijkheid hiermee was dat de gebruiker per functionaliteit een nieuwe *mark* moest leren

om er snel mee te kunnen werken. Kurtenbach stelt voor om de gebruiker een radiaal menu aan te bieden dat hij rond zijn cursor kan laten verschijnen door de muis een tijd in te drukken. De markeringen in dit systeem zouden dan neerkomen op de richting die de gebruiker met zijn cursor beweegt om de selectie te maken: een lijn in een bepaalde richting. De achterliggende bedoeling is dat gebruikers markeringen oefenen door eenvoudigweg het radiale menu te gebruiken. In zijn research werkt Kurtenbach vooral met een digitale pen als invoermethode maar hij haalt ook aan dat deze functionaliteit perfect na te bootsen is met de muis (G. P. Kurtenbach, 1993).

Kurtenbach's marking-menu's lijken heel sterk op taartmenu's. Ze werken ook met een circulair selectiesysteem met het visuele verschil dat er een *inktlijntje* achter de cursor getrokken wordt. We kunnen ons dus voor een deel baseren op Callahan's resultaten (Callahan et al., 1988). Deze resultaten geven een sterk vermoeden dat marking-menu's ook beter presteren dan lineaire menu's.



Figuur 2.4: De werking van Kurtenbach's marking-menu's. Links wordt de *novice mode* getoond, rechts de *expert mode*. (overgenomen uit (G. P. Kurtenbach, 1993)).

Kurtenbach voerde enkele casestudy's uit om het nut van zijn marking-menu's te verifiëren (G. P. Kurtenbach & Buxton, 1994). Hij onderzocht de gedragingen van twee gebruikers terwijl ze werkten met een bestaand programma voor videoanalyse waaraan een marking-menu-interface was toegevoegd. Er werd meer dan honderd uur aan data verzameld over het programmagebruik. De eerste gebruiker was een expert, de andere was minder bekend met het gebruik van computers. Uit de data bleek dat het gebruik van *marks*

tot significant snellere interactie met de computer leidde dan wanneer de gebruiker het radiale menu gebruikte (vier tot zeven keer sneller). Ook bleek dat de expertgebruiker vaker met *marks* werkte dan de *novice*. De novice-gebruiker begon na verloop van tijd wel meer en meer naar het gebruik van *marks* te grijpen. Omdat dit een casestudy was, waren de gewenste resultaten van elke actie niet van tervoren gekend. Men kon dus geen foutenanalyse maken. Wel heeft men de gebruikers na afloop van de test gevraagd naar de frequentie vande gemaakte fouten. Ze meldden dat het aantal gemaakte fouten acceptabel was. De gebruikers vertelden ook dat ze het gebruik van *marks* als veel efficiënter ervoeren dan het bestaande menu. Uit deze studie is ook gebleken dat een expert sporadisch teruggrijpt naar het gebruik van het pop-up-menu om na te kijken waar een bepaalde optie staat als zijn geheugen het even laat afweten.

Kurtenbach (1993) (1994) beschrijft dat marking-menu's ook werden toegepast in interfaces voor mobiele apparaten (Weiser, 1995), whiteboards (Elrod et al., 1992) en in combinatie met andere technieken met de zogenaamde *HotBox* (G. P. Kurtenbach, Fitzmaurice, Owen & Baudel, 1999).

2.1.4.2 Hiërarchische marking-menu's

Men heeft dus kunnen aantonen dat *marks* een tijdswinst opleveren in vlakke menu's. Dit soort menu's komt vaak voor binnen programma's om een uit te voeren actie te selecteren. Om door een structuur te navigeren heeft men echter nood aan hiërarchische marking menu's, omdat we het aantal items in een taartmenu niet ongelimiteerd kunnen opvoeren zonder dit onbruikbaar te maken. Kurtenbach beschrijft het ontwerp en gebruik van hiërarchische marking-menu's (G. Kurtenbach & Buxton, 1993). Kurtenbach beschrijft dat een novice-gebruiker de cursor ingedrukt houdt om het circulaire menu op te roepen en vervolgens de cursor beweegt naar het gekozen item. Indien dit item een submenu bevat, zal dat worden weergegeven. De gebruiker kan dan weer op dezelfde manier verder navigeren binnen dit submenu. Wanneer de gebruiker stopt met de cursor ingedrukt te houden, zullen de geselecteerde keuzes verwerkt worden. De gebruiker kan een keuze ongedaan maken door met de cursor terug te bewegen op het reeds afgelegde pad, terug naar het centrum van het radiale menu. Op deze manier kan hij *naar boven* navigeren binnen de hiërarchie (G. Kurtenbach & Buxton, 1993).

Kurtenbach voerde experimenten uit om een aantal vragen te beantwoorden (G. Kurtenbach & Buxton, 1993). Zo wilde hij in de eerste plaats weten

of hiërarchische marking-menu's nuttig zijn. Daarnaast vroeg hij zich af wat de maximale dimensies (breedte en diepte) van de hiërarchische structuur zouden zijn. Hij wou ook weten of het gebruiken van verschillende breedtes binnen eenzelfde menu de prestaties met dit menu zouden verminderen. Tot slot vroeg hij zich af of men met een computermuis voldoende complexe markeringen kan maken in vergelijking met een digitale pen. Het experiment werd uitgevoerd met twaalf gebruikers. Omdat men geïnteresseerd was in het gedrag van experts, gebruikte men een hiërarchisch menu bestaande uit geneste exemplaren van een windroos. Aan de gebruikers werd gevraagd om sequenties van windrichtingen te selecteren. Gebruikers kregen menu's met variërende dieptes en breedtes voorgeschoteld. De snelheid waarmee men selecties maakte en het aantal gemaakte fouten werd bijgehouden. Door de vrij lage foutenfrequentie kon men afleiden dat hiërarchische marking-menu's een haalbaar idee zijn. Men merkte, zoals verwacht, dat de gebruikers mindere prestaties neerzetten wanneer de breedte en de diepte van het menu omhoog gingen. Verder besloot men dat tot diepte twee het aantal gemaakte fouten erg klein was. Wanneer men dieper ging in een hiërarchie met breedte acht of meer, doken er echter meer fouten bij het gebruik van markeringen. Dit gold zelfs voor experts. Als de breedte kleiner was, kon de hiërarchie dieper worden voordat dit effect zich voordeed. Zo kon men in een structuur met breedte vier, tot vier niveaus diep gaan alvorens het aantal gemaakte fouten problematisch werd. Dit probleem zou deels verholpen kunnen worden door de veelgebruikte items te plaatsen volgens de assen. Uit de data kon men ook afleiden dat gebruikers iets sneller werken met een pen dan met een muis.

2.1.4.3 Voordelen en tekortkomingen

Marking menu's kennen alle voordelen van taartmenu's. Zoals aangetoond kunnen gebruikers aan de hand van marking-menu's sneller navigeren binnen de opties van een menu (Callahan et al., 1988; G. Kurtenbach & Buxton, 1993; G. P. Kurtenbach & Buxton, 1994). Gebruikers blijven vermoedelijk ook aandachtiger bij hun werk omdat hun *workflow* niet gebroken wordt doordat ze steeds hun aandacht moeten verplaatsen naar een statisch menu bovenaan de pagina. Men heeft ook kunnen aantonen dat marking-menu's gebruikers makkelijk de overgang maken van *novice*- naar het erg snelle expertgedrag (G. P. Kurtenbach, 1993; G. P. Kurtenbach & Buxton, 1994).

Kurtenbach beschrijft echter enkele moeilijkheden bij het implementeren en het gebruik van marking-menu's (G. P. Kurtenbach, 1993). Zo doet het *segmentatieprobleem* zich voor bij het analyseren van de gemaakte te-

kens. Selectie binnen een marking-menu bestaat veelal uit het maken van samengestelde *marks*, bv. om door verschillende lagen van het menu te navigeren. Het segmentatieprobleem is de benaming van de problematiek rond de afbakening van individuele tekens. Deze afbakening is nodig om de ingegeven commando's correct te interpreteren en een correct resultaat te verkrijgen. Omdat marking-menu's in eerste instantie geen rekening houden met de lengte van de markeringen, is het mogelijk om met dezelfde *mark* toch verschillende opties te selecteren. Dit doet zich echter enkel voor wanneer het menu niet opgeroepen wordt. Door het segmentatieprobleem worden marking menu's onbruikbaar wanneer de breedte of diepte van het hiërarchisch menu te groot wordt (G. Kurtenbach & Buxton, 1993). Door het grote aantal items wordt het voor de gebruiker moeilijk zijn *mark* in de juiste hoek te tekenen om het gewenste item te selecteren. Dit zorgt ervoor dat hiërarchische marking-menu's slechts een beperkt aantal items (64, breedte 8 en diepte 2) kunnen bevatten alvorens selecties problematisch worden (G. Kurtenbach & Buxton, 1993).

Een ander probleem is dat de gebruiker nagenoeg rechte lijnen moet trekken en vrij correcte hoeken moet hanteren bij het tekenen van de *marks*. Dit kan problematisch worden wanneer de gebruiker enkel inaccurate invoerapparaten voorhanden heeft (een trackpad van een laptop bijvoorbeeld). Ook zijn marking-menu's niet toepasselijk wanneer de locatie van items dynamisch wordt aanpast binnen de menu's (G. P. Kurtenbach & Buxton, 1994).

2.1.4.4 Verfijningen op marking-menu's

Kurtenbach ging later zelf verder in op zijn onderzoek en zocht naar een manier om het probleem met de beperkte breedte van marking-menu's op te lossen. Hij diende een patent in over een techniek die marking-menu's met lineaire menu's combineert (G. P. Kurtenbach, 1999). Verder onderzoek probeert de tekortkomingen van marking-menu's verder weg te werken (Tapia & Kurtenbach, 1995; Zhao & Balakrishnan, 2004; Bailly et al., 2007, 2008).

Tapia voegt enkele *verfijningen* toe aan Kurtenbach's marking-menu's (Tapia & Kurtenbach, 1995). Drie ideeën liggen aan de basis van zijn werk:

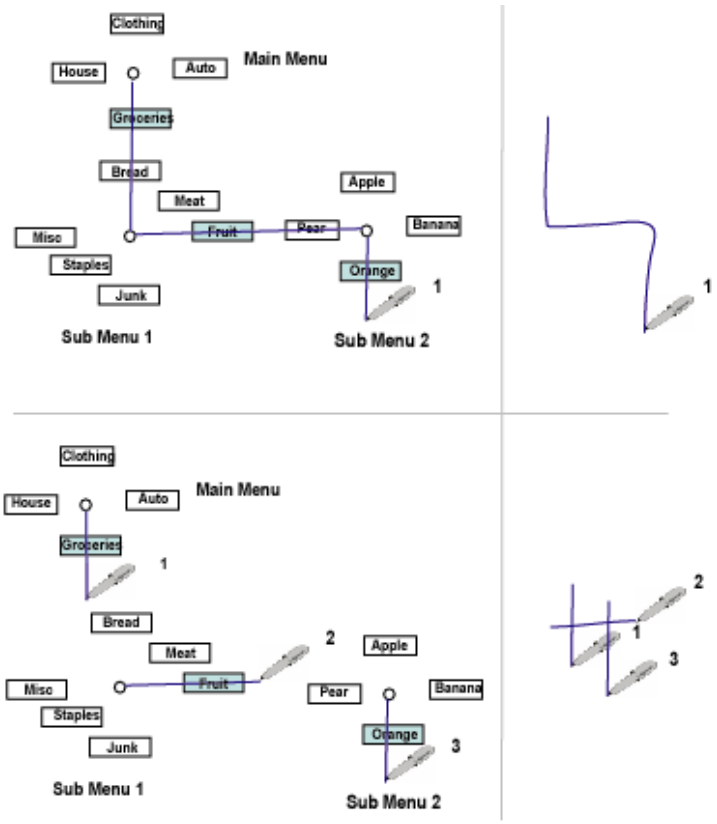
- Behouden van de visuele context
- Verbergen van onnodige informatie
- Vaardigheidsontwikkeling door grafische terugkoppeling

Deze principes worden nagestreefd bij het opstellen van enkele ingrepen in het origineel ontwerp. Men verandert echter niets ingrijpend omdat het meestal over grafische aanpassingen gaat. Zo pleit Tapia bijvoorbeeld voor het weergeven van enkel een label per menu-item, in plaats van het volledig (opgevulde) taartmenu. Op deze manier wordt er minder verborgen van de onderliggende context (het document waarboven het menu geopend wordt). Hiernaast moet wel de functionaliteit van het oorspronkelijke taartmenu behouden worden. Zo moet een item nog steeds geselecteerd worden als de cursor binnen de *taartschijf* van dat label, of binnen het label zelf, wordt losgelaten. De andere aanbevelingen gaan over de symmetrie, de plaatsing en de hoeveelheid van de selectie-opties, het verbergen van parent-menu's en de manier waarop het inktspoor wordt weergegeven. Deze aanpassingen werden vermoedelijk opgesteld en bijgestuurd aan de hand van gebruikersobservaties, maar in Tapia's paper wordt geen empirische verificatie vermeld (Tapia & Kurtenbach, 1995).

Simple marks

Zhao stelt een variant op marking-menu's voor die enkele van de problemen van de standaard implementatie verhelpen: *simple marks*, ook wel een *multi-stroke-menu* genoemd (Zhao & Balakrishnan, 2004). Hierbij werkt de gebruiker met discontinue *marks* in plaats van de aaneengeschakelde *marks* die Kurtenbach (G. P. Kurtenbach, 1993; G. Kurtenbach & Buxton, 1993) voorstelde. Het verschil tussen deze technieken is voorgesteld in afbeelding 2.5. Volgens Zhao kan men met *simple marks* nauwkeurig selecteren in een marking-menu met een grotere diepte, wat bij de standaard-implementatie een groot probleem was (G. Kurtenbach & Buxton, 1993; Zhao & Balakrishnan, 2004). Daarnaast wordt het probleem van de ambiguïteit door de variabele lengtes van *marks* opgelost. Ook heeft de gebruiker minder fysieke plaats nodig om de *marks* te tekenen. Om deze beweringen kracht bij te zetten, voerde Zhao een gebruikerstest uit. Men probeerde vooral na te gaan hoe de selectie-snelheid van de gebruiker verschilde tussen de *simple* en *compound marks* en of men met *simple marks* diepere hiërarchieën kon ondersteunen. Men voerde het experiment uit met twaalf gebruikers, die geen van allen eerdere kennis van marking-menus hadden. Om expertgedrag te simuleren werd er gebruikt gemaakt van de windrichtingen als menu-opties. Gebruikers werkten zowel met *simple* als *compound* hiërarchische marking-menu's. Men kon aantonen dat gebruikers significant minder fouten maakten met de *simple marks* dan met de *compound marks*, ook voor diepere structuren. Gebruikers bleken gemiddeld gezien zelfs sneller te presteren met de

simple marks. Ook de hypothese dat gebruikers minder fysieke ruimte nodig hadden om *marks* te tekenen bleek te kloppen. Aan de deelnemende gebruikers werden na afloop van het onderzoek gevraagd een vragenlijst in te vullen om hun voorkeur tussen de twee technieken te kunnen bepalen. Gebruikers bleken geen uitgesproken voorkeur te hebben voor één van de alternatieven. Wel bleek dat de prestaties - hoewel ze in de *expert mode* verbeterden - in de *novice mode* slechter werden.

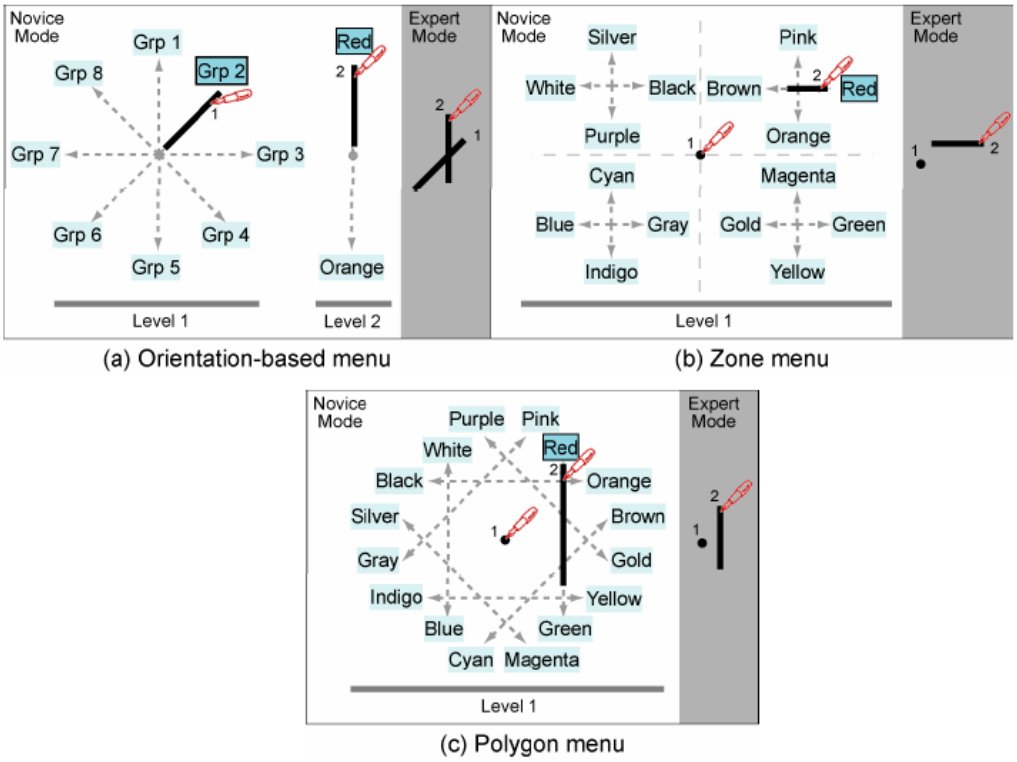


Figuur 2.5: *Compound* of samengestelde *marks* (a) tegenover *Simple marks* (b). Links wordt de *novice mode* uitgebeeld, rechts de *expert mode*. (overgenomen uit (Zhao & Balakrishnan, 2004)).

Zone- en polygon-menu's

Zhao stelt in een latere studie twee nieuwe varianten op multi-stroke marking-menu's voor: *Zone-menu's* en *Polygon-menu's* (Zhao et al., 2006). Met deze menu's probeert hij de maximumbreedte van het marking-menu uit te breiden

door naast de oriëntatie ook rekening te houden met de positie van getekende markeringen. De nieuwe menu's vergen een extra actie voor het kalibreren van het menu: de gebruiker moet eerst een stip plaatsen op het scherm om de oorsprong van het menu te bepalen. Een vergelijking tussen de drie menu's wordt afgebeeld in figuur 2.6.



Figuur 2.6: Overzicht van drie soorten marking-menu's: een multi-mark-menu (a), een zone-menu (b) en een polygon-menu (c) (overgenomen uit (Zhao et al., 2006)).

