



Vrije Universiteit Brussel

Faculteit Wetenschappen
Departement Informatica
Software Languages Lab

On the spot applicatie ontwikkeling op een iPhone aan de hand van workflows

Proefschrift ingediend met het oog op het behalen
van de graad van Master in de Ingenieurswetenschappen: computerwetenschappen

Denis Coppens

Promotor: Viviane Jonckers
Begeleiders: Niels Joncheere
Ragnhild Van Der Straeten

JUNI 2011



On the spot applicatie ontwikkeling op een iPhone aan de hand van workflows

Denis Coppens

JUNI 2011

Samenvatting

In dit document wordt een nieuwe programmeertaal en -omgeving voorgesteld om het mogelijk te maken voor niet-programmeurs om een applicatie on the spot te bouwen op een smartphone.

In de voorgestelde programmeertaal, een grafische workflowtaal, is het mogelijk om de verschillende functionaliteiten, die beschikbaar zijn op een smartphone, te combineren tot een nieuwe applicatie.

De grafische workflowtaal bestaat uit bouwblokken (acties, gebeurtenissen en samengestelde gebeurtenissen) die verbonden kunnen worden aan de hand van verschillende patronen (sequentieel, parallel, conditioneel en iteratief patroon). Acties kunnen gecombineerd worden met guards en stopgebeurtenissen om een nieuw bouwblok te maken. Om data tussen de verschillende bouwblokken uit te wisselen, wordt er gebruik gemaakt van een globale omgeving.

Naast de programmeertaal, wordt er ook een programmeeromgeving voor smartphones voorgesteld. Deze programmeeromgeving laat toe om op een intuïtieve manier, aan de hand van multitouch gestures, een workflowapplicatie bouwen.

Een prototype van de voorgestelde programmeertaal en -omgeving is geïmplementeerd voor iPhones in Objective-C. Een eerste reeks scenario's is dan geïmplementeerd op de iPhone om het geheel te demonstreren.

Tenslotte is de voorgestelde programmeertaal en -omgeving uitgetest door verschillende testpersonen met een verschillende achtergrond. Jongeren vanaf 14 jaar tot oudere mensen bleken geen probleem te hebben om een opgedragen reeks scenario's te implementeren in de aangeboden workflowtaal en -omgeving.

Abstract

This document introduces a new programming language and environment that makes it possible for non-programmers to develop an application on the spot directly on their smartphone.

In the proposed programming language, a graphical workflow language, it is possible to combine the various functionalities available on a smartphone into a new application.

The graphical workflow language consists of building blocks (actions, events and composite events) that can be connected using different patterns (sequential, parallel, conditional and iterative). Actions can be combined with guards and stop events into a new building block. To transfer data between the different building blocks, a global environment was adopted.

Besides the programming language, a programming environment is proposed for smartphones. This programming environment allows the building of a workflow application, in an intuitive way; by using multitouch gestures.

A prototype of the proposed programming language and environment is implemented for the iPhone in Objective-C. A first set of scenarios has already been implemented on the iPhone to demonstrate the proposed programming language and environment.

Finally, the proposed programming language and environment is tested by several test persons with different backgrounds. Youngsters of only 14 years old and even elderly people seem not to have a problem to implement the presented set of scenarios in the proposed workflow language and environment.

Dankwoord.

Ik heb dit onderzoek kunnen realiseren met de zeer gewaarde hulp van

- Professor Viviane Jonckers, mijn promotor
- Niels Joncheere, mijn begeleider
- Ragnhild Van Der Straeten, mijn begeleider
- Hugo Coppens, mijn vader

Voor hun constructieve opmerkingen en hulp.

Inhoudsopgave

1	Inleiding	8
1.1	On the spot ontwikkeling	9
1.2	Scenario's	9
1.3	Scope van de scenario's	10
1.4	Eindgebruiker ontwikkeling	12
1.4.1	General purpose talen	12
1.4.2	Script talen	13
1.4.3	Workflow talen	13
1.5	Doel van dit onderzoek	14
2	Achtergrondinformatie	15
2.1	Smartphones	15
2.1.1	Karakteristieken	15
2.1.2	Mogelijkheden	16
2.1.3	Ontwikkeling	17
2.1.4	Objective-C	17
2.1.5	Benodigdheden	18
2.1.6	Ontwikkelen van een applicatie	18
2.1.7	Uitbrengen van een applicatie	20
2.2	On the spot	20
2.2.1	Design principes	20
2.2.2	Gestalt psychologie	22
3	Verwant werk	23
3.1	Script talen	23
3.1.1	Tekstuele script talen	24
3.1.2	Grafische script talen	25
3.2	Workflow	28
3.2.1	Business Process Execution Language (BPEL)	29
3.2.2	Yet Another Workflow Language (YAWL)	30
3.2.3	Business Process Model and Notation (BPMN)	31
3.3	Modelleertalen	33
3.3.1	UML activiteitendiagram	33
3.3.2	UML toestandsdiagram	35
3.4	Grafische talen	36
3.4.1	Apple Automator	36
3.4.2	Lego Mindstorms	38
3.4.3	Yahoo Pipes	40
3.4.4	Applicatie uitvinder voor Android	42
3.5	Besluit	43

4	Workflow bouwblokken	44
4.1	Gebeurtenissen	44
4.2	Samengestelde gebeurtenissen	51
4.2.1	Niet schakelaar	53
4.2.2	Logische combinaties	54
4.3	Acties	55
4.3.1	Stopgebeurtenissen op acties	56
4.3.2	Gebeurtenissen als guards op acties	58
5	Control flow en data patronen	59
5.1	Start- en eindknopen	59
5.2	Sequentieel patroon	60
5.3	Parallele takken	61
5.3.1	Parallele start	61
5.3.2	Synchronisatie	62
5.4	Conditionele takken	63
5.4.1	Voorwaarden	64
5.4.2	Fusie	64
5.4.3	Uitgebreid voorbeeld	66
5.5	Iteratie	68
5.6	Variabelen	70
5.7	Browsen van reeds bestaande configuratieparameters van bouwblokken	72
5.7.1	Dynamische referentie	74
6	Gebruikersinterface	75
6.1	Bestaande gebruikersinterfaces	75
6.2	Workflow interface elementen	77
6.3	Interface elementen voor een samengestelde gebeurtenis . . .	79
6.4	Specifieke handelingen op de workflow	81
6.4.1	Aanpassen van de look en feel	81
6.4.2	Toevoegen van guards en stopgebeurtenissen op acties	81
6.5	Bouwen en bewerken van workflows aan de hand van multi- touchgestures	82
6.5.1	Voordeel	82
6.5.2	Toevoegen van bouwblokken	83
6.5.3	Toevoegen van patronen.	84
6.5.4	Verwijderen van een element	85
6.5.5	Verplaatsen van elementen	86
6.5.6	Knippen en plakken van elementen	87
6.5.7	Bouwen van een groep	88
6.5.8	Het uiteenvouwen van een groep	89
6.6	Gebruikersklassen	91
6.6.1	Ontwikkelaar	91

6.6.2	Workflow maker	91
6.6.3	Workflow gebruiker	92
6.7	Gouden regels	93
7	Implementatie	95
7.1	Implementatie van de workflow	95
7.1.1	Grafische weergave	95
7.1.2	Actie en gebeurtenis	96
7.1.3	Patronen en knopen	99
7.1.4	Functionaliteit van een workflow bouwblok	100
7.1.5	Voeg nieuw workflow bouwblok toe	101
7.2	Implementatie van samengestelde gebeurtenissen	102
7.2.1	Grafische weergave	102
7.2.2	Gebeurtenissen	103
7.2.3	EN en OF schakelaars	103
7.2.4	Functionaliteit van een gebeurtenis in een samengestelde gebeurtenis	104
7.3	Controleren van de uitvoering van een workflow	105
7.3.1	Verschillende bouwblokken die tegelijkertijd uitgevoerd worden	105
7.3.2	Uitvoeren van samengestelde gebeurtenissen	106
7.4	Design keuzes	108
7.4.1	Graf	108
7.4.2	Dubbel gelinkte lijsten	108
7.4.3	Dubbele dispatch	109
7.4.4	Enkel één gebeurtenis wordt getriggerd, observer patroon	110
7.4.5	Meerdere looks en feels, abstract factory patroon	111
7.4.6	Workflow elementen, composite patroon	113
7.4.7	Factory method patroon om functionaliteit aan een actie, gebeurtenis toe te voegen	114
7.4.8	State en visitor patroon voor de uitvoering van een workflow	115
7.5	Multitouch gestures	118
7.6	Meta model	120
8	Nagaan op het correct gebruik van variabelen	122
8.1	In een sequentieel patroon	122
8.2	In een parallel patroon	124
8.3	In een conditioneel patroon	125
8.4	In een parallel patroon met guards	127
8.5	Visitor patroon.	128

9	Validatie	129
9.1	Hoe	129
9.2	De gevraagde scenario's	130
9.3	Resultaten	131
10	Besluit & toekomstig werk	133
10.1	Samenvatting	133
10.2	Discussie	135
10.3	Kritische blik	135
10.3.1	Voordeel van guards	136
10.4	Toekomstig werk	138
10.4.1	Automatisch suggereren van guards	138
10.5	Tegelijkertijd uitvoeren van verschillende workflows	139

1 Inleiding

Smartphones worden meer en meer gebruikt in de hedendaagse wereld. Smartphones evolueren snel en bieden ook steeds meer extra technologieën zoals bewegingssensoren, multitouchschermen, gps, enzovoort. Naast de aanwezige technologieën, zijn er ook basisfunctionaliteiten aanwezig zoals het afspelen van muziek, het verzenden en ontvangen van SMS-berichten en telefoongesprekken. De processorkracht van de laatste modellen evenaart die van goedkope laptops. Naast al deze technologieën, hebben smartphones echter het nadeel een klein scherm te hebben, alsook minder geheugen en een kleinere opslagcapaciteit.

Om het gebruik van smartphones te vergemakkelijken, zijn er een hele boel applicaties ontwikkeld en er komen er elke dag nog bij. Deze applicaties hebben betrekking tot een brede waaier van domeinen zoals opvoeding, navigatie, spellen, opzoeken, tekstverwerkers, tijdschriften, enzovoort.

Dankzij de technologieën zoals Wi-Fi, 3G-netwerk, die in de smartphones aanwezig zijn, is het mogelijk om e-mails, notities, foto's enzovoort met het internet te synchroniseren. Dit laat toe dat smartphones informatie uitwisselen, nieuwe applicaties installeren en nog andere mogelijkheden bieden, zonder noodzakelijk op een desktop of ander toestel beroep te moeten doen. Om een nieuwe applicatie uit te brengen op smartphones, worden er online applicatiewinkels gebruikt. Deze winkels laten toe om gemakkelijk en onmiddellijk nieuwe applicaties op de smartphone te installeren.

Smartphones kunnen een onafhankelijk bestaan leiden, behalve bij het ontwikkelen van nieuwe applicaties. Het ontwikkelen van nieuwe applicaties gebeurt op een computer.

Voor het ontwikkelen van een nieuwe applicatie, is een goede voorkennis van programmeren noodzakelijk. In alle verschillende ontwikkelingsstadia moet er, zoals bij alle andere applicaties, bij een smartphone applicatie ook uitvoerig getest worden. Het testen van applicaties op smartphones gebeurt meestal via een smartphone simulator. Deze smartphone simulators kunnen echter niet alles aanbieden, zoals het nabootsen van een precieze beweging die het toestel ondergaat of het inkomen van een telefoongesprek, enzovoort. Hiervoor moet een smartphone ingeschakeld worden en moet de programmeur de applicatie na elke wijziging opnieuw op het toestel installeren en testen. Uit ervaring wordt opgemerkt dat dit een omslachtig proces is.

Eens de applicatie uitvoerig getest is, moet de applicatie openbaar geplaatst worden in de applicatiewinkel. Eens de applicatie aanvaard wordt in de winkel, kunnen gebruikers het programma downloaden en gebruiken.

Wij willen echter ook niet-programmeurs aanzetten om bestaande functionaliteiten te combineren in een nieuwe applicatie. De ontwikkeling van een nieuwe applicatie zou dan ook op de smartphone zelf gebeuren.

1.1 On the spot ontwikkeling

Het ontwikkelingsproces van applicaties voor smartphones is heel omslachtig. Als nieuwe applicaties rechtstreeks op de smartphone ontwikkeld worden, laat dit toe om onmiddellijk de applicatie te testen en te gebruiken zonder de noodzaak om via een computer te gaan.

Dit verlaagt de moeilijkheidsgraad voor het aanmaken en nadien het uitbrengen van nieuwe applicaties.

Ook is het aantrekkelijker om een nieuwe applicatie te bouwen op het toestel zelf (zittend in de zetel of op de trein). De gebruiker heeft het toestel in handen en weet wat het kan, hij kan dan onmiddellijk met deze technologieën experimenteren, zonder naar zijn computer te gaan.

Met on the spot ontwikkeling bedoelen we dus dat de applicatie op de smartphone zelf wordt geïmplementeerd en getest.

1.2 Scenario's

Om de scope van ons onderzoek af te bakenen, worden in deze sectie enkele scenario's beschreven. Deze scenario's zijn voorbeelden van applicaties die we, aan de hand van de nieuw ontwikkelde taal en omgeving, op het einde van dit onderzoek willen kunnen implementeren.

Scenario 1 :

Elke avond, op weg naar huis, moet Kim door een donker en verlaten steegje wandelen. Bij het wandelen door dit steegje, heeft ze altijd schrik dat ze overvallen zou worden. Ze stelt zich voor dat, als er plots iemand naar haar toe rent, ze geen tijd meer zou hebben om naar haar man te bellen. Ze zou het best handig vinden dat haar smartphone automatisch naar haar man belt, als ze haar toestel hevig schudt.

Scenario 2 :

Een rust- en verzorgingsinstelling vindt het belangrijk dat de oudere mensen in contact kunnen blijven met familie en vrienden. Hiervoor besloot de directie om aan alle inwoners een smartphone te geven. Dank zij deze smartphone, kunnen ze hun e-mails lezen, met hun familie bellen en op het internet surfen.

Ook ziet de instelling de mogelijkheid om deze smartphone te voorzien van een applicatie die, bij een valbeweging, onmiddellijk de verpleegster van wacht op de hoogte brengt door het zenden van een SMS. Ondertussen begint de smartphone ook muziek af te spelen om de eigenaar op de hoogte te brengen van de door de smartphone gevoelde valbeweging. Het kan echter zijn dat de gebruiker per ongeluk de smartphone heeft laten vallen. Indien

dit het geval zou zijn, kan de eigenaar, terwijl het muziekstuk afgespeeld wordt tijdens de eerste 20 seconden, door een knop op het scherm, aanduiden dat er niets ernstig gebeurd is. Eens de gebruiker op deze knop gedrukt heeft, wordt er een SMS gestuurd naar de verpleegsters om te melden dat alles in orde is.

Indien de gebruiker niet binnen de 20 seconden op de knop duwt, kan enkel de verpleegster de applicatie op de smartphone ontgrendelen. Dit is om zeker te zijn dat er medisch personeel bij de gevallen persoon komt.

Scenario 3 :

NoiseTube is een project dat statistieken maakt van het geluidsniveau in verschillende steden. Dit gebeurt aan de hand van geluidsopnameapparatuur die op specifieke plaatsen geluid opneemt en doorstuurt naar de centrale servers. Op de centrale servers worden deze data geanalyseerd en kan men het lawaaiverloop op de meetplaatsen in kaart brengen.

Tom is een medewerker aan het project NoiseTube. Hij bezit een smartphone met een ingebouwde GPS en geluidsrecorder. Hij zou graag aan de hand van zijn smartphone geluid willen opmeten en dan deze geluidsmeting om de 10 seconden doorsturen naar de NoiseTube server, samen met de locatie van de meting.

1.3 Scope van de scenario's

Deze 3 beschreven scenario's maken allen gebruik van een combinatie van verschillende technologieën.

Scenario 1 :

Bij scenario 1 is de door Kim gewenste applicatie een sequentie van 2 stappen, namelijk:

1. De smartphone wordt aan een hevige beweging blootgesteld
2. De smartphone belt de man van Kim op

Een onderscheid bestaat tussen stap 1 en 2. Bij stap 1, is de smartphone aan het wachten op het plaatsvinden van een gebeurtenis, waargenomen door de bewegingssensor. Bij stap 2, voert de smartphone een actie uit; deze actie, het bellen van een bepaald nummer is een functionaliteit die op de smartphone sowieso beschikbaar is.

Scenario 2 :

1. De smartphone voelt een hevige neerwaartse beweging.
Op dit moment gebeuren er 2 zaken tegelijkertijd, namelijk:
 - 1.1 De smartphone stuurt een SMS naar de verpleegster van wacht
 - 1.2 De smartphone speelt muziek.
Tijdens het afspelen van het muziekstuk, kunnen zich 2 mogelijkheden voordoen:
 - 1.2.1 De gebruiker drukt op de 'stop' knop op het scherm (het muziekstuk wordt automatisch stilgezet)
 - 1.2.1.1 Er wordt een SMS naar de verpleegster van wacht gestuurd om te melden dat alles in orde is.
 - 1.2.2 De 20 seconden zijn voorbij, zonder dat er op de knop wordt gedruwd (het muziekstukje wordt automatisch stilgezet)
 - 1.2.2.1 Het toestel vergrendelt zich.
 - 1.2.2.2 De smartphone wacht op ontgrendelen door de verpleegster van wacht.

Zoals in scenario 1, kunnen we dit scenario ook in stappen uitschrijven, hier voldoet een sequentiële aanpak echter niet. De verschillende takken moeten tegelijkertijd uitgevoerd worden en een beslissing moet genomen worden afhankelijk of een bepaalde gebeurtenis plaatsvindt of niet.

Ook in dit scenario, herkennen we verschillen tussen de stappen.

Bij stap 1 is de smartphone aan het wachten op het plaatsvinden van een gebeurtenis, waargenomen door een sensor. Bij stap 1.1, 1.2, 1.2.1.1 en 1.2.2.1 voert de smartphone een actie uit.

Bij stap 1.2.1 en 1.2.2.2 is de smartphone aan het wachten op een interactie van de gebruiker op het scherm. Bij stap 1.2.2 wordt er gewacht tot er 20 seconden voorbij zijn; ondertussen wordt er muziek afgespeeld.

Scenario 3 :

Bij scenario 3, is de door Tom gewenste applicatie ook een combinatie van de aanwezige technologieën in zijn smartphone, namelijk :

1. De smartphone neemt 10 seconden geluid op
2. De smartphone bepaalt zijn locatie.
3. De smartphone analyseert het opgenomen fragment en stuurt de geluidswaarden samen met de lokalisatiegegevens naar de NoiseTube server.

4. Hierna start de applicatie opnieuw met punt 1

We merken op dat stap 1, 2 en 3 acties zijn die ondernomen worden door het toestel.

Doorheen deze 3 scenario's merken we op dat we verschillende toepassingen kunnen ontwikkelen door het samenstellen van verschillende bouwblokken. We merken op dat er een verschil bestaat tussen de bouwblokken zelf, namelijk:

1. Gebeurtenissen : de smartphone is aan het wachten op een gebeurtenis die zich voordoet. Deze gebeurtenissen kunnen sensor gebonden, tijdsgebonden, gebruikersinteracties, en zo verder, zijn.
2. Acties : de smartphone voert een actie uit; de duur van de actie is afhankelijk van hoe lang de smartphone erover doet om de actie uit te voeren. Een actie kan bijvoorbeeld, speel muziek, neem geluid op en stuur SMS zijn.

Voor een gedetailleerdere uitleg over acties en gebeurtenissen, wordt er verwezen naar sectie 4.

Door deze bouwblokken (acties en gebeurtenissen) te combineren, kan een applicatie gebouwd worden.

1.4 Eindgebruiker ontwikkeling

1.4.1 General purpose talen

Om niet-programmeurs toe te laten applicaties te ontwikkelen, moeten we een eenvoudige en intuïtieve programmeertaal en -omgeving aanbieden.

Het ontwikkelen van een programma, aan de hand van een general purpose taal, is te complex. De gebruiker moet eerst een geschikte taal aanleren, wat op zich al geen gemakkelijke taak is, zelfs voor ervaren programmeurs. Daarnaast moet hij ook leren omgaan met alle bibliotheken die nodig zijn om de diverse technologieën op de smartphones te gebruiken. Voor elke sensor, zoals de bewegingssensor en lokalisatiesensor, is er een afzonderlijke bibliotheek.

Naast de hoge complexiteit van een general purpose taal, is het ook niet mogelijk om de applicatie op het toestel zelf te ontwikkelen. Het scherm en het toetsenbord aanwezig op het scherm zijn te klein om degelijk met een general purpose taal om te gaan.

1.4.2 Script talen

Omdat general purpose talen te complex zijn en teveel vrijheid bieden aan de programmeurs, zijn er script talen ontstaan. Script talen bieden dezelfde basis taal elementen als general purpose talen, waarbij ganse bibliotheken soms al geabstraheerd zijn aan de hand van functies; zo kan de programmeur sneller en gemakkelijker applicaties bouwen.

Script talen zijn talen die meestal werken met bestaande applicaties, hardware of het operating systeem. Deze script talen bieden de programmeur de functionaliteiten van de bestaande applicaties, hardware of bestuursstelsel. Deze functionaliteiten kunnen dan opgeroepen en gebruikt worden in de script taal. De meeste script talen zijn nog steeds tekstueel. Het programmeren op een smartphone met een klein scherm met een tekstuele taal is noch aantrekkelijk noch gemakkelijk voor de gebruiker. Ook is het nadeel van een tekstuele taal dat de eindgebruiker alle verschillende syntaxelementen moet kennen. Indien de gebruiker deze taal niet veel gebruikt, vergeet hij de exacte syntax. Door het vergeten van de syntax kan de gebruiker telkens ontmoedigd geraken.

Naast tekstuele script talen, bestaan er ook grafische script talen. Er bestaan bijvoorbeeld grafische script talen die het mogelijk maken om spelomgevingen te bouwen in een spelmotor (zie sectie 3.1.2).

Een grafische taal is gemakkelijker om te programmeren op een klein scherm; de eindgebruiker hoeft dan al niet meer te typen. De eindgebruiker hoeft dan enkel de verschillende blokken te slepen en te combineren naargelang zijn gewenste applicatie.

1.4.3 Workflow talen

Een andere grafische manier om bestaande functies te combineren voor het vormen van een applicatie, zijn workflows. Aan de hand van een workflow, kan men een applicatie bouwen door verschillende taken te combineren aan de hand van patronen. Deze patronen (workflow patterns) bieden een gestructureerde manier voor het combineren van individuele taken. Indien we terugdenken aan scenario één, kunnen we bijvoorbeeld de verschillende stappen combineren aan de hand van het sequentie patroon.

In de Lego Mindstorms ontwikkelingsomgeving (zie sectie 3.4) is aangetoond dat, aan de hand van deze patronen, 7-jarige kinderen in staat zijn om applicaties te bouwen.

Verschillende workflow talen zijn zo ontstaan om het toegankelijker te maken voor niet-eindprogrammeurs om zelf applicaties te bouwen. Voorbeelden zijn Automator door Apple en App Inventor door Google (zie sectie 3.4).

Indien we nu naar onze scenario's kijken, merken we dat we verschillende types blokken hebben, namelijk acties en gebeurtenissen. Om de scenario's op een gemakkelijke manier te ontwikkelen, moeten we deze bouwblokken

combineren en liefst op een zo gemakkelijk mogelijke manier.

Het bouwen van een workflow taal lijkt hiervoor een goed uitgangspunt, omdat het grafische aspect een aantrekkelijker vorm biedt voor de eindgebruiker en een workflow taal een lagere instapdrempel kan hebben.

1.5 Doel van dit onderzoek

Het doel van dit onderzoek is om aan niet-programmeurs een nieuwe programmeertaal en -omgeving aan te bieden voor smartphones. De ontworpen taal moet gemakkelijk overweg kunnen met de verschillende technologieën van de smartphones, zodat deze gebruikt kan worden voor uiteenlopende toepassingen. De taal moet flexibele combinatie-mogelijkheden toelaten, zodat gebruikers interessante applicaties kunnen bouwen. Verder moet deze taal programmeerbaar en bruikbaar zijn op de smartphone zelf.

Het onderzoek zal in volgende onderdelen verdeeld worden:

1. Workflow taal: de eerste stap bestaat eruit om een workflow taal te ontwerpen, die aan alle tot nu toe beschreven eisen voldoet: een flexibele, gemakkelijk te gebruiken "taal" voor on the spot ontwikkeling op smartphones. Hiervoor moet dus nagedacht worden over welke workflow patronen cruciaal zijn en hoe deze worden weergegeven en gebruikt op een beperkt scherm van een smartphone. Naast een nieuwe workflow taal zal er ook een aangepaste programmeeromgeving ontworpen worden om het mogelijk maken de workflow on the spot te bouwen. Voor het aanmaken, verwijderen, bewerken van de workflow in de programmeer omgeving zullen multitouch gestures gebruikt worden.
2. Infrastructuur: in een tweede fase zal de nodige infrastructuur moeten ontwikkeld worden om deze nieuwe taal te kunnen gebruiken en uitvoeren op een mobile device. Als target platform is er gekozen voor de iPhone van Apple. Er zal dus een grafische workflow interface gebouwd worden en een interpreter om workflows uit te voeren; dit zal gebeuren in Objective-C.
3. Demonstratie en proof of concept: in deze fase zal nagezien worden of de beschreven scenario's implementeerbaar zijn en uitgevoerd kunnen worden in de voorgestelde workflow taal.
4. Validatie: in de laatste fase zal de aangeboden programmeertaal en -omgeving, op beperkte schaal, uitgeprobeerd worden door niet programmeurs. Deze zullen, na een introductie, een reeks scenario's voorgeschoteld krijgen en zelfstandig de corresponderende applicatie moeten bouwen.

2 Achtergrondinformatie

2.1 Smartphones

Smartphones worden met de dag populairder. Vroeger diende een GSM enkel om te bellen en om berichten te versturen. Hiernaast had de gebruiker verschillende toestellen voor verschillende doeleinden zoals een fototoestel, een muzikspeler, een navigatiesysteem enzoverder. Nu worden deze verschillende functies allemaal in één toestel samengebracht. Een dergelijk toestel heet een smartphone.

Tegenwoordig is de markt verdeeld tussen verschillende smartphone operating systemen marktleiders, zoals Microsoft Windows Mobile, Google Android, iOS en BlackBerry.

2.1.1 Karakteristieken

Smartphones bezitten tegenwoordig over bijna evenveel rekenkracht als goedkope laptops. Bovendien bezit een smartphone veel meer technologieën dan de meeste computers, zoals bewegingssensoren, GPS, een cellulaire connectie... Al deze verschillende technologieën kunnen in de applicaties gebruikt worden.

Smartphone applicaties kunnen als verkleinde computerapplicaties gezien worden. Deze applicaties bieden dezelfde mogelijkheden als computerapplicaties, met de extra bekommernis door het kleine scherm aanwezig op een smartphone.

Wanneer de gebruikers het apparaat de hele dag dragen, wordt het een deel van hun dagelijkse gewoontes. Het is dan ook belangrijk dat deze toepassingen alle verwachtingen van de individuele gebruiker vervullen.

Omdat de gebruiker zijn smartphone overal en op het even welk tijdstip kan gebruiken, moeten de applicaties overweg kunnen met verschillende situaties (the carry principle) zoals geen verbinding, veel omgevingsgeluid, enzovoort. Een smartphone is een mobiel toestel; het woord mobiel refereert naar de gebruiker en niet naar het toestel. Dit betekent dat verschillende toestellen gemakkelijk draagbaar kunnen worden genoemd zoals een laptop, smartphone, enzoverder. Alvorens een toestel in de mobiele categorie zou vallen, moet dit toestel aan verschillende criteria voldoen, zoals bijvoorbeeld dat het toestel met 1 hand bruikbaar is. [10].

De verschillende karakteristieken van een smartphone zijn:

1. Persoonlijk: een smartphone wordt meestal door één gebruiker gebruikt
2. Communicatief: de smartphone kan op verschillende manieren een verbinding maken met netwerken of andere toestellen.

3. Handelbaar: een smartphone is draagbaar en kan gebruikt worden met één hand
4. Snel uit slaapstand te halen: een smartphone is snel opgestart en klaar voor gebruik.

2.1.2 Mogelijkheden

Zoals reeds vermeld, zitten smartphones boordevol technologieën, zowel hardware- als softwarematige.

1. Foto- en video-opname en afspelen : smartphones beschikken bijna allemaal over minimaal 1 camera, waarmee men foto's kan nemen en video's maken en hierna kunnen deze afgespeeld worden op het scherm
2. Microfoon: hiermee kan de smartphone geluid opnemen of waarnemen in de omgeving van het toestel
3. Luidspreker: hiermee kan de smartphone geluidsopnames afspelen
4. Cellulaire connectie: met deze connectie kan de smartphone zich verbinden met het mobiel telefoonnetwerk. Het is dan mogelijk om berichten te verzenden en ontvangen en om telefoongesprekken te voeren.
5. WIFI en 3G: met deze connectie kan de smartphone zich met een netwerk verbinden wat meestal leidt tot internettoegang.
6. Bluetooth: met deze connectie kan de smartphone zich verbinden met een naburig toestel dat ook over de bluetooth verbindingstechnologie beschikt.
7. Locatie: de locatie (GPS) ontvanger geeft een precieze positie van de actuele ruimtelijke situering van het toestel.
8. Drie-assen gyroscoop: aan de hand van deze sensor is het mogelijk om waar te nemen aan welke drie dimensionele bewegingen het toestel onderworpen wordt.
9. Versnellingsmeter: aan de hand van deze sensor is het mogelijk om waar te nemen aan welke snelheid (of met welke kracht) het toestel verplaatst wordt.
10. Nabijheidssensor: dank zij deze sensor is het mogelijk om waar te nemen hoever een object gelegen is van de smartphone. Let wel op dat, indien het toestel maar over één sensor beschikt (aan bijvoorbeeld de bovenzijde van het toestel) het dan enkel mogelijk is om langs deze zijde van het toestel waar te nemen of een object naburig is.

11. Sensor voor omgevingslicht: aan de hand van deze sensor is het mogelijk om waar te nemen hoeveel omgevingslicht er rond de sensor aanwezig is. Let op dat indien de sensor aan de onderkant van smartphone zit en deze dan op tafel ligt, de sensor zal waarnemen dat het heel donker is.
12. Aansluiting voor extern scherm: langs deze connectie kan een extern scherm aangesloten worden.

2.1.3 Ontwikkeling

In dit project is er gekozen om een demonstratie-applicatie te ontwikkelen op de iPhone.

Om op de iPhone te programmeren, moet enkele stappen overbrugd worden zoals het aanleren van de standaard programmeertaal (Objective C). Het doornemen van de handleidingen van de verschillende benodigdheden (computer, handleidingen, iPhone SDK). Ook moet er gedacht worden hoe de applicatie moet ontwikkeld worden en over welke bibliotheken hiervoor gehanteerd moeten worden. Al deze stappen zullen in de volgende sectie behandeld worden; aan de hand van alle bovenvermelde stappen, wordt er al opgemerkt dat het programmeren op de iPhone niet voor iedereen weggelegd is.

2.1.4 Objective-C

Objective-C is formeel de enige programmeertaal waarmee het mogelijk is om nieuwe applicaties voor de iPhone te ontwikkelen. Objective-C is een superset van de programmeertaal C en heeft een deel van zijn syntax te danken aan Smalltalk.

Objective-C is een reflectieve, objectgeoriënteerde programmeertaal die boodschappen zoals in Smalltalk toevoegt aan de programmeertaal C. De laatste jaren is er in Objective-C afgestapt van het handmatig moeten doen aan geheugenbeheer; dit is een vereenvoudiging voor de gebruiker. Echter werd automatisch geheugenbeheer niet ingevoerd voor de iPhone waar nog altijd handmatig de objecten gealloceerd en vrijgegeven moeten worden.

Om applicaties te ontwikkelen voor de iPhone, komen er extra bibliotheken bij. Voor elke hardware of software technologie is er een bibliotheek aanwezig; aan de hand van elke bibliotheek kan je elke technologie tot in detail gebruiken.

2.1.5 Benodigdheden

Om te kunnen programmeren in Objective-C, moet je over de laatste versie van Xcode beschikken. Dit is de IDE die Apple aanbiedt voor iPhone-ontwikkeling. Hierbij komen er enkele problemen naar voor. Je moet namelijk over een Macintosh computer beschikken met een Intel-processor en over het laatste besturingssysteem van Apple (Mac OS X Snow Leopard). Naast Xcode, moet de gebruiker ook de laatste iPhone SDK installeren om applicaties voor een iPhone te kunnen ontwikkelen.

Zoals de naam iPhone SDK het aangeeft, is dit een "Software Development Kit"; dit is een verzameling van hulpmiddelen die handig zijn bij het ontwikkelen van programma's voor de iPhone. Deze hulpmiddelen zijn onder andere de volgende :

1. iPhone simulator: biedt een simulatie programma voor de iPhone aan; hierin kan de gebruiker zijn programma's draaien en testen zonder de noodzaak om via een fysieke iPhone te gaan. Echter kunnen alle iPhone mogelijkheden hier niet in worden getest, bijvoorbeeld het verbinden met een cellulair netwerk, specifieke bewegingen waaraan het toestel onderworpen wordt, enzoverder.
2. iPhone Interface builder: biedt een manier om grafische aspecten van een iPhone programma te construeren; de functionaliteit achter de grafische interface moet geprogrammeerd worden in Objective-C.

Het volledig beheersen van een general purpose taal is al een moeilijke zaak; hier komen dan nog alle bibliotheken bij die extra moeilijkheden brengen. Zo toont een studie aan dat er zes grote obstakels zijn die een programmeur moet overwinnen om een taal en bibliotheken onder te knie te hebben; deze obstakels zijn: design van de taal, selectie, coördinatie, gebruik, verstaanbaarheid en informatie. [34]

2.1.6 Ontwikkelen van een applicatie

Naast het leren van de programmeertaal en -omgeving, is het ontwikkelen van een applicatie op zich geen gemakkelijke tak. Om deze tak te vereenvoudigen, zijn er verschillende design-processen gebouwd, om zo efficiënt en gemakkelijk mogelijk doorheen de ontwikkeling te geraken [11, 13].

In het algemeen worden de volgende stadia herkend in de verschillende design processen :

1. Definitiestudie/analyse: er wordt nagegaan wat de vereisten en specificaties zijn voor de applicatie en hoe deze door de verschillende gebruikers gebruikt kunnen worden. Indien de gewenste applicatie in een specifiek domein gespecialiseerd is, bijvoorbeeld de medische sector, moet de analist zich eerst in dit domein inwerken.

2. Basisontwerp: de vereisten, die naar voor zijn gekomen bij het maken van de analyse, worden in een basisontwerp van de applicatie uitgewerkt.
3. Technisch ontwerp/detailontwerp: aan de hand van de voorgaande fases, wordt het programma volledig uitgewerkt. Het plan van hoe het programma gecodeerd gaat worden, wordt gemaakt.
4. Bouw/implementatie: in dit stadium wordt het programma geprogrammeerd. Het gemaakte ontwerp wordt hier als basis gebruikt om het volledig programma te implementeren.
5. Testen: in deze laatste fase, wordt er getest of alles werkt zoals beschreven staat in de analyse van de applicatie. Naast het correct werken van de applicatie, wordt er ook nagegaan indien de gebruikersinterface voldoet aan eisen van de eindgebruikers.

Voor het beheersen van deze vijf stadia, heeft een programmeur veel praktijk en kennis nodig. In het algemeen wordt software gebouwd door een team van verschillende specialisten, die zich elk bezighouden met één of enkele stap van het proces.

Deze vijf fasen worden herhaaldelijk overlopen tot de applicatie volledig afgewerkt is. Zo een herhaal proces heet een software life cycle ; naargelang verschillende aspecten zoals tijd, kost, aanpak, bestaan er verschillende software life cycles . Eén van de eenvoudigste is het watervalproces [38].

2.1.7 Uitbrengen van een applicatie

Als je uiteindelijk alle nodige technologieën, ervaring, studie van Objective-C en bibliotheken onder de knie hebt, kan je je applicatie implementeren en uitgeven. Hiervoor moet je bij Apple een certificaat kopen om je programma te mogen uitbrengen. Het programma zal degelijk worden getest door Apple op geheugenlekken, op correct functioneren en er zal worden nagegaan of je programma niet tegen hun principes ingaat.

Aan de hand van dit certificaat, kan je ook je programma op je eigen iPhone testen.

2.2 On the spot

Naast het traditionele ontwikkelingsproces, zoals hierboven beschreven, zou het ook mogelijk kunnen zijn om op de smartphone zelf te ontwikkelen en onmiddellijk uit te voeren. Dit laat toe dat de gebruiker het toestel in handen heeft en hiermee dan overal kan werken zonder de noodzaak te hebben om de certificaten, programmeeromgeving en bibliotheken te downloaden en te installeren op zijn computer. De gebruiker kan dan elke dag op de weg naar zijn werk op de trein aan applicaties werken.

Het nadeel om te programmeren op een smartphone is het kleine scherm. Hiervoor moet dus een aangepaste gebruikersomgeving gemaakt worden.

Om gebruikersomgevingen te bouwen, is er al meermaals onderzoek gedaan; dit onderzoek zal in volgende secties geïntroduceerd worden.

2.2.1 Design principles

Ben Shneiderman heeft onderzoek gedaan om goede gebruikersomgevingen te bouwen; hij definieerde hierbij dan acht gouden regels, die als een goede basis kunnen dienen [48]. Deze regels zijn de volgende:

1. Hou de gebruikersomgeving zo consistent mogelijk.
2. Laat kortere wegen toe.
3. Geef informatieve feedback
4. Indien gebruik van dialogen, zorg voor een goede opbouw.
5. Voorkom het kunnen maken van fouten; als fouten mogelijk zijn geef een zo gemakkelijke mogelijke errorcorrectie.
6. Laat de gebruiker toe om stappen ongedaan te maken.
7. Pas de controle en interface aan voor verschillende types gebruikers.
8. Verminder geheugenlading op korte termijn.

Aanvullend onderzoek omtrent smartphones is gedaan, dankzij het toenemend succes van smartphones. De hierboven beschreven 8 gouden regels blijven steeds van toepassing, maar sommige regels moeten worden aangepast aan het kleine scherm en omgeving.

Een bijkomend probleem is dat de gebruiker in een drukke en afleidende omgeving kan zitten. De gebruikersinterface moet hints geven, die duidelijk maken waarmee de gebruiker bezig is. Bij regel 5 moet er bijvoorbeeld aan gedacht worden dat de gebruiker extra fouten kan maken omdat het scherm klein is en dus ook het toetsenbord dat er zich op bevindt. Zo kan de gebruiker bij het typen veel foute toetsen aanslaan. Dit is een probleem dat bij een computerapplicatie minder bestaat. Bij een mobile device, kan er geopteerd worden om andere input voorzieningen te geven aan de gebruiker. Naast deze hogergenoemde richtlijnen, komen er nieuwe richtlijnen naar voor: er wordt aangeraden om voor een “top-down-interface” te kiezen. De reden hiervoor zijn de te kleine schermen. Een top-down-interface kan gezien worden als verschillende dozen. Op een doos staat er een korte beschrijving van de inhoud. Eens je deze doos opent, zie je de inhoud en meer gedetailleerde inhoud van de doos. Indien dit een grote doos is, kan deze bestaan uit nieuwe dozen, die zo nieuwe dozen openen en steeds meer en meer gedetailleerde informatie bieden. Men kan dan op één klein scherm informatie bieden in kleine dozen aan de gebruiker. De gebruiker kan dan op een doos drukken en dan wordt het hoofdscherm met de verschillende dozen vervangen door de inhoud van de geselecteerde doos.

Naast deze bijkomende richtlijn, is er een andere richtlijn om de gebruiker zo weinig mogelijk te laten typen. In plaats van te typen, wordt er aangeraden om de gebruiker de keuze te laten om zo gemakkelijk mogelijk informatie te kunnen selecteren uit lijsten. Deze richtlijn geeft ook de voorkeur om, waar mogelijk, default waarden te gebruiken. De gebruiker kan de default waarden aanpassen, indien hij wenst, maar in meeste situaties zouden de default waarden moeten voldoen [48] [10] [24].

De grootste bekommernis van een smartphone is het scherm: met dit klein scherm is het niet mogelijk om verschillende applicaties naast elkaar te tonen. Het is dan ook cruciaal dat alle nodige informatie aanwezig is in de applicatie, zodat de gebruiker de applicatie niet hoeft af te sluiten om informatie te vinden.

Uiteindelijk is een belangrijke richtlijn om lijsten alfabetisch te sorteren. Zoekfuncties doorheen lijsten hoeven ingeschakeld te worden indien de lijsten langer zijn dan wat op het scherm kan getoond worden.

2.2.2 Gestalt psychologie

Naast de 8 gouden regels van Shneiderman, wordt de gestalt psychologie gebruikt om gebruikersinterfaces te verbeteren. De gestalt psychologie beschrijft hoe mensen objecten groeperen die voorkomen in zekere patronen. Deze wetten zijn:

1. De wet van gelijkenis: wanneer elementen dezelfde karakteristieken uiten (zoals kleur, vorm, ...) zal de gebruiker tussen deze elementen een verband zien.
2. De wet van nabijheid: indien enkele elementen dichter bij elkaar staan dan andere, zal de gebruiker deze dicht bij elkaar staande elementen als een groep waarnemen.
3. De wet van continuïteit: door een zekere stroom doorheen de elementen te bouwen, zal de gebruiker elementen in dezelfde stroom tot één groep aanschouwen. Een voorbeeld van zo een stroom is de uitlijning in websites.
4. De wet van afsluiting: indien elementen een onvolledige vorm weergeven, zal de gebruiker deze onvolledige vorm toch waarnemen zonder dat deze er expliciet staat.
5. De wet van de symmetrie: indien objecten in een gebruikersinterface afgebakend zijn door symmetrische grenzen, zal de gebruiker elementen binnenin één dezelfde grens als een groep aanschouwen.

Voor de gebruikersinterface die wij zullen bouwen, zijn er sommige beperkingen door de smartphone, zoals de grootte van het scherm. Het mobiele milieu is minder voorspelbaar in de meeste gevallen dan de omgeving rondom computers. Dit is een gevolg van het feit dat de gebruiker de hele dag het apparaat in verschillende situaties gebruikt.

In dit onderzoek zullen wij ons beperken tot mobiele apparaten, namelijk smartphones met het multitouchscherm (voor het gebruikersinterface). De voorgestelde omgeving en de engine zullen ook op andere mobiele apparaten bruikbaar zijn.

3 Verwant werk

Onderzoek toont aan dat het leren van een general purpose taal zoals namelijk Objective-C en het onder de knie krijgen van het hele ontwikkelingsproces tien jaar in beslag neemt [38]. Om deze reden zijn general purpose talen niet geschikt voor niet-programmeurs.

Naast het feit dat een general purpose taal moeilijk te leren is, is deze ook niet gewenst om on the spot te ontwikkelen. De reden hiervoor is dat een tekstuele taal niet past op een klein scherm en dito toetsenbord.

Om deze reden gaan we in deze sectie andere bestaande programmeer mogelijkheden onderzoeken.

3.1 Script talen

Script talen zijn talen die gemaakt zijn voor het maken van kleine scripts. Een script beschrijft een veel, door de gebruiker uitgevoerde, computer handeling. Een computer kan, door het script uit te voeren, deze handeling herhalen.

In een script taal zijn alle basiselementen van een general purpose taal aanwezig. De kracht van script talen is dat ze complexe functionaliteiten weergeven in één functie; dit wil zeggen dat bijvoorbeeld in een script taal een functie "speel liedje A" aanwezig is. In een general purpose taal zou je eerst het pad naar het liedje moeten inladen, hierna de muzikspeler, vervolgens het liedje aan de muzikspeler toevoegen. Eens het liedje is afgespeeld, moet het muzikspelers geheugen vrijgeven worden. Al deze stappen uit de general purpose taal zitten dan geabstraheerd in de speelfunctie in de script taal. Dit zorgt ervoor dat script talen meestal een lagere instapdrempel hebben dan general purpose talen.

De meeste script talen worden gebouwd rondom een zeker domein. Zo is AppleScript [52] een script taal die operating systeem functionaliteiten abstraheert en bruikbaar maakt in een script taal. In een ander domein is er JavaScript [50] dat een script taal is, die veel gebruikt wordt om webpagina's interactiever te maken.

3.1.1 Tekstuele script talen

Als voorbeeld van een tekstuele script taal, wordt hier AppleScript aangehaald. Aan de hand van Applescript, kunnen we operating systeem functionaliteiten gebruiken, zoals muziek afspelen, vensters weergeven, tekst voorlezen enzoverder.

In het volgende voorbeeld wordt er getoond hoe we aan de hand van AppleScript een script kunnen schrijven, dat eerst een pop up venster weergeeft met de tekst “Example!!” en 2 knoppen: “Beep” en “Talk”.

Indien op de knop “Beep” gedrukt wordt, start er een iteratief proces dat 5 keer een beeptoon gaat afspelen. Indien de “Talk” knop ingedrukt wordt, dan zal aan de hand van het “say” commando de tekst: “hallo” worden gelezen door de computer.

Het script ziet er als volgt uit:

```
display alert "Example!!" buttons { "Beep", "Talk" }
set the Answer to button returned of the result
if the Answer is "Beep" then
    beep 5
else
    say "hallo"
end if
```

In een tweede voorbeeld gaan we, aan de hand van een script, een zekere applicatie het sluitcommando sturen. Dit script ziet er als volgt uit:

```
Tell application "Microsoft Word" to quit
```

We merken op in deze 2 voorbeelden dat we aan de hand van AppleScript verschillende commando's zoals say, display, tell... eenvoudig verschillende functies kunnen uitvoeren. We kunnen deze dan combineren in een script om veel voorkomende handelingen te automatiseren (zoals het tweede voorbeeld) of eenvoudige programma's te ontwerpen (zoals voorbeeld 1).