Bij een *zone-menu* worden de selectiemogelijkheden opgedeeld in vier afzonderlijke menu's. Het scherm wordt gesplitst rond een geselecteerd schermpunt en op deze manier opgedeeld in zones. Elk van de submenu's kan nu afgebeeld worden op een bepaalde zone. Afhankelijk van de positie van de getekende mark ten opzichte van de oorsprong van het menu, zal het teken in een bepaalde zone vallen. Zo kan er een optie uit het overeenkomstige submenu geselecteerd worden. De positie van de mark bepaalt uit welk submenu de selectie zal komen en de richting van de mark bepaalt de uiteindelijk geko-

zen menu-optie. Op deze manier kan het menu n*m menu-items per niveau hebben, met n het aantal zones en m de breedte van de submenu's (Zhao et al., 2006).

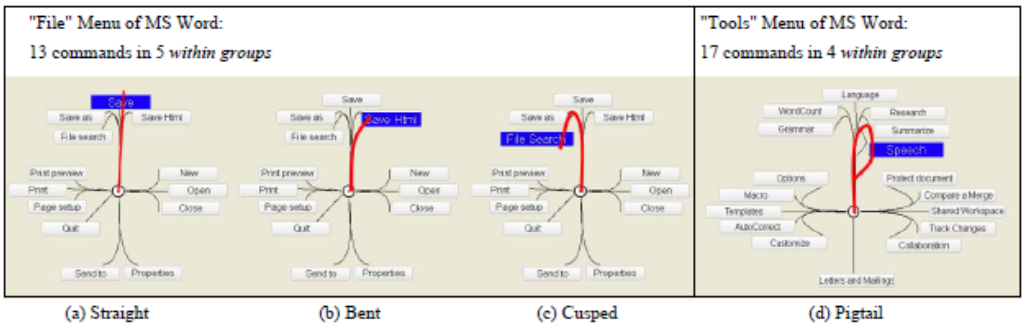
Een nadeel van de zone-menu's is dat de gebruiker de oorsprong van het menu zorgvuldig moet kiezen, om het menu werkbaar te houden: een slecht gekozen oorsprong kan ervoor zorgen dat enkele zones erg klein en moeilijk bereikbaar zijn. Om dit op te lossen stelt Zhao de *polygon-menu's* voor (Zhao et al., 2006). Bij een polygon-menu tekent de gebruiker markeringen die overeenkomen met de zijden van een n-zijdige veelhoek. De menu-opties zijn geordend rond de zijden van deze veelhoek. Zo "*hangt*" er aan weerszijden van een zijde een selectie-optie. De hoek en positie ten opzichte van de oorsprong die de door de gebruiker getekende markering heeft, bepalen dan welke zijde van de veelhoek geselecteerd wordt. De richting waarin de mark getekend wordt, bepaalt dan welke van de twee opties die aan die zijde verbonden zijn gekozen wordt. Op deze manier kan het menu breedte 2n hebben, met n het aantal zijden van de veelhoek (Zhao et al., 2006).

Zhao (2006) voerde een gebruikerstest uit met negen vrijwilligers om hun prestaties met de zone en polygon-menu's te vergelijken met hun prestaties met multi-stroke-menu's. Geen van de gebruikers had ervaring met marking-menu's. Expertgedrag werd gesimuleerd door gebruikers de gewenste markeringen te tonen. Men hield de snelheid en foutenfrequentie van de gebruikers bij. Uit de experimenten bleek dat polygon- en zone-menu's accurater maar trager zijn dan standaard multi-stroke-menu's voor een klein aantal menu-items. Bij een groter aantal menu-items bleken alle menu's ongeveer even accuraat te zijn, maar waren de polygon- en zone-menu's significant sneller. Deze tijdswinst bij grotere menu's komt waarschijnlijk omdat zone- en polygon-menu's minder diep zijn dan de overeenkomstige standaard multi-stroke-menu's. Verdere experimenten bewezen dat zone-menu's met een breedte van 64 niet mogelijk waren door een te lage nauwkeurigheid (d.i. een te hoge foutenfrequentie). Zone-menu's met een breedte van 32 bleken wel een mogelijkheid. Ook testen de onderzoekers het effect van een gefixeerde oorsprong, waarbij de extra markering die de gebruiker moet maken om de oorsprong te selecteren, overbodig is. De resultaten van deze tests kwamen overeen met die van de tests met een variabele oorsprong, afgezien van een kleine snelheidswinst. Bailly haalt aan dat gebruikers de actie van het selecteren van de oorsprong van het menu niet aangenaam vonden (Bailly et al., 2008). Ook merkte hij in zijn experimenten dat gebruikers moeite hadden met het leren werken met polygon-menu's (Bailly et al., 2008).

Flower-menu's

Het aantal items dat in een standaard hiërarchisch marking-menu weergegeven kan worden is beperkt. Dit kan opgelost worden door de maximum diepte te verhogen (door middel van *simple marks*) of door de maximum breedte op te voeren. Om de problematiek rond de beperkte breedte van hiërarchische marking-menu's op te lossen, stelt Bailly zijn *flower-menu* voor (Bailly et al., 2008). De werking van dit menu wordt afgebeeld in figuur 2.7. Dit menu is een variatie op het hiërarchische marking-menu, waarbij ook gebogen *marks* mogelijk zijn. Hierdoor kan men in het menu veel meer selectieopties per niveau weergeven, tot een theoretisch maximum van 56 items (7 soorten gebogen *marks* in 8 richtingen). Een ander voordeel is dat men binnen flower-menu's kan werken met *within groups*. Dit zijn groeperingen van items die bij elkaar horen binnen hetzelfde hiërarchische niveau (bv. *save*, *save as* en *save html*). Hiërarchische flower-menu's werken, zoals Zhao's multi-stroke-menu (Zhao & Balakrishnan, 2004), met afzonderlijke discontinue *marks*.

Bailly toonde in een eerste experiment aan dat de verschillende *marks* verschillende prestaties tot gevolg hadden (Bailly et al., 2008). Zo waren selecties met rechte *marks* sneller dan die met gebogen *marks*. In een tweede experiment vergeleek men de prestaties van experts wanneer ze werkten met flower-menu's tegenover lineaire menu's (met *keyboard hotkeys*) en polygon-menu's. Men voerde deze studie uit met 18 deelnemers. Men gebruikte een menu met een breedte van 16 en een diepte van 1. Verschillende configuraties van de *within groups* werden getest. Gebruikers bleken de locatie van items in het menu significant beter te onthouden in het flower-menu dan in de andere menu's. In totale selectietijd waren flower-menu's (2.4 s) sneller dan lineaire menu's (3.5 s) en polygon-menu's (3.8 s). Na het experiment werd aan de gebruikers gevraagd een vragenlijst in te vullen om hun voorkeuren te bepalen. De gebruikers bleken een voorkeur te hebben voor de flower-menu's.

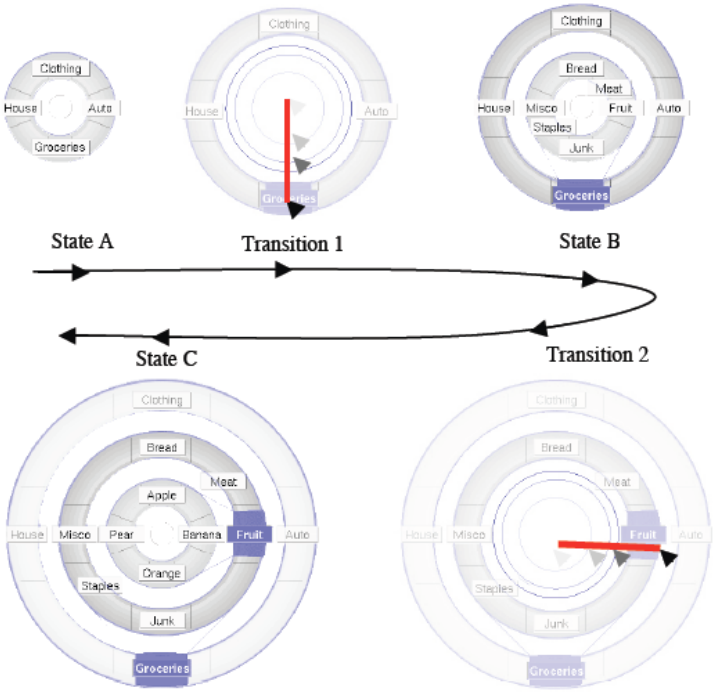


Figuur 2.7: Een flower-menu met *within groups* en het effect van verschillende soorten *marks* binnen dit menu. (overgenomen uit (Bailly et al., 2008)).

Wave menu's

Daar waar de eerder besproken verfijningen toegespitst zijn op het verbeteren van de *expert mode* van de marking-menu's, richt Bailly (Bailly et al., 2007) zich op het verbeteren van de *novice mode* van deze menu's met zijn *wave-menu*. Bailly verklaart deze keuze door aan te halen dat de *novice mode* de eerste kennismaking is tussen de gebruiker en het marking-menu. Indien deze interactie niet positief ervaren wordt, zal de gebruiker minder geneigd zijn zich te verdiepen in het werken met het marking-menu. Indien dit gebeurt zal het erg efficiënte expertgedrag nooit vertoond worden. Een andere reden om zich te concentreren op de *novice mode* is dat uit vorige onderzoeken (G. P. Kurtenbach & Buxton, 1994) (G. Kurtenbach & Buxton, 1993) blijkt dat zelfs ervaren gebruikers vaak teruggrijpen naar de *novice mode* om minder gebruikte commando's te lokaliseren.

Wave-menu's zijn een variant op de multi-stroke-menu's, gekend van Zhao (2004). De menu's verschillen enkel in *novice mode*. In een wave-menu worden de selectie-opties ook radiaal rond de cursor weergegeven. Afbeelding 2.8 toont de werking van het wave-menu. Wanneer er een item geselecteerd wordt, en zodoende naar een submenu genavigeerd wordt, wordt het parent-menu uitvergroot en komt het submenu centraal binnen dit parent-menu in beeld. Het voordeel van deze methode is dat het selectiepad (d.i. de eerder gemaakte selecties) duidelijk in beeld blijft. Ook kan op deze manier gebruik gemaakt worden van *pre-visualisatie*. Pre-visualisatie is het tonen van de inhoud van een submenu wanneer de gebruiker met zijn cursor over een item zweeft, zonder een selectie te maken. Dit is een groot hulpmiddel voor nieuwe gebruikers, die de structuur van het menu nog moeten verkennen.



Figuur 2.8: De werking van een wave-menu (overgenomen uit (Bailly et al., 2007)).

Bailly voerde een gebruikerstest uit met 12 gebruikers, die allen computerervaring hadden (Bailly et al., 2007). Aan de gebruikers werd gevraagd een item te selecteren uit een hiërarchie. Om deze taak te laten corresponderen met een reële situatie werd de naam "*Naam1*" gegeven aan parent-menu-items die mogelijk het item "*Naam2*" bevatten. De gebruikers moesten alle iteraties van een bepaald item ("*Naam2*") vinden in het menu. Deze taak werd uitgevoerd op wave-menu's, compound-stroke-menu's en multi-stroke-menu's. Als één van de enigen gebruikte Bailly een computermuis in plaats van een digitale pen als invoerapparaat tijdens het experiment. Men hield de tijd die nodig was voor het uitvoeren van de taak en het aantal gemaakte fouten bij. Men ondervond dat prestaties met compound-stroke-menu's en wave-menu's significant sneller waren dan die met multi-stroke-menu's. Daarnaast bleek ook dat gebruikers aanzienlijk (8 keer) minder fouten maakten met wave-menu's dan met multi-stroke-menu's. Het is belangrijk te vermelden dat deze resultaten enkel opgaan voor de *novice mode* van de menu's.

In deze sectie bespraken we fisheye-menu's, split-menu's, pie-menu's en

marking menu's. We gaven steeds een korte synthese van de voor- en nadelen van deze menutypes. Deze syntheses worden later hernomen in de samenvattende tabellen 2.1 en 2.2. In de volgende sectie bespreken we een aantal eigenschappen van menu's die een impact hebben op de gebruikerservaring.

2.2 Algemene menu-eigenschappen

Naast het menutype zijn er andere eigenschappen die de goede werking van een menu bepalen. We bespreken de aanpasbaarheid van de menu's, het visualiseren van het selectiepad en de diepte en breedte van het menu.

2.2.1 Statische, adaptieve of aanpasbare menu's?

Iedereen is bekend met statische menu's. Het is immers de standaard binnen user-interface-design. Statische menu's zijn menu's die niet wijzigen en die niet aanpasbaar zijn tijdens hun levensloop. Ze houden dus dezelfde vorm en inhoud die ze hadden na hun implementatie. Tegenwoordig bevat software enorm veel functionaliteit. Zodanig veel dat ze soms zelfs *bloated* (opgeblazen) wordt genoemd (Kaufman & Weed, 1998; McGrenere, Baecker & Booth, 2002). Men merkt dat gebruikers vaak slechts een deel van de mogelijke opties binnen een programma gebruiken. Dit deel is afhankelijk van de taak die de gebruiker met de applicatie uitvoert. Dit zorgt ervoor dat gebruikers moeten bladeren door ellenlange menu's met functionaliteit die ze nooit zullen gebruiken voor ze bij de items komen die nuttig zijn voor hen (Findlater & McGrenere, 2004). Om dit te verhelpen wordt er gewerkt met **personalisatie** van de interface. Verschillende gebruikers (of gebruikersgroepen) gebruiken hetzelfde programma, maar de interface kan voor ieder van hen verschillen. Dit kan gedaan worden op een aantal verschillende manieren. Men kan de interfaces ofwel adaptief, ofwel aanpasbaar door de gebruiker (Engels: *adaptable*) maken (Findlater & McGrenere, 2004).

2.2.1.1 Adaptieve Menu's

Adaptieve menu's zijn menu's waarvan de inhoud, meestal is dit de volgorde van de items in het menu, aangepast wordt naargelang van het gedrag en de voorkeuren van gebruikers in het systeem (Selvidge, 2002). Een adaptief menu kan gebruiker-afhankelijk zijn, wat wil zeggen dat elke gebruiker een ander menu krijgt, of het kan *system-wide* zijn, waarbij elke gebruiker hetzelfde menu krijgt. Het systeem zelf past het menu dynamisch aan door

rekening te houden met de selectiefrequentie van bepaalde items binnen dit menu. Meer gebruikte items worden zo meer in de kijker geplaatst en minder gebruikte items krijgen een minder voorname plaats. Op deze manier zouden gebruikers gemiddeld gezien veel sneller moeten kunnen werken met de applicatie, aangezien de meest gebruikte links makkelijk bereikbaar zijn. Enkel in randgevallen, waar de gebruiker een weinig gebruikt item nodig heeft, zou het menu vergelijkbaar aan niet-adaptieve menu's presteren.

Adaptieve menu's kennen echter een aantal nadelen. Findlater en Gajos halen aan dat het menu sneller zou moeten worden door de menu-items dynamisch te herschikken, maar dat gebruikers geen leereffect zullen ondervinden. Zo zullen ze nooit kunnen voorspellen op welke plaats in het menu een bepaald, gekend item zich zal bevinden. De gebruiker moet dus iedere keer dat het menu zich aanpast opnieuw vertrouwd raken met de nieuwe ordening (Gajos, Czerwinski, Tan & Weld, 2006; Findlater & Gajos, 2009). In het werk van Greenberg vinden we het empirisch bewijs voor deze stelling (Greenberg & Witten, 1985).

Gajos vergeleek verschillende aanpakken voor het maken van adaptieve menu's om te bepalen welke aspecten dit soort menu's al dan niet succesvol maken (Gajos et al., 2006). Hij vergeleek drie soorten adaptieve menu's met een niet- adaptief referentiemenu. De gebruikte adaptieve menu's waren:

- een **split-interface** waarbij opties uit het hoofdmenu gekopieerd werden naar een adaptieve *toolbar* in de hoofdbalk.
- een **moving-interface**, geïnspireerd op een split-interface, waarbij enkele veel gebruikte items in de hoofdbalk van het menu beschikbaar zijn en de andere items in een dropdown-paneel. De volgorde van de items in het dropdown-paneel wijzigt dynamisch.
- een **visual-popout-interface** waarbij de items in een dropdown-paneel geschikt staan. Vaak gebruikte items worden *"opgewaardeerd"*. Opgewaardeerde items krijgen een opvallende kleur binnen het paneel. Indien het menu een opgewaardeerd item bevat, wordt de knop die het dropdown-paneel oproept ook opvallend gekleurd.

Gajos voerde gebruikerstesten uit op 26 testpersonen tussen de leeftijden 25 en 55 (Gajos et al., 2006). Deze gebruikers hadden allen een gemiddelde tot hoge kennis van computersystemen. Aan de testpersonen werd gevraagd een reeks opdrachten uit te voeren met de verschillende interfaces. Uit de resultaten van deze tests bleek dat gebruikers een significante voorkeur hadden

voor de split-interface. Verder bleek dat de visual-popout-interface als erg afleidend werd ervaren. Het is belangrijk om op te merken dat de resultaten van de tijdsanalyse aantoonden dat er geen significant verschil bestond tussen de uitvoeringstijden met de verschillende interfaces. Gajos besloot dat het succes van een adaptief design afhangt van de ervaren *benefit/cost*-ratio (Gajos et al., 2006). Dit wil enerzijds zeggen dat adaptieve menu's die een grote snelheidswinst opleveren maar door de gebruikers als storend ervaren worden geen geslaagde interfaces opleveren. Anderzijds kunnen menu's met een lage snelheidswinst maar die een beperkt negatief effect hebben op de gebruikerservaring als erg succesvol geklassificeerd worden.

Al-Omar haalt verder ook aan dat de grootte van een menu een belangrijke rol speelt in de selectie van het type adaptief menu (Al-Omar & Rigas, 2009). Sommige menu's worden door gebruikers immers niet gesmaakt als het menu niet voldoende groot is. Het is dus belangrijk om het juiste type adaptief menu te kiezen voor de gewenste *formfactor*. Zo zal een adaptief split-menu niet wenselijk zijn voor een menu dat gebruikt moet worden op een kleine schermruimte (bv. een mobiele telefoon) (Al-Omar & Rigas, 2009).

2.2.1.2 Aanpasbare menu's

Naast menu's die zichzelf dynamisch aanpassen, bestaat er ook een andere manier om gebruikers een gepersonaliseerd menu te geven. Dit zijn de **aanpasbare menu's**. Sommige aanpasbare menu's zijn initieel leeg of schaars ingevuld en worden door de gebruiker zelf opgevuld met menu-items en werken in tandem met een standaard-interface. In andere versies van aanpasbare menu's hebben gebruikers de mogelijkheid om zelf de menu-items van de standaard-interface te rangschikken. De kern van dit soort menu's is simpelweg dat de gebruiker de keuze krijgt hoe zijn menu eruit moet zien'.