Script talen worden in de meerderheid van de gevallen niet gebruikt door onervaren programmeurs, ondanks de verlaagde instapdrempel.. Er zijn inderdaad script talen die de instapdrempel niet veel verlagen en dus geen grote productiviteitswinsten bieden [3].

Voor deze reden, worden er pogingen ondernomen om bovenop script talen een grafische gebruikersinterface te bieden. Zo is er Automator (zie sectie 3.4) ontstaan, een grafische workflow script taal, met als basis AppleScript. Om het systeem te vergemakkelijken, is het in Automator enkel mogelijk om sequenties te bouwen en geen condities, iteraties, en zo verder, wat in AppleScript wel mogelijk is.

3.1.2 Grafische script talen

Om de instapdrempel van script talen te verlagen, worden grafische script talen voorgesteld. Grafische talen spreken niet ervaren programmeurs meer aan. Grafische script talen vinden we onder andere terug in spel engines zoals “graphical scripting language” voor de c4 engine en de “Kismet graphical language” voor de unreal engine.

Graphical scripting language voor de c4 spel engine

De makers van de script taal voor de c4 engine, hebben gekozen om hun script taal grafisch voor te stellen. Dit laat toe dat de gebruiker bestaande bouwblokken kan combineren op een grafische manier.

In de script taal wordt gewacht op gebeurtenissen, die voorkomen in de spel-omgeving. Gebeurtenissen worden meestal geactiveerd door de avatar. Een avatar is een virtueel personage in een spelomgeving dat door een persoon of door de computer wordt bestuurd.

De gebruiker kan verschillende bestaande bouwblokken combineren. De gebruiker kan ook nieuwe bouwblokken definiëren; dit gebeurt aan de hand van een tekstuele programmeertaal.

De taalelementen, aanwezig in de script taal voor de c4 engine, bevatten lokale en globale variabelen, condities, iteraties en sequenties [14].

In figuur 1 zien we dat bij het ingeven van een code wordt nagegaan of de code 1234 is. Zo ja, wordt de groene pijl gevolgd, die eerst een boodschap gaat weergeven; vervolgens gaat er een node uitgeschakeld worden. Deze node kan bijvoorbeeld een lamp in het spel voorstellen.

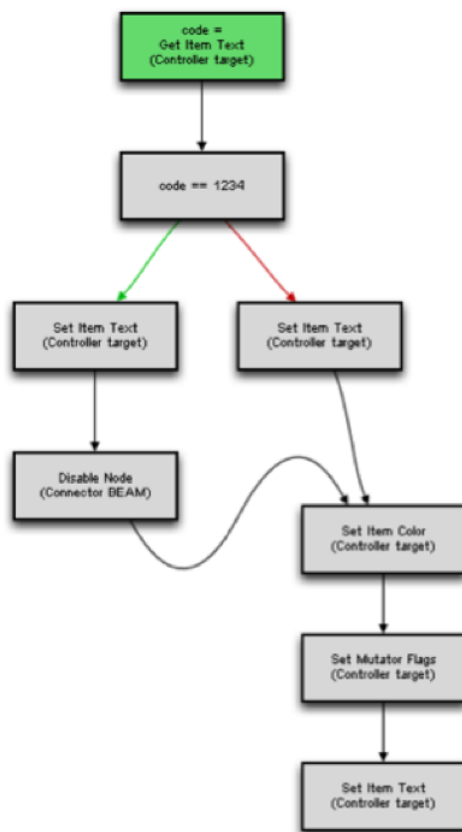
Indien een foute code ingegeven werd, wordt er een fout boodschap weergegeven.

In beide gevallen wordt de tekst in kleur gezet.

Kismet voor de unreal engine

De makers van de Kismet taal bieden, net zoals de grafische script taal voor de c4 engine, een script taal die ingaat op activiteiten die gebeuren in de virtuele wereld. Zo kan de programmeur bijvoorbeeld een schakelaar definiëren, die een lamp aan en uit doet.

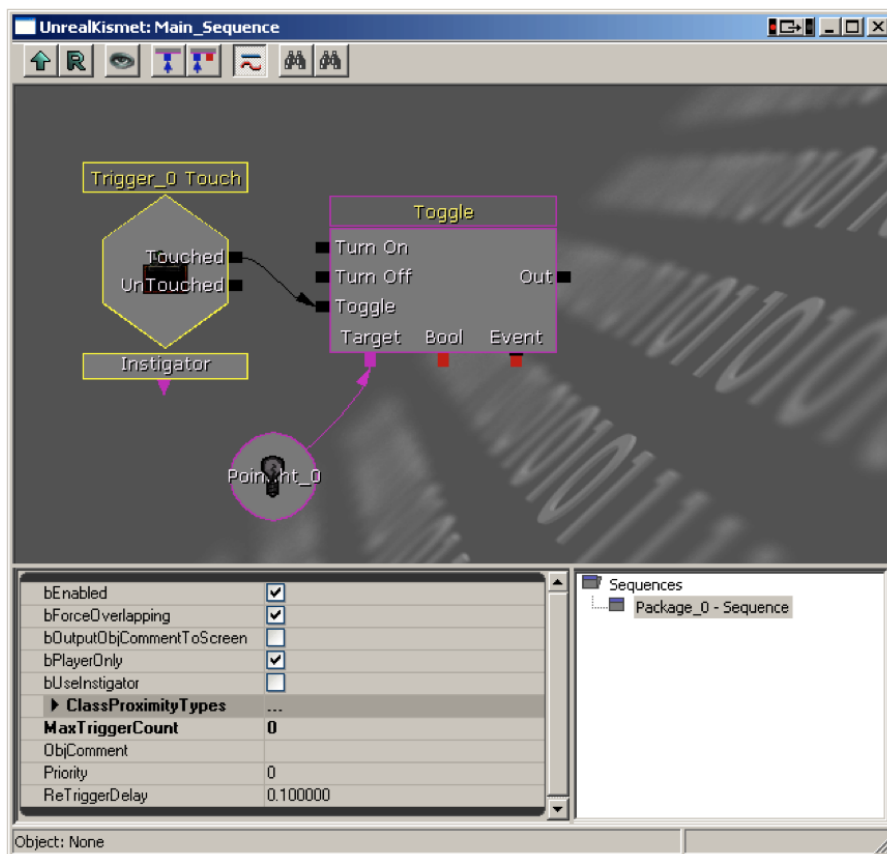
Dit gebeurt door een bestaand grafisch element te nemen dat een poort voorstelt. Zoals afgebeeld in figuur 2, heeft een poort twee stadia, namelijk: de poort wordt ingeduwed of niet. Bij het induwen van een poort, wil de programmeur een lamp laten aangaan; hiervoor kan hij een wisselblok gebruiken dat automatisch de lamp aandoet, indien ze uit was en omgekeerd. De taal bestaat uit verschillende voorgedefinieerde grafische blokken met verschillende functies; deze functies kunnen dan gecombineerd worden met een ander blok in de taal.



Figuur 1: Grafische script taal voor de c4 spel engine

In deze script taal is elk object (bijvoorbeeld een poort, een lamp, een pi-
 stoel) een voorgedefinieerde bouwblok van de grafische taal. Nieuwe bouw-
 blokken kunnen toegevoegd worden aan de hand van tekstuele programma-
 tiecode.

Door deze blokken te combineren, kan de programmeur een spelwereld be-
 schrijven en ook hoe alle objecten reageren op wat de avatar doet [22].



Figuur 2: Kismet script voor de unreal engine

3.2 Workflow

De definitie van een workflow is :

The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules [55].

Aan de hand van een workflow is het dus mogelijk om businessprocessen te beschrijven. Deze workflows worden in meeste workflow talen gebouwd aan de hand van verschillende patronen. Een voorbeeld van een dergelijk patroon is een sequentie patroon, dat sequentieel verschillende taken één voor één gaat uitvoeren.

Deze patronen bieden oplossingen voor veel voorkomende problemen.

Een overzicht van de patronen wordt gegeven in de volgende papers: “Workflow control flow patterns” en “Workflow Control Flow Patterns: A Revised View” [47].

Alle bestaande patronen worden niet in alle workflow talen ondersteund. De reden hiervoor is dat sommige patronen te moeilijk zijn voor het gewenste doelpubliek of dat het patroon geen bijkomende waarde biedt voor de gewenste workflow taal.

Een workflow bestaat uit:

1. Activiteiten: een activiteit beschrijft een stuk werk dat één onderdeel is van groter proces.
Een manuele activiteit kan niet volledig afgehandeld worden door een computer maar vereist de tussenkomst van een gebruiker.
Automatische activiteiten daarentegen kunnen volledig door een computer worden afgehandeld.
2. Patronen: beschrijft hoe de coördinatie tussen de verschillende activiteiten gebeurt. Deze coördinatie zorgt ervoor dat activiteiten verbonden worden om een gemeenschappelijk doel te bereiken.

3.2.1 Business Process Execution Language (BPEL)

BPEL [8] is één van de meeste gebruikte workflow talen in het domein van web services. Aan de hand van BPEL is het mogelijk om het bedrijfsproces beter te monitoren. Aanpassingen kunnen sneller worden doorgevoerd en de verschillende services zijn herbruikbaar. BPEL is een tekstuele workflow taal die met de XML syntax gebouwd is [51]. Aan de hand van BPEL en XML, kunnen business processen gedeeld worden tussen verschillende platformen. De verschillende bouwstenen van BPEL zijn:

1. Receive: ontvangt een boodschap van bijvoorbeeld een andere webservice
2. Reply: antwoordt op een gekregen boodschap van bijvoorbeeld een webservice
3. Invoke: roept een operatie aan van een andere webservice,
4. Throw: genereert een fout
5. Assign: past de waarde van een variabele aan
6. Terminate: stopt het bedrijfsproces
7. Wait: wacht een gegeven tijdsduur

Deze bouwstenen kunnen vervolgens gecombineerd worden tot een proces aan de hand van volgende taaleigenschappen:

1. Sequence: verschillende activiteiten worden sequentieel uitgevoerd
2. Condition: er wordt een specifieke tak gekozen afhankelijk of de conditie waar of vals is
3. While: een activiteit moet herhaaldelijk uitgevoerd worden totdat de conditie vals is
4. Flow: verschillende activiteiten moeten in parallel gebeuren

In het hieronder getoonde voorbeeld, zien we hoe deze verschillende patronen tekstueel verbonden worden. Een iteratie wordt uitgevoerd zo lang als de query waar is. Het proces dat uitgevoerd moet worden (zolang als de query waar is), is een sequentie die eerst op een boodschap wacht en vervolgens een variabele gaat updaten.

```
<while>
<condition> query </condition>
<sequence>
<receive> ... </receive>
<assign> ... </assign>
</sequence>
</while>
```

3.2.2 Yet Another Workflow Language (YAWL)

YAWL is een workflow taal die complexe data, transformaties, integratie met organisatorische middelen en web services biedt. Door al deze functionaliteiten te combineren, kunnen bedrijfsprocessen beschreven worden.

YAWL voorziet een grafische omgeving voor het maken van workflows; deze worden in XML syntax opgeslagen.

Verschillende activiteiten worden verbonden aan de hand van workflow patronen.

Alle ondersteunde patronen worden geïntroduceerd op de volgende website : <http://www.yawlfoundation.org/resources/patterns#state>.

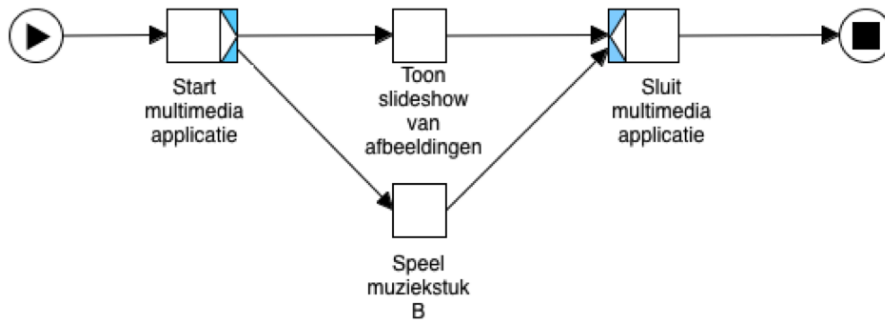
YAWL biedt alle basispatronen zoals sequentie, parallel splitsing, synchronisatie, exclusieve keuze, enzoverder.

YAWL biedt ook meer gevorderde patronen voor complexere situaties; deze worden niet besproken in dit onderzoek, omdat deze uit onze scope vallen.

Een voorbeeld van een proces beschreven aan de hand van de YAWL syntax, wordt getoond in figuur 3. Dit voorbeeld gaat eerst een multimedia applicatie opstarten; eens deze opgestart is, gaat deze 2 activiteiten tegelijkertijd uitvoeren, namelijk:

1. Het afspelen van een muziekstuk B
2. Het afspelen van een slideshow

Eens deze 2 activiteiten afgelopen zijn, wordt de applicatie afgesloten.



Figuur 3: Voorbeeld van een YAWL workflow.

3.2.3 Business Process Model and Notation (BPMN)

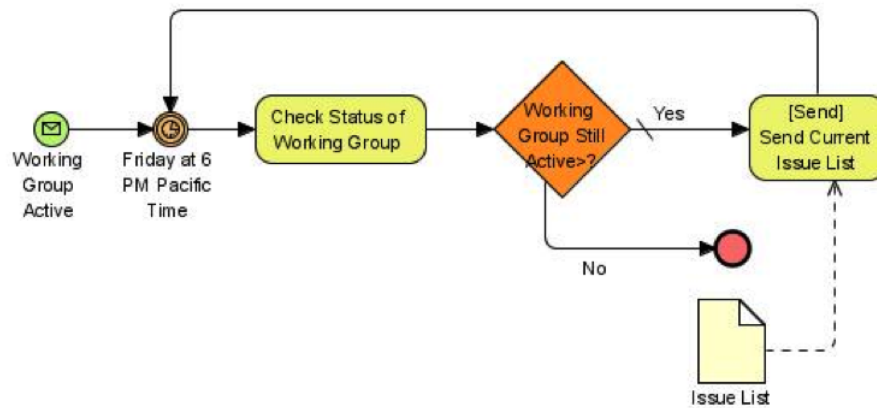
Business Process Model and Notation is een grafische voorstelling en model voor het definiëren van business processen in een business proces model. Het doel van BPMN is om business processen te beschrijven voor technische en business gebruikers, door een intuïtieve en gebruiksvriendelijke notatie te bieden.

Een BPMN model is opgebouwd uit verschillende delen, namelijk:

1. Objecten
 - (a) Gebeurtenissen: een gebeurtenis wordt weergegeven door een cirkel en stelt iets voor dat gebeurt op, bijvoorbeeld, een zeker tijdstip
 - (b) Activiteiten: een activiteit wordt weergegeven door een afgeronde rechthoek en stelt iets voor dat gedaan moet worden.
 - (c) Gateways: wordt door een ruit weergegeven en stelt splitsingen (parallel of conditioneel) en samenvoegingen van verschillende stromen voor.
2. Connecties tussen objecten
 - (a) Sequentiële stroom: door objecten te verbinden in een sequentiële stroom, wordt er getoond in welke volgorde objecten moeten worden uitgevoerd.
 - (b) Boodschap stroom: aan de hand van boodschappen, kan er weergegeven worden welke boodschappen tussen verschillende processen gestuurd worden.
 - (c) Associaties: een associatie geeft weer dat er een verband bestaat tussen verschillende objecten.
3. Zwembanen (swimlanes): een zwembaan wordt rondom objecten geplaatst. Voor een zwembaan is een bepaalde actor verantwoordelijk, bijvoorbeeld voor het maken van een report is de manager verantwoordelijk
4. Artefacten: de gebruiker kan aan de hand van annotaties meer informatie in het model voorzien.

Het voorbeeld op figuur 4 toont een BPMN proces. Als een nieuwe werkgroep actief wordt, wordt er op vrijdag, om 18 uur, de gebeurtenis getriggert dat nagaat of deze werkgroep nog actief is.

1. Zo ja, wordt er een actie ondernomen; deze actie zal een lijst met de problemen doorsturen naar alle mensen die in deze groep zitten en wordt de gebeurtenis weer geactiveerd en zal de volgende vrijdag om 18 uur dit proces opnieuw starten.
2. Indien deze werkgroep niet meer actief is, wordt dit proces afgesloten.



Figuur 4: BPMN voorbeeld [27]

3.3 Modelleertalen

Modelleertalen zijn formele talen. Deze talen worden gebruikt om informatie, kennis van een systeem weer te geven. Een dergelijke modelleertaal bestaat uit verschillende regels. Elke regel definieert de verschillende bouwblokken van de modelleertalen. In deze sectie wordt het UML activiteiten-diagram en het toestandsdiagram geïntroduceerd.

3.3.1 UML activiteitendiagram

Een activiteitendiagram is een manier om een verband te leggen tussen verschillende activiteiten binnenin een proces. Dit laat toe om een proces uit te schrijven aan de hand van de verschillende activiteiten. Ook kan er gezien worden hoe deze activiteiten onderling moeten werken. Om de onderlinge structuur tussen activiteiten te beschrijven, wordt er gebruik gemaakt van volgende elementen:

1. Gevulde cirkel: startpunt van het diagram
2. Gevulde cirkel omringd door een grotere cirkel: eindpunt van het diagram
3. Afgeronde rechthoek: representatie van een activiteit, en activiteit beschrijft een stuk werk.
4. Pijlen: aan de hand van pijlen kunnen de verschillende activiteiten verbonden worden. Eens een activiteit afgerond is, wordt de pijl gevolgd en de volgende activiteit uitgevoerd.
5. Keuzeknoppen hebben de vorm van een ruit: op een keuzeknoop staat een conditie; indien deze conditie waar is, wordt de waar-pijl gevolgd en indien deze conditie niet waar is, wordt de vals-pijl gevolgd.
6. Indien één pijl splitst en naar verschillende activiteiten wijst, wil dit zeggen dat er verschillende activiteiten tegelijkertijd uitgevoerd worden. Later zal deze splitsing terug samenkomen.
7. Zwembanen (swimlanes): een zwembaan wordt een deel van het activiteitendiagram geplaatst. Voor een zwembaan is een bepaalde actor verantwoordelijk; bijvoorbeeld voor het maken van een report is de manager verantwoordelijk

In het gegeven voorbeeld op figuur 5 zien we dat, wanneer de applicatie gestart wordt, er een muziek afgespeeld wordt, namelijk nummer X. Eens dit nummer afgelopen is, is de volgende activiteit een keuzeknoop. In deze keuzeknoop kan de gebruiker binnen de 20 seconden op een opnieuw knop drukken. Indien de gebruiker dit doet, wordt het nummer opnieuw afgespeeld.

Indien de gebruiker niet binnen de 20 seconden op de opnieuw knop drukt, wordt de vals-pijl gevolgd en is de applicatie afgelopen.



Figuur 5: Voorbeeld van een UML activiteitendiagram

3.3.2 UML toestandsdiagram

Aan de hand van een toestandsdiagram [42, 36, 16] is het mogelijk om de verschillende toestanden in een systeem weer te geven. Deze toestanden kunnen worden verbonden door middel van pijlen. Door deze toestanden te verbinden, kunnen we een proces op een grafische manier weergeven.

De basiselementen van een toestandsdiagram zijn:

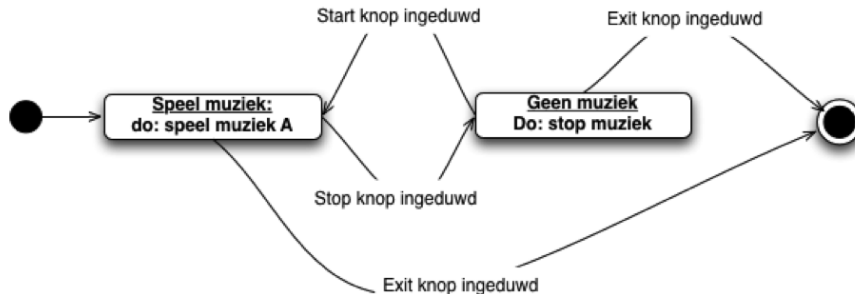
1. Gevulde cirkel: startpunt van het diagram
2. Gevulde cirkel omringd door een grotere cirkel: eindpunt van het diagram
3. Afgeronde rechthoek: representatie van een toestand. Het bovenste deel van de rechthoek bevat de naam van de toestand; hieronder worden de activiteiten beschreven, die door de toestand ondernomen worden.
4. Pijlen tussen acties kunnen verschillende dingen betekenen, namelijk:
 - (a) Een gebeurtenis. Een gebeurtenis is iets dat gebeurt in ruimte en tijd, bijvoorbeeld, een toets wordt ingedrukt. Een gebeurtenis wordt weergegeven als parameter van een pijl; indien deze gebeurtenis plaatsvindt, wordt de toestand verlaten en wordt de pijl van de gebeurtenis gevolgd naar de volgende toestand. Indien een gebeurtenis plaatsvindt in een toestand waarin het systeem zich momenteel niet bevindt, wordt deze toestand genegeerd.
 - (b) Guard condities zijn booleaanse expressies. Net zoals gebeurtenissen, zijn dit parameters van pijlen tussen 2 toestanden. Een guard wordt weergegeven tussen rechthoekige haken. Indien een guard waar is, wordt de huidige toestand verlaten en wordt de pijl van de guard gevolgd naar de volgende toestand.
 - (c) Indien een guard en gebeurtenis op dezelfde pijl staan, moet en de gebeurtenis plaats gevonden hebben en op hetzelfde moment de guard waar zijn.

In figuur 6 zien we 2 toestanden, namelijk

1. Er speelt muziek
2. Er speelt geen muziek

Om van toestand 1 naar toestand 2 te gaan, moet de gebeurtenis, “de stop knop wordt ingedrukt”, gebeuren. Eens we in toestand 2 zijn, wordt alle muziek gestopt; indien er nu op de start knop gedrukt wordt, wordt nummer A weer afgespeeld.

Indien in beide toestanden op de exit knop gedrukt wordt, wordt het proces gestopt.



Figuur 6: Voorbeeld van een UML toestandsdiagram

3.4 Grafische talen

In deze sectie worden verschillende grafische talen en omgevingen geïntroduceerd. Deze talen zijn ontworpen om de drempel te verlagen en zo niet-programmeurs aan te trekken om applicaties te bouwen.

3.4.1 Apple Automator

Automator [53] is een door Apple ontwikkelde applicatie. Automator biedt een grafische user interface rond AppleScript. Het doel van Automator is dat de gebruiker op een gemakkelijkere manier scripts zou kunnen maken. Net zoals in AppleScript, kan de gebruiker in Automator bijvoorbeeld aan de besturingssysteem functionaliteiten; deze kan de gebruiker dan combineren in een script.

Omdat de makers van Automator een alledaagse gebruiker zonder programmeer ervaring wilden aanspreken, is Automator niet even krachtig als Applescript. Het is niet mogelijk om in Automator condities of iteraties (over verschillende activiteiten) uit te drukken. In Automator kan je enkel sequentiële scripts maken.