Het aanbieden van deze functionaliteit heeft als nadeel dat de ontwikkelaars veel meer tijd en moeite moeten steken in het implementeren van de software. Ook is er de moeilijkheid dat het vaak de ontwikkelaars zullen zijn die bepalen welke items aanpasbaar zijn en in welke mate ze gewijzigd kunnen worden (Stuerzlinger, Chapuis, Phillips & Roussel, 2006). Als antwoord op deze problematiek werkte Stuerzlinger in zijn paper een methode uit om gebruikers de mogelijkheid te geven om een interface naar hun hand te zetten via zijn *user-interface façades* (Stuerzlinger et al., 2006). Stuerlinger baseerde zich hiervoor op vier principes:

- snelle, eenvoudige, *just-in-time* aanpassingen

- de mogelijkheid voor zowel globale als lokale aanpassingen
- diepe aanpasbaarheid, dus de mogelijkheid om niet enkel voorgedefinieerde maar ook nieuwe modificaties te kunnen maken
- *cross-application* aanpasbaarheid

Façades zijn vensters die nieuwe of bestaande interface-opties bevatten. Met de *façades* krijgt de gebruiker de mogelijkheid om bestaande elementen te selecteren en deze in een nieuw venster te kopiëren, delen van vensters (semi)-transparant te maken en externe applicaties op te roepen vanuit de *façade*. In Stuerzlinger's paper wordt echter geen onderzoek gedaan naar de impact van *façades* op de gebruikerservaring (Stuerzlinger et al., 2006).
Mackay voerde een experiment uit om het gedrag van gebruikers te bestuderen wanneer ze werkten met aanpasbare menu's (Mackay, 1991). Het experiment werd uitgevoerd met 51 gebruikers met verschillende computerervaring. Gebruikers werkten over een periode van 4 maanden met software waaraan ze verscheidene aanpassingen konden maken. Men verzamelde gegevens tijdens het gebruik van de software. Gebruikers werden ook geïnterviewd en hen werd gevraagd om een vragenlijst in te vullen. Uit de resultaten van dit experiment bleek dat het merendeel van de gebruikers verkoos om de software niet aan te passen. De meeste gebruikers pasten de software enkel aan nadat die, door het uitkomen van een nieuwe versie, vanzelf wijzigde. De gebruikers kozen er dan voor om deze nieuwe aanpassingen terug te draaien, zodat ze verder met hun vertrouwde interface konden werken. Een veel aangehaalde reden om de software niet aan te passen was het gebrek aan tijd (Mackay, 1991). Men leerde ook dat het al dan niet aanpassen van de software een sociaal aspect kent: gebruikers waren geneigd om aanpassingen die andere mensen uitvoerden over te nemen (Mackay, 1991). Het aanpassen van software leek enkel te gebeuren wanneer gebruikers vonden dat de voordelen van deze aanpassingen (uitvoeringsnelheid) opwogen tegen de kosten (de tijd die nodig is om de aanpassingen te maken) (Mackay, 1991).

Voor een analyse van de haalbaarheid van aanpasbare menu's in user-interfaces verwijzen we naar sectie 2.2.1.3 van dit hoofdstuk.

2.2.1.3 De vergelijking

McGrenere voerde een experiment uit waarbij hij een adaptief menu tegenover een aanpasbaar menu plaatste. Hij vergeleek de adaptieve interface van Microsoft Word 2000 met een eigen prototype van een aanpasbaar menu

(McGrenere et al., 2002). Wanneer het opgeroepen wordt, biedt het adaptieve menu van *MS Word 2000* de gebruiker een kort menu met de meest gebruikte items. Wanneer de gebruiker met zijn cursor over dit menu blijft zweven, komt het langere, uitgebreide menu tevoorschijn. Indien er een item uit dit uitgebreide menu wordt gekozen, wordt het toegevoegd aan de korte lijst.
Het prototype van het aanpasbaar menu bestaat uit twee delen: een menu dat door de gebruiker ingevuld wordt en het volledig origineel menu (het uitgebreide menu uit de *MS Word 2000*-implementatie). De gebruiker beschikt over een manier om tussen de twee menu's te wisselen.
McGrenere voerde een gebruikerstest uit met 20 deelnemers met een gemiddelde en uitgebreide kennis van computersystemen, die allen ervaring hadden met de *MS Word 2000*-software (McGrenere et al., 2002). Gebruikers werkten een tijd met de aanpasbare interface, om op het einde terug te moeten overschakelen op de standaard implementatie. De acties van gebruikers in de software werden gedurende een periode opgevolgd. Gebruikers vulden op regelmatige momenten vragenlijsten in en hadden geregeld een afspraak met een onderzoeker om hun ervaringen te bespreken. Uit de verzamelde data bleek dat de meerderheid van de gebruikers de aanpasbare interface gebruikte en dat het aanpassen van het menu vlot leek te verlopen. De vragenlijsten en interviews toonden aan dat een significant deel van de gebruikers (7 van de 20) niet begreep hoe de adaptieve menu's werkten. De meerderheid (13) van de gebruikers verkoos het aanpasbare menu boven het adaptieve menu. (McGrenere et al., 2002)

Findlater vergelijkt in haar onderzoek de performantie van statische, aanpasbare en adaptieve menu's (Findlater & McGrenere, 2004). Men voerde een experiment uit met 27 gebruikers met uiteenlopende kennis van computers. Deze gebruikers voerden selectietaken uit op de 3 menutypes:

- een **statisch split-menu**: een implementatie van de split-menu's, zoals beschreven in sectie 2.1.2. De bovenste partitie van het split-menu bevat de 4 frequentst geselecteerde items.

- een **adaptief split-menu**: een split-menu waarvan de bovenste partitie 4 items bevat. Van deze items worden er 2 gekozen op basis van de selectiefrequentie. De andere 2 items zijn de twee recentst geselecteerde items.

- een **aanpasbaar split-menu**: een split-menu waarin de gebruikers de volgorde van de items kunnen aanpassen binnen de bovenste zowel als de onderste partitie. Aanpassingen worden gedaan door items naar een

andere positie te verplaatsen met behulp van de pijltjestoetsen. De bovenste partitie van dit menu is initieel leeg. Het is aan de gebruiker om dit inhoud te geven.

Tijdens het experiment werden de selectiesnelheid en het aantal gemaakte fouten gemeten. Na uitvoering van de opdrachten werd aan de gebruikers gevraagd om een vragenlijst in te vullen. Met deze vragenlijst polste men naar de voorkeur van de gebruikers, de ingeschatte efficiëntie, de foutenfrequentie, de frustraties en het gebruiksgemak. Gebruikers kregen slechts eenmalig de mogelijkheid om het aanpasbare menu naar hun hand te zetten. Uit de resultaten bleek dat het merendeel van de gebruikers ervoor koos om het aanpasbaar menu te wijzigen (24 van de 27 gebruikers). Verder bleek ook dat gebruikers snellere selecties maakten met de statische menu's dan met de adaptieve menu's. Er werd geen significant verschil gevonden in de selectiesnelheden bij gebruik van de aanpasbare menu's enerzijds en de statische menu's anderzijds. Men vond ook geen significante verschillen in de foutenfrequentie bij gebruik van de verschillende menu's. Uit de vragenlijsten bleek dat de gebruikers een voorkeur hadden voor de aanpasbare menu's (Findlater & McGrenere, 2004).

Park vergelijkt in zijn onderzoek de efficiëntie van adaptieve en aanpasbare menu's voor desktoptoepassingen (Park, Han, Park & Cho, 2007). Verder geeft hij ook kritiek op voorgaande studies. Hij stelt dat veel van de studies gebaseerd zijn op onrealistische assumpties (Park et al., 2007). De onderzoekers in deze studies gingen ervan uit dat men kon weten welke items vaak geselecteerd zouden worden. Park stelt dat dit gegeven onbekend is tot het systeem in de praktijk wordt gebruikt. Park verwijst ook naar een eerder onderzoek waar het begrip "*concept drift*" wordt aangehaald (Webb, Pazzani & Billsus, 2001). *Concept drift* binnen adaptieve menu's houdt in dat het systeem een verandering in gebruiksfrequentie niet weergeeft totdat die door het systeem herkend wordt (Webb et al., 2001). Park poogt met zijn onderzoek om op een kwantitatieve manier een vergelijking te maken tussen aanpasbare en adaptieve menu's (Park et al., 2007). Park voerde een experiment uit met 32 gebruikers, waarvan de leeftijd lag tussen 18 en 27 jaar. De gebruikers werden in 2 gelijke groepen verdeeld. Deze groepen voerden de gebruikerstest uit met verschillende selectiefrequentieverdelingen (Park et al., 2007). In het experiment voerden de gebruikers selectieopdrachten uit op 4 menutypes:

- een **traditioneel menu**: dit is een typisch, statisch menu waarin de gebruiker geen aanpassingen kan maken.

- een **aanpasbaar menu**: de gebruiker kan dit menu kan door aangepassen door items in het menu te verslepen naar een nieuwe locatie.
- een **adaptief split-menu**: dit is een split-menu zoals beschreven in sectie 2.1.2, waarbij de inhoud van het bovenste deel van het menu afgestemd is op de selectiefrequentie van items binnen de laatste 50 selecties. Dit menu wordt na elke selectie aangepast.
- een **adaptief highlight-menu**: in dit menu worden de meeste frequent geselecteerde items in vetdruk weergegeven en niet verplaatst.

Deze menu's bevatten 60 verschillende zelfstandige naamwoorden uit 4 categorieën (lichaamsdelen, landen, dranken en sporten) (Park et al., 2007). De gebruikers voerden 50 selecties uit in elk van de menu's. Men hield de selectietijden en het aantal gemaakte fouten bij. Uit de resultaten van de experimenten bleek dat de gebruikers over het algemeen het snelste werkten met het aanpasbaar menu, gevolgd door het adaptief split-menu en het split-menu. Het traditioneel menu bleek het traagste te werken. Er bleek geen significant verschil te bestaan in het aantal gemaakte fouten bij het gebruik van de verschillende menu's. De gebruikers hadden de minste voorkeur voor het traditionele menu. Er was echter geen verschil in gebruikersvoorkeur tussen de andere menu's (Park et al., 2007).
Park stelt in zijn paper dat hoewel het aanpasbaar menu uit zijn studie het meest geschikte menu leek, er wel een kost aan verbonden is. Gebruikers moeten dit menu eerst aanpassen voordat er een positief effect is op de selectiesnelheid met dit menu (Park et al., 2007). Hiermee herhaalt hij de bevindingen uit het werk van Mackay (1991).

Uit deze studies blijkt dat gebruikers over het algemeen de aanpasbare menu's verkiezen. Wel moeten we hierbij vermelden dat het gebruik van deze menu's een keerzijde heeft. In de eerste plaats vragen aanpasbare menu's meer ontwikkelingstijd (Stuerzlinger et al., 2006). Daarnaast moeten de gebruikers ook tijd en moeite steken in het naar hun hand zetten van het menu, alvorens ze er snellere selecties mee kunnen maken. Het alternatief van het gebruik van adaptieve menu's kent deze specifieke nadelen niet. Wel hebben deze menu's een andere tekortkoming: gebruikers ervaren geen leereffect bij het werken met deze menu's omdat ze bij elk gebruik anders zijn (Findlater & Gajos, 2009) (Gajos et al., 2006). Bovendien heeft onderzoek uitgewezen dat prestaties van gebruikers met adaptieve menu's niet sneller zijn dan de prestaties met een statisch menu (Findlater & McGrenere, 2004).

2.2.2 Bijhouden van het selectiepad: sequentieel versus uitbreidende weergave

Na de vergelijking van de eigenschappen van statische, adaptieve en aanpasbare menu's hebben we het in deze sectie als tweede algemene menueigenschap over de visualisatie van het selectiepad.

Zaphiris maakt de afweging tussen het gebruik van sequentiële tegenover uitbreidende menu's (Zaphiris, Shneiderman & Norman, 1999). Dit zijn beide hiërarchische menu's, die verschillen in de manier waarop voorafgaande keuzes aan de gebruiker weergegeven worden. Zo behouden uitbreidende hiërarchische menu's de volledige context van het selectiepad dat de gebruiker heeft afgelegd. De lijst met bovenliggende niveaus blijft zichtbaar en de niveaus *"klappen open"*, om de onderliggende mogelijkheden toe te laten. Daartegenover staan sequentiële menu's die enkel het huidige selectieniveau weergeven, al dan niet met een optie om terug naar een bovenliggende categorie te navigeren.

Zaphiris' experiment werd uitgevoerd met 22 studenten (Zaphiris et al., 1999). Deze studenten waren allemaal ervaren computergebruikers, die minstens 2 keer per week op het web actief waren. Bij dit experiment varieerde zowel het menutype (sequentieel of uitbreidend) als de diepte van dit menu (2, 3 en 4 niveaus), terwijl de 457 keuzes van het laagste niveau constant werden gehouden.
Gebruikers werden getest op reactietijden en zoekefficiëntie. Ze werden verzocht na het experiment een enquête in te vullen om hun ervaringen te quoteren. Uit de resultaten van de experimenten bleek dat opzoekingen met de uitbreidende menu's trager verliepen dan die met de sequentiële menu's en dat de diepte van de hiërarchie het minst invloed had op de sequentiële menu's.

Hierbij merken we op dat sequentiële menu's compacter weergegeven kunnen worden dan uitbreidende menu's. Dit kan de doorslaggevende factor zijn in de keuze tussen deze twee alternatieven wanneer we een mogelijke uitbreiding naar mobiele apparaten in gedachten moeten houden.

2.2.3 Diepte/breedte van het menu

We haalden eerder aan dat de hiërarchie die gebruikt wordt in het bestaande menu van de Belgische Geïntegreerde Politie om praktische redenen niet aangepast zal kunnen worden. Toch is het voor de volledigheid en de kadering van dit onderzoek nuttig om de problematiek rond de eigenschappen van deze classificatiestructuur te bespreken. Zo handelt een groot deel van het onderzoek dat verricht wordt in verband met het ontwerpen van menu's voor hiërarchieën over het zoeken naar de optimale afweging tussen de **diepte** en de **breedte**. Deze informatie kan nuttig zijn als men in de toekomst het menu zou willen uitbreiden of voor lezers die hun eigen menu willen ontwerpen.

2.2.3.1 Diepte/breedte en snelheid

Kiger voerde experimenten uit in verband met de optimale diepte van een hiërarchisch menu (Kiger, 1984). Hij analyseerde het aantal gemaakte fouten en de snelheid van gebruik bij een groep testpersonen. Deze personen kregen verschillende menu's aangeboden. De hiërarchieën van deze menu's hadden steeds dezelfde eindknopen, maar het aantal keuzemogelijkheden per niveau verschilde. Dit leidde tot een verschil in diepte in de algemene structuur van het menu. Hij merkte op dat het aantal gemaakte fouten en de zoektijd groter wordt naargelang de menustructuur dieper wordt.

Jacko bouwde verder op dit onderzoek (Jacko, Salvendy & Koubek, 1995). Men stelde dat de prestaties van gebruikers afhankelijk zijn van 3 variabelen: de dimensies van het menu, de complexiteit van de taak en de kennis van de gebruiker. Men voerde experimenten uit met 24 gebruikers waarbij men de snelheid van het uitvoeren van de taak en het aantal fouten analyseerde. Men toonde aan dat *expert users* relatief foutloos konden werken met zowel diepe als ondiepe menu's en dit bij taken met een hoge zowel als een lage complexiteit. *Novice users* daarentegen presteerden significant beter wanneer ze met ondiepe menu's en lage complexiteit werden geconfronteerd (Kiger, 1984).

De hypothese dat de complexiteit van de opzoeking toeneemt naarmate de diepte van een hiërarchisch menu toeneemt, blijkt ook te gelden voor het navigeren binnen web-gebaseerde platformen (Zaphiris, 2000).

De meeste onderzoekers zijn het er over eens dat een ondiepe hiërarchie de zoektijden en ervaren complexiteit van het menu verlagen. Gebruikers leken ook een voorkeur te hebben voor de ondiepe structuren. Er zijn echter stemmen die opgaan voor het gebruik van diepere en smallere hiërarchieën

in bepaalde omstandigheden. Landauer voerde een onderzoek naar de reactietijd die gebruikers nodig hadden om een correct item te selecteren uit een lijst van opties binnen een hiërarchisch menu (Landauer & Nachbar, 1985). De onderzoekers lieten de gebruikers werken op menu's met variërende breedtes. Ze kwamen tot de conclusie dat de reactietijd veel groter was bij bredere menu's. Ze haalden wel aan dat de gecumuleerde reactietijd, over het hele zoekproces, hoger is voor opzoekingen in smallere structuren. Paap spreekt over het *funneling*-effect (Paap & Roske-Hofstrand, 1986). Dit houdt in dat gebruikers stapsgewijs geleid worden naar de juiste keuze door hun steeds te laten kiezen tussen minder en vaak gerelateerde opties. Deze techniek is erop gebaseerd de gebruiker niet te overspoelen met keuzes, aangezien, volgens de onderzoeker, te veel keuzes kunnen leiden tot meer gemaakte fouten. We kunnen hieruit afleiden dat een smallere structuur sneller kan zijn als gebruikers snel verloren raken in de vele keuzes. Ze hebben meer tijd nodig om een keuze te maken (langer dan de reactietijd om de correcte keuze te selecteren). Het verkleinen van het aantal mogelijkheden per keuzeniveau kan dan een versnelling opleveren.

We merken dat we in ons menu-ontwerp een afweging zullen moeten maken. Willen we eerder nieuwere gebruikers gidsen naar juiste oplossingen door hen steeds te laten kiezen tussen weinig alternatieven of willen we *"cateren"* naar de *expert users* en de totale zoektijd beperken door het gebruik van een breder menu? Een alternatief zou de gulden middenweg tussen deze mogelijkheden kunnen zijn: een menustructuur zo opstellen dat hij op geen enkel punt overdreven breed is, wat immers slecht is voor nieuwere gebruikers. Aan de andere kant moeten we ervoor zorgen dat de diepte van de menustructuur niet te groot wordt, zodat de af te leggen weg beperkt blijft. Uit meerdere bronnen halen we dat de optimale breedte ergens tussen de vier en tien mogelijkheden per niveau ligt. De middenweg van acht opties wordt ook vaak aangeraden (Paap & Roske-Hofstrand, 1986; Jacko et al., 1995; Zaphiris, Kurniawan & Ellis, 2003; Geven, Sefelin & Tscheligi, 2006).

2.2.3.2 Diepte/breedte en leeftijd van de gebruikers

Zaphiris onderzocht de invloed van leeftijd op de prestaties van gebruikers in een hiërarchisch menu (Zaphiris et al., 2003). Hij voorspelde dat oudere gebruikers meer moeite zouden hebben met het navigeren in menu's en websites naar gelang hun visuele en motorische vaardigheden geleidelijk aan achteruit gaan. Dit werd bevestigd in zijn onderzoek. Bovendien kon hij mathematisch aantonen dat de optimale breedte van een menu onafhankelijk was van de leeftijd van de gebruikers. Hun zoeksnelheid zou natuurlijk wel verschillen

zelfs bij de optimale breedte. Men kwam uit dat optimale resultaten behaald werden tussen de vijf en tien keuzemogelijkheden per niveau.

2.2.3.3 Diepte/breedte en mobiele platforms

We hebben in de voorbije paragrafen reeds besproken dat een beperkte diepte een positief effect heeft op de navigatie doorheen hiërarchische menu's, voor zowel *expert* als *novice users* en voor zowel jongere als oudere gebruikers. We vragen ons nu af of deze hypotheses blijven gelden wanneer we het menu zouden overbrengen op mobiele apparaten. Het is immers de bedoeling van de politie om de web-gebaseerde software ook op tablets en mobiele telefoons beschikbaar te stellen.

De voornaamste verschillen tussen pc's en mobiele platformen zijn de **schermgrootte** en de **rekenkracht**. Voor het weergeven van menu's is er echter niet veel rekenkracht nodig, dus de enige beperkende factor waarmee we rekening moeten houden is het kleinere schermformaat. Dit maakt dat bv. uitklapbare menu's moeilijker weer te geven zijn. Er bestaan tal van mobiele alternatieven voor de weergave van menu's. We zullen hier echter niet dieper op ingaan aangezien we in ons onderzoek geen specifieke oplossing zoeken voor mobiele apparaten. We willen immers dat onze oplossing op zowel vaste als mobiele toestellen bruikbaar is.
In het kader van het eigen project vragen we ons wel af welk effect de grootte van het scherm heeft op de optimale structuur van het menu. Concreet stellen we ons de volgende vraag: "*Is het beter in de breedte of in de diepte te werken bij het weergeven van menu's op mobiele apparaten?*".

Parush voerde een experiment uit waarbij men 2 versies van een webplatform ontwikkelde (Parush & Yuviler-Gavish, 2004). Van de versies had de ene een diepe menu-structuur, de andere een brede en ondiepe structuur. Het experiment werd uitgevoerd met 96 gebruikers. Ze werden verdeeld in een groep die op een desktopcomputer werkte en een groep die op een mobiele telefoon werkte. De groepen werden dan nog eens verdeeld over de webplatformen met diepe en brede structuur. De gebruikers wisselden op bepaalde tijdsintervallen van groep. Men bestudeerde de snelheid van uitvoering en het aantal gemaakte fouten. Het aantal gemaakte fouten met zowel de diepe als de brede structuur bleek niet significant. Wel was er de succesfrequentie op de computers hoger dan op de mobiele apparaten. De snelheid waarmee de navigatie gebeurde, bleek wel afhankelijk te zijn van de structuur van het menu. Zo wees het onderzoek uit dat de brede structuur kortere navigatie-

tijden tot gevolg had, zowel op de pc als op de mobiele platformen (Parush & Yuviler-Gavish, 2004).

Geven stelde daarentegen dat, hoewel bredere menustructuren leiden tot kortere navigatietijden op de meeste platformen, het karakteristieke kleine scherm bij mobiele apparaten ertoe leidt dat een diepere structuur mogelijk beter geschikt is (Geven et al., 2006). Hij verwachtte ook dat gebruikers wel een brede structuur verkiezen op de desktop, maar deze voorkeur niet behouden op mobiele apparaten. Geven maakte voor zijn experiment onderscheid tussen *expert* en *novice users*. Men werkte met een groep van 15 testgebruikers die het systeem als *novice user* gebruikten. Hiervan kwamen er 14 later terug om dezelfde test uit te voeren als *expert users*. De gebruikers werkten met 4 verschillende hiërarchieën, variërend van erg smal tot erg breed. Tijdens de test werden het aantal fouten en het aantal toetsaanslagen bijgehouden voor analyse. Na de test werd aan de gebruikers gevraagd om de hiërarchieën te quoteren. Geven kon aan de hand van de resultaten van de experimenten enkel deels aannemen dat de smallere hiërarchie beter presteerde (Geven et al., 2006). Er was enkel een significant verschil tussen de erg smalle en de erg brede structuren, waarbij de erg smalle structuur het inderdaad beter deed. Gebruikers prefereerden de smallere structuur. Zoals men had verwacht, bleek het aantal aanslagen lager te zijn bij het gebruik van de smallere hiërarchie dan bij gebruik van de bredere. Dit was voornamelijk omdat de gebruikers moesten scrollen om alle opties van de brede structuur te kunnen zien. Het eindresultaat van deze studie was dat mobiele gebruikers beter presteerden met een smalle hiërarchie (4 tot 8 items per niveau) en dat er geen verschil was in voorkeur tussen *expert* en *novice users*.

We leerden dat hoewel veel studies over menu-ontwerp aanraden om brede structuren te gebruiken, mobiele gebruikers blijkbaar toch de voorkeur geven aan een smallere hiërarchie. Bij het ontwerp van een mobiele versie van software moet er dus wel degelijk overwogen worden om een volledig ander menu op te stellen dan wat gebruikt wordt voor de desktop-applicatie. Ook is het belangrijk op te merken dat gebruikers op de computer sneller kunnen navigeren dan op mobiele platformen. Het verschil in de resultaten van Parush en Geven is waarschijnlijk te wijten aan het feit dat Parush het aantal aanslagen niet in rekening heeft gebracht bij zijn analyse.

2.3 Conclusie

In dit hoofdstuk bespaken we, op basis van een literatuurstudie, bestaande menustijlen die de gebruiker toelaten om een keuze te maken uit een grote hoeveelheid opties. We bespraken ook een aantal onderliggende eigenschappen van verschillende menutypes. In de tabel 2.1 vatten we nu de voordelen en tekortkomingen van de besproken types samen in een verduidelijkende vergelijking. We overlopen ook enkele menu-eigenschappen in tabel 2.2.

Onze literatuurstudie toonde ook aan dat gebruikers met een menu dat het selectiepad sequentieel weergeeft sneller selecties maakten dan met een menu dat het selectiepad uitbreidend weergeeft (Zaphiris et al., 1999). We leerden eveneens dat men de hiërarchie van een menu beter in de breedte uitbreidt dan in de diepte, op zowel vaste als mobiele toestellen (Kiger, 1984; Parush & Yuviler-Gavish, 2004). Dit laatste wordt wel tegengesproken door een recenter onderzoek (Geven et al., 2006) waarbij het aantal aanslagen mee in rekening gebracht werd. Bij het ontwerp van een mobiele versie van software moet er dus wel degelijk overwogen worden om een volledig ander menu op te stellen dan wat gebruikt wordt voor de desktop-applicatie.

Menu-type	**Voordelen**	**Tekortkomingen**
Fisheye-menu's	• mogelijkheid om een grote hoeveelheid aan opties weer te geven; item plaatsbesparend	• niet alle opties zijn onmiddellijk leesbaar • de gebruiker moet de exacte naam van het gezochte item kennen • moeilijk om alle opties te overlopen • wordt niet gesmaakt door gebruikers

Split-menu's	• kleine groep frequent geselecteerde items makkelijk te zien • theoretisch snellere werking dan klassieke menu's	• in de praktijk niet consequent sneller dan klassieke menu's • werking sterk afhankelijk van selectiealgoritmes
Pie-menu's	• sneller dan klassieke menu's • nemen weinig schermruimte in • geen aparte schermpartitie nodig voor weergave menu	• beperkt aantal weergegeven items
Marking-menu's	• sneller dan klassieke menu's • nemen weinig schermruimte in • geen aparte schermpartitie nodig voor weergave menu • geoptimaliseerd voor expertgebruikers • groot aantal verfijningen die performantie vergroten en tekortkomingen wegwerken (bv. flower-menu's en zone-menu's)	• herkenning van markeringen soms problematisch (segmentatieprobleem) • beperkte breedte en diepte

Tabel 2.1: Vergelijking van de menutypes besproken in sectie 2.1. Deze tabel toont de positieve en negatieve aspecten van elk besproken menutype .

Eigenschap	Voordelen	Tekortkomingen
Adaptieve menu's	• frequent geselecteerde items duidelijk weergegeven	• onmogelijk de locatie van een item te voorspellen • in de praktijk geen significant verschil met statische menu's
Aanpasbare menu's	• gebruiker heeft controle over hoe het menu eruit ziet	• langere ontwikkelingstijd en moeite bij implementatie • veel gebruikers passen menu niet aan • snelheidswinst niet sluitend aangetoond

Tabel 2.2: Overzicht van de menu-eigenschappen besproken in sectie 2.2. Deze tabel toont de positieve en negatieve aspecten van de besproken eigenschappen.

3
Voorgestelde oplossing

In dit hoofdstuk bespreken we eerst onze bedenkingen en overwegingen over de eerder beschreven menu-oplossingen die courant zijn in een interface ontwerp. Dan leiden we uit onze gemaakte overwegingen een eigen oplossing af die de voordelen van deze oplossingen behoudt maar bovendien een aantal kleine aanpassingen voorstelt die onze oplossing hopelijk beter zal laten presteren in de gegeven context.

3.1 Overwegingen

In deze sectie bespreken we de overwegingen die we maakten bij het kiezen van een geschikte oplossing voor ons probleem. We overlopen de twee geschiktste kandidaten en sommen hun voor- en nadelen op. We voegen onze eigen commentaren hieraan toe en bespreken eventuele oplossingen voor bestaande problemen.

Uit de literatuurstudie halen we de twee kandidaten die ons het geschiktst lijken voor ons specifiek probleem. Enerzijds overwegen we de implementatie van een split-menu, anderzijds overwegen we het gebruik van een marking-menu.

3.1.1 Split-menu

De eerder besproken split-menu's (Sears & Shneiderman, 1994) zouden de frustratie, die het huidige menu bij sommige gebruikers opwekt, kunnen verhelpen door de veel gebruikte elementen apart weer te geven in een *quick select*. Op deze manier zouden de meeste gebruikers vermoedelijk sneller door het menu kunnen gaan. Het interessante aan de split-menu's is dat ze onafhankelijk van de andere oplossingen werken: men kan altijd een split-menu bovenop een bestaand menu maken. We maken toch enkele bedenkingen. Zo vinden we split-menu's vooral een manier om het inefficiënt gebruik van het bestaande menu te omzeilen. Dit verhelpt wel de symptomen van de problemen met het menu, maar niet de onderliggende oorzaak. Ook is het belangrijk te vermelden dat de selectiefrequentie van bepaalde items heel tijd- en locatie-afhankelijk is.

De problematiek van de tijd- en locatiegebondenheid van de selectiefrequentie kan eventueel opgelost worden door het menu adaptief te maken. Uit de literatuur leerden we echter dat het menu sneller zou moeten worden door de menu-items dynamisch te herschikken, maar dat gebruikers geen leereffect zullen ondervinden. Zo zullen ze nooit kunnen voorspellen op welke plaats in het menu een bepaald gekend item zich zal bevinden. De gebruiker moet dus iedere keer dat het menu zich aanpast opnieuw vertrouwd raken met de nieuwe ordening (Greenberg & Witten, 1985; Gajos et al., 2006; Findlater & Gajos, 2009). Dit lijkt ons een groot nadeel, aangezien we willen dat expert-gebruikers op een snelle manier met het systeem kunnen werken. Bovendien zou het implementeren van een split-menu ervoor zorgen dat de oplossing aanzienlijk meer schermruimte zou innemen dan het bestaande menu. Dit is niet wenselijk, aangezien we een mogelijke uitbreiding naar mobiele platformen in gedachten moeten houden.

3.1.2 Marking-menu

In Kurtenbach's marking-menu's heeft de gebruiker de keuze uit twee mogelijke voorstellingsmodi: *novice mode* en *expert mode* (G. P. Kurtenbach, 1993). Ofwel kan hij het radiale menu opvragen en hierbinnen een selectie maken (d.i. *novice mode*), ofwel kan hij markeringen tekenen zonder dit menu op te roepen (d.i. *expert mode*). We kunnen de gulden tussenweg bewandelen. Gebruikers zouden dan ofwel een eenvoudige selectie kunnen uitvoeren uit het opengeklapte menu ofwel *marks* kunnen hanteren, waarbij het menu tijdens het tekenen nog steeds beperkt zichtbaar is. Intuïtief lijkt het ons dat

deze extra begeleiding de *novice users* nog meer zal helpen om het gebruik van *marks* aan te leren en dat de extra informatie de expertgebruikers niet zal hinderen. Aan de andere kant zal het weergeven van dit menu extra tijd kosten wat de snelheidswinst voor expertgebruikers negatief kan beïnvloeden.

3.1.2.1 Probleem met hiërarchische marking-menu's

Een belangrijke bedenking is dat Kurtenbach (G. Kurtenbach & Buxton, 1993) stelde dat bij het gebruik van hiërarchische marking-menu's men de diepte van de hiërarchie best niet groter dan twee maakt (G. P. Kurtenbach, 1993). Het aantal gemaakte fouten gaat immers sterk omhoog wanneer men dieper in de structuur navigeert. Een reden voor deze fouten is dat gebruikers het menu niet zagen terwijl ze aan het tekenen waren en daardoor vaker foute items selecteerden, ook al hadden ze een goed idee van waar de items stonden. We kunnen dit probleem op een aantal manieren verhelpen. Eerst kunnen we het radiale menu groter maken, zodat de selectie-opties verder uit elkaar liggen. Daarnaast kunnen we werken met een dode zone rond het centrum van dit menu en komen we tot een selectieband, zoals ook beschreven in Kurtenbach's thesis (G. P. Kurtenbach, 1993). Door de combinatie van meer verspreide opties en de dode zone zou ambiguïteit minder parten spelen bij de selectie van een item. Toch blijft in de standaard implementatie het aantal menu-items beperkt. Hierbovenop komt ook dat de standaard implementatie van marking-menu's, met zijn *compound strokes*, veel schermruimte vraagt voor het maken van een selectie in een hiërarchisch menu. Dit kan tot tien keer zoveel horizontale schermruimte innemen als met lineaire menu's (Bailly et al., 2007).

3.1.2.2 Oplossingen voor problemen met hiërarchische marking-menu's

Een groot probleem met de standaardmarking-menu's is dat ze slechts bruikbaar zijn indien de hiërarchie een beperkte breedte en diepte heeft. We kunnen de maximale diepte van het menu opvoeren door gebruik te maken van *simple marks* en het menu om te vormen tot een multi-stroke menu. Uit de literatuur blijkt dat dit soort menu's superieur is aan de standaardmarking-menu's (Zhao & Balakrishnan, 2004). Ook neemt het multi-stroke-menu significant minder schermruimte in beslag bij het maken van selecties. Naast het verhogen van de diepte willen we de breedte van ons menu ook opvoeren. Onze hiërarchie is immers erg uitgebreid. De beste opties hiervoor lijken het

zone-menu (Zhao et al., 2006) en het flower-menu (Bailly et al., 2008). Hierbij lijken de flower-menu's eenvoudiger in gebruik aangezien de gebruikers geen extra actie moeten uitvoeren voor het definiëren van een oorsprong. Ook lijkt het concept van flower-menu's simpeler. Deze menu's kunnen daarenboven grotere breedtes aan. Hiertegenover staat dat zone-menu's eenvoudiger te implementeren lijken aangezien er weinig aandacht besteed moet worden aan de *gesture*-herkenning.

Belang van *novice mode*

Bailly wijst op het belang van de *novice mode* bij marking-menu's, omdat dit de manier is waarop alle gebruikers initieel met het systeem zullen werken (Bailly et al., 2007). De voorgestelde wave-menu's (Bailly et al., 2007) zijn interessant, maar we kunnen de gebruikte technieken moeilijk toepassen in de zone-menu's. Wel zijn ze bruikbaar bij flower-menu's. Wat we uit de resultaten van dit onderzoek zeker moeten meenemen is dat we bij onze eigen experimenten rekening moeten houden met de prestaties en het comfort van de gebruikers terwijl ze werken in *novice mode*.

Alternatieve versies van marking-menu's

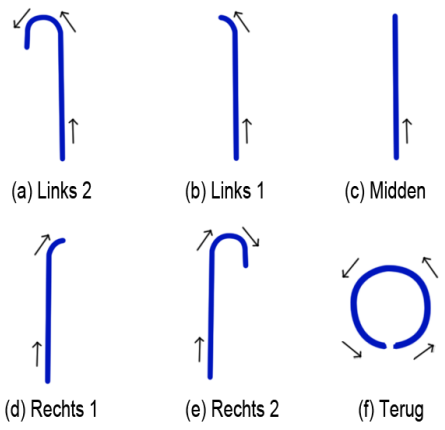
Het probleem met het gebruik van uitsluitend marking-menu's is dat de structuur van de gegeven hiërarchie bestaat uit een aantal geneste categorieën die leiden tot een groot aantal verschillende eindknopen per categorie. We kunnen het grootste deel van de hiërarchie wel voorstellen in een marking-menu of hiërarchisch taartmenu, maar we kunnen voor sommige categorieën niet alle eindknopen binnen dit menu inpassen, gezien de beperkte breedte en diepte van dit soort menu's (G. Kurtenbach & Buxton, 1993). We hebben meerdere teksten gevonden waarbij onderzoekers radiale menu's combineerden met lineaire menu's, om het probleem met de beperkte haalbare breedte van een radiaal menu te verhelpen. Zo vinden we patenten van Kurtenbach (1999) en Atkinson (1997) die deze combinatorische methode voorstellen. Ook beschrijft Kurtenbach met zijn *Hotbox* een interface die verschillende soorten menu's combineert (G. P. Kurtenbach et al., 1999) en gebruikt Roudaut concepten van marking-menu's in combinatie met andere menu's voor mobiele toepassingen (Roudaut, Bailly, Lecolinet & Nigay, 2009). De aanpak van het combineren van verschillende technologieën kan voor ons dus ook mogelijk een uitkomst bieden.

3.2 Menumodel

In dit onderdeel bespreken we de functionaliteit en eigenschappen van ons voorgesteld menu. In het volgend hoofdstuk bespreken we de implementatie van dit menu.

Op basis van de besproken overwegingen, kiezen we voor het model van het flower-menu (Bailly et al., 2008). We kiezen voor acht *within groups* in de acht windrichtingen. Deze groepen zullen elk vijf menu-items bevatten, elk aanspreekbaar door een unieke *gesture*. Op deze manier kunnen we elk niveau van de gebruikte hiërarchie voorstellen. De voorgestelde *gestures* zijn te zien in afbeelding 3.1. We voorzien in *gestures* voor:

- (a) de meest linkse selectie
- (b) de midden-linkse selectie
- (c) de middelste selectie
- (d) de midden-rechtse selectie
- (e) de meest rechtse selectie
- (f) het terugkeren naar het vorige niveau



Figuur 3.1: De voorgestelde *gestures* die gebruikt worden bij de bediening van het flower-menu. De *marks* zijn afgebeeld in het blauw. De tekenrichting wordt in het zwart weergegeven.

Hiernaast kiezen we er ook voor om de functionaliteit van de wave-menu's (Bailly et al., 2007) toe te voegen. Het menu zal, na een selectie, aan zijn buitenzijde het vorig bezochte niveau en de gemaakte selectie daarbinnen weergeven. Op deze manier heeft de gebruiker steeds een zicht op waar hij zich bevindt binnen het menu.