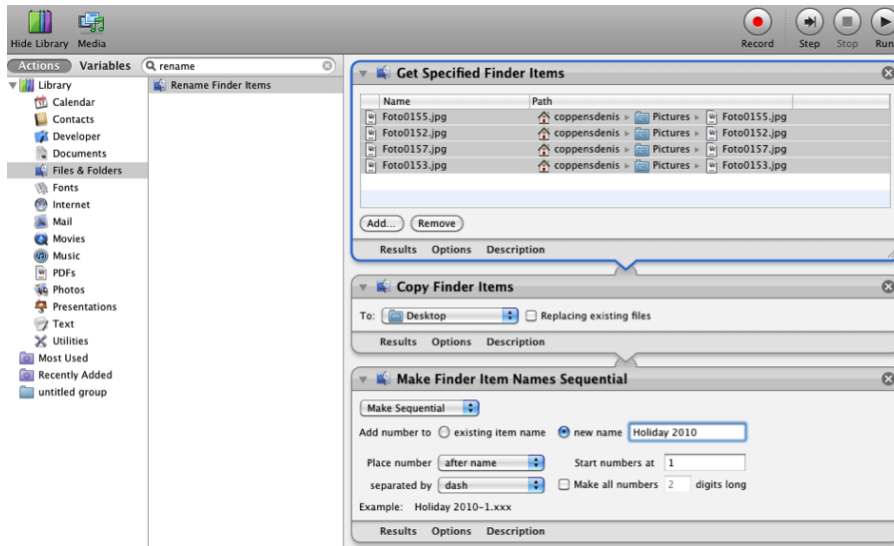
De makers van Automator hebben gekozen voor de gebruiksvriendelijkheid door alle mogelijke functies te abstraheren in bouwblokken, waarvan de parameters kunnen aangepast worden.

Een voorbeeld van zo een bouwblok kan zijn: het afspelen van een liedje. De parameters van dit bouwblok zijn dan: welk liedje, hoeveel keer moet het liedje gespeeld worden...

Deze bouwblokken worden dan achter elkaar geplakt. Zo bouwt de programmeur een sequentie van bouwblokken, die een script vormen.

Vervolgens is het mogelijk om dit script op te slaan en uit te voeren wanneer de gebruiker dit wenst.

In het getoonde voorbeeld op figuur 7 gaan er foto's geselecteerd worden. Dit gebeurt in het bovenste bouwblok. Vervolgens worden, in het middelste bouwblok, de afbeeldingen gekopieerd naar het bureaublad. Eens deze afbeeldingen gekopieerd zijn, wordt hen een nieuwe naam toegekend, "Holiday 2010" met elk een uniek nummer.



Figuur 7: Automator

3.4.2 Lego Mindstorms

Lego Mindstorms [54] zijn kleine robots van Lego waaraan verschillende motorische elementen en sensoren kunnen gekoppeld worden. Deze robots kunnen geprogrammeerd worden, zodat de ontworpen Lego robot een zeker proces kan uitvoeren.

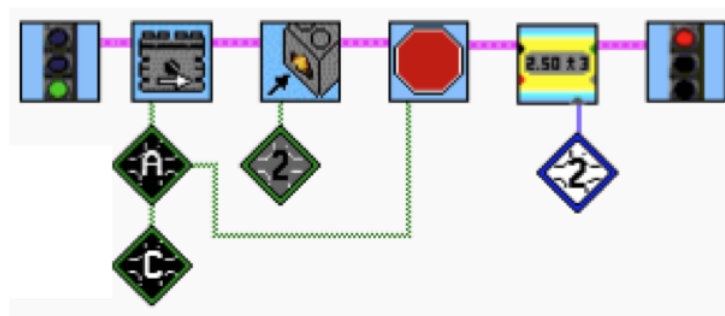
Het zou zo mogelijk zijn om je eerste robot in 30 minuten te bouwen [54].

Om programma's te schrijven voor de Lego Mindstorm, kunnen general purpose talen gebruikt worden.

In deze sectie gaan we deze talen niet bespreken omdat general purpose geen gewenste taal is voor ons doelpubliek.

Naast general purpose talen, kan je ook met Robolab [23] programmeren; deze taal, die een domein specifieke abstractie is van LabView [6], biedt aan de gebruiker een grafische workflow taal aan. Aan de hand van deze workflow taal, zou het mogelijk zijn, voor 9 jarige kinderen, om een nieuwe robot en applicatie te bouwen [4].

Indien we deze taal analyseren, merken we op dat deze specifiek voor Mindstorms gebouwd is. Op een Mindstorm zijn er 8 poorten aanwezig, genummerd van 1 tot 4 en van A tot D; op elke poort kan je een motor of sensor zetten, die geprogrammeerd kan worden.



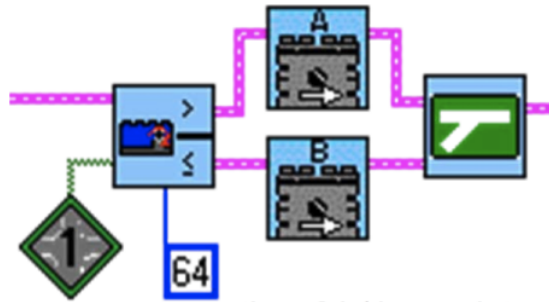
Figuur 8: Voorbeeld van een applicatie voor de Lego Mindstorm robot

In Robolab is het mogelijk elke poort individueel aan te spreken. Op figuur 8 zien we hoe je een programma neerschrijft in Robolab.

Elk symbool heeft een welbepaalde betekenis, het groene verkeerslicht staat voor het begin van het proces, het rode verkeerslicht voor het einde ervan. Het icoon boven A en C duidt aan dat men de motoren wil starten op poort A en C.

Het voorbeeld zal de motoren op poorten A en C starten tot de input op poort 2 waar is; dit is bijvoorbeeld een druksensor; indien de druksensor ingedrukt wordt, geeft deze een nieuwe waarde, de input op poort 2 is dan

waar. Vervolgens worden motor A en C stilgezet en wordt de sterkte waarmee de druksensor van poort 2 ingedruwd werd op het display getoond.



Figuur 9: Voorbeeld van een conditionele applicatie voor de Lego Mindstorm robot

Naast het sequentieel gedrag, is het ook mogelijk om conditioneel gedrag te beschrijven; dit wordt getoond in figuur 9. In dit voorbeeld wordt motor A in gang gezet, indien de rotatie van de sensor op poort 1 groter is dan 64. Indien de rotatie van de sensor kleiner of gelijk was dan 64, wordt de motor op poort B in gang gezet. Hierna worden deze 2 takken samengebracht. Het verloop van de rest van de applicatie is hetzelfde voor beide takken.

Naast condities, informatiedoorgaven (zoals aangetoond in het voorbeeld) is het ook mogelijk om te itereren in Robolab. Zo kan de gebruiker het programma terugzetten naar een zekere activiteit in de workflow.

Subroutines (oproep naar een ander programma) zijn ook mogelijk; de gebruiker kan dan, tijdens het uitvoeren van de workflow, een oproep doen naar een applicatie op de computer. Zo kan de lego Mindstorm aan de verbonden computer een commando sturen om muziek te laten afspelen.

Het ontwikkelen van deze workflows gebeurt op een computer; het gemaakte proces wordt dan naar de robot gestuurd door de bluetooth connectiviteit. Het ontwikkelen gebeurt dus niet on the spot.

3.4.3 Yahoo Pipes

Grafische talen worden gebruikt voor het ontwikkelen in game engines, robots, operating systemen. Yahoo heeft een visuele taal aangeboden om mashups te maken: deze taal heet Yahoo Pipes [28]. Een mashup is een applicatie die informatie uit verschillende bronnen haalt en aan de gebruiker gegroepeerd toont.

Deze informatie kan bovendien gesorteerd worden per categorie, trefwoorden, enzoverder.

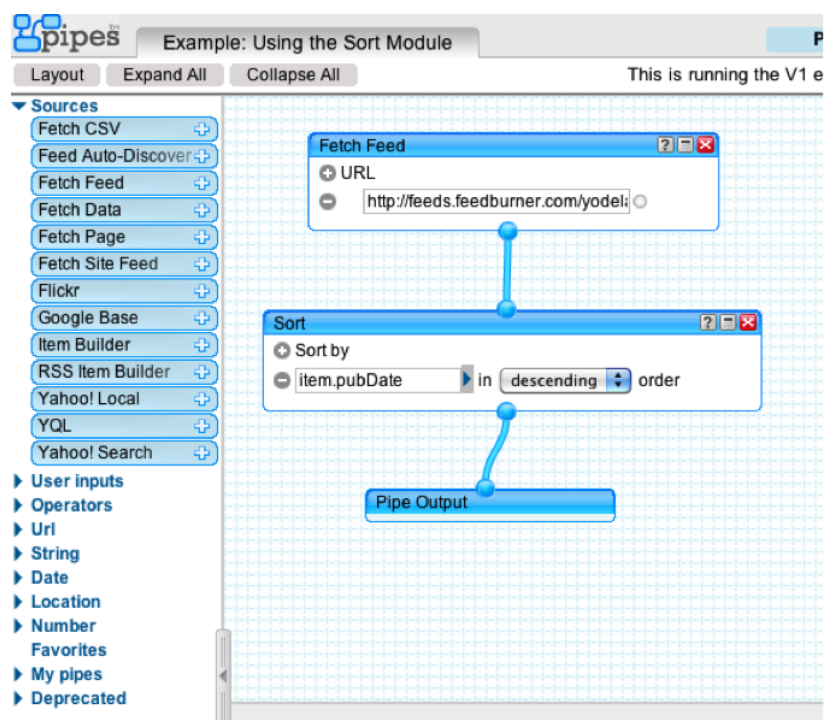
De grafische omgeving van Yahoo Pipes laat de gebruiker toe om informatie vanuit een zekere bron te filteren, combineren...

Dit wordt mogelijk gemaakt door informatie door verschillende blokken te laten gaan. Elke blok gaat dan iets specifiek doen met de inkomende informatie. Een subset van alle verschillende blokken is:

1. Een combiner: gaat informatie uit verschillende bronnen combineren.
2. Een collector: collecteert gelijkaardige informatie vanuit de verschillende bronnen.
3. Een searcher: zoekt specifieke informatie uit alle inkomende informatie. Deze zoekfunctie kan bijvoorbeeld alle informatie sorteren op trefwoorden. Omdat het web constant nieuwe informatie biedt, is het mogelijk dat nieuwe informatie beschikbaar wordt; hiervoor kan er een iteratie ingebouwd worden en zal de *zoekdoos* automatisch informatie opnieuw zoeken op een website.
4. Een selector: de gebruiker kan handmatig uit alle informatie de voor hem interessante informatie selecteren.
5. ...

Door al deze blokken aan elkaar te hangen, kan de gebruiker alle informatie filteren en organiseren om uiteindelijk enkel de interessante informatie over te houden; dit aan elkaar hangen van blokken wordt een pipe genoemd.

Eens een pipe gemaakt is, kan deze gebruikt worden om elke dag informatie uit de gegeven bronnen af te halen. Zo kan een gebruiker bijvoorbeeld al het sportnieuws van een zekere ploeg filteren uit een nieuwssite. De gebruiker kan dan snel zien of er nieuwe artikels gepost zijn op de nieuwssite over zijn lievelingsploeg.



Figuur 10: Voorbeeld van een Pipe gehaald van het officiële Yahoo pipe leerprogramma

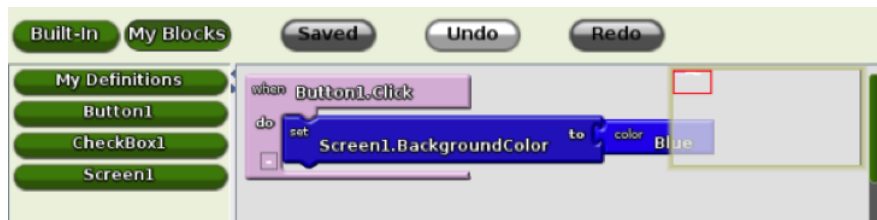
3.4.4 Applicatie uitvinder voor Android

Applicatie uitvinder voor Android [25] is een grafische workflow taal die toelaat om applicaties te bouwen voor het Android operating system. Deze applicaties kunnen spellen, kwissen, enzoverder zijn.

Een door Google aangehaalde statement zegt dat, om een applicatie te maken aan de hand van deze grafische workflow taal, geen professionele ontwikkelaars nodig zijn.

To use App Inventor, you do not need to be a professional developer. This is because instead of writing code, you visually design the way the app looks and use blocks to specify the app's behavior. [26]

Google gebruikte als basis voor hun gebruikersinterface het onderzoek “Open blocks” gedaan door meneer Roque [44]. Dit is een Java bibliotheek om blok-gebaseerde gebruikersinterfaces te bouwen. Deze verschillende blokken stellen activiteiten voor, die deel zijn van een groter project. Deze verschillende activiteiten kunnen aan elkaar gekoppeld worden door blokken in de gebruikersinterface aan elkaar te koppelen. Door deze blokken aan elkaar te koppelen, vormt men een proces zoals getoond op figuur 11.

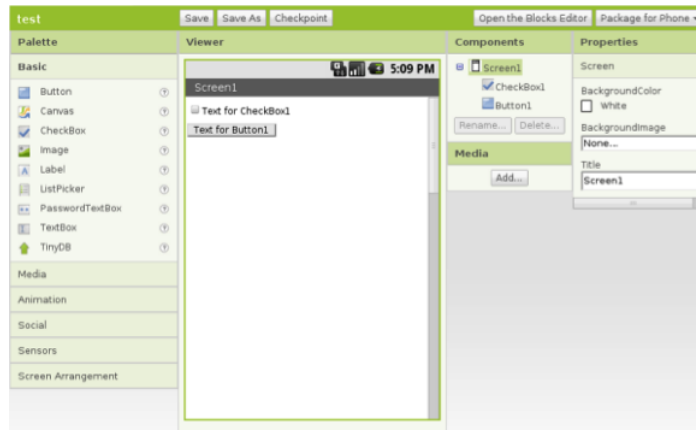


Figuur 11: Voorbeeld van de blok interface voor het maken van applicaties voor Android

Het nadeel van deze is aanpak is dat de gebruiker steeds langs zijn computer moet om een applicatie te maken en dus niet op zijn smartphone zelf een programma kan maken.

Hiernaast kan de programmeur gebruik maken van een interface builder (zie figuur 12) om vensters te maken in de applicatie.

De gebruiker kan aan deze nieuwe vensters blokken hangen, om gedrag te definiëren tussen het venster en de gebruiker; zo kan aan een “opslaan knop” in de interface een blok gehangen worden dat alle informatie gaat opslaan.



Figuur 12: User interface bouwer voor de applicatie uitvinden voor Android

3.5 Besluit

In deze sectie merken we op dat grafische talen en workflow talen een lagere instapdrempel bezitten dan tekstuele talen. Zo kunnen 9 jarige kinderen met workflow talen werken. Als er over een business proces wil gesproken worden met niet-programmeurs, wordt er altijd voor een visuele taal gekozen.

Hierbij bieden grafische talen en workflow talen een oplossing voor het combineren van verschillende blokken. Het combineren van dergelijke blokken in een proces, op een intuïtieve manier, is één van de doeleinden van ons onderzoek.

In alle bestaande grafische talen wordt er echter altijd geprogrammeerd op een computer.

In dit onderzoek zal er dus gekeken worden hoe we een grafische taal kunnen combineren met een klein scherm om op the spot ontwikkeling aan te kunnen bieden.

Naast een aangepaste interface, moet er ook nagegaan worden welke verschillende taalelementen in onze taal aanwezig hoeven te zijn.

4 Workflow bouwblokken

De grafische taal die we voor ogen hebben, biedt verschillende bouwblokken, die door verschillende patronen kunnen verbonden worden om als eindresultaat een workflow te geven.

In deze sectie worden de verschillende bouwblokken, die beschikbaar zijn in onze taal, voorgesteld.

Zoals reeds aangegeven aan de hand van scenario's in de inleiding, wordt er een onderscheid gemaakt tussen acties en gebeurtenissen: deze zullen een andere visuele weergave krijgen.

Acties en gebeurtenissen worden ook in andere talen gebruikt; de externe input en interacties in tijd en ruimte zijn gebeurtenissen. Acties worden uitgevoerd door de machine zelf; deze worden zelfstandig uitgevoerd [39, 7, 9].

Door een visueel onderscheid te bieden tussen de verschillende bouwblokken, verhogen we de leesbaarheid en bruikbaarheid van de taal en ontwikkelsomgeving.

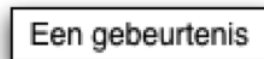
4.1 Gebeurtenissen

Definitie: Een gebeurtenis is iets dat zich voordoet in tijd en ruimte, bijvoorbeeld

1. Een gebeurtenis kan een gebruikersinteractie met de smartphone voorstellen, zoals onder andere:
 - (a) Een knop wordt ingedrukt
 - (b) De gebruiker raakt het scherm met twee vingers aan
2. Een gebeurtenis kan sensor afhankelijk zijn, zoals:
 - (a) De bewegingssensoren voelen een hevige beweging.
 - (b) De lichtsensoren nemen waar dat er geen licht rondom het toestel meer aanwezig is
3. Een gebeurtenis kan ook netwerk afhankelijk zijn, bijvoorbeeld:
 - (a) De smartphone ontvangt een nieuwe SMS
 - (b) De smartphone heeft geen netwerkconnectie meer
4. Een gebeurtenis kan tijdsgebonden zijn, bijvoorbeeld:
 - (a) Een zekere tijdsduur is verlopen (bijvoorbeeld 10 seconden)
 - (b) Het is een bepaald tijdstip (bijvoorbeeld 18:00 uur)

Semantiek: bij het uitvoeren van een workflow, zal bij een gebeurtenis bouwblok gewacht worden tot de gebeurtenis plaatsvindt.

Representatie: een gebeurtenis wordt in onze taal voorgesteld aan de hand van een rechthoek (zie figuur 13).



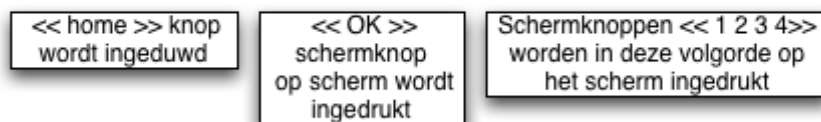
Figuur 13: Grafische weergave van een gebeurtenis

Voorbeelden: De ontwikkelaar van een nieuwe applicatie wil inspelen op een specifieke gebeurtenis; bijvoorbeeld de gebruiker drukt op de 'home knop'. In deze workflowtaal wordt voor elke specifieke gebeurtenis een nieuw *gebeurtenis bouwblok* gebruikt. Veel van de beschikbare gebeurtenissen zijn configureerbaar. Bijvoorbeeld de gebeurtenis 'druk op een fysieke knop' is configureerbaar door de maker van de workflow; hij kan specificeren welke van de fysieke knoppen, die op de smartphone beschikbaar zijn, ingedrukt moet worden. Enkele voorbeelden van gebeurtenissen zullen meer in detail besproken worden.

1. Een gebeurtenis kan een gebruikersinteractie met de smartphone voorstellen, zoals, onder andere:
 - (a) **Knoppen en knoppencombinaties:** de maker van de workflow wil inspelen op het feit dat de gebruiker op een knop of knoppencombinatie induwt. Er zijn verschillende mogelijke knoppen zoals:
 - i. Fysieke knop: bijvoorbeeld de home knop, eens de gebruiker deze fysieke knop induwt, vindt gebeurtenis plaats.
 - ii. Knop op het scherm: een knop wordt weergegeven op het scherm. Deze gebeurtenis vindt plaats wanneer de gebruiker het scherm op de juiste plaats aanraakt, wat dan gelijk staat met op de knop te duwen.
 - iii. Een combinatie van knoppen: tijdens het uitvoeren van deze gebeurtenis, moet de gebruiker een sequentie van knoppen (fysieke of op het scherm) indrukken. Indien de gebruiker de correcte knoppen in de correcte volgorde induwt, vindt deze gebeurtenis plaats.

- (b) **Aanrakingen en gestures:** de maker van de workflow wil inspelen op het feit dat de gebruiker het scherm aanraakt of een gesture op het scherm uitvoert. Verschillende soorten aanrakingen bestaan, zoals:
- i. Aanraken: de gebruiker hoeft enkel het scherm aan te raken met een vooraf bepaald aantal vingers. Eens de gebruiker bij het uitvoeren van de gebeurtenis het scherm met het juiste aantal vingers aanraakt, vindt deze gebeurtenis plaats.
 - ii. Gesture uitvoeren: de gebruiker moet het scherm met een bepaald aantal vingers aanraken en hierna een welbepaalde beweging uitvoeren, bijvoorbeeld van de linkerkant van het scherm naar de rechterkant glijden. Deze gebeurtenis vindt plaats wanneer de gebruiker de gesture correct uitvoert.

Voorbeelden van knoppen en knopcombinaties



Voorbeelden van aanrakingen en gestures

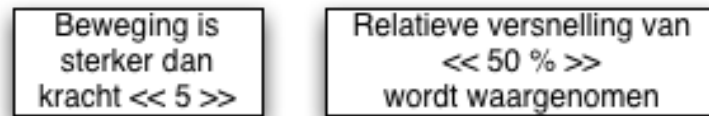


Figuur 14: Grafische weergave van gebeurtenissen die een gebruikersinteractie met de smartphone voorstellen. De variabelen, die de maker van de workflow kan configureren in een gebeurtenis, worden voorgesteld tussen de symbolen << en >>

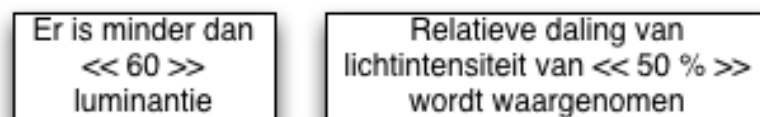
2. Een gebeurtenis kan sensorafhankelijk zijn, zoals:

- (a) **Bewegingen:** de maker van een workflow wil inspelen op een beweging die het toestel ondergaat. Deze beweging kan aan verschillende criteria moeten voldoen, zoals:
 - i. Sterker dan een zekere kracht: de gevoelde beweging moet heviger zijn dan een opgegeven kracht, eens deze beweging plaatsgevonden heeft, vindt de gebeurtenis plaats
 - ii. Een plots verschil van sterkte van beweging, groter dan een zekere waarde: bij het starten van het uitvoeren van de gebeurtenis wordt de beginwaarde waargenomen. Hierna moet deze sterkte van beweging (bijvoorbeeld) halveren. Eens de sterkte gehalveerd is, vindt deze gebeurtenis plaats.
- (b) **Lichtsterkte:** de maker van een workflow wil inspelen op een variatie in lichtintensiteit. Deze daling of stijging kan waargenomen worden op verschillende manieren, bijvoorbeeld:
 - i. Lichtintensiteit moet kleiner zijn dan een zekere drempelwaarde: wanneer de lichtintensiteit lager wordt dan de bepaalde waarde, zal deze gebeurtenis plaatsvinden.
 - ii. Lichtintensiteit moet dalen met een zeker percentage: bij het starten van het uitvoeren van de gebeurtenis, wordt de beginwaarde van de lichtsterkte waargenomen; hierna moet deze lichtsterkte (bijvoorbeeld) halveren. Eens de sterkte gehalveerd is, vindt deze gebeurtenis plaats.

Voorbeelden van bewegingen



Voorbeelden van lichtsterkte



Figuur 15: Grafische weergave van gebeurtenissen die sensorafhankelijk zijn. De variabelen, die de maker van de workflow kan configureren in een gebeurtenis, worden voorgesteld tussen de symbolen << en >>

3. Een gebeurtenis kan ook netwerk afhankelijk zijn, zoals:

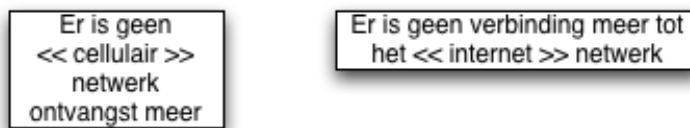
- (a) **De smartphone ontvangt een nieuwe SMS:** de maker van de workflow wil inspelen op een SMS die ontvangen wordt door de smartphone; de SMS moet echter voldoen aan een bepaald criterium; voorbeelden van criteria zijn:
 - i. Van een zekere persoon: de SMS moet vanaf een welbepaald nummer verzonden zijn. Eens de smartphone een boodschap van het gegeven nummer ontvangt, vindt de gebeurtenis plaats.
 - ii. Met zekere sleutelwoorden in de boodschap: de ontvangen boodschap moet één of meerdere sleutelwoorden bevatten. Eens de smartphone deze boodschap met de specifieke sleutelwoorden ontvangt, vindt de gebeurtenis plaats.
- (b) **De smartphone heeft geen netwerkconnectie meer:** in een applicatie wil de maker inspelen op het wegvallen van het netwerk. Een smartphone kan verbonden zijn aan verschillende netwerken, enkele voorbeelden zijn:
 - i. Cellulair netwerk: de gebeurtenis vindt plaats wanneer de smartphone geen ontvangst meer heeft van het cellulaire netwerk.

- ii. Geen verbinding met internet: de netwerkconnectie, die leidt tot het internet, is weggefallen; een smartphone kan verbonden zijn aan de hand van een 3G netwerk en/of een WIFI connectie. Alvorens er geen connectie meer gemaakt kan worden met internet, moeten beide connecties weggefallen zijn. Eens er geen netwerkconnectie aanwezig meer is, zal deze gebeurtenis plaatsvinden.

Voorbeelden van het ontvangen van een SMS



Voorbeelden van netwerkconnectie



Figuur 16: Grafische weergave van gebeurtenissen die netwerk afhankelijk zijn. De variabelen, die de maker van de workflow kan configureren in een gebeurtenis, worden voorgesteld tussen de symbolen << en >>

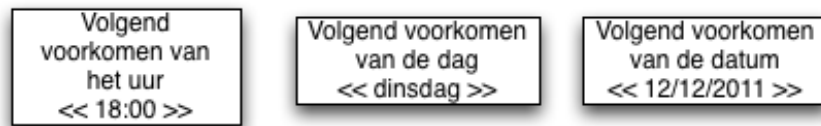
4. Een gebeurtenis kan tijdsgebonden zijn, bijvoorbeeld:

- (a) **Een bepaald tijdstip:** aan de hand van deze gebeurtenis, kan de maker van een applicatie een tijdstip gebruiken. Een tijdstip kan verschillende interpretaties hebben, zoals:
 - i. Het volgende voorkomen van dit tijdstip: indien er bijvoorbeeld een uur gegeven is, zoals 18:00 uur, zal de volgende keer dat het 18:00 uur is, deze gebeurtenis plaatsvinden.
 - ii. Op een specifieke dag in de week: de maker wil beschrijven dat een gebeurtenis plaatsvindt, bijvoorbeeld op dinsdag.

- iii. Op een specifieke datum: de maker wil beschrijven dat een gebeurtenis plaatsvindt op een zekere datum, bijvoorbeeld op 12 december 2011.
- (b) **Een zekere tijdsduur is verlopen:** aan de hand van deze gebeurtenis kan de maker van een applicatie tijdsverloop gebruiken, zoals:
 - i. 3 uur na het starten van het uitvoeren van deze gebeurtenis vindt de gebeurtenis plaats.

Eens de vooropgestelde tijdsspanne verlopen is, zal deze gebeurtenis plaatsvinden.

Voorbeelden van een bepaald tijdstip



Voorbeelden van verlopen tijdsduur



Figuur 17: Grafische weergave van gebeurtenissen die tijdsgebonden zijn. De variabelen, die de maker van de workflow kan configureren in een gebeurtenis, worden voorgesteld tussen de symbolen << en >>

4.2 Samengestelde gebeurtenissen

Definitie: een samengestelde gebeurtenis is een combinatie van verschillende gebeurtenissen, gecombineerd aan de hand van een aantal gegeven samenstellingsoperatoren of schakelaars

1. **EN schakelaar:** verschillende gebeurtenissen kunnen aan de hand van de *EN schakelaar* verbonden worden en kunnen zich voordoen in om het even welke volgorde. Eens alle gebeurtenissen hebben plaatsgevonden, is de samengestelde gebeurtenis afgerond.
2. **OF schakelaar:** verschillende gebeurtenissen kunnen aan de hand van een *OF schakelaar* verbonden worden. Zodra één van deze gebeurtenissen heeft plaatsgevonden, is de samengestelde gebeurtenis afgerond.
3. Combinatie van de **EN en OF schakelaar:** verschillende samengestelde gebeurtenissen aan de hand van de *EN schakelaar*, kunnen met elkaar verbonden worden aan de hand van een *OF schakelaar*. Indien alle gebeurtenissen van één *of tak* gebeurd zijn, is deze samengestelde gebeurtenis afgerond.

Voorbeelden: enkele voorbeelden van samengestelde gebeurtenissen aan de hand van *EN of OF schakelaars* zullen meer in detail besproken worden

1. **Er is plots veel lawaai rondom het toestel en de smartphone voelt een hevige beweging:** in deze samengestelde gebeurtenis herkennen we 2 gebeurtenissen, namelijk:
 - (a) Er is plots veel lawaai rondom het toestel
 - (b) De smartphone voelt een hevige beweging

Deze twee gebeurtenissen zijn verbonden aan de hand van de *EN schakelaar*. De achterliggende gedachte achter dit voorbeeld is dat, vooreer deze samengestelde gebeurtenis plaatsvindt, het toestel 2 verschillende dingen moet waarnemen: veel lawaai en een hevige beweging. Deze 2 gebeurtenissen kunnen in een onafhankelijke volgorde gebeuren, de 2 mogelijke scenario's zijn:

- (a) Er wordt eerst veel lawaai rondom het toestel waargenomen; pas wanneer de smartphone hevige beweging voelt wordt deze samengestelde gebeurtenis afgerond
- (b) Er wordt eerst een hevige beweging door de smartphone gevoeld, wanneer er later plots veel lawaai rondom het toestel waargenomen wordt, is deze gebeurtenis afgerond.

2. **De gebruiker drukt op een knop en een tijdsduur moet verlopen zijn:** in deze samengestelde gebeurtenis, herkennen we 2 gebeurtenissen, namelijk

- (a) de gebruiker drukt op een knop
- (b) een tijdsduur moet verlopen zijn

Deze twee gebeurtenissen zijn verbonden aan de hand van de *EN schakelaar*. De achterliggende gedachte achter dit voorbeeld is dat, vooraleer deze samengestelde gebeurtenis plaatsvindt, de gebruiker op een knop moet duwen maar dat er ook een minimale tijdsduur moet verlopen zijn. Deze 2 gebeurtenissen kunnen in onafhankelijke volgorde gebeuren, de mogelijke scenario's zijn

- (a) Indien de gebruiker op de knop drukt alvorens de tijdsduur verlopen is, zal de gebeurtenis pas beëindigd zijn wanneer de tijdsduur verlopen is
- (b) Indien de tijdsduur verlopen is maar de gebruiker nog niet op de knop heeft gedruwd, zal deze samengestelde gebeurtenis pas plaatsvinden wanneer de gebruiker op de knop drukt

3. **De smartphone ontvangt een nieuwe SMS of er is geen cellulaire ontvangst:** in deze samengestelde gebeurtenis herkennen we 2 gebeurtenissen, namelijk:

- (a) De smartphone ontvangt een nieuwe SMS
- (b) Er is geen cellulaire ontvangst

Deze twee gebeurtenissen zijn verbonden aan de hand van de *OF schakelaar*. De achterliggende gedachte bij dit voorbeeld is als er geen cellulaire ontvangst aanwezig is, er geen nieuwe SMS ontvangen kan worden; aan de hand van de *OF schakelaar*, kunnen we voorkomen dat bij geen cellulaire ontvangst deze samengestelde gebeurtenis eeuwig vastzit.

Zolang er wel cellulaire ontvangst is, is er wel kans op een binnenkomende SMS.

4. **Een zekere tijdsduur is verlopen of de gebruiker geeft een correcte code in:** in deze samengestelde gebeurtenis herkennen we 2 gebeurtenissen, namelijk:

- (a) Een zekere tijdsduur is verlopen
- (b) De gebruiker geeft een correcte code in

Deze twee gebeurtenissen zijn verbonden aan de hand van de *OF schakelaar*. De betekenis achter deze samengestelde gebeurtenis is dat, als de gebruiker vroegtijdig de gebeurtenis wil laten plaatsvinden, hij dit kan doen door de correcte code in te geven. Indien de gebruiker geen correcte code ingeeft, zal de gebeurtenis pas plaatsvinden wanneer de tijdsduur verlopen is.

4.2.1 Niet schakelaar

Er is gekozen om geen *niet schakelaar* toe te laten bij het maken van samengestelde gebeurtenissen, omdat de semantiek achter een genegeerde gebeurtenis niet altijd triviaal is.

Wat de betekenis van bijvoorbeeld een tijdsduur die genegeerd wordt aan de hand van de *niet schakelaar*? Indien de gebeurtenis is “na een tijdsverloop van 5 seconden” is de negatie dan: er zijn geen 5 seconden verlopen? Maar neemt deze genegeerde gebeurtenis dan plaats na 0, 1, 2, 3 of 4 seconden? Een tweede voorbeeld: wat als de gebeurtenis “de smartphone ontvangt een SMS” genegeerd wordt. Het is technisch niet mogelijk dat de smartphone een ononderbroken stroom van SMS'en binnenkrijgt. Tussen 2 boodschappen is er altijd een (kleine) tussenpauze. Willen we hiermee dan uitdrukken dat, geen SMS'en ontvangen langer is dan deze pauzetijd? Deze betekenis is ook niet eenduidig.

Er is dus gekozen om geen *niet schakelaar* te voorzien. Indien er nood is aan het tegengestelde van een specifieke gebeurtenis, zoals: “de smartphone voelt plots een hevige beweging”, dan moet er een nieuwe gebeurtenis voorzien worden in de workflow, bijvoorbeeld: “de smartphone voelt dat de beweging plots wegvalt”.

Op deze manier worden ambiguïteiten vermeden.

4.2.2 Logische combinaties

Er is gekozen om de maker van de workflow enkel een beperkte mogelijkheid te laten om verschillende gebeurtenissen te combineren aan de hand van EN en OF schakelaars. De maker van de workflow kan verschillende gebeurtenissen combineren aan de hand van de EN schakelaar. Daarna kan hij ook EN gecombineerde gebeurtenissen combineren met een OF schakelaar. Bijvoorbeeld:

$(A \text{ EN } C) \text{ OF } (B \text{ EN } C)$

Het is dus niet mogelijk om OF gecombineerde gebeurtenissen te combineren aan de hand van een EN schakelaar. Bijvoorbeeld:

$(A \text{ OF } B) \text{ EN } C$

De voorbeelden hierboven zijn logisch equivalent aan de hand van de regel van de distributiviteit.

Niet alle logische combinaties van gebeurtenissen zijn logisch equivalent met de gekozen combinatie van EN en OF schakelaars. Het volgend voorbeeld kan dus niet (zonder de niet schakelaar [31]) herschreven worden in onze beperkte mogelijkheid logische combinatie:

$(A \text{ OF } B) \text{ EN } (C \text{ OF } D)$

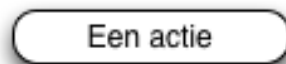
Er is dus gekozen om verschillende EN schakelaars te combineren aan de hand van OF schakelaars, omdat dit in onze ogen een intuïtieve manier is voor het combineren van verschillende gebeurtenissen, die door niet programmeurs zonder training, kan begrepen worden.

4.3 Acties

Definitie: acties worden uitgevoerd door de smartphone zelf; deze kunnen zelfstandig uitgevoerd worden zonder gebruikersinteractie, bijvoorbeeld:

1. Neem geluid op
2. Speel muziek of video af
3. Laat de smartphone vibreren
4. Stuur een SMS

Representatie: een actie wordt in onze taal voorgesteld aan de hand van een afgeronde rechthoek (zie figuur 18).



Figuur 18: Grafische weergave van een actie

Voorbeelden: enkele voorbeelden van acties worden nader besproken:

1. **Speel muziek:** deze actie zal specifiek muziekstuk afspelen; dit muziekstuk kan door de maker van de workflow geselecteerd worden. Eens, bij het uitvoeren van de workflow, het muziekstuk volledig afgespeeld is, is deze actie afgerond.
2. **Stuur een SMS:** deze actie zal een SMS versturen; bij het maken van de workflow kan de maker een nummer en boodschap aan deze actie koppelen, zodat bij het uitvoeren van deze actie deze laatste volledig automatisch door de smartphone uitgevoerd kan worden. Eens, bij het uitvoeren van de workflow, de SMS verzonden is, is deze actie afgerond.
3. **Start een applicatie:** deze actie zal een andere applicatie starten; bij het maken van de workflow, kan de maker deze applicatie selecteren. Eens, bij het uitvoeren van de workflow, de applicatie afgesloten wordt, wordt deze actie afgerond.

4. **Stuur een notificatie:** deze actie zal een notificatie sturen naar een andere applicatie; bij het maken van de workflow, kan de maker de applicatie selecteren waarnaar de notificatie gestuurd wordt, alsook de gezonden boodschap kan ingevuld worden. Eens, bij het uitvoeren van de workflow, deze boodschap verzonden is, is deze actie afgerond.
5. **Neem een foto:** deze actie zal, met de camera aanwezig op de smartphone, een foto nemen en opslaan in een album. De maker van de workflow kan bepalen in welk album deze afbeelding opgeslagen moet worden. Eens, bij het uitvoeren van de workflow, de foto genomen en opgeslagen is, is deze actie afgerond.

Onze workflowtaal bevat een brede waaier van acties en gebeurtenissen, die overeenkomen met uiteenlopende functionaliteiten, die op een smartphone aanwezig zijn.

Nieuwe functionaliteiten, die op de smartphone beschikbaar worden, kunnen aan de workflowtaal toegevoegd worden om de workflowtaal te verrijken. Samengestelde acties bestaan niet; acties worden samengesteld aan de hand van workflow patronen geïntroduceerd in sectie 5.

4.3.1 Stopgebeurtenissen op acties

Acties worden uitgevoerd totdat hun normaal verloop beëindigd is. Bijvoorbeeld, indien een actie het afspelen van een muziekstuk is, is deze actie slechts afgelopen wanneer het muziekstuk volledig is afgespeeld.

Smartphones worden in verschillende omgevingen gebruikt; het kan dus mogelijk zijn dat dit muziekstuk storend is voor de gebruiker in zijn actuele situatie. De maker van een workflow zou aan de hand van een gebeurtenis de mogelijkheid willen geven aan de gebruiker om een actie vroegtijdig te stoppen. Om een muziekstuk te stoppen, kunnen we verschillende manieren bedenken, namelijk:

1. De gebruiker drukt op een stopknop
2. De gebruiker schudt met zijn smartphone
3. De gebruiker raakt het scherm met 2 vingers aan
4. Er is veel lawaai in de omgeving
5. ...

We merken op dat al deze scenario's om de actie te stoppen gebeurtenissen zijn. Indien we deze gebeurtenissen met een actie koppelen tot een nieuw type actie, zullen we veel combinaties nodig hebben.

Bijvoorbeeld:

1. Speel muziek tot de gebruiker op de stopknop duwt ¹
2. Speel muziek tot de gebruiker met zijn smartphone schudt
3. Enzoverder...

Het kan ook zijn dat de maker van de workflow verschillende stopcondities wil definiëren op één actie, bijvoorbeeld:

1. De actie wordt vroegtijdig gestopt indien de gebruiker op de stopknop drukt en met zijn smartphone schudt
2. De actie wordt vroegtijdig gestopt indien de gebruiker het scherm met 2 vingers aanraakt of er veel lawaai in de omgeving is

Wat we merken is dat we een hele hoop bouwblokken krijgen die dezelfde actie bevatten, maar met telkens een andere stopgebeurtenis of combinatie van stopgebeurtenissen.

We kunnen ons inbeelden dat zelfs als een hele waaier acties gecombineerd met stopgebeurtenissen zou opgenomen worden in de workflowtaal, de maker van een workflow toch nog een actie zou willen combineren met een andere stopconditie

Wat we dus willen, is dat de maker van de workflow zelf stopcondities op een actie kan vastleggen. Dit betekent dat de maker van een workflow door een gebeurtenis een actie vroegtijdig kan stoppen.

Definitie: een stopgebeurtenis is een samengestelde of eenvoudige gebeurtenis, die vastlegt hoe een actie vroegtijdig afgerond kan worden. Indien de (samengestelde) gebeurtenis zich niet voordoet, wordt de actie op zijn normale wijze afgerond.

¹ Merk op dat de gebeurtenis "druk op een knop" aan de hand van een sequentie patroon (zie sectie 5) achter de actie "speel muziek" plaatsen een andere betekenis heeft, namelijk de muziek speelt helemaal ten einde en hierna wordt dan gewacht tot de gebruiker op de stopknop duwt.

4.3.2 Gebeurtenissen als guards op acties

Op een smartphone gebeuren er veel zaken tegelijkertijd; zo kunnen er bijvoorbeeld verschillende applicaties tegelijkertijd draaien. Dit zorgt ervoor dat op elk moment enkele functionaliteiten van de smartphone in gebruik kunnen zijn.

Zo kan bijvoorbeeld een applicatie al muziek aan het afspelen zijn op het moment dat een andere applicatie ook een actie "speel muziek" wilt starten. Indien de maker van een applicatie muziek wil afspelen, maar wil wachten tot de muzikspeler beschikbaar is, kan hij gebruik maken van een guard.² Guards zorgen ervoor dat, voor een actie uitgevoerd wordt, sommige condities moeten voldaan zijn, bijvoorbeeld

1. Er is geen muziek aan het afspelen op de smartphone
2. Er is geen luid lawaai rondom het toestel
3. Er zijn 10 seconden voorbij vooraleer de actie uitgevoerd wordt
4. ...

De maker van een applicatie kan als guard van een actie opnieuw een combinatie van verschillende gebeurtenissen gebruiken zoals bijvoorbeeld:

1. Er is momenteel geen andere muziek aan het afspelen op de smartphone en er is geen luid lawaai rondom het toestel
2. Er zijn 30 seconden voorbij of er is geen luid lawaai rondom het toestel.

Net zoals stopgebeurtenissen, zijn guards eenvoudige of samengestelde gebeurtenissen gebonden aan een actie. Zo krijgt de maker van een workflow de vrijheid om vrijuit acties te combineren met guards. Guards bieden extra uitdrukingskracht bij het beschrijven van een specifieke actie.

Definitie: een guard is een samengestelde of eenvoudige gebeurtenis, die beschrijft welke gebeurtenissen moeten plaatsgevonden hebben, vooraleer een actie uitgevoerd wordt.

² Merk op dat het mogelijk is om verschillende acties tegelijkertijd te laten uitvoeren aan de hand van een parallel patroon (zie sectie 5). Indien de maker van een workflow wil verzekeren dat dezelfde functionaliteit niet tegelijkertijd gebruikt wordt door 2 gebeurtenissen, zou hij deze in een sequentie kunnen plaatsen; bijvoorbeeld: speel muziek 1 en zodra muziek 1 afgelopen is speel muziek 2. Dit heeft een andere betekenis als een guard, die zegt speel muziek 2 enkel als er geen andere applicatie de muzikspeler gebruikt.

5 Control flow en data patronen

De grafische taal, die we voor ogen hebben, biedt een aantal verschillende bouwblokken, die door verschillende patronen kunnen verbonden worden om als eindresultaat een workflow te geven. In de vorige sectie werden de verschillende bouwblokken van de workflow geïntroduceerd. In deze sectie zullen de verschillende patronen besproken worden die ervoor zorgen dat de verschillende bouwblokken gecombineerd kunnen worden om een volledige workflow te vormen.

Elke workflow bestaat uit een start- en eindpunt; dit zal als eerste in deze sectie besproken worden.

Vervolgens worden de control flow patronen besproken, namelijk : sequentie, exclusieve of parallelle takken en iteratie.

Hierna wordt er uitgelegd voor welk data patroon gekozen is [49, 47].

In het laatste deel van deze sectie, wordt uitgelegd hoe er statische en dynamische referenties gemaakt kunnen worden.

Zoals in vorige sectie geïntroduceerd is, bestaan onze bouwblokken uit acties, gebeurtenissen en samengestelde gebeurtenissen. Voorbeelden in deze sectie zullen de notatie en bouwblokken uit onze taal gebruiken.

5.1 Start- en eindknopen

Elke workflow bestaat uit een eind- en beginpunt. Deze geven weer waar de workflow definitie begint en eindigt. Er is gekozen om het mogelijk te maken om aan een startknoop een naam te geven. Deze naam stelt de naam voor van de workflow. Aan de hand van deze naam, kan de maker van de workflow gemakkelijk identificeren aan welke workflow hij aan het werken is.

Voor een startknoop en na een eindknoop is het niet mogelijk om andere patronen toe te voegen.



Figuur 19: Grafische weergave van een start- en eindknoop. De maker van de workflow kan een naam aan een startknoop geven, deze aanpasbare naam wordt voorgesteld tussen de symbolen << en >>

5.2 Sequentieel patroon

In scenario 1 uit sectie 1.2 is het noodzakelijk om verschillende acties en gebeurtenissen na elkaar te kunnen uitvoeren.

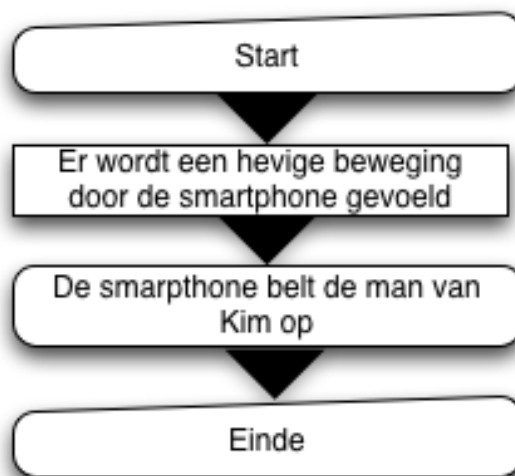
Om dit uit te drukken, maken we gebruik van het sequentie patroon. Dit patroon gaat in een workflow verschillende bouwblokken verbinden en een volgorde aanbrengen. Uit elke bouwblok, dat een element is van de sequentie, vertrekt één pijl naar een volgende activiteit: deze pijl wordt weergegeven op figuur 20 aan de hand van de zwarte driehoek onder een bouwblok en ook onder de startknoop. Wanneer verschillende bouwblokken verbonden zijn, vormen deze een sequentie.