Tijdens zijn onderzoek merkte Bailly dat gebruikers moeite hadden met terug te keren naar een bovenliggend hiërarchisch niveau, wanneer ze werkten met multi-stroke-menu's (Bailly et al., 2007). Zoals te zien in afbeelding 3.1 lossen we dit op door een aparte *mark* te creëren voor het terugkeren naar een vorig niveau.

Uit onderzoek door Kiger (1984) en Jacko (1995) leerden we dat we ons menu beter zo ondiep mogelijk houden. Dit kan perfect met de eerdere keuze voor het flower-menu. We beperken de maximum diepte van het menu tot **drie niveaus**.

3.3 Conclusie

In dit hoofdstuk overwogen we twee mogelijke oplossingen voor het geschetste probleem: een adaptief split-menu en een waving flower-menu. Beide menu's laten ons toe om een grote hoeveelheid selectie-opties weer te geven en ze te ordenen. Uit onze literatuurstudie leerden we echter dat het split-menu verwarrend overkomt voor de gebruikers. Bovendien zou het implementeren van een split-menu ervoor zorgen dat de oplossing aanzienlijk meer schermruimte zou innemen dan het bestaande menu. Beide gevolgen van het implementeren van een split-menu leken ons niet wenselijk. We kozen daarom voor het gebruik van een flower-menu. Dit menu laat ons toe om de nodige informatie op een kleinere schermoppervlakte weer te geven (Bailly et al., 2008). Bovendien is dit een nieuw en interessant menutype. We kozen binnen dit flower-menu voor acht *within groups* in de acht windrichtingen. Deze groepen bevatten elk vijf menu-items, aanspreekbaar door een unieke *gesture*. Op deze manier kunnen we elk niveau van de gebruikte hiërarchie voorstellen. De voorgestelde gestures zijn te zien in afbeelding 3.1.

4 Implementatie

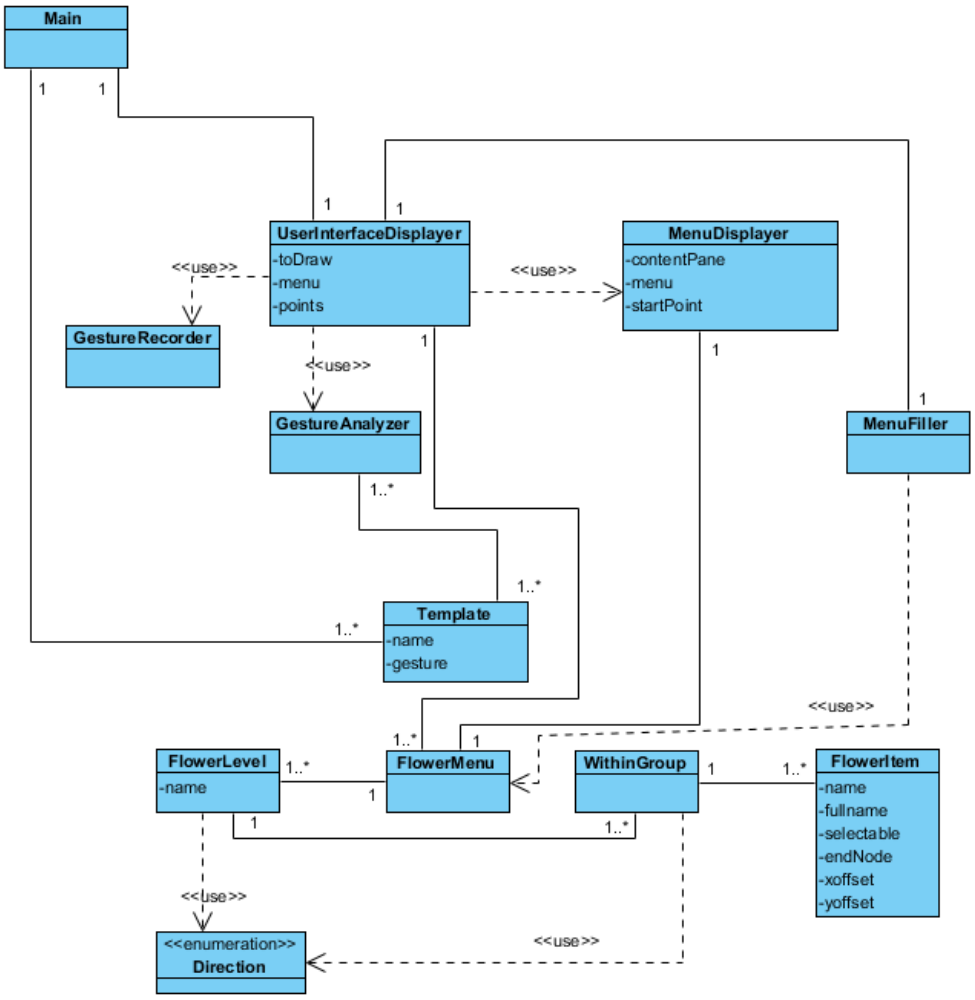
In dit hoofdstuk bespreken we de implementatie: werkwijze en gebruikte technologieën, gemaakte afwegingen, uitdagingen en hoe we daarmee zijn omgegaan.

4.1 Gebruikte technologieën, algoritmes en structuren

We kiezen om gebruik te maken van de programmeertaal Java voor onze implementatie. Eerder vermeldden we dat het uiteindelijke menu binnen een webbrowser zou moeten werken. Dit is met Java mogelijk door het wijzigen van de applicatie naar een Java web-applet of het omzetten (*porten*) van het programma naar een webgebaseerde taal (zoals javascript). Het is voor dit onderzoek echter enkel de bedoeling om te komen tot een werkende *proof-of-concept*-applicatie, waarvan de performantie en het gebruik geëvalueerd kunnen worden. We gebruiken de *Eclipse IDE voor Java* als ontwikkelingsomgeving.

De applicatie bestaat uit drie grote delen, elk met hun eigen verantwoordelijkheden. Het eerste deel spitst zich toe op het verwerken van de achterliggende data en deze aan de gebruiker weer te geven. Het tweede deel dient

voor het herkennen en verwerken van de door de gebruiker gemaakte *marks* (*gestures*). Het laatste deel bevat achterliggende programma-logica. Het bevat onder andere de weer te geven data en de structuur van het menu. De structuur van de applicatie is te zien in het UML-diagram in afbeelding 4.1. We volgden de aanbevelingen van Larman bij het opstellen van dit diagram (Larman, 2005). De werking van voorgenoemde programmadelen wordt in de volgende paragrafen beschreven.



Figuur 4.1: UML-diagram van de structuur van de applicatie.

4.1.1 Gesture-herkenning

Om de gestures te herkennen implementeren we een *$1 recognizer*, zoals beschreven door Wobbrock (2007). De *$1 Recognizer* werkt in verschillende stappen om uiteindelijk een *gesture* (*mark*) te herkennen. Eerst wordt er een pre-processing uitgevoerd op de getekende *gestures*, waarna ze vergeleken worden met een reeks voorbeeld-*gestures*, de zogenaamde *templates*. De *pre-processing* werkt als volgt: eerst wordt de getekende *gesture gesampled* naar een vast, door de programmeur gekozen, aantal punten. Hierna wordt deze puntenverzameling geroteerd, zodat haar indicatieve hoek op 0° valt. De indicatieve hoek is de hoek die gevormd wordt tussen de horizontale as en de vector, bestaande uit de centroïde van de puntenverzameling en het eerste punt uit deze verzameling (Wobbrock et al., 2007). Vervolgens wordt de puntenwolk geschaald, om binnen een door de ontwerper vastgestelde *bounding square* te vallen. Aangezien deze voorbereidende stappen ook uitgevoerd worden op de voorbeeld-*gestures*, de *templates*, kunnen we ervan uitgaan dat zowel de getekende *gesture* als de *templates* hetzelfde aantal punten hebben en binnen dezelfde *bounding square* vallen. Hierdoor is het een stuk eenvoudiger om de door de gebruiker gemaakte *mark* te vergelijken met de *templates*. Deze *templates* zijn elk opgeslagen in hun eigen tekstbestand en worden ingelezen bij het opstarten van het programma door de klasse *Main*. Het vergelijken van de invoer met de *templates* wordt gedaan door de afstand tussen de punten van de getekende *mark* en de punten van elke *template-mark* paarsgewijs te vergelijken. De *template* waarvoor de afstand het kortste is, wordt dan geselecteerd als best passende *gesture* (Wobbrock et al., 2007). De pseudo-code van het algoritme is te zien in afbeelding 4.2.

De uiteindelijke herkenning van gemaakte *gestures* gebeurt in twee stappen. Eerst bepaalt het programma aan de hand van de onbewerkte puntenverzameling (dus zonder pre-processing stappen uit te voeren) in welke richting de *mark* getrokken werd. Dit gebeurt door een punt in de tweede helft van de *gesture* te nemen en de richtingcoëfficient beginnend van de oorsprong van de *mark* tot dit punt te berekenen. Aan de hand van deze coëfficient wordt dan de windrichting van het gemaakte teken bepaald, en dus de geselecteerde *within group* van het menu. Hierna wordt de *gesture* doorgegeven aan de *$1 recognizer*, die dan uitwijst welk item uit de groep geselecteerd moet worden.

$1 GESTURE RECOGNIZER

Step 1. Resample a *points* path into *n* evenly spaced points.

```
RESAMPLE(points, n)
 1  I ← PATH-LENGTH(points) / (n − 1)
 2  D ← 0
 3  newPoints ← points_0
 4  foreach point p_i for i ≥ 1 in points do
 5    d ← DISTANCE(p_{i-1}, p_i)
 6    if (D + d) ≥ I then
 7      q_x ← p_{i-1_x} + ((I − D) / d) × (p_{i_x} − p_{i-1_x})
 8      q_y ← p_{i-1_y} + ((I − D) / d) × (p_{i_y} − p_{i-1_y})
 9      APPEND(newPoints, q)
10      INSERT(points, i, q)   // q will be the next p_i
11      D ← 0
12    else D ← D + d
13  return newPoints
PATH-LENGTH(A)
 1  d ← 0
 2  for i from 1 to |A| step 1 do
 3    d ← d + DISTANCE(A_{i-1}, A_i)
 4  return d
```

Step 2. Rotate *points* so that their indicative angle is at 0°.

```
ROTATE-TO-ZERO(points)
 1  c ← CENTROID(points)   // computes (x̄, ȳ)
 2  θ ← ATAN (c_y − points_{0_y}, c_x − points_{0_x})   // for -π ≤ θ ≤ π
 3  newPoints ← ROTATE-BY(points, -θ)
 4  return newPoints
ROTATE-BY(points, θ)
 1  c ← CENTROID(points)
 2  foreach point p in points do
 3    q_x ← (p_x − c_x) COS θ − (p_y − c_y) SIN θ + c_x
 4    q_y ← (p_x − c_x) SIN θ + (p_y − c_y) COS θ + c_y
 5    APPEND(newPoints, q)
 6  return newPoints
```

Step 3. Scale *points* so that the resulting bounding box will be of *size*² dimension; then translate *points* to the origin. BOUNDING-BOX returns a rectangle according to (min_x, min_y), (max_x, max_y). For gestures serving as templates, Steps 1-3 should be carried out once on the raw input points. For candidates, Steps 1-4 should be used just after the candidate is articulated.

```
SCALE-TO-SQUARE(points, size)
 1  B ← BOUNDING-BOX(points)
 2  foreach point p in points do
 3    q_x ← p_x × (size / B_width)
 4    q_y ← p_y × (size / B_height)
 5    APPEND(newPoints, q)
 6  return newPoints
TRANSLATE-TO-ORIGIN(points)
 1  c ← CENTROID(points)
 2  foreach point p in points do
 3    q_x ← p_x − c_x
 4    q_y ← p_y − c_y
 5    APPEND(newPoints, q)
 6  return newPoints
```

Step 4. Match *points* against a set of *templates*. The *size* variable on line 7 of RECOGNIZE refers to the *size* passed to SCALE-TO-SQUARE in Step 3. The symbol φ equals ½(-1 + √5). We use θ=±45° and θ_Δ=2° on line 3 of RECOGNIZE. Due to using RESAMPLE, we can assume that *A* and *B* in PATH-DISTANCE contain the same number of points, i.e., $|A|=|B|$.

```
RECOGNIZE(points, templates)
 1  b ← +∞
 2  foreach template T in templates do
 3    d ← DISTANCE-AT-BEST-ANGLE(points, T, -θ, θ, θ_Δ)
 4    if d < b then
 5      b ← d
 6      T' ← T
 7  score ← 1 − b / 0.5√(size² + size²)
 8  return ⟨T', score⟩
DISTANCE-AT-BEST-ANGLE(points, T, θ_a, θ_b, θ_Δ)
 1  x_1 ← φθ_a + (1 − φ)θ_b
 2  f_1 ← DISTANCE-AT-ANGLE(points, T, x_1)
 3  x_2 ← (1 − φ)θ_a + φθ_b
 4  f_2 ← DISTANCE-AT-ANGLE(points, T, x_2)
 5  while |θ_b − θ_a| > θ_Δ do
 6    if f_1 < f_2 then
 7      θ_b ← x_2
 8      x_2 ← x_1
 9      f_2 ← f_1
10      x_1 ← φθ_a + (1 − φ)θ_b
11      f_1 ← DISTANCE-AT-ANGLE(points, T, x_1)
12    else
13      θ_a ← x_1
14      x_1 ← x_2
15      f_1 ← f_2
16      x_2 ← (1 − φ)θ_a + φθ_b
17      f_2 ← DISTANCE-AT-ANGLE(points, T, x_2)
18  return MIN(f_1, f_2)
DISTANCE-AT-ANGLE(points, T, θ)
 1  newPoints ← ROTATE-BY(points, θ)
 2  d ← PATH-DISTANCE(newPoints, T_points)
 3  return d
PATH-DISTANCE(A, B)
 1  d ← 0
 2  for i from 0 to |A| step 1 do
 3    d ← d + DISTANCE(A_i, B_i)
 4  return d / |A|
```

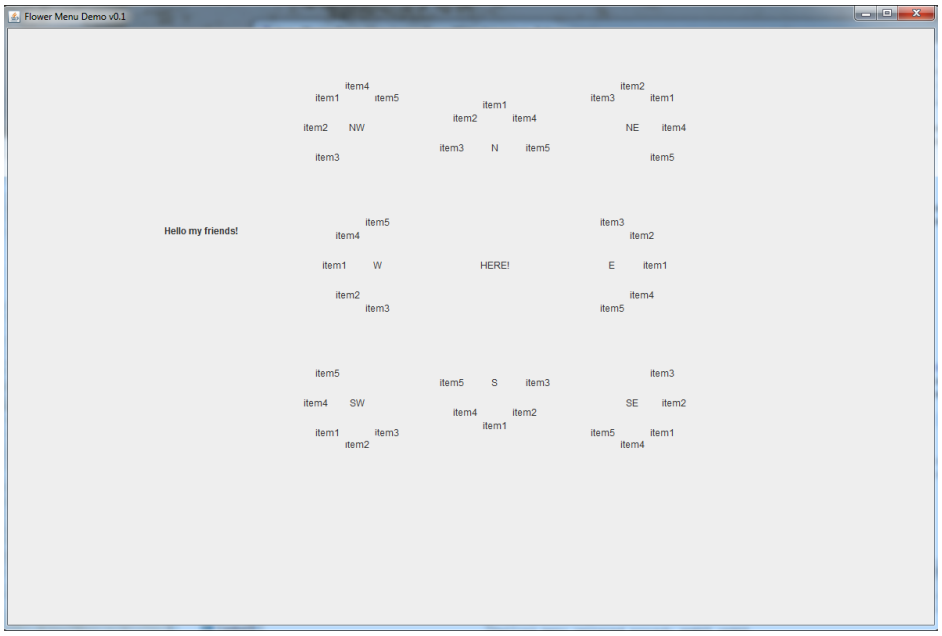
Figuur 4.2: Pseudo-code voor het *$1 recognizer*-algoritme. (overgenomen uit (Wobbrock et al., 2007))

4.1.2 Menustructuur

Om de uitwerking van het flower-menu te realiseren hebben we een aantal klassen uitgewerkt. Ze zijn te zien in afbeelding 4.1. Onze implementatie van het flower-menu is gebaseerd op een **boomstructuur** (Cormen, 2007).

De hoofdklasse van het menu heet *FlowerMenu* en ze bevat onder andere het huidige *FlowerLevel*, het basis *FlowerLevel* en de code voor het verwerken van een selectie binnen het menu. Wanneer het menu verteld wordt dat er een item geselecteerd werd en dit item een subniveau bevat, wordt het huidige niveau vervangen door het subniveau dat overeenkomt met de gemaakte keuze. De klasse *FlowerLevel* bevat een naam en voorziet plaats voor maximaal zeven *FlowerItems*. Dit aantal items laat ons toe alle items binnen een hiërarchisch niveau weer te geven, zonder de *within groups* te groot en onoverzichtelijk te maken. Een *FlowerLevel* bevat ook een referentie naar het bovenliggende niveau. Dit om de implementatie van de functionaliteit voor het terugkeren naar een vorig niveau eenvoudiger te maken. Een *FlowerItem* heeft als taak het voorstellen van een menuoptie. Deze items bevatten een naam en, indien de menu-optie geen eindknoop is, een referentie naar de kinderen van het *FlowerItem* (d.i. het onderliggende *FlowerLevel*).

Zoals bepaald in het menumodel bestaat het menu uit verschillende *within groups*. Deze groepen bevatten elk een aantal menuopties. In onze implementatie is er plaats voor acht *within groups* die elk tot zeven items kunnen bevatten. We hebben het aantal *within groups* en items overgenomen uit het werk van Bailly (2008). Daar werd aangehaald dat dit het maximaal aantal groepen en items is dat weergegeven kan worden zonder de interface te druk te maken (Bailly et al., 2008). Door tot zeven items per *within group* mogelijk te maken, in plaats van de vijf vermeld in het model, bereiden we de implementatie ook voor op toekomstige uitbreidingen van de hiërarchie. Het menu bevat naast de domeinlogica (namen en subniveaus) ook informatie over de plaatsing binnen de uiteindelijke weergave naar de gebruiker. We hebben ervoor gekozen om die dynamisch te genereren. We hebben eerst de oostelijke *within group* geprogrammeerd. Hierin hebben we de verschillende menu-items een vastgelegde plaats gegeven. Met deze groep als basis worden de zeven andere groepen gegenereerd, aan de hand van een opsomming die elke windrichting aan een bepaalde hoek koppelt. De items in de andere windrichtingen worden geroteerd volgens deze hoeken. Een eerste implementatie van een volledig *flower level* kan gezien worden in afbeelding 4.3.