Aan de hand van dit sequentieel patroon, kunnen we nu scenario 1 uit sectie 1.2 uitdrukken (zie figuur 20).

In dit voorbeeld wordt het sequentieel proces gestart na de startknoop. Hierna wordt er gewacht tot een gebeurtenis zich voordoet, namelijk dat de smartphone een hevige beweging ondergaat.

Eens deze gebeurtenis plaats heeft gevonden, wordt er een actie uitgevoerd. Deze actie bestaat uit het automatisch opbellen van de man van Kim.

Eens deze actie afgerond is, is dit proces afgelopen.



Figuur 20: voorbeeld van een sequentieel patroon dat correspondeert met scenario 1 uit sectie 1.2

5.3 Parallele takken

Scenario 2 uit sectie 2.2 begint met het wachten op een hevige neerwaartse beweging. Na het voelen van zo een hevige beweging worden 2 takken tegelijkertijd opgestart. Als voorbeeld voor deze sectie, wordt enkel dit deel uit het scenario behouden.

1. *De smartphone voelt een hevige neerwaartse beweging* Op dit moment gebeuren er 2 zaken tegelijkertijd, namelijk:
 - (a) *De smartphone stuurt een SMS naar de verpleegster van wacht*
 - (b) *De smartphone speelt muziek*

Dit voorbeeld wordt uitgetekend in figuur 21. We zien dat voor het grafisch weergeven van deze parallelle takken, een nieuw control flow patroon geïntroduceerd wordt dat uitgeschreven wordt aan de hand van 2 nieuwe knopen, namelijk: de parallelle start knoop en de parallelle synchronisatie knoop.

5.3.1 Parallele start

Tot nu toe hebben we enkel een sequentie van bouwblokken beschreven. Bij het uitvoeren van een sequentie, wordt er maar één bouwblok tegelijkertijd uitgevoerd.

Indien we nu verschillende deelworkflows tegelijkertijd willen uitvoeren, is een nieuw control flow patroon nodig.

Het beginpunt van de parallelle takken wordt aangeduid aan de hand van de *parallelle start knoop* op afbeelding 21.

Bij het uitvoeren van een workflow, worden er verschillende nieuwe parallelle takken gestart bij het bereiken van de *parallelle start knoop*.

In het voorbeeld op afbeelding 21 wordt de workflow gestart bij de start-knoop. Hierna wordt door middel van een sequentieel patroon de gebeurtenis "de smartphone voelt een hevige neerwaartse beweging" gestart. Eens de gebeurtenis "de smartphone voelt een hevige neerwaartse beweging" plaatsvindt, worden er aan de hand van de parallelle start knoop 2 verschillende subworkflows gestart die in het voorbeeld elk bestaan uit een heel simpel sequentieel patroon, namelijk 2 acties (de smartphone "speelt muziek" en "stuurt een SMS"). Deze 2 deelworkflows worden gelijktijdig uitgevoerd. In het algemeen kunnen deelworkflows bestaan uit alle andere ondersteunde bouwblokken en patronen.

5.3.2 Synchronisatie

Op de figuur 21 zien we dat, na het uitvoeren van de parallelle takken, deze aan de hand van een parallelle synchronisatie knoop weer samensmelten tot eenzelfde tak. Het synchroniseren van parallelle deelworkflows gebeurt enkel indien alle deelworkflows afgelopen zijn. De synchronisatie van de verschillende deelworkflows wordt weergegeven aan de hand van de *parallelle synchronisatie knoop*.

In figuur 21 zien we hoe 2 concurrente deelworkflows gesynchroniseerd worden:

De eerste deelworkflow bestaat uit de actie: “de smartphone speelt muziek”.

De tweede deelworkflow bestaat uit de actie: “de smartphone stuurt een SMS naar de verpleegster van wacht”.

Deze twee deelworkflows worden vervolgens gesynchroniseerd in de *parallelle synchronisatie knoop*.

Beide takken moeten voltooid zijn, alvorens het vervolg van de workflow uitgevoerd wordt.



Figuur 21: voorbeeld van een parallel patroon

5.4 Conditionele takken

Beslissingen in programmeertalen worden meestal gemaakt aan de hand van het "waar" of "niet waar" zijn van een conditie, zoals bijvoorbeeld in een klassiek *als (conditie) dan(...)* *anders(...)* controlestructuur.

Indien de conditie waar is, wordt de *dan tak* uitgevoerd; indien de conditie niet waar is, wordt de *anders tak* uitgevoerd. Zo kunnen we in sectie 3 zien hoe in de kismet grafische script taal, BPMN en UML activiteitendiagrammen het programma anders verloopt naargelang een conditie waar is of niet. In onze voorgestelde taal bestaat, in de context van gebeurtenissen, de notie van "waar" en "niet waar" niet. Hierdoor is het niet mogelijk om beslissingen te nemen naargelang een bouwblok (gebeurtenis of een actie met een stopgebeurtenis) waar is of niet. Er is een manier nodig om uit te drukken dat een deelworkflow conditioneel uit te voeren is.

De voorgestelde bouwblokken (acties, (samengestelde) gebeurtenissen) zijn afgelopen indien de gebeurtenis plaatsvindt of indien de actie afgelopen is. Indien de maker van de workflow nu bij het ontwikkelen van een applicatie volgende situatie wil uitdrukken, moet er een nieuw patroon ingevoerd worden.

1. Indien de gebruiker binnen de 20 seconden op de "home knop" drukt, wordt een video afgespeeld.
2. Indien de gebruiker niet binnen de 20 seconden op de "home knop" heeft gedrukt, wordt een muziekstukje afgespeeld.

We merken op dat, naargelang de gebruiker op de knop duwt of niet, een andere actie uitgevoerd wordt. Ook merken we dat in dit voorbeeld de 2 takken met een gebeurtenis beginnen. In tak 1 is de gebeurtenis "op de home knop duwen". In tak 2 is de gebeurtenis "er zijn 20 seconden verlopen". Naargelang welke gebeurtenis eerst plaatsvindt, wordt een andere deelworkflow uitgevoerd. Indien we dit nu grafisch willen uitdrukken, krijgen we het resultaat weergegeven op figuur 22.

We zien dat voor het grafisch weergeven van deze conditionele takken, een nieuw control flow patroon geïntroduceerd wordt dat uitgeschreven wordt aan de hand van verschillende nieuwe deelworkflows, namelijk: de voorwaarde en de conditionele uit te voeren deelworkflow.

5.4.1 Voorwaarden

Indien we verschillende conditionele takken willen opstellen, moet er duidelijk weergegeven worden waar de *voorwaarden* beginnen. Hiervoor werd er gebruik gemaakt van de *conditionele start* knoop. Na deze knoop worden er verschillende voorwaarden beschreven, deze voorwaarden kunnen bestaan uit de verschillende patronen en bouwblokken.

Om dit duidelijk weer te geven aan de maker van de workflow, moet er een duidelijk onderscheid aanwezig zijn tussen de *voorwaarde* en de *conditionele uit te voeren deelworkflow*. Hiervoor wordt er gebruik gemaakt van een *eind voorwaarde* knoop (zie figuur 22) in elke tak.

Voorwaarden worden dus beschreven tussen de *conditionele start* knoop conditie en de *eind voorwaarde* knoop. Er wordt gekozen om een maximale expressiviteit te geven aan de makers van de workflow, om voorwaarden te beschrijven. Voorwaarden kunnen bestaan uit deelworkflows, die bestaan uit acties en (samengestelde) gebeurtenissen, die verbonden worden door de verschillende voorgestelde patronen. Bij het uitvoeren van conditionele takken, is de semantiek dat alle voorwaarden tegelijkertijd gestart worden en eens één voorwaarde voltooid is (met andere woorden één tak de *eind voorwaarde knoop* bereikt) alle andere voorwaarden gestopt worden. Eens dit laatste gebeurd is, wordt de *conditionele uit te voeren deelworkflow*, de tak van de eerst bereikte *eind voorwaarde knoop*, uitgevoerd.

In het voorbeeld op figuur 22 betekent dit dat:

1. Indien de gebruiker binnen de 20 seconden op de knop duwt, wordt *eind voorwaarde 1* als eerste bereikt en wordt de andere voorwaarde stopgezet (met andere woorden, de gebeurtenis, "er zijn 20 seconden verlopen" kan niet meer plaatsvinden). Hierna wordt een video afgespeeld.
2. Indien de gebruiker niet binnen de 20 seconden op de home knop duwt, wordt *eind voorwaarde 2* als eerste bereikt en wordt de andere voorwaarde stopgezet (met andere woorden, de gebeurtenis, "op de home knop duwen" kan niet meer plaatsvinden). Hierna wordt muziek afgespeeld.

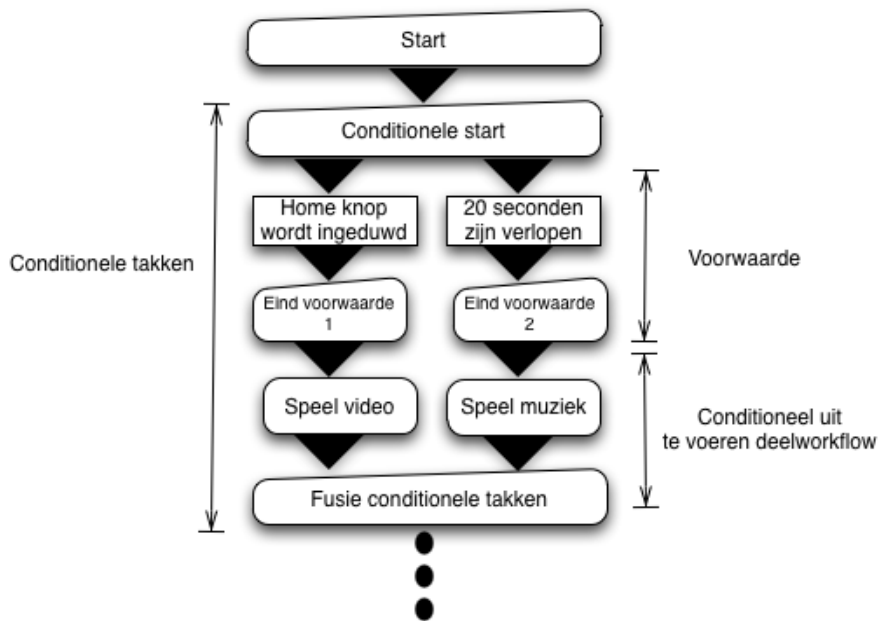
5.4.2 Fusie

Op de figuur 22 zien we dat de verschillende conditionele takken en hun specifieke deelworkflows aan de hand van de *fusie van conditionele takken knoop* weer samenkomen. De semantiek hierachter is dat het vervolg van de workflow voor alle verschillende conditionele takken hetzelfde is. Bij het uitvoeren van de workflow, kan er maar één conditionele tak afgelopen worden (omdat er maar één voorwaarde, als eerste kan plaatsvinden en dus slechts één *conditionele uit te voeren deelworkflow* uitgevoerd kan worden).

Hierdoor kan de *fusie van conditionele takken knoop* maar langs één tak bereikt worden.

Eens dat de *fusie van conditionele takken knoop* bereikt is, wordt de rest van de workflow uitgevoerd.

In het voorbeeld op figuur 22 betekent dit dat, als de actie “speel video” of de actie “speel muziek” afgerond is, de rest van de workflow uitgevoerd wordt.



Figuur 22: Voorbeeld van een conditioneel patroon

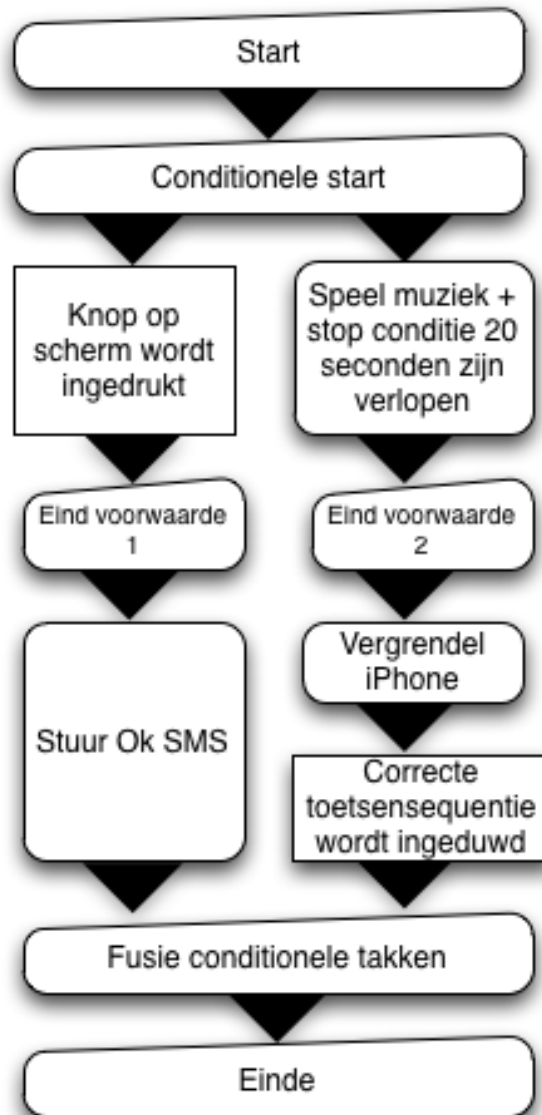
5.4.3 Uitgebreid voorbeeld

Het begin van scenario 2 uit sectie 1.2 werd in vorige sectie behandeld, namelijk het parallel gebeuren van 2 takken. In dit scenario worden twee conditionele takken gebruikt.

Het scenario is het volgende :

- 1 De smartphone speelt muziek.
Tijdens het afspelen van het muziekstuk, kunnen zich 2 mogelijkheden voordoen:
 - 1.1 De gebruiker drukt op de 'stop' knop op het scherm, het muziekstuk wordt automatisch stopgezet
 - 1.1.1 Hierna wordt er een SMS naar de verpleegster van wacht gestuurd om te melden dat alles in orde is.
 - 1.2 De 20 seconden zijn voorbij zonder dat er op de 'stop' knop werd gedrukt, het muziekstukje wordt automatisch stilgezet;
 - 1.2.1 Hierna vergrendelt het toestel zich.
 - 1.2.2 Hierna kan enkel de verpleegster van wacht het toestel ontgrendelen.

Deze conditionele takken kunnen uitgedrukt worden aan de hand van ons conditioneel patroon. Dit voorbeeld wordt uitgewerkt in figuur 23. Er is gebruik gemaakt van een stop gebeurtenis op de "speel muziek" die de voorwaarde voorstelt van een conditionele tak. Dit betekent dat voorwaarde 1.2 (de rechter voorwaarde op de afbeelding) plaatsvindt na 20 seconden en het muziekstuk gestopt wordt. Indien de gebruiker binnen 20 seconden op de knop op het scherm duwt, worden alle andere voorwaarden gestopt. In dit voorbeeld betekent het dat de actie die muziek speelt wordt stopgezet.



Figuur 23: Uitgebreid voorbeeld van een conditionele tak

5.5 Iteratie

In scenario 3 uit sectie 1.2 hebben we ook een iteratief patroon nodig. Bij een iteratie in andere workflow talen, wordt herhaaldelijk dezelfde deelworkflow uitgevoerd als de *loopconditie* waar is. Een voorbeeld van dergelijk iteratief patroon wordt uitgebeeld in sectie 3.3.1 over UML activiteitendiagrammen. In onze voorgestelde taal bestaat, opnieuw in de context van gebeurtenissen, de notie van "waar" en "niet waar" niet, een klassieke *while (cond) do (body)* is dus niet aangewezen. Daarom is er gekozen om te werken aan de hand van voorwaarden, net zoals in conditionele takken uit vorig hoofdstuk. Er worden 2 voorwaarden gebruikt om een iteratie uit te drukken, namelijk:

- een loop voorwaarde: indien deze voorwaarde als eerste afgelopen is, wordt de body uitgevoerd; wanneer deze body afgelopen is, worden de voorwaarden weer concurrent van elkaar gestart.
- een stop voorwaarde: als deze voorwaarde als eerste afgelopen is, wordt de iteratie stopgezet en wordt de rest van de workflow gestart.

De verschillende knopen en deelworkflows (de stop en loop voorwaarde en de body) worden op figuur 24 weergegeven.

Het kiezen tussen de stop en loop voorwaarde gebeurt op dezelfde manier als bij de conditionele takken, namelijk: de 2 voorwaarden worden bij het bereiken van de *conditionele start knoop* tegelijkertijd concurrent van elkaar opgestart. Afhankelijk van welke voorwaarde als eerste afgerond is, wordt voor de stop of loop voorwaarde gekozen.

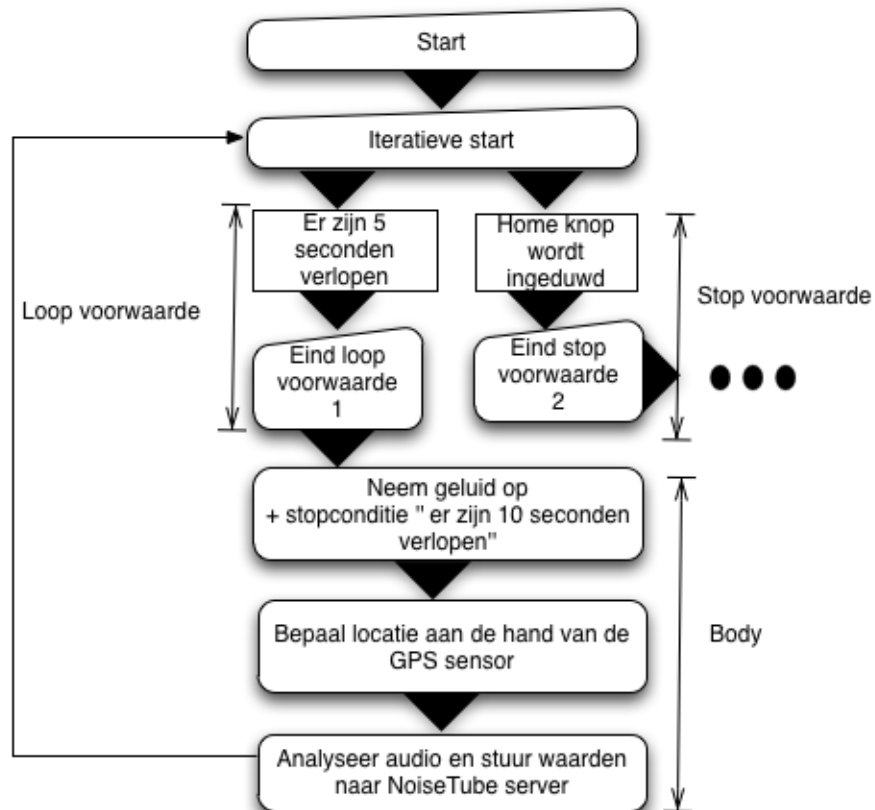
Zowel de voorwaarden als de iteratieve deelworkflow, kunnen bestaan uit alle verschillende beschikbare patronen en bouwblokken.

In figuur 24 wordt de volgende iteratie uitgebeeld:.

1. Er worden 2 voorwaarden gestart
 - 1.1 De gebruiker heeft 5 seconden om op de "home knop" te duwen; Indien de gebruiker dit doet, wordt het volgend sequentieel patroon, dat de iteratieve deelworkflow van de iteratie voorstelt, gestart:
 - 1.1.1 Een actie wordt uitgevoerd die gedurende 10 seconden geluid gaat opnemen
 - 1.1.2 Hierna wordt de actie uitgevoerd die de huidige locatie gaat bepalen aan de hand van de GPS sensor
 - 1.1.3 Hierna wordt de actie uitgevoerd die de data analyseert en verstuurt naar de NoiseTube server.
Eens de data verstuurd zijn, wordt er weer een keuze gemaakt tussen de loop en stop voorwaarde.

1.2 Er zijn 5 seconden verlopen.

Indien deze gebeurtenis als eerste gebeurt is de iteratie afgelopen en wordt de rest van de workflow gestart.



Figuur 24: Voorbeeld van een iteratief patroon

5.6 Variabelen

Om data door te geven tussen de verschillende bouwblokken, worden er verschillende workflow patronen voorgesteld. Deze patronen worden gecatalogeerd in volgende groepen [29, 46]:

1. Data interactie (data interaction): aan de hand van dit patroon, gaan er data gestuurd worden over de pijlen, die de patronen tussen bouwblokken voorstellen. Deze data kunnen dan de verdere uitvoering van de workflow beïnvloeden. Een conditie bij een *exclusieve of* zou bijvoorbeeld kunnen afhangen van de data aanwezig. Data die doorgestuurd worden over de transities, kunnen ook dienen voor de interne uitvoering van de volgende activiteit op de transitie.
2. Data overdracht (data transfer): dit patroon kunnen we zien als een interactie patroon en als uitbreiding op het vorige patroon; in dit patroon gaat er gekeken worden hoe de data behandeld moeten worden tussen de verschillende bouwblokken (bijvoorbeeld, indien data gestuurd moet worden als een referentie, als een kopie ...). Aan de hand van dit patroon kunnen de verschillende manieren waarop data overgedragen worden tussen de verschillende bouwblokken gedefinieerd worden.

Er is gekozen om deze patronen niet te gebruiken, omdat we een zo eenvoudig mogelijk gebruikersinterface willen bieden.

Op figuur 25 zien we hoe de maker van een workflow een sequentiële vragenlijst wil opbouwen aan de hand van een data overdracht patroon. Op het einde van de vragenlijst wil de maker van de workflow alle antwoorden doorsturen naar een server. Dit wil zeggen dat hij, bij elk bouwblok, het antwoord moet doorsturen over de sequentiële pijl, zodat de informatie tot het laatste bouwblok gestuurd kan worden. Bij elke nieuwe vraag, komt er een nieuw antwoord, op het einde van dit voorbeeld moet de maker van een workflow 4 antwoorden doorsturen naar de actie.

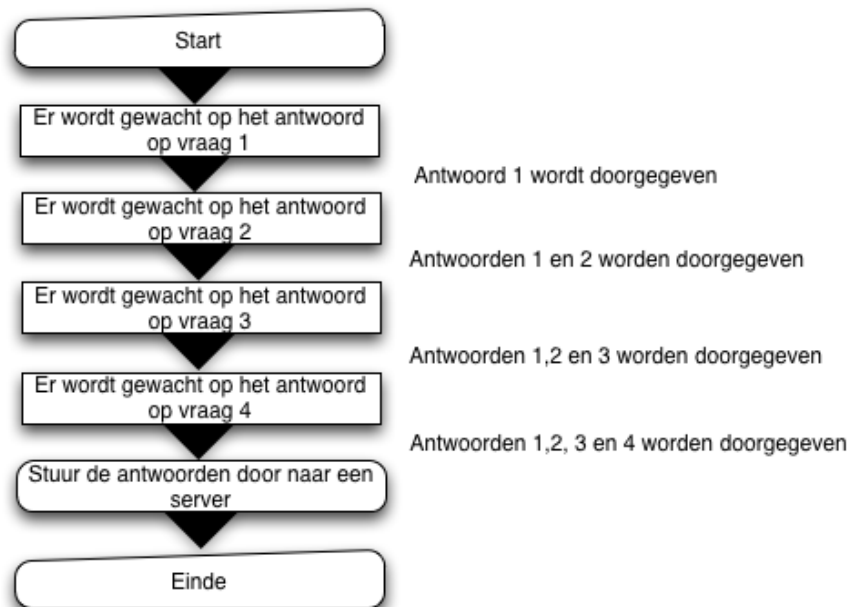
In ons project is er gekozen voor een derde soort patroon, namelijk

- 3 Data zichtbaarheid (data visibility): dit patroon maakt gebruik van een globale omgeving waar informatie opgeslagen kan worden. In elke bouwblok is het mogelijk om informatie op te slaan in en op te halen uit deze omgeving.