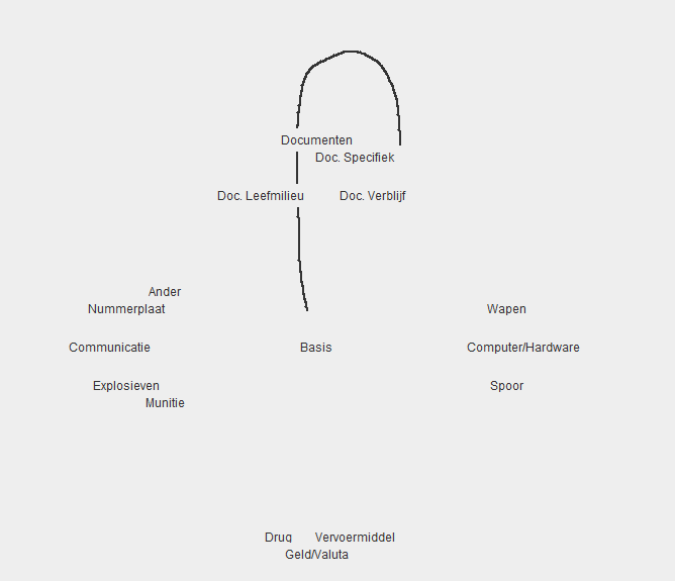
Figuur 4.3: Een eerste implementatie van een *FlowerLevel*. Men kan *within groups* zien in de verschillende richtingen, elk met vijf menu-items.

De inhoud van het flower-menu wordt verzorgd door de klasse *MenuFiller*. Deze klasse bevat alle commando's om de namen van de menu-items te plaatsen en om ze selecteerbaar te maken. Hiernaast zorgen deze commando's er ook voor dat de subniveaus aan de overeenkomstige items gekoppeld worden.

4.1.3 Weergave

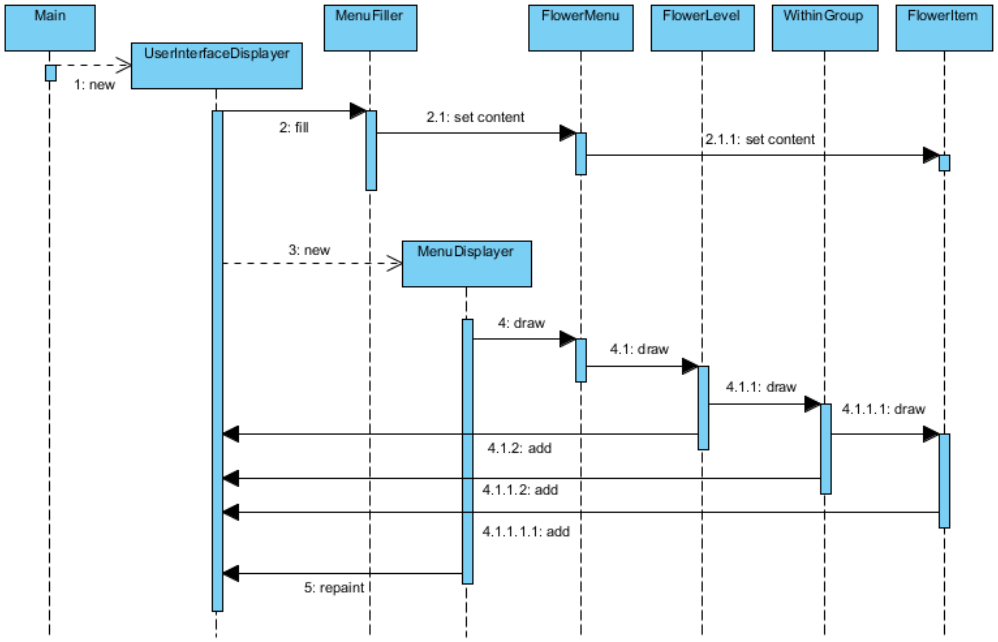
Het weergeven van de gebruikersinterface gebeurt op basis van het *Java Swing*-pakket. We hebben bij de implementatie gebruikt gemaakt van de handleiding van Zukowski (Zukowski, 2005). De klasse *UserInterfaceDisplayer*, zoals te zien in afbeelding 4.1, is een subklasse van de Java *built-in* klasse *JPanel*. Dit wil dus zeggen dat deze klasse op zichzelf een weergavepaneel is. *UserInterfaceDisplayer* is verantwoordelijk voor het initialiseren en weergeven van het flower-menu. De displayer houdt een lijst bij van alle items die afgebeeld moeten worden. Daarnaast bevat de displayer twee *MouseListeners*, elk met hun eigen timer, die zorgen voor de respons op gebruikeracties. Eén van deze *MouseListeners* is verantwoordelijk voor het opslaan van de getekende *gesture*. De andere wordt pas geactiveerd als de gebruiker de muisknop 0.3 seconden ingedrukt houdt zonder de cursor van positie te veranderen. Deze laatste *MouseListener* geeft bij activatie een *draw()*-

commando door aan het flower-menu, waarna het menu zichzelf toevoegt aan de af te beelden items binnen de *UserInterfaceDisplayer*. Hiernaast zorgt de *UserInterfaceDisplayer* voor de terugkoppeling van de gemaakte markering. Deze terugkoppeling houdt in dat de gebruiker tijdens het uitvoeren van een *gesture* onmiddellijk kan zien wat hij getekend heeft. Een voorbeeld van deze functionaliteit is te zien in afbeelding 4.4.



Figuur 4.4: Het tekenen van een markering binnen de applicatie. Het systeem geeft de getekende markering weer aan de hand van een zwarte lijn.

De *Main*-klasse is verantwoordelijk voor het initialiseren van een primair weergave-paneel. Dit paneel bevat de titel van de applicatie en een nieuw aangemaakte *UserInterfaceDisplayer*. Een sequentie-diagram, dat de werking van de weergave van het menu binnen onze applicatie illustreert, is te zien in afbeelding 4.5. Dit diagram werd opgesteld aan de hand van het boek van Larman (2005).

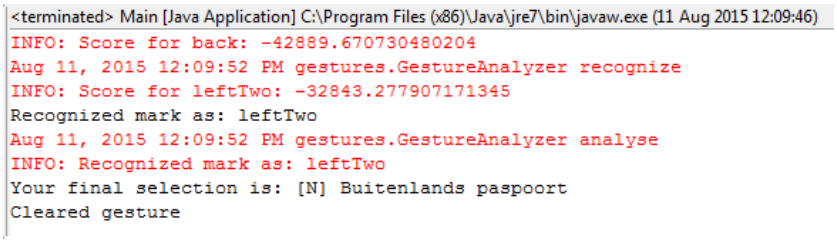


Figuur 4.5: Een sequentie-diagram dat de werking van de weergave van het menu binnen onze applicatie weergeeft.

4.2 Uitdagingen bij de implementatie

Probleem met afstanden van groepen/items en lange namen

Tijdens de implementatie merkten we een probleem op met de weergave van ons flower-menu. De namen van de menu-items bleken elkaar te overlappen. Dit had twee oorzaken: in de eerste plaats waren de afstanden tussen de verschillende *within groups* en de afzonderlijke menu-items te klein. Het tweede probleem was dat de namen van sommige items simpelweg te lang waren om volledig weergegeven te worden in het menu. De oplossing lag voor de hand. We hebben de spatie tussen de verschillende menu-items vergroot. Daarnaast werkten we in de laatste versie van de implementatie met afgekorte namen voor de langere menu-items. Zodra de gebruiker ze selecteert krijgt hij de volledige naam in het terugkoppelingspaneel van de *Eclipse IDE* (d.i. de *console*). Afbeelding 4.6 toont een voorbeeld van deze terugkoppeling.

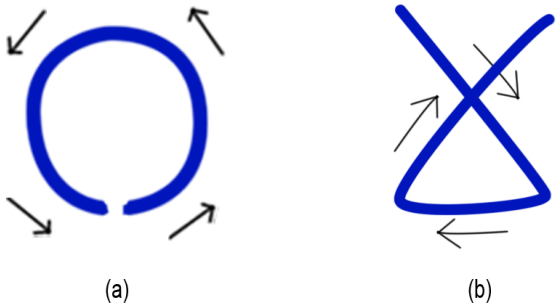


Figuur 4.6: Het terugkoppelingspaneel van de *Eclipse IDE*.

Probleem met herkenning

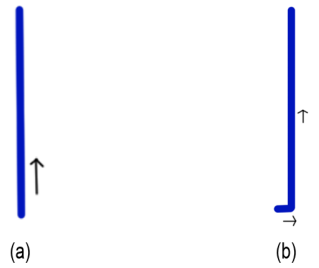
We merkten ook dat er een probleem was met de herkenning van sommige *marks*. Vaak werd een verkeerde herkenning gemaakt (een andere *mark* dan de doel-*mark* werd herkend), of werd een *gesture* niet herkend. We losten het eerste deel van dit probleem, nl. de incorrecte herkenning, op door het aantal *gesampelde* punten te verhogen. Het niet herkennen van *gestures* bleek zich enkel voor te doen bij het symbool om terug te gaan in de selectie. We hadden bij het verwerken van de *mouse events* ingesteld dat de *gesture*-herkenning pas in werking mocht treden als de muisaanwijzer daadwerkelijk van positie veranderd was. Dit betekende dat de *gesture* die we gekozen hadden voor het terugkeren in de selectie, zijnde een cirkel, nooit geregistreerd zou worden als hij volledig getekend werd. De enige mogelijkheid om dit probleem te verhelpen, zonder de werking van de herkenner te moeten aanpassen, was het veranderen van het gebruikte symbool. We besloten een “kruis"te gebruiken

als nieuw symbool, zoals te zien in afbeelding 4.7. Na deze aanpassingen werden de meeste *gestures* correct herkend.



Figuur 4.7: De oude (a) en nieuwe (b) *marks* voor het terugkeren in de selectie. De *mark* zelf is afgebeeld in het blauw. De tekenrichting wordt aangeduid door de zwarte pijlen.

We merkten hierna dat de gebruikte gesture-herkenner problemen had met het onderscheid maken tussen de *gesture* voor de selectie van het *middelste* item en die voor de selectie van de *eerste linkse/rechtse* items. Dit kwam waarschijnlijk doordat deze *gestures* heel erg op elkaar lijken. We hebben dit probleem deels opgelost door de *gestures* meer verschillend te maken. Dit leidde echter nog steeds tot foute herkenningen voor *gestures* om de middelste items te selecteren. We hebben er daarom voor gekozen om de *gesture* voor een middelste selectie aan te passen, zoals te zien in figuur 4.8. De nieuwe *gesture* heeft aan zijn basis een kleine uitwijking om het verschil met de andere *gestures* groter te maken. Zoals eerder beschreven werkt de initiële gesture-herkenning met de hoek waarin de ingegeven *gesture* getekend werd. Voor het bepalen van deze hoek gebruiken we de *inverse tangens*-functie van Java. Er deed zich echter een probleem voor wanneer de gebruiker een rechte noordelijke of zuidelijke mark maakt. De hieruit voorkomende hoek is dan in beide gevallen immers 0 °. We losten dit op door voor deze selecties de richtingscoëfficient van de *gesture* te bepalen. Daarna konden we het onderscheid tussen noordelijke en zuidelijke *marks* herkennen door te kijken naar het teken van de richtingscoëfficient. Indien dat negatief is, weten we dat de gebruiker een zuidelijke selectie maakte. In het geval van een positieve richtingscoëfficient tekende de gebruiker een noordelijke gesture.



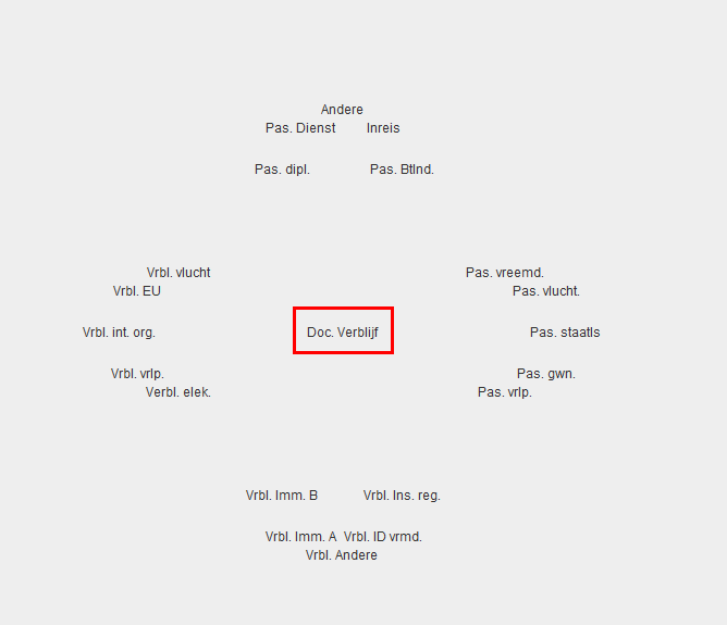
Figuur 4.8: De oude (a) en nieuwe (b) *marks* voor het selecteren van een middelste item. De *mark* zelf is afgebeeld in het blauw. De tekenrichting wordt aangeduid door de zwarte pijlen.

Probleem met wave-menu

Zoals beschreven in het model van het door ons voorgestelde menu, wilden we tevens een *wave*-functionaliteit implementeren. De werking hiervan is te zien in afbeelding 2.8. Na een initiële implementatie bleek echter dat er te weinig schermruimte was om de vorige menuniveaus zichtbaar te maken en het geheel toch overzichtelijk te houden. We hebben daarom ervoor gekozen om een beperktere functionaliteit in ons uiteindelijke menu op te nemen. We plaatsen nu in het centrum van het menu een label met de titel van het openstaande menu, geïllustreerd in afbeelding 4.9. Op deze manier weet de gebruiker steeds in welk subniveau hij zich bevindt zonder dat de interface overladen is met informatie.

Probleem met schuine richtingen

Oorspronkelijk was het de bedoeling om het menu *within groups* te geven voor acht windrichtingen. Nadat we experimenteerden met een vroege implementatie die deze acht groepen bevatte, merkten we dat de gekozen gestureherkenner problemen had met het herkennen van *gestures* in de *schuine* richtingen. De items in de noordoostelijke, zuidoostelijke, zuidwestelijke en noordwestelijke richtingen waren niet selecteerbaar. Na vele pogingen om de oorzaak van dit probleem te vinden, konden we het probleem niet verhelpen. We hebben in de finale versie van het menu ervoor gekozen om de problematische windrichtingen achterwege te laten en verder te gaan met slechts vier *within groups*. Dit leidde niet tot problemen met de hiërarchie aangezien geen enkel subniveau meer dan twintig items hoefde te bevatten.



Figuur 4.9: De weergave van het huidige subniveau binnen het flower-menu. Het label met de huidige locatie is in het rood aangeduid.

De klasse *MenuFiller* en trage implementatie

Onze oplossing om de inhoud van het menu te verzorgen was de klasse *MenuFiller* te gebruiken. Hierin worden alle items en subniveaus gedefinieerd. Het implementeren van deze items vergde echter erg veel werk. De totale grootte van deze klasse in de uiteindelijke implementatie bedroeg 1.215 lijnen code. Een greep uit deze implementatie is te zien in figuur 4.10. We hebben ervoor gekozen geen alternatieve implementatie uit te werken om dit probleem te verhelpen. Hierdoor hebben we echter veel tijd moeten spenderen aan het invullen van alle items in het menu. Een oplossing voor dit probleem zou zijn om de inhoud en de structuur van het menu in een apart bestand te noteren (bv. een XML-bestand) en dit in te lezen in de software. De software geeft het menu dan dynamisch zijn inhoud.

```
documentLevel.groups.get(Direction.NORTH).mid.name = "";
documentLevel.groups.get(Direction.NORTH).mid.selectable = false;
documentLevel.groups.get(Direction.NORTH).leftOne.name = "Arbeidskaart";
documentLevel.groups.get(Direction.NORTH).leftTwo.name = "Beroepskaart";
documentLevel.groups.get(Direction.NORTH).rightOne.name = "At. v. verlies";
documentLevel.groups.get(Direction.NORTH).rightOne.fullname = "Attest van verlies/diefst
documentLevel.groups.get(Direction.NORTH).rightTwo.name = "Andere";

documentLevel.groups.get(Direction.EAST).mid.name = "ID Belg Btnlnd";
documentLevel.groups.get(Direction.EAST).mid.fullname = "Identiteitskaart Belg verblijve
documentLevel.groups.get(Direction.EAST).leftOne.name = "ID militair";
documentLevel.groups.get(Direction.EAST).leftOne.fullname = "Militaire identiteitskaart",
documentLevel.groups.get(Direction.EAST).leftTwo.name = "ID nationaal";
documentLevel.groups.get(Direction.EAST).leftTwo.fullname = "Nationale Identiteitskaart",
documentLevel.groups.get(Direction.EAST).rightOne.name = "ID voorlopig";
documentLevel.groups.get(Direction.EAST).rightOne.fullname = "Voorlopige Identiteitskaar
documentLevel.groups.get(Direction.EAST).rightTwo.name = "ID kind";
documentLevel.groups.get(Direction.EAST).rightTwo.fullname = "Identiteitsdocument voor k
```

Figuur 4.10: Een greep uit de code van de klasse *MenuFiller*.

Probleem met Java-klasse *Point*

In een vroege versie van de implementatie gebruikten we de ingebouwde Java-klasse *Point*. Deze klasse heeft als doel het voorstellen van een punt in een twee-dimensionale ruimte. De *Point*-klasse kent slechts één constructor methode, met name *Point(int x, int y)*. We kunnen dus enkel gehele getallen ingeven als x- en y-waarden. Dit leidde tot problemen met de *resample*-methode van onze gesture-herkenner (afgebeeld in figuur 4.2). De *resample*-methode gaf bij elke uitvoering een ander aantal punten terug, in plaats van het vooraf vastgelegde doel. Daardoor konden de door de gebruiker ingegeven *gestures* niet vergeleken worden met de *templates*. We losten dit probleem op door een zelfgeschreven klasse te implementeren voor het voorstellen van punten: de klasse *Punt*. Deze neemt als invoerwaarden van de *constructor* wel reële getallen aan. Na deze aanpassing werkte de *resample*-methode zoals gepland.

Probleem met display-size

In een vroegere versie van de implementatie werd de weergave geregeld in een enkel *JPanel*. Dit paneel zorgde voor de weergave van het menu en de registratie van de *mouse events*. Bij het toevoegen van de terugkoppeling van de door de gebruiker getekende markering moesten we echter ons ontwerp aanpassen. We vervingen de eenlagige structuur van de weergavepanelen door een tweelagige structuur. Toen we de applicatie daarna testten bleek dat er niets gebeurde wanneer de gebruiker een actie uitvoerde. Na enig zoekwerk bleek dat de *mouse events* niet geregistreerd werden door het binnenste paneel omdat we bij het aanmaken van het binnenste paneel vergeten waren een dimensie in te stellen. Daardoor zat dit registratiepaneel wel bevat zat

in het hoofdscherm, maar het bedekte geen oppervlakte. Het probleem werd verholpen door de dimensie van het binnenste paneel in te stellen. Dit was een vrij beperkt en simpel probleem maar toch kostte het ons behoorlijk wat tijd voor we überhaupt snapten wat er aan de hand was.