Bij data zichtbaarheid is het niet meer noodzakelijk om te kunnen communiceren van 2 bouwblokken; elke bouwblok kan aan alle data in de omgeving. Er werd gekozen om, aan de hand van een omgeving te werken omdat, dit ook dicht bij de manier van werken staat van niet programmeurs. Zo worden in veel verschillende opslagmethoden data bijgehouden aan de hand

van een keyword. Zo wordt in een telefoonboek, aan de hand van een naam (die het keyword is) een telefoonnummer opgezocht. In een woordenboek wordt een woord (dat het keyword is) opgezocht om de betekenis of synoniem hiervan te vinden. Zo worden dus data bijgehouden en gebonden aan een keyword; deze keywords zijn de variabelen. Deze variabelen worden in de voorgestelde taal gebruikt om data tussen de verschillende bouwblokken te delen. Elk bouwblok kan nieuwe variabelen in de globale omgeving toevoegen en opzoeken.

Het nadeel van deze aanpak is dat het mogelijk is dat een variabele in een bouwblok gebruikt wordt en dat deze nog niet aanwezig in de omgeving. Er moet dus nagegaan worden indien alle variabelen, die gebruikt worden op dat ogenblik, ook aanwezig zijn in de omgeving. Een oplossing voor dit probleem wordt voorgesteld in sectie 8.



Figuur 25: Voorbeeld van een sequentieel vragenlijst

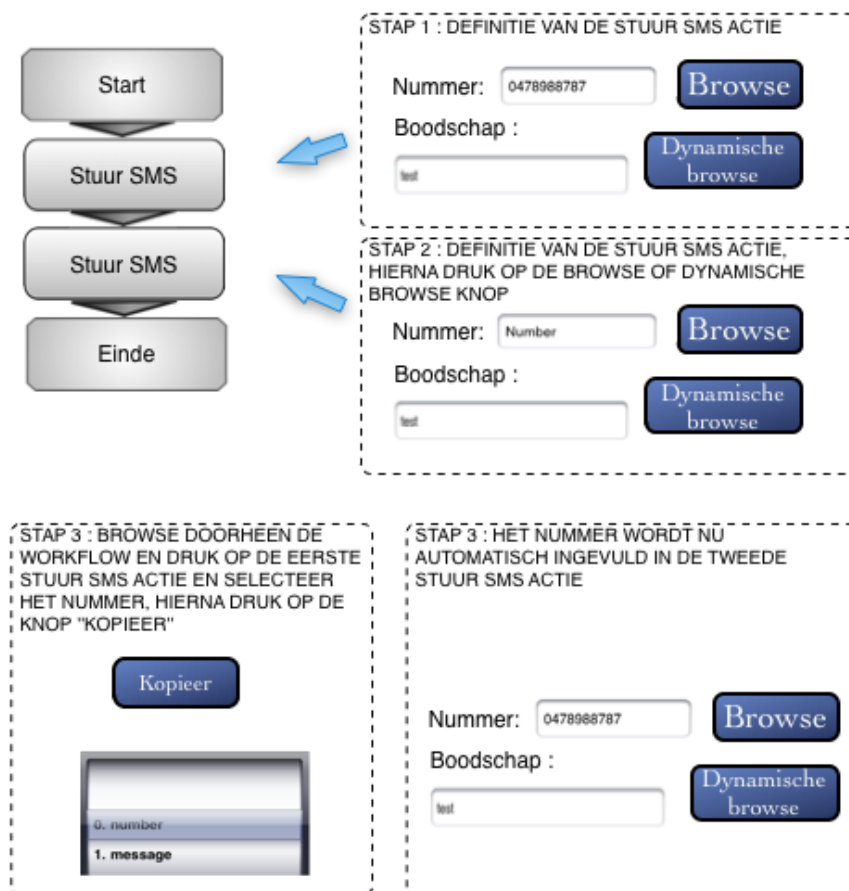
5.7 Browsen van reeds bestaande configuratieparameters van bouwblokken

Bij het definiëren van een workflow, zou een de maker van de workflow 2 SMS'en willen sturen in 2 verschillende acties. Hiervoor moet er een gsm nummer ingevuld worden. De maker van de workflow zou 2 maal het zelfde nummer kunnen ingeven.

Een andere oplossing zou zijn dat de maker van de workflow, bij het ingeven van het 2^{de} nummer, doorheen de bouwblokken zou kunnen browsen en het reeds gebruikte gsm nummer kan overnemen. Dit voorkomt dat de maker van de workflow 2 maal het telefoonnummer hoeft in te geven.

Het browsen van configuratieparameters van bouwblokken laat de maker van de workflow toe om, bij het invullen van een veld, doorheen de workflow te browsen. Eens de maker van de workflow een data-element heeft geselecteerd uit een ander element, wordt dit automatisch gekopieerd naar het veld dat de maker van de workflow aan het invullen was.

Een voorbeeld van hoe het browsen naar een reeds ingevuld telefoonnummer werkt, wordt afgebeeld in figuur 26.



Figuur 26: Browsen van gegevens

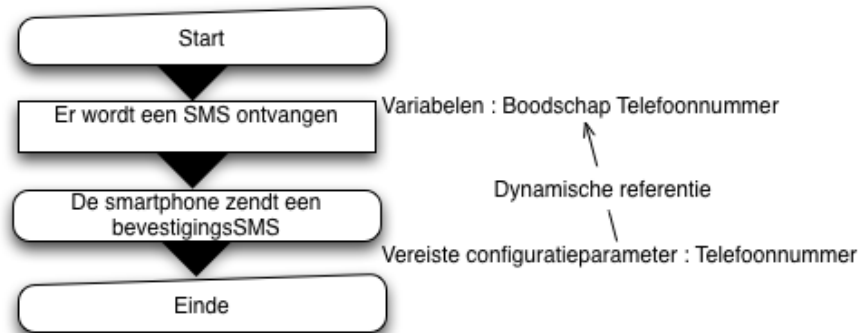
5.7.1 Dynamische referentie

Configuratieparameters van bouwblokken worden niet altijd ingevuld bij het definiëren van de workflow.

Bijvoorbeeld, bij het uitvoeren van een applicatie, wordt er een SMS ontvangen. Hierna zou de applicatie automatisch een SMS terug moeten sturen naar de afzender van de SMS, om bijvoorbeeld te bevestigen dat de boodschap ontvangen werd. Deze workflow wordt uitgebeeld op figuur 27.

Dit betekent dat, voor het terugsturen van de bevestigingsboodschap, het telefoonnummer nodig is van de afzender van de SMS. Hiervoor moet er een referentie geplaatst worden vanuit de 2^{de} actie (die een bevestigingsboodschap gaat sturen) naar het telefoonnummer uit de eerste actie. De manier waarop dynamische referenties geplaatst kunnen worden, is gelijkaardig aan de manier getoond in figuur 26. Dit betekent dat de maker van de workflow zowel dynamische als statische referenties doorheen de workflow kan plaatsen.

Het maken van een dynamische referentie gebeurt op diezelfde manier als uitgebeeld op figuur 26, enkel in stap 4 moet gekozen worden voor de knop "dynamische browse" om een dynamische referentie te plaatsen.



Figuur 27: Voorbeeld van een sequentieel patroon met dynamische referenties

6 Gebruikersinterface

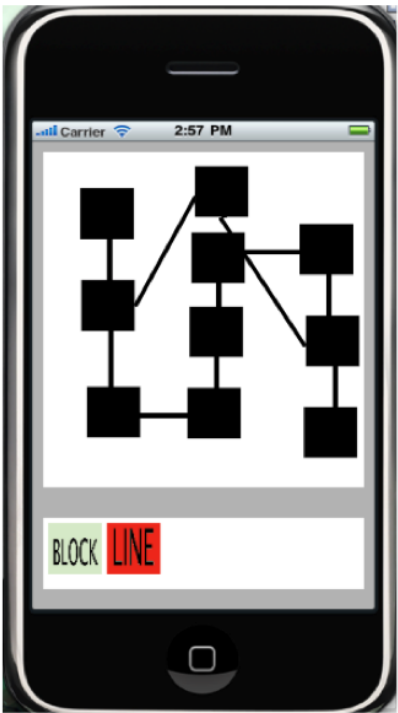
In deze sectie wordt de concrete syntax van de bouwblokken, patronen en knopen voorgesteld. Hierna worden de bewerkingen op de workflows uitgedrukt aan de hand van multitouch gestures.

6.1 Bestaande gebruikersinterfaces

In deze sectie wordt er gekeken naar bestaande gebruikersinterfaces van workflow talen.

Zo zien we op de linker zijde van figuur 28 dat een aanpak met een tekstuele taal geen oplossing is. Het keyboard neemt al bijna 50 percent van het scherm in. Zelfs al neemt het toetsenbord zoveel plaats in, bij het gebruiken is het nog steeds niet gemakkelijk om de goede toetsen aan te slaan. Een tweede nadeel van een tekstuele taal is dat de gebruiker amper 13 regels tekst op het scherm kan plaatsen. Dit biedt geen overzicht van het programma waarmee de gebruiker bezig is.

Een tweede aanpak, die te zien is op de rechter zijde van figuur 28, was de eerste iteratie van dit onderzoek. Er werd nagegaan hoe een gebruikersinterface gelijkaardig aan de gebruikersinterface van YAWL en BPMN zou weergegeven worden op een smartphone. Hier botsten we op het probleem dat het scherm te klein is om een gestructureerd overzicht van de workflow aan te bieden. Zo merken we dat 10 verschillende blokken al bijna een volledig scherm vullen. Hierbij komt de palet, die een deel van de werkomgeving inneemt om de bouwelementen van de taal beschikbaar te stellen voor de maker van de workflow. Naast het weinig aantal beschikbare blokken op één scherm is er het probleem van de pijlen tussen de bouwblokken. Deze pijlen, zoals te zien op figuur 28, kunnen mekaar overlappen. Dit heeft als gevolg dat er geen gebruiksvriendelijke interface aangeboden wordt.



76

6.2 Workflow interface elementen

In deze sectie wordt de concrete syntax van de verschillende bouwblokken, patronen en knopen voorgesteld voor de workflow uit de vorige secties van dit document.

Bij het maken van de concrete syntax en programmeeromgeving voor de gebruikersinterface op de smartphone is er een beperking aangebracht op het parallel en conditioneel patroon. Het is mogelijk om slechts 2 parallelle takken of conditionele takken in één patroon te definiëren. Het smartphone scherm wordt in 2 verdeeld en de maker van de workflow kan aan de linker en rechterkant deelworkflows definiëren.

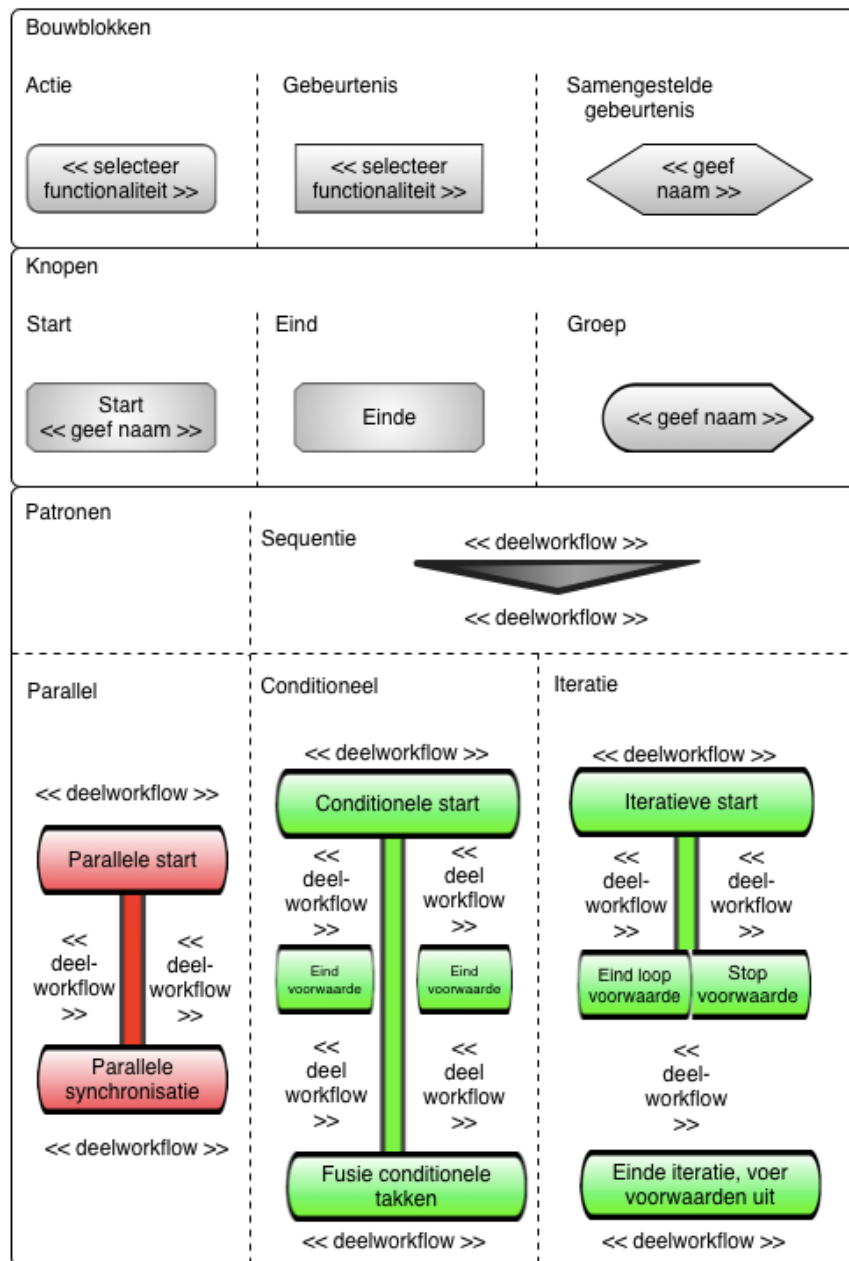
De reden waarom er enkel 2 parallel takken mogelijk zijn, is om een zo proper mogelijk visuele weergave te bieden aan de gebruiker. Indien bijvoorbeeld 3 takken zouden worden toegelaten, zou het scherm in 3 verdeeld moeten worden; hierdoor zouden de verschillende takken te klein zijn en quasi niet meer bruikbaar voor de maker van de workflow. Door twee toegelaten takken naast elkaar weer te geven, kan de maker van de workflow snel een overzicht krijgen van welke taken of voorwaarden tegelijkertijd gebeuren.

Naast deze beperking tot 2 takken in de parallelle en conditionele patronen, wordt er een nog niet besproken knoop geïntroduceerd, die de maker van de workflow extra functionaliteit biedt. Deze knoop stelt een groep voor. Aan de hand van een groep, kan de maker van de workflow een deelworkflow verbergen in één element. Dit laat toe om delen van de workflow te categoriseren in verschillende deelworkflows. Ook kan de maker van de workflow door delen van de workflows te abstraheren in een groep, de lengte van de workflow samenbrengen in kleinere delen.

Indien de gebruiker de knoop, die de groep voorstelt, aanraakt, wordt de deelworkflow uit de groep weergegeven. De maker van de workflow kan terug naar de algemene workflow aan de hand van een "terug" knop die wordt weergegeven wanneer de maker van de workflow in een groep navigeert.

Tenslotte is het mogelijk om aan een groep een naam te geven, zodat de maker van de workflow zich later op een eenvoudige en snelle manier kan herinneren wat de inhoud van de groep is.

De concrete syntax van alle bouwblokken, knopen en patronen wordt weergegeven in figuur 29.

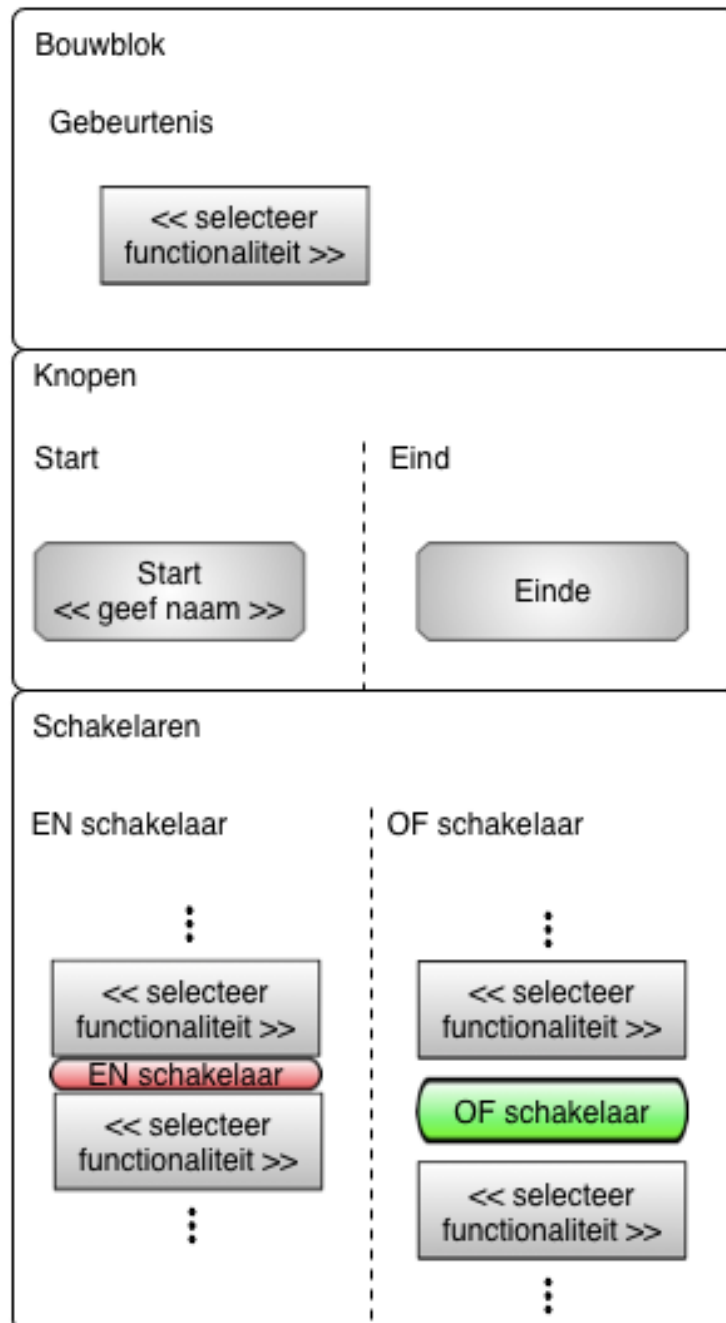


Figuur 29: Concrete syntax van de workflow. De maker van de workflow kan de variabelen tussen de << en >> zelf bepalen bij het maken van de workflow.

6.3 Interface elementen voor een samengestelde gebeurtenis

In deze sectie wordt de concrete syntax voor het maken van een samengestelde gebeurtenis voorgesteld (voor meer informatie over samengestelde gebeurtenissen zie sectie 4.2). De concrete syntax voor een samengestelde gebeurtenis is voorgesteld op figuur 30.

Een speciale eigenschap van de *EN schakelaar* is dat deze enkel wordt gebruikt om verschillende gebeurtenissen te groeperen; alle gebeurtenissen die gescheiden zijn door een *EN schakelaar* moeten telkens afgelopen zijn alvorens deze samengestelde gebeurtenis zelf afgelopen is. Hierdoor kan een *EN schakelaar* alleen tussen 2 gebeurtenissen toegevoegd worden. Een *EN schakelaar* wordt hierdoor automatisch toegevoegd wanneer er 2 gebeurtenissen onder elkaar geplaatst worden.



Figuur 30: Concrete syntax voor een samengestelde gebeurtenis. De maker van de samengestelde gebeurtenis kan de variabelen tussen de `<<` en `>>` zelf bepalen bij het maken van de workflow.

6.4 Specifieke handelingen op de workflow

In deze sectie zal er beschreven worden hoe de maker van de workflow een workflow kan bouwen en bewerken aan de hand van de verschillende beschikbare functies en multitouch gestures.

6.4.1 Aanpassen van de look en feel

De maker van de workflow is in staat om de kleur en de vorm van de verschillende bouwblokken, knopen en patronen te veranderen door bijvoorbeeld een ander thema te gebruiken. Dit laat toe dat de gebruiker in een voor hem aantrekkelijke omgeving kan werken. Hiervoor moet de gebruiker op het start element van de workflow drukken; hier kan hij de optie vinden om de kleur te veranderen.

6.4.2 Toevoegen van guards en stopgebeurtenissen op acties

Het toevoegen van een nieuwe guard of stopgebeurtenis gebeurt door op een actie te duwen. Hierna krijgt de maker van de workflow de mogelijkheid om de specifieke functionaliteit van de actie te selecteren, maar ook om de guard en stopgebeurtenis te definiëren. Indien de maker van de workflow bijvoorbeeld op de *guard definitie* knop drukt, opent zich de grafische weergave van een samengestelde gebeurtenis; hier kan de maker van de workflow dan de guards definiëren voor de geselecteerde actie.

6.5 Bouwen en bewerken van workflows aan de hand van multitouchgestures

6.5.1 Voordeel

Ben Shneiderman heeft verschillende concepten geïntroduceerd over het voordeel van touchscreens ten opzichte van andere input manieren [1]. Deze voordelen zijn:

1. Aanraken van een grafische weergave, vereist weinig denkwerk en is een vorm van directe manipulatie die gemakkelijk leerbaar is.
2. Aanraakgevoelige schermen zijn een snelle en intuïtieve manier om input te kunnen geven.
3. Aanraakgevoelige schermen hebben een gemakkelijker hand-oog coördinatie dan muis en toetsenbord.
4. Er is geen bijkomende apparatuur nodig om met het systeem te werken.

In een paper over 'Ambient Touch: "Designing Tactile Interfaces for Hand-held Devices" geschreven door "Ivan Poupyrev, Shigeaki Maruyama and Jun Rekimoto" [41] [37] tonen de auteurs, aan de hand van een experiment, aan dat aanraakgevoelige schermen een stijging in werksnelheid opleveren van 22 percent.

Gestures worden beschreven aan de hand van verschillende stadia, bijvoorbeeld: de gebruiker raakt het scherm aan, hierna doet hij een neerwaartse beweging en tenslotte heft hij zijn vinger(s) op van het scherm. [45, 33, 5, 40].