4.3 Conclusie

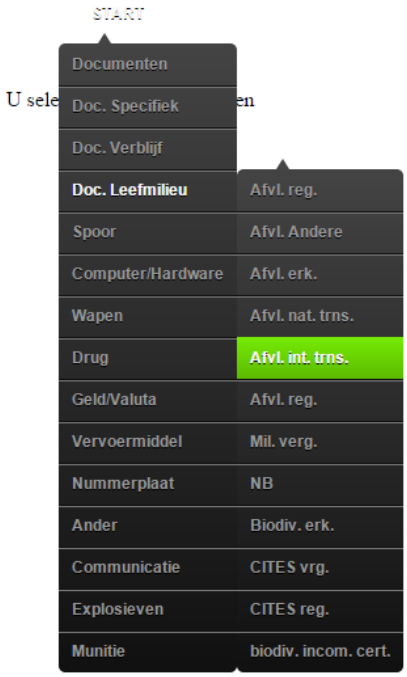
We bespraken de structuur van de applicatie (te zien in afbeelding 4.1) en schetsten enkele uitdagingen die we tegenkwamen tijdens de implementatie. Ten gevolge van deze uitdagingen waren we gedwongen om de uiteindelijke werking van ons flower-menu licht te herzien. In de uiteindelijke versie werken we met vier *within groups*, die elk vijf items bevatten. Het weergeven van het selectiepad werd beperkt tot het tonen van het subniveau waarin de gebruiker zich bevindt. Verder moesten we constateren dat enkele van de gekozen gestures niet de gewenste resultaten leverden. We moesten ze aanpassen om de werking van de applicatie te verbeteren. De oude en nieuwe gestures zijn te zien in afbeeldingen 4.7 en 4.8. Ondanks de problemen die we tijdens het implementatieproces ervoeren, zijn we er in geslaagd een werkende *proof-of-concept*-applicatie te maken.

5 Evaluatie

Aangezien het ons doel was om de bruikbaarheid van het selecteren van een voorwerpstype te verhogen moeten we de bruikbaarheid van ons voorstel verifiëren. We voerden een experiment uit met reële testpersonen waarbij we de snelheid en de accuraatheid in verschillende menu's vergeleken. Daarnaast hebben we gepeild naar het oordeel van de gebruiker (*satisfaction* en *attitude*). In dit hoofdstuk bespreken we de resultaten van de uitgevoerde gebruikerstests en interpreteren we deze resultaten.

5.1 Opstelling

We hebben naast onze implementatie van het flower-menu ook een referentiemenu geïmplementeerd, zoals te zien in afbeelding 5.1. Dit referentiemenu is een klassiek dropdown-menu zoals degene die courant gebruikt worden in web-interfaces. De hiërarchie gebruikt voor het referentiemenu is identiek aan die van het flower-menu. Dit wil zeggen dat het referentie menu dezelfde diepte en breedte heeft als het flower-menu. De volgorde van de items is ook identiek. Het referentiemenu werd opgesteld in *php* en *css* en is toegankelijk via een webbrowser.



Figuur 5.1: Het referentiemenu zoals gebruikt in de gebruikerstests.

We hebben gebruikerstesten uitgevoerd met tien gebruikers, met een leeftijd tussen twintig en veertig jaar. Deze personen hebben gemiddelde tot uitgebreide ervaring met het algemene gebruik van computers en met webinterfaces. Aan de testgebruikers werd verteld dat we een vergelijkende studie gingen maken tussen twee menutypes, zonder te vermelden dat het flower-menu een nieuw voorstel was, om het effect van eventuele *pleasers* tegen te gaan. We vroegen de gebruikers om twintig opdrachten uit te voeren op elk van de menu's. Deze opdrachten bestonden uit het selecteren van een bepaald item. Bij het uitvoeren van de opdrachten met het flower-menu werd aan de gebruikers gevraagd om elke opdracht twee keer uit te voeren. Bij de eerste uitvoering mochten ze het flower-menu oproepen (*novice mode*). Bij de tweede uitvoering moesten ze de selectie maken zonder het menu op te roepen (*expert mode*). Om er zeker van te zijn dat gebruikers nog wisten welke markeringen ze moesten maken om de gewenste items te selecteren, werden deze iteraties direct na elkaar uitgevoerd, dus niet in aparte sessies. Om vakkennis te simuleren beschikten de gebruikers steeds over het pad dat ze doorheen de hiërarchie moesten volgen. De testpopulatie werd in twee gelijke groepen opgedeeld. De ene groep ("groep 1") voerde eerst opdrachten

uit op het flower-menu en daarna op het referentiemenu. De andere groep ("groep 2") deed dit in omgekeerde volgorde, vanwege een mogelijk effect van de volgorde waarin de twee menu's gebruikt werden.

De tests werden uitgevoerd op een laptop met een computermuis. Tijdens het uitvoeren van de opdrachten werden de selectiesnelheid en het aantal gemaakte fouten geregistreerd en werd het scherm opgenomen door middel van de *Open Broadcaster Software* (OBS-developers, 2015). Na het uitvoeren van de opdrachten werd aan de gebruikers gevraagd een beperkte vragenlijst in te vullen. Hierin polsten we naar de eventuele voorkeur van de gebruikers voor één van de menu's, welk menu de gebruikers het meest geschikt vonden voor kleinere schermen en wat volgens de gebruikers de grootste tekortkomingen van beide menu's zijn. Tot slot gaven we de gebruikers de optie om nog andere commentaren of opmerkingen mee te geven.

5.2 Datacollectie

Tijdens het uitvoeren van de gebruikerstests werd de invoer van de gebruikers opgeslagen in logbestanden. Hierin werden per gebruiker de tijd die men nodig heeft om een selectie te maken (de selectietijd), de geselecteerde items en een aantal systeemvariabelen opgenomen. Aan de hand van deze bestanden achterhaalden we nadien de tijden en het aantal gemaakte fouten. Deze data werden bijgestuurd aan de hand van de opgenomen beelden. Zo konden we fouten die gemaakt werden als gevolg van problemen met de implementatie elimineren. Deze fouten zouden immers een vertekend beeld geven.

De vragenlijsten die de gebruikers invulden, werden bijgehouden voor latere analyse. Hieruit haalden we de cijfers over de voorkeuren van de gebruikers, hun mening over welk menu het best zou presteren op mobiele toestellen en hun opmerkingen.

5.3 Resultaten

Uit de gebruikerstest haalden we data over de selectiesnelheid en het aantal gemaakte fouten per menu. Bij de verwerking van de data werden de fouten afkomstig uit problemen met de implementatie buiten beschouwing gelaten. De gemiddelde selectiesnelheden per gebruiker zijn te zien in tabel 5.1 (snelheden in seconden). De data over het gemiddelde aantal gemaakte fouten per gebruiker zijn te zien in tabel 5.2. Dit gemiddelde werd berekend door het totaal aantal fouten te delen door het aantal opdrachten (20).

	Dropdown-menu	**Flower novice**	**Flower expert**	**Groep**
Gebruiker 1	7.65 s	8.77 s	3.09 s	1
Gebruiker 2	8.43 s	10.3 s	4.17 s	1
Gebruiker 3	7.74 s	9.38 s	3.87 s	1
Gebruiker 4	9.01 s	6.95 s	3.42 s	1
Gebruiker 5	6.29 s	6.54 s	3.18 s	1
Gebruiker 6	7.89 s	8.33 s	2.84 s	2
Gebruiker 7	7.44 s	6.76 s	3.52 s	2
Gebruiker 8	9.88 s	6.66 s	2.95s	2
Gebruiker 9	9.57 s	6.96 s	2.92 s	2
Gebruiker 10	7.95 s	7.69 s	3.49 s	2

Tabel 5.1: Tabel met de data van de selectiesnelheden in seconden, waarin de gemiddelde waarden over alle opdrachten per gebruiker weergegeven worden.

	Dropdown-menu	**Flower novice**	**Flower expert**	**Groep**
Gebruiker 1	0.05	0.1	0	1
Gebruiker 2	0.05	0.15	0	1
Gebruiker 3	0	0.25	0	1
Gebruiker 4	0.15	0.05	0	1
Gebruiker 5	0.05	0.05	0	1
Gebruiker 6	0	0	0	2
Gebruiker 7	0	0	0	2
Gebruiker 8	0	0	0.05	2
Gebruiker 9	0.05	0	0	2
Gebruiker 10	0.05	0.05	0	2

Tabel 5.2: Tabel met de data van het aantal gemaakte fouten, waarin de gemiddelde waarden over alle opdrachten per gebruiker weergegeven worden.

Uit deze data groeit ons vermoeden dat het flower-menu in *expert mode* significant sneller werkt dan zowel het flower-menu in *novice mode* als het dropdown-menu. De snelheden bij het gebruik van het flower-menu in *novice mode* en het dropdown-menu lijken ongeveer gelijk te zijn. We analyseren deze data aan de hand van statistische significantietests.

5.3.1 Significantietests van het verschil tussen de groepen

We zullen eerst bekijken of de volgorde waarin de testpersonen met de verschillende menu's hebben gewerkt een invloed had op de snelheden en het aantal gemaakte fouten. Dit doen we door de data uit tabellen 5.1 en 5.2 te onderwerpen aan een *ANOVA-test*. Als afhankelijke variabelen gebruiken we de benodigde tijden en het aantal fouten. Als onafhankelijke variabelen (factoren) gebruiken we de groepen waarin de gebruikers verdeeld werden. Als significantieparameter gebruiken we de waarde *0.05*. Dit betekent dat we de nulhypothese moeten verwerpen als het significantieniveau van de tests lager ligt dan deze parameter.

We testen eerst of er een significant verschil bestaat tussen de selectietijden van de twee groepen voor elk van de menu's. Onze nulhypothese luidt: H_0 : *Er is geen significant verschil tussen de selectietijden van beide groepen.* De resultaten van de *ANOVA*-test zijn te vinden in tabel 5.3. Geen van de significantieniveaus valt onder de significantiegrens. Aldus wordt de nulhypothese niet verworpen. We besluiten dat er **geen significant verschil** bestaat tussen de selectietijden van de verschillende gebruikersgroepen.

	F	Sig.
Dropdown tijden tussen groepen	1.151	0.315
Flower-menu (novice) tijden tussen groepen	1.982	0.197
Flower-menu (expert) tijden tussen groepen	2.486	0.154

Tabel 5.3: Tabel met de resultaten van de *ANOVA-test* die onderzoekt of er significante verschillen zijn tussen de selectiesnelheden van de verschillende gebruikersgroepen.

We testen ook of er een significant verschil bestaat tussen het aantal gemaakte fouten van beide gebruikersgroepen. Onze nulhypothese luidt:
H_0 : *Er is geen significant verschil tussen het aantal gemaakte fouten van beide groepen.*
De resultaten van de *ANOVA-test* zijn te vinden in tabel 5.4. We merken dat het significantieniveau van de gemaakte fouten bij het flower-menu in *novice mode* onder de significantiegrens van 0.05 valt. We moeten dus onze nulhypothese verwerpen en besluiten dat er een **significant verschil** bestaat in het aantal gemaakte fouten tussen beide groepen voor het flower-menu in *novice mode*. De testpersonen die eerst opdrachten uitvoerden met het flower-menu en daarna op het referentiemenu maakten meer fouten in *novice mode* dan testpersonen die eerst het referentiemenu gebruikten.

	F	Sig.
Dropdown fouten tussen groepen	2.133	0.182
Flower-menu (novice) fouten tussen groepen	8.067	0.022
Flower-menu (expert) fouten tussen groepen	1.000	0.347

Tabel 5.4: Tabel met de resultaten van de *ANOVA-test* die onderzoekt of er significante verschillen zijn tussen het aantal gemaakte fouten van de verschillende gebruikersgroepen.

5.3.2 Significantietests van het verschil tussen de menu's

We onderzoeken nu of er significante verschillen bestaan in de selectiesnelheden en het aantal gemaakte fouten tussen de verschillende menu's. Dit doen we door de data uit tabellen 5.1 en 5.2 te onderwerpen aan een *ANOVA-test*. Als afhankelijke variabelen gebruiken we de benodigde tijden en aantal fouten. Als onafhankelijke variabelen (factoren) gebruiken we de verschillende menu's. Als significantieparameter gebruiken we de waarde *0.05*. Dit betekent dat we de nulhypothese moeten verwerpen als het significantieniveau van de tests lager ligt dan deze parameter.

We testen eerst of er significante verschillen bestaan tussen de selectiesnelheden bij gebruik van de verschillende menu's. Onze nulhypothese luidt:
H_0 : *Er is geen significant verschil tussen de selectietijden van de menu's.*
De resultaten van de *ANOVA-test* zijn te vinden in tabel 5.5. We merken dat het significantieniveau van de selectietijden onder de significantiegrens van 0.05 valt. We moeten dus de nulhypothese verwerpen en besluiten dat

er een **significant verschil** bestaat tussen de selectiesnelheden van de verschillende menu's. Aan de hand van de *Bonferroni post-hoc test* gaan we na tussen welke menu's de verschillen liggen. De resultaten van deze post-hoc test zijn te vinden in tabel 5.6. Uit deze data leiden we af dat het flower-menu in *expert mode* significant sneller is dan het dropdown-menu en het flower-menu in *novice mode*. We zien ook dat er geen significant verschil bestaat tussen de selectiesnelheden binnen het flower-menu in *novice mode* en het dropdown-menu.

	F	Sig.
Tijden tussen groepen	71.737	0.000

Tabel 5.5: Tabel met de resultaten van de *ANOVA-test* die onderzoekt of er significante verschillen zijn tussen de selectiesnelheden bij gebruik van de verschillende menu's.

(I) Menu	(J) Menu	Gemiddeld verschil (I - J)	Sig.
Dropdown-menu	Flower-menu (novice)	0.352	1.000
	Flower-menu (expert)	4.841	0.000
Flower-menu (novice)	Dropdown-menu	-0.352	1.000
	Flower-menu (expert)	4.891	0.000
Flower-menu (expert)	Dropdown-menu	-4.841	0.000
	Flower-menu (novice)	-4.891	0.000

Tabel 5.6: Tabel met de resultaten van de *Bonferroni-test* die onderzoekt of er significante verschillen zijn tussen de selectiesnelheden bij gebruik van de verschillende menu's.

We analyseren het aantal gemaakte fouten tussen de verschillende menu's. We zoeken wederom naar het bestaan van een significant verschil. Onze nulhypothese luidt:
H_0 : *Er is geen significant verschil tussen het aantal gemaakte fouten bij het gebruik van de verschillende menu's.*
De resultaten van de *ANOVA*-test zijn te vinden in tabel 5.7. Geen van de significantieniveaus valt onder de significantiegrens. Aldus wordt de nulhypothese niet verworpen. We besluiten dat er **geen significant verschil** bestaat tussen het aantal gemaakte fouten bij het gebruik van de verschillende menu's.

	F	Sig.
Fouten tussen groepen	3.009	0.066

Tabel 5.7: Tabel met de resultaten van de *ANOVA-test* die onderzoekt of er significante verschillen zijn tussen het aantal gemaakte fouten bij gebruik van de verschillende menu's.

5.3.3 Resultaten van de vragenlijsten

Uit de vragenlijsten blijkt dat de voorkeur van 70% van de gebruikers uitgaat naar het dropdown-menu. De meeste gebruikers geven als reden hiervoor aan dat ze reeds omvangrijke ervaring met dit soort menu's hebben en dat het gebruik van het dropdown-menu erg natuurlijk aanvoelt. De gebruikers die een voorkeur hadden voor het flower-menu gaven in hun opmerkingen aan dat dropdown-menu's hen, vooral op mobiele toestellen, enorm stoort. Op de vraag welk menu het meest geschikt zou zijn voor kleinere schermen en mobiele platformen gaf een grote meerderheid van de gebruikers (80%) aan dat het flower-menu volgens hen het meest geschikt leek aangezien het menu minder schermruimte inneemt.

De commentaren leren ons dat de gebruikers vonden dat bij het het flower-menu de *gesture* heel precies uitgevoerd moet worden om tot een juiste herkenning te komen. Sommige gebruikers haalden ook aan dat het flower-menu een vrij hoge leercurve heeft. Volgens hen duurt het bij een uitgebreide hiërarchie een aanzienlijke tijd voordat een gebruiker markeringen *"blind"* (in *expert mode*) kan uitvoeren. Ze merkten ook op dat een betere aanduiding van welke *gestures* overeenkomen met welke items de implementatie vooruit zou helpen. Ze raadden aan hiervoor kleuren of voorbeeldlijnen te gebruiken.

Informele commentaar leerde ons dat gebruikers de weergave van de titel van het subniveau waarin ze zich bevinden enorm apprecieerden.

Daarnaast vermeldden een aantal gebruikers dat ze het potentieel van het flower-menu zeker inzien en dat met genoeg oefening gebruikers die dit menu hanteren sneller zullen werken dan gebruikers die een standaardmenu gebruiken.

5.4 Bespreking van de resultaten

De data uit onze gebruikerstest wijzen erop dat er wel degelijk een significant verschil bestaat tussen de snelheden waarmee gebruikers selecties kunnen uitvoeren in de verschillende menu's. De uitvoering blijkt het snelste te gaan met het flower-menu, gebruikt in *expert mode*. Daarnaast merken we dat er geen significante verschillen bestaan tussen het flower-menu in *novice mode* en het dropdown-menu. We moeten wel vermelden dat dit laatste een gevolg kan zijn van onze beperkte testgroep. De mogelijkheid bestaat dat een uitgebreider gebruikersonderzoek zou uitwijzen dat er wel degelijk een significant verschil bestaat tussen deze twee menu's. Ons onderzoek duidt vooral bestaande trends aan, die eventueel verder verfijnd kunnen worden in meer diepgaande analyses.

We merkten tijdens het analyseren van de data ook op dat er tussen de verschillende groepen een significant verschil bestaat in het aantal gemaakte fouten bij het gebruik van het flower-menu in *novice mode*. Dit komt waarschijnlijk door het bestaan van uitschieters in groep 1. De gebruikte statistische significantietests zijn immers gevoelig aan het bestaan van dergelijke uitschieters. Het feit dat dit significant verschil bestaat doet geen afbreuk aan de resultaten van de tests. Ons vermoeden wordt verder gesterkt door het uitblijven van andere significante verschillen tussen de fouten gemaakt door beide groepen. Indien de testvolgorde een invloed zou hebben op de prestaties van de gebruikers zouden we dit op meerdere plaatsen merken.

De analyse wees uit dat het aantal gemaakte fouten bij het gebruik van de verschillende menu's niet significant verschillen. Volgens dit resultaat kunnen de gebruikers even nauwkeurig werken met de verschillende menu's. Opnieuw moeten we erop wijzen dat het uitblijven van een significant verschil hier veroorzaakt kan zijn door de beperkte testgroep. We verwachten echter niet dat een uitgebreider onderzoek in een andere trend zou resulteren.

De voorkeur van gebruikers gaat echter nog steeds uit naar het klassieke dropdown-menu, hoewel men algemeen akkoord is dat het flower-menu beter zou werken voor kleinere schermen en mobiele dragers. Uit verdere gebruikersbevragingen bleek dat deze keuze vooral afkomstig is uit de vertrouwdheid die gebruikers reeds hebben met het dropdown-menu en niet zozeer met het onaangenaam zijn van het flower-menu. Uit de schriftelijke bevraging bleek dat sommige gebruikers liever met het flower-menu werken. Deze gebruikers gaven aan dat dropdown-menu's hen storen. Deze mening zou beïnvloed kunnen zijn door het feit dat deze gebruikers frequent met mobiele apparaten werken.

6

Conclusie en discussie

In dit hoofdstuk blikken we terug op het uitgevoerde onderzoek en trekken we conclusies uit onze bevindingen. We kaarten ook nog steeds bestaande problemen aan en geven een voorzet voor eventueel verder onderzoek.

Het doel van dit onderzoek was, op vraag van de Belgische Geïntegreerde Politie, het vinden van een oplossing die zowel nieuwe als ervaren gebruikers de mogelijkheid geeft om op een goed bruikbare wijze een object te selecteren uit een uitgebreide bestaande hiërarchie van objecten. We zochten naar een oplossing die zowel op vaste als mobiele toestellen bruikbaar is.

Ons onderzoek leverde interessante overwegingen op, die men in het achterhoofd kan houden bij het ontwerp van hiërarchische menu's. We leerden dat menu's met een hoge breedte over het algemeen beter presteren dan menu's met een hoge diepte (Kiger, 1984). Onze implementatie is op dit gegeven gebaseerd en hanteert daarom een vrij hoge breedte. We kozen voor een flower-menu omdat deze grotere breedte daarin mogelijk is en omdat gebruikers in eerdere studies een voorkeur hadden voor dit menu (Bailly et al., 2007). We hebben echter geen onderzoek gedaan naar het effect van een lagere breedte en hogere diepte op de prestaties die gebruikers leveren met ons voorgestelde menu.

Tot onze spijt moeten we besluiten dat onze *proof-of-concept*-implementatie wat te wensen overliet. We merkten immers uit de gebruikersbevraging dat veel personen het gevoel hadden dat de *gestures* te precies getekend moesten worden om herkend te worden. Dit is het gevolg van de problemen die we hebben gehad met het herkennen van de *gestures* voor het selecteren van het *middelste item* uit een bepaalde richting. We denken dan ook dat de gekozen *$1 gesture recognizer* niet optimaal is voor dit soort *gestures*. De herkenner moet immers accuraat zijn bij verschillende *gestures* die weinig van elkaar verschillen. Daarnaast zou meer gebruikerssturing bij het werken met het menu een stap vooruit zijn. We denken bijvoorbeeld aan voorbeeldlijnen naar elk item die gebruikers duidelijker maken welke *gesture* ze moeten uitvoeren om een bepaalde selectie te maken.

Uit de analyse van de data uit het experiment bleek dat er wel degelijk een significant verschil bestaat tussen de verschillende menu's. We stelden vast dat het flower-menu minstens even goed presteert als een standaard dropdown-menu. Het flower-menu presteerde zelfs aanzienlijk veel beter wanneer het in *expert mode* gebruikt werd. Dit leidt ons tot de conclusie dat ons voorgestelde menu een waardig alternatief kan zijn voor het bestaande, problematische menu en dat het in de toekomst gebruikt zou kunnen worden in nieuwe implementaties in plaats van een standaard dropdown-menu. We moeten hier wel bij vermelden dat uit de gebruikersbevraging bleek dat veel van de gebruikers een voorkeur hadden voor het klassieke dropdown-menu. Dit kwam vooral omdat ze vertrouwd zijn met dit courant menutype. De gebruikers meldden wel dat ze hun twijfels hadden bij de leercurve die gepaard gaat met het gebruik van een flower-menu. Toch zagen de testpersonen het potentieel van ons voorgestelde menu in. De gebruikers hebben echter geen ervaring opgedaan met het spontane gebruik van het flower-menu. Uit het onderzoek van Kurtenbach weten we dat in een reële omgeving de *transition component* van het gebruik van het flower-menu ook een plaats zou krijgen (G. P. Kurtenbach, 1993). Dit zou betekenen dat gebruikers de *gestures* die ze reeds kennen kunnen uitvoeren zonder het menu te openen en voor de ongekende bewegingen kunnen *"spieken"* door het menu op te vragen.

We besluiten dat het gebruik van flower-menu's zeker tot de mogelijkheden behoort in applicaties waar het merendeel van de gebruikers expertgebruikers zijn, die veel en lang met de applicatie zullen werken. Op deze manier wordt het leerproces immers spontaan uitgevoerd en zal het menu de beste resultaten halen. De tijdswinst voor expertgebruikers is reden genoeg om flower-menu's te implementeren: de negatieve effecten op novice-

gebruikers blijken erg mee te vallen. Daarnaast vermoeden we dat zulke menu's vlotter aangeleerd kunnen worden als de hiërarchie beperkt gehouden wordt. Ons onderzoek toont echter aan dat zelfs bij uitgebreide menustructuren het flower-menu degelijk presteert. We raden de mensen van de Belgische Geïntegreerde Politie dan ook aan om flower-menu's zeker te overwegen voor hun software. Deze aanbeveling wordt versterkt door het feit dat de politie overweegt om *PaCoS* op mobiele dragers beschikbaar te maken. Onze gebruikersbevragingen wezen immers uit dat veel mensen flower-menu's verkiezen voor hun mobiele apparaten. De bruikbaarheid van het flower-menu op een mobiel apparaat moet wel nog verder onderzocht worden. Onze resultaten en besluiten zijn gebaseerd op onze beperkte gebruikerstests. Dit wil zeggen dat we er in geslaagd zijn om een reeks mogelijke trends te bepalen die dan later geverifieerd kunnen worden in meer diepgaande onderzoeken.

Verder onderzoek zou gevoerd kunnen worden naar het gebruik van flower-menu's en de realisatie ervan. De *proof-of-concept*-implementatie moet nog meer in detail uitgewerkt worden. Daarnaast zou een onderzoek op een grotere testgroep kunnen bepalen of de trends die wij vastgesteld hebben binnen ons onderzoek behouden blijven. Het gebruik van flower-menu's op mobiele dragers is ook een interessante vraag, die in de toekomst onderzocht kan worden. De compatibiliteit van dit menu met de *gesture-based* bediening die gebruikers gewoon zijn op mobiele platformen zou mooie resultaten kunnen opleveren. Uitvoerigere testen met dezelfde testpersonen kunnen interessante bevindingen opleveren met betrekking tot het leerproces en de evolutie van hun voorkeur. Uit onze literatuurstudie leerden we dat het succes van een nieuw soort menu afhangt van de afweging tussen de voordelen van het menu tegenover de ervaren inspanning die de gebruikers moeten leveren bij het gebruik van dit menu (Gajos et al., 2006). Men kan het toegankelijker maken van de *novice mode* van het flower-menu verder onderzoeken. Een positieve gebruikerservaring is immers een belangrijk onderdeel van een geslaagde interface.

Nawoord

Bij het maken van deze thesis leerde ik niet alleen veel over menu-ontwerp, ik leerde ook doelgericht en consistent werken aan een project van grote omvang. Na een moeilijke start van wikken en wegen bij de verschillende instanties, ben ik toch tevreden dat ik dit onderzoek gestart heb. Ik ben erg ingenomen met de resultaten van het geleverde onderzoek, want ze gingen mijn verwachtingen te boven. Ik had nooit gehoopt dat mijn voorgestelde flowermenu even goed zou presteren als het referentiemenu in de gebruikerstests. De grootste hindernis bij het maken van deze scriptie was het halen van de zelf opgelegde deadlines. Deze deadlines waren een noodzakelijk kwaad. Ik denk immers dat ik anders nooit tot dit resultaat gekomen was.

Ik hoop met dit werkstuk een (bescheiden) wetenschappelijke bijdrage te hebben geleverd aan het onderzoek rond het gebruik van nieuwe soorten menu's. Een aangename gebruikerservaring aanbieden lijkt me persoonlijk nog steeds een groot struikelblok bij het ontwerp van veel hedendaagse software. Ik vond het daarom ook aangenaam om tijdens mijn literatuurstudie te merken dat er rond deze problematiek toch vrij veel wetenschappelijk onderzoek gedaan wordt. Ik heb ook beseft dat het maken van de perfecte interface nog steeds uiterst moeilijk is wegens de grote verschillen in de gebruikerspopulatie.

Referenties

Al-Omar, K. & Rigas, D. (2009). The effect of size of personalised menus on user satisfaction. In *Proceedings of the 11th wseas international conference on mathematical methods and computational techniques in electrical engineering* (pp. 322–327).

Bailly, G., Lecolinet, E. & Nigay, L. (2007). Wave menus: improving the novice mode of hierarchical marking menus. In *Human-computer interaction–interact 2007* (pp. 475–488). Springer.

Bailly, G., Lecolinet, E. & Nigay, L. (2008). Flower menus: a new type of marking menu with large menu breadth, within groups and efficient expert mode memorization. In P. Bottoni (red.), *Proceedings of the working conference on advanced visual interfaces* (pp. 15–22).

Bederson, B. B. (2000). Fisheye menus. In *Proceedings of the 13th annual acm symposium on user interface software and technology* (pp. 217–225).

Callahan, J., Hopkins, D., Weiser, M. & Shneiderman, B. (1988). An empirical comparison of pie vs. linear menus. In J. O'Hare (red.), *Proceedings of the sigchi conference on human factors in computing systems* (pp. 95–100).

Cormen, T. H. (2007). *Introduction to algorithms, second edition.* MIT press.

Elrod, S., Bruce, R., Gold, R., Goldberg, D., Halasz, F., Janssen, W., ... others (1992). Liveboard: a large interactive display supporting group meetings, presentations, and remote collaboration. In P. Bauersfeld, J. Bennett & G. Lynch (red.), *Proceedings of the sigchi conference on human factors in computing systems* (pp. 599–607).

Findlater, L. & Gajos, K. Z. (2009). Design space and evaluation challenges of adaptive graphical user interfaces. *AI Magazine*, *30*(4), 68.

Findlater, L. & McGrenere, J. (2004). A comparison of static, adaptive, and adaptable menus. In E. Dykstra-Erickson & M. Tscheligi (red.), *Proceedings of the sigchi conference on human factors in computing systems* (pp. 89–96).

Fitzmaurice, G., Khan, A., Pieké, R., Buxton, B. & Kurtenbach, G. (2003). Tracking menus. In *Proceedings of the 16th annual acm symposium on user interface software and technology* (pp. 71–79).

Gajos, K. Z., Czerwinski, M., Tan, D. S. & Weld, D. S. (2006). Exploring the design space for adaptive graphical user interfaces. In A. Celentano (red.), *Proceedings of the working conference on advanced visual interfaces* (pp. 201–208).

Geven, A., Sefelin, R. & Tscheligi, M. (2006). Depth and breadth away from the desktop: the optimal information hierarchy for mobile use. In *Proceedings of the 8th conference on human-computer interaction with mobile devices and services* (pp. 157–164).

Greenberg, S. & Witten, I. H. (1985). Adaptive personalized interfaces - a question of viability. *Behaviour & Information Technology*, *4*(1), 31–45.

Gutwin, C. & Fedak, C. (2004). Interacting with big interfaces on small screens: a comparison of fisheye, zoom, and panning techniques. In W. Heidrich & R. Balakrishnan (red.), *Proceedings of graphics interface 2004* (pp. 145–152).

Hornbæk, K. & Hertzum, M. (2007). Untangling the usability of fisheye menus. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *14*(2), 6.

Jacko, J. A., Salvendy, G. & Koubek, R. J. (1995). Modelling of menu design in computerized work. In *Interacting with computers* (Dl. 7, pp. 304–330).

Karlson, A. K., Bederson, B. B. & SanGiovanni, J. (2005). Applens and launchtile: two designs for one-handed thumb use on small devices. In W. Kellogg, S. Zhai, C. Gale & G. C. Veer (red.), *Proceedings of the sigchi conference on human factors in computing systems* (pp. 201–210).

Kaufman, L. & Weed, B. (1998). Too much of a good thing?: identifying and resolving bloat in the user interface. In *Chi 98 conference summary on human factors in computing systems* (pp. 207–208).

Kiger, J. I. (1984). The depth/breadth trade-off in the design of menu-driven user interfaces. *International Journal of Man-Machine Studies*, *20*(2), 201–213.

Kurtenbach, G. & Buxton, W. (1993). The limits of expert performance using hierarchic marking menus. In S. Ashlund (red.), *Proceedings of the interact'93 and chi'93 conference on human factors in computing systems* (pp. 482–487).

Kurtenbach, G. P. (1993). *The design and evaluation of marking menus* (Academisch proefschrift). University of Toronto.

Kurtenbach, G. P. (1999, jul). *Display and control of menus with radial and linear portions.* Google Patents. (US Patent 5,926,178)

Kurtenbach, G. P. & Buxton, W. (1994). User learning and performance with marking menus. In B. Adelson, S. Dumais & J. Olson (red.), *Proceedings of the sigchi conference on human factors in computing systems* (pp. 258–264).

Kurtenbach, G. P., Fitzmaurice, G. W., Owen, R. N. & Baudel, T. (1999). The hotbox: efficient access to a large number of menu-items. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 231–237).

Landauer, T. K. & Nachbar, D. W. (1985). Selection from alphabetic and numeric menu trees using a touch screen: breadth, depth, and width. In L. Borman & B. Curtis (red.), *Chi '85 proceedings of the sigchi conference on human factors in computing systems* (pp. 73–78).

Larman, C. (2005). *Applying uml and patterns: an introduction to object-oriented analysis and design and iterative development.* Pearson Education India.

Mackay, W. E. (1991). Triggers and barriers to customizing software. In S. B. Robertson, G. M. Olson & J. R. Olson (red.), *Proceedings of the sigchi conference on human factors in computing systems* (pp. 153–160).

Masui, T. (1998). Lensbar-visualization for browsing and filtering large lists of data. In E. Banissi, F. Khosrowshahi & M. Sarfraz (red.), *Ieee symposium on information visualization, 1998. proceedings.* (pp. 113–120).

McGrenere, J., Baecker, R. M. & Booth, K. S. (2002). An evaluation of a multiple interface design solution for bloated software. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 164–170).

Nielsen, J. (2003). *Usability 101: Introduction to usability.*

OBS-developers. (2015). *Open broadcaster software website.* Geraadpleegd op 16 juli 2015, van https://obsproject.com/.

Paap, K. R. & Roske-Hofstrand, R. J. (1986). The optimal number of menu options per panel. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *28*(4), 377–385.

Park, J., Han, S. H., Park, Y. S. & Cho, Y. (2007). Adaptable versus adaptive menus on the desktop: Performance and user satisfaction. *International journal of industrial ergonomics*, *37*(8), 675–684.

Parush, A. & Yuviler-Gavish, N. (2004). Web navigation structures in cellular phones: the depth/breadth trade-off issue. *International Journal of Human-Computer Studies*, *60*(5), 753–770.

Peffers, K., Tuunanen, T., Rothenberger, M. A. & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of management information systems*, *24*(3), 45–77.

Roudaut, A., Bailly, G., Lecolinet, E. & Nigay, L. (2009). Leaf menus: Linear menus with stroke shortcuts for small handheld devices. In

Human-computer interaction–interact 2009 (pp. 616–619). Springer.

Schaffer, D., Zuo, Z., Greenberg, S., Bartram, L., Dill, J., Dubs, S. & Roseman, M. (1996). Navigating hierarchically clustered networks through fisheye and full-zoom methods. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *3*(2), 162–188.

Sears, A. & Shneiderman, B. (1994). Split menus: effectively using selection frequency to organize menus. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *1*(1), 27–51.

Selvidge, P. (2002). Menu design: to adapt, or not to adapt. *Usability News*, *4*(1).

Smith, S. L. & Mosier, J. N. (1986). *Guidelines for designing user interface software*. Mitre Corporation Bedford, MA.

Spence, R. & Apperley, M. (1982). Data base navigation: an office environment for the professional. *Behaviour & Information Technology*, *1*(1), 43–54.

Stone, D., Jarrett, C., Woodroffe, M. & Minocha, S. (2005). *User interface design and evaluation*. Morgan Kaufmann.

Stuerzlinger, W., Chapuis, O., Phillips, D. & Roussel, N. (2006). User interface façades: towards fully adaptable user interfaces. In *Proceedings of the 19th annual acm symposium on user interface software and technology* (pp. 309–318).

Tapia, M. A. & Kurtenbach, G. (1995). Some design refinements and principles on the appearance and behavior of marking menus. In *Proceedings of the 8th annual acm symposium on user interface and software technology* (pp. 189–195).

Webb, G. I., Pazzani, M. J. & Billsus, D. (2001). Machine learning for user modeling. *User modeling and user-adapted interaction*, *11*(1-2), 19–29.

Weiser, M. (1995). The computer for the 21st century. *Scientific American*, *272*(3), 78–89.

Wieringa, R. (2009). Design science as nested problem solving. In V. Vaishanvi (red.), *Proceedings of the 4th international conference on design science research in information systems and technology* (p. 8).

Wiseman, N. E., Lemke, H. & Hiles, J. (1969). Pixie: A new approach to graphical man-machine communication. In *Proc. 1969 cad conf., iee conf. pub* (Dl. 51, p. 463).

Wobbrock, J. O., Wilson, A. D. & Li, Y. (2007). Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual acm symposium on user interface software and technology* (pp. 159–168).

Zaphiris, P. (2000). Depth vs breath in the arrangement of web links. In *Proceedings of the human factors and ergonomics society annual meeting* (Dl. 44, pp. 453–456).

Zaphiris, P., Kurniawan, S. H. & Ellis, R. D. (2003). Age related differences and the depth vs. breadth tradeoff in hierarchical online information systems. In *Universal access theoretical perspectives, practice, and experience* (pp. 23–42). Springer.

Zaphiris, P., Shneiderman, B. & Norman, K. L. (1999). *Expandable indexes versus sequential menus for searching hierarchies on the world wide web.* Geraadpleegd op 16 maart 2014, van http://hcil2.cs.umd.edu/trs/99-15/99-15.html.

Zhao, S., Agrawala, M. & Hinckley, K. (2006). Zone and polygon menus: using relative position to increase the breadth of multi-stroke marking menus. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 1077–1086).

Zhao, S. & Balakrishnan, R. (2004). Simple vs. compound mark hierarchical marking menus. In *Proceedings of the 17th annual acm symposium on user interface software and technology* (pp. 33–42).

Zukowski, J. (2005). *The definitive guide to java swing.* Apress.