6.5.2 Toevoegen van bouwblokken

De eerste multitouch gesture die beschreven wordt, staat in voor het toevoegen van acties, gebeurtenissen of samengestelde gebeurtenissen in de workflow.

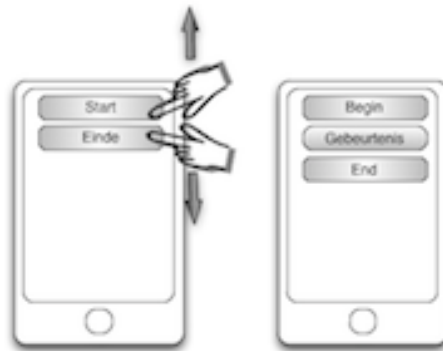
De multitouch gesture wordt uitgevoerd op de volgende manier en uitgebeeld op figuur 31:

1. De maker van de workflow plaatst zijn vingers op 2 naburige bouwblokken. Eén vinger op het onderste bouwblok, de andere vinger op het bovenste bouwblok.
2. De maker van de workflow glijdt met zijn vinger, die op het bovenste bouwblok aanwezig is naar de bovenkant van het scherm en ondertussen met de andere vinger, op het onderste bouwblok, naar de onderkant van het scherm.
3. De vingers moeten verschoven worden tot elke vinger buiten de grafische weergave van de start bouwblokken is.
4. De maker van de workflow kan nu zijn vingers tegelijkertijd van het scherm heffen.

Eens deze stappen voltooid zijn, kan de maker van de workflow kiezen tussen verschillende bouwblokken.

In een workflow zal de keuze bestaan uit een actie, gebeurtenis of samengestelde gebeurtenis.

In een samengestelde gebeurtenis, wordt deze gesture automatisch gebonden aan het toevoegen van een gebeurtenis.



Figuur 31: Grafische weergave van een *toevoegen gesture* in een samengestelde gebeurtenis.

6.5.3 Toevoegen van patronen.

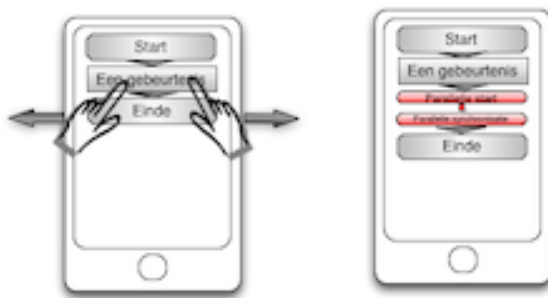
Na het kunnen toevoegen van de bouwblokken, wordt de gesture geïntroduceerd, die het mogelijk maakt om de verschillende patronen toe te voegen in de workflow, zoals: een parallel patroon, een conditioneel patroon, een iteratie patroon. Dezelfde gesture kan gebruikt worden voor het toevoegen van een *OF schakelaar* in een samengestelde gebeurtenis.

De multitouch gesture wordt uitgevoerd op de volgende manier en uitgebeeld op figuur 32:

1. De maker van de workflow plaatst zijn vingers op hetzelfde bouwblok; onder dit bouwblok zal het patroon ingevoerd worden.
2. De maker van de workflow glijdt met zijn vinger die op de meest linker kant van het bouwblok staat naar de linker kant van het scherm. Met zijn andere vinger glijdt hij naar de rechter kant van het scherm.
3. De vingers moeten verschoven worden en ondertussen altijd in de grafische weergave van dit element blijven.
4. De maker van de workflow kan nu zijn vingers tegelijkertijd van het scherm heffen.

Eens deze stappen voltooid zijn, kan de maker van de workflow kiezen tussen de verschillende bouwblokken. In de workflow zal de keuze bestaan uit een parallel patroon, een conditioneel patroon of een iteratie patroon.

In een samengestelde gebeurtenis, wordt deze gesture gebonden aan het toevoegen van een *OF schakelaar*. Hier wordt ook geen keuze gelaten omdat de *EN schakelaar* automatisch geplaatst wordt tussen twee naburige gebeurtenissen.



Figuur 32: Grafische weergave van het toevoegen van een *parallel patroon*

6.5.4 Verwijderen van een element

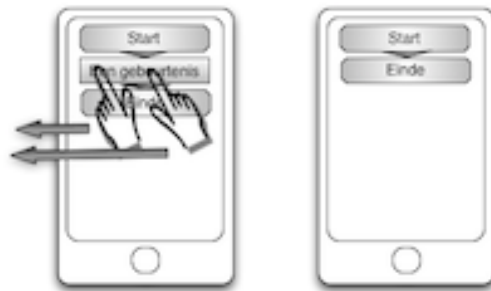
Indien de maker van de workflow een bouwblok of patroon wenst te verwijderen, kan hij dit aan de hand van de volgende gesture (deze gesture wordt uitgebeeld op figuur 33):

1. De maker van de workflow plaatst 2 vingers op hetzelfde bouwblok aan de meest rechtse kant.
2. De maker van de workflow glijdt met zijn beide vingers van de rechterkant van het bouwblok naar de linkerkant.
3. De vingers moeten verschoven worden tot de meest linkse kant van het bouwblok.
4. De maker van de workflow kan nu zijn vingers tegelijkertijd van het scherm heffen.

Indien de maker van de workflow deze gesture uitvoert op, bijvoorbeeld, het starten van een parallel of conditioneel patroon, wordt het hele patroon verwijderd.

Indien de maker van de workflow deze gesture op een actie, gebeurtenis of samengestelde gebeurtenis binnen de workflow uitvoert, wordt enkel dit bouwblok verwijderd.

De maker van de workflow kan in een samengestelde gebeurtenis op deze manier ook gebeurtenissen en *OF schakelaars* verwijderen.



Figuur 33: Grafische weergave van een verwijder gesture

6.5.5 Verplaatsen van elementen

Indien de maker van de workflow een bouwblok of patroon wenst te verplaatsen, kan hij dit doen aan de hand van de “verplaats gesture”. Deze gesture kan gebruikt worden in de workflow en in een samengestelde gebeurtenissen. De multitouch gesture wordt uitgevoerd op de volgende manier en uitgebeeld op figuur 34:

1. De maker van de workflow plaatst 2 vingers op hetzelfde bouwblok.
2. De maker van de workflow glijdt met zijn beide vingers naar een andere positie in de workflow
3. De vingers moeten verschoven worden tot deze de grafische interface van het huidige bouwblok verlaten.
4. De maker van de workflow kan nu zijn vingers tegelijkertijd van het scherm heffen; het element wordt toegevoegd onder het bouwblok waarop de maker van de workflow zijn beweging gestopt is

Indien de maker van de workflow deze gesture uitvoert op, bijvoorbeeld, het starten van een parallel of conditioneel patroon, wordt het hele patroon verplaatst.

Indien de maker van de workflow deze gesture op een actie, gebeurtenis of samengestelde gebeurtenis binnen de workflow uitvoert, wordt enkel dit element verplaatst.

De maker van de workflow kan in een samengestelde gebeurtenis op deze manier ook gebeurtenissen en *OF schakelaars* verplaatsen.



Figuur 34: Grafische weergave van een verplaats gesture

6.5.6 Knippen en plakken van elementen

Indien de maker van de workflow een bouwblok of patroon wenst te knippen om deze vervolgens (bijvoorbeeld) in een groep te plakken, kan hij gebruik maken van het volgende patroon:

1. De maker van de workflow plaatst 2 vingers op hetzelfde bouwblok.
2. De maker van de workflow heft zijn vingers tegelijkertijd van het scherm.

Op dit moment kan de maker van de workflow kiezen tussen de knip- en plakfunctie.

Indien hij een voor de knipfunctie kiest, wordt het element of patroon verwijderd en kan hij dit element ergens anders plakken. Indien hij 2 elementen na elkaar knipt, gaat het eerst geknipte element verloren.

Indien de maker van de workflow deze gesture uitvoert op, bijvoorbeeld, het starten van een parallel of conditioneel patroon, wordt het hele patroon geknipt en eventueel later geplakt. Indien de maker van de workflow deze gesture op een actie, gebeurtenis of samengestelde gebeurtenis binnen de workflow uitvoert, wordt enkel dit element geknipt en eventueel later geplakt. De maker van de workflow kan in een samengestelde gebeurtenis op deze manier ook gebeurtenissen en *OF schakelaars* knippen en plakken.

6.5.7 Bouwen van een groep

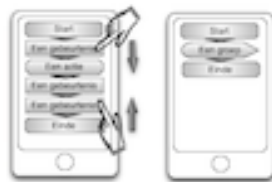
Wanneer de maker van de workflow een groep wil maken in een workflow, zal hij de hier beschreven gesture moeten gebruiken; deze gesture wordt uitgebeeld op figuur 35:

1. De maker van de workflow plaatst zijn vingers op 2 bouwblokken uit dezelfde sequentie. Groepen kunnen dus niet gemaakt worden tussen de verschillende parallelle of conditionele takken. Een groep kan wel gemaakt worden in één tak van een conditioneel of parallel patroon.
2. De maker van de workflow glijdt met zijn vinger, die op het bovenste bouwblok staat, naar beneden en ondertussen met de andere vinger, op het onderste bouwblok, naar boven.
3. De maker van de workflow glijdt met zijn vingers tot beide vingers elkaar raken.
4. De maker van de workflow kan nu zijn vingers tegelijkertijd van het scherm heffen.

Alle elementen tussen de 2 bouwblokken geselecteerd in de eerste stap, zullen de nieuwe groep vormen, indien de gesture correct afgehandeld wordt. De beperking van deze gesture is dat groepen enkel gemaakt kunnen worden uit de elementen zichtbaar op het scherm.

Als oplossing voor de beperking van het maken van een groep met enkel de zichtbare elementen op het scherm, kan de maker van de workflow groepen in verschillende stappen maken.

Bijvoorbeeld, de maker van de workflow maakt een groep A met 3 elementen, hierna maakt hij een 2^{de} groep B met de 3 elementen. De 2 aangemaakte groepen A en B worden samengebracht. In de nieuwe groep bestaande uit 2 groepen worden de 2 groepen uit elkaar gevouwen. Er blijft dan slechts één groep over met 6 elementen. Voor een uitleg van het uiteenvouwen van een groep, zie volgende sectie.

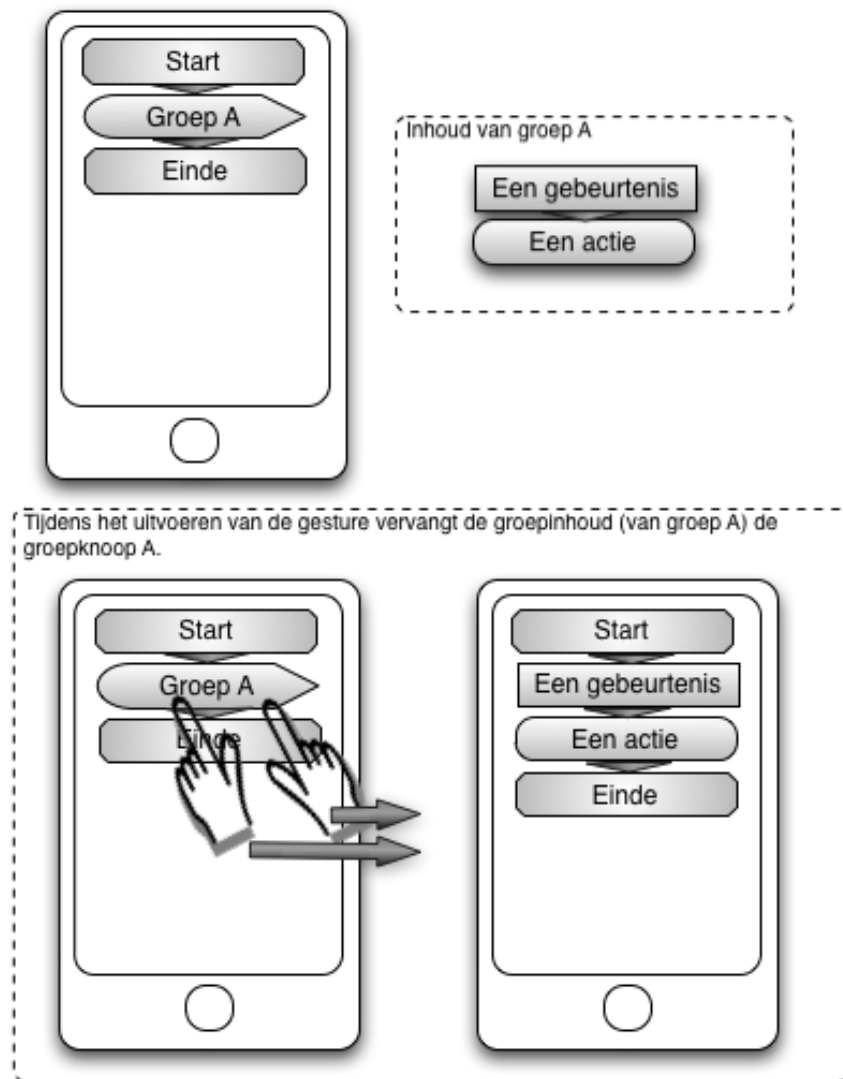


Figuur 35: Grafische weergave van het bouwen van een groep

6.5.8 Het uiteenvouwen van een groep

Op figuur 36 zien we hoe het uiteenvouwen van een groep kan gebeuren. De groep A met een sequentieel proces als inhoud wordt verwijderd. We merken op dat de inhoud van de groep de groepknoop vervangt.

1. De maker van de workflow plaatst 2 vingers op dezelfde "groeppknoop" aan de meest linkse kant.
2. De maker van de workflow glijdt met zijn beide vingers van de linker kant van de knoop naar de rechter kant.
3. De vingers moeten verschoven worden tot de meest rechtse kant van de groeppknoop.
4. De maker van de workflow kan nu zijn vingers tegelijkertijd van het scherm heffen.



Figuur 36: Grafische weergave van het uitenenvouwen van een groep. De groep in de linker workflow bestaat uit een sequentieel proces beginnend met een gebeurtenis, gevolgd door een actie.

6.6 Gebruikersklassen

In deze sectie worden de verschillende gebruikers geïntroduceerd, die met dit systeem zullen werken, namelijk: de ontwikkelaar (voegt nieuwe acties en gebeurtenissen toe aan de workflow), de workflow maker (maakt een workflow) en een gebruiker (gebruikt een workflow).

6.6.1 Ontwikkelaar

De ontwikkelaar is de persoon die de workflow gaat uitbreiden met nieuwe acties en gebeurtenissen. Aan de hand van programmeercode, die niet on the spot ontwikkeld kan worden, breidt hij het systeem uit.

1. Type van gebruiker: directe gebruiker
2. Motivatie/doeleinden: het implementeren van nieuwe acties en gebeurtenissen die de workflow rijker zal maken.
3. Ervaring: gevorderde programmeer-skills met kennis van de gebruikte bibliotheken en technologieën
4. Takenkennis: hoog
5. Gebruik: niet verplicht, hobby, interesse
6. Bewust van de interne werking van de applicatie: hoog
7. Frequentie van gebruik: nu en dan, tot regelmatig
8. Smartphone ervaring: hoog, inzicht in het gebruik en werking van smartphones en de gebruikte technologieën
9. Training: geen

6.6.2 Workflow maker

De workflow maker is een persoon, die, aan de hand van de voorgestelde programmeertaal en omgeving, een nieuwe applicatie gaat bouwen en testen en hoogstwaarschijnlijk ook gebruiken.

1. Type van gebruiker: directe gebruiker
2. Motivatie/doeleinden: het implementeren of aanpassen van workflows voor het eigen gebruik of voor het uitbrengen van een applicatie voor, bijvoorbeeld, een bedrijf.
3. Ervaring: geen programmeerervaring nodig. Kennis van de workflow-taal en -omgeving is wel nodig.

4. Takenkennis: middelmatig
5. Gebruik: niet verplicht, hobby, interesse, maken van een applicatie voor een bedrijf.
6. Bewust van de interne werking van de applicatie: neen
7. Frequentie van gebruik: nu en dan, tot regelmatig
8. Smartphone ervaring: middelmatig, de workflow maker moet weten welke verschillende activiteiten er bestaan en welke technologieën deze van de smartphone gebruikt.
9. Training: geen

6.6.3 Workflow gebruiker

De gebruiker is de persoon die bestaande workflows gaat uitvoeren en gebruiken op zijn smartphone.

1. Type van gebruiker: (in)directe gebruiker
2. Motivatie/doeleinden: het gebruik van voorgedefinieerde workflows
3. Ervaring: geen
4. Takenkennis: geen
5. Gebruik: niet verplicht, enkel voor het gebruiken van aangeboden applicaties
6. Bewust van de interne werking van de applicatie: neen
7. Frequentie van gebruik: nu en dan, tot regelmatig
8. Smartphone ervaring: geen tot hoog
9. Training: geen

6.7 Gouden regels

In deze sectie zal een overzicht gegeven worden van hoe de gouden regels uit sectie 2.2.1, omtrent de grafische weergave, in dit onderzoek geïntegreerd werden.

1. Hou de gebruikersomgeving zo consistent mogelijk: consistentie doorheen de programmeeromgeving en -taal is behouden door consistent dezelfde vormen en kleur aan te bieden voor de elementen.
2. Laat kortere wegen toe: kortere wegen worden mogelijk gemaakt aan de hand van gestures: het is niet meer nodig voor de maker van de workflow om door menu's te navigeren om een element toe te voegen.
3. Geef informatieve feedback: feedback over de fout geïnitieerde variabelen wordt voorzien bij het willen opslaan of uitvoeren van de workflow. Ook kan de maker van de workflow door de verschillende vormen en kleuren van de bouwblokken, knopen en patronen informatie krijgen over het type blok. Tenslotte is het mogelijk voor de maker van de workflow een naam te geven aan een groep. Op deze manier kan de maker van de workflow, door een betekenisvolle naam te gebruiken, feedback voorzien.
4. Indien gebruik gemaakt wordt van dialogen, zorg voor een goede opbouw: bij het toevoegen van een element kan de maker van de workflow kiezen tussen een actie en gebeurtenis; er is gekozen om een klein dialoogvenster te tonen, waar de maker van de workflow tussen de verschillende bouwblokken kan kiezen. Eens de maker van de workflow zijn keuze heeft aangeduid, wordt het venster verwijderd en het bouwblok automatisch toegevoegd. Dialogen worden zo kort en duidelijk mogelijk gehouden.
5. Voorkom het kunnen maken van fouten; als fouten mogelijk zijn, geef een zo gemakkelijk mogelijke errorcorrectie: er werd gekozen voor error controle; aan de hand van het statische controle mechanisme voor variabelen, worden alle foute geïnitieerde variabelen aan de maker van de workflow weergegeven.
Bij het gebruik van de gestures, is er gekozen voor error preventie: indien de maker van de workflow een gesture uitvoert dat niet bestaat of betekenisloos in de gegeven context, wordt deze genegeerd. De maker van de workflow kan hier dus niets verkeerd doen.
6. Laat de gebruiker toe om stappen ongedaan te maken: de maker van de workflow kan ongewenste blokken verwijderen aan de hand van gestures. Indien hij deze terug wil toevoegen, kan dit even snel aan de hand van "voeg element gesture". In een volgende iteratie, zou er een

redo en undo gesture kunnen toegevoegd worden aan de taal om redo en undo volledig te ondersteunen.

7. Pas de controle en interface aan voor verschillende types gebruikers: dit onderzoek heeft zich toegespitst met bedoeling een zo gemakkelijke taal en omgeving aan te bieden, voor deze reden zijn er geen verschillende controle en interface mogelijkheden.
8. Verminder geheugenlading op korte termijn: aan de hand van de visuele hints, namelijk de kleuren en vormen, kan de maker van de workflow op een snelle manier zien waarmee hij bezig is.

Naast deze gouden regels, werd er ook gekozen om de mobile guidelines te volgen en gebruik te maken van een top down interface. Zo kan de maker van de workflow op een element uit de workflow duwen. Hierna wordt er meer informatie over dit specifiek element gegeven. Tenslotte is het gebruik van het toetsenbord bijna overbodig in het gebruik van de voorgestelde taal en omgeving. Er werd gekozen om aan de hand van lijsten het aan de maker van de workflow mogelijk te maken om de gewenste keuzes te maken. Indien deze lijsten te lang worden, is een zoekfunctie voorzien.

7 Implementatie

In deze sectie zal eerst de implementatie van de programmeertaal en -omgeving voorgesteld worden, alsook de designkeuzes die gemaakt werden. Hierna zal de interface besproken worden die dient om de uitvoering van de workflow te controleren. Deze interface is voornamelijk gericht op het mogelijk maken van het testen op een correct verloop van de gemaakte workflow.

7.1 Implementatie van de workflow

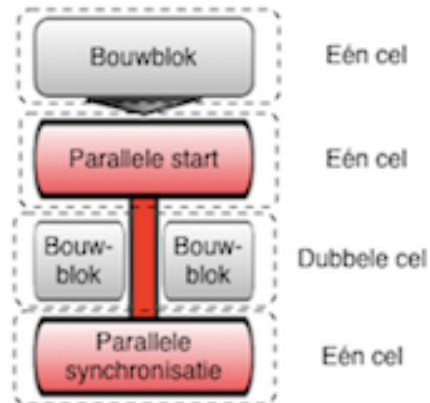
Bij het beschrijven van bouwblokken en patronen uit de workflow taal, wordt er een onderscheid gemaakt tussen acties/gebeurtenissen, taalelementen en samengestelde gebeurtenissen. Al deze elementen hebben allen een andere betekenis, maar hebben, op enkele verschillen na, dezelfde implementatie. Het grootste verschil is de functionaliteit van de patronen en bouwblokken. De functionaliteit van een patroon of bouwblok beschrijft de uitvoering ervan.

7.1.1 Grafische weergave

Voor de grafische weergave van alle bouwblokken en patronen in de programmeeromgeving, zijn er 2 verschillende voorstellingen zoals afgebeeld op figuur 37, namelijk:.

1. Eén cel: er wordt één cel op de hele breedte van het scherm getoond, in een sequentie waar geen parallelle takken bestaan. Deze cellen worden gebruikt voor het weergeven van de verschillende bouwblokken en knopen van de patronen.
2. Dubbele cel (twee cellen): het scherm wordt in 2 verdeeld om 2 verschillende bouwblokken en/of knopen weer te geven. Deze worden gebruikt om, onder andere, de 2 takken uit de *parallele* en *conditionele* patronen naast elkaar weer te geven.

Een UML weergave van de grafische elementen wordt in figuur 38 getoond. We merken op in deze figuur dat acties guards en stopgebeurtenissen bevatten. Guards en stopgebeurtenissen zijn samengestelde gebeurtenissen. Elk workflowelement bezit ook een functionaliteit: dit is de klasse die gaat beschrijven hoe een specifiek workflow element uitgevoerd moet worden. Tenslotte bezit elk workflow element een `typeOfElement`; deze laatste wordt gebruikt om snel, aan de hand van een jump tafel, een element te kunnen herkennen. Als voorbeeld: een actie is van type 0, een gebeurtenis van type 1.



Figuur 37: Voorbeeld van de voorstelling van één cel en een dubbele cel.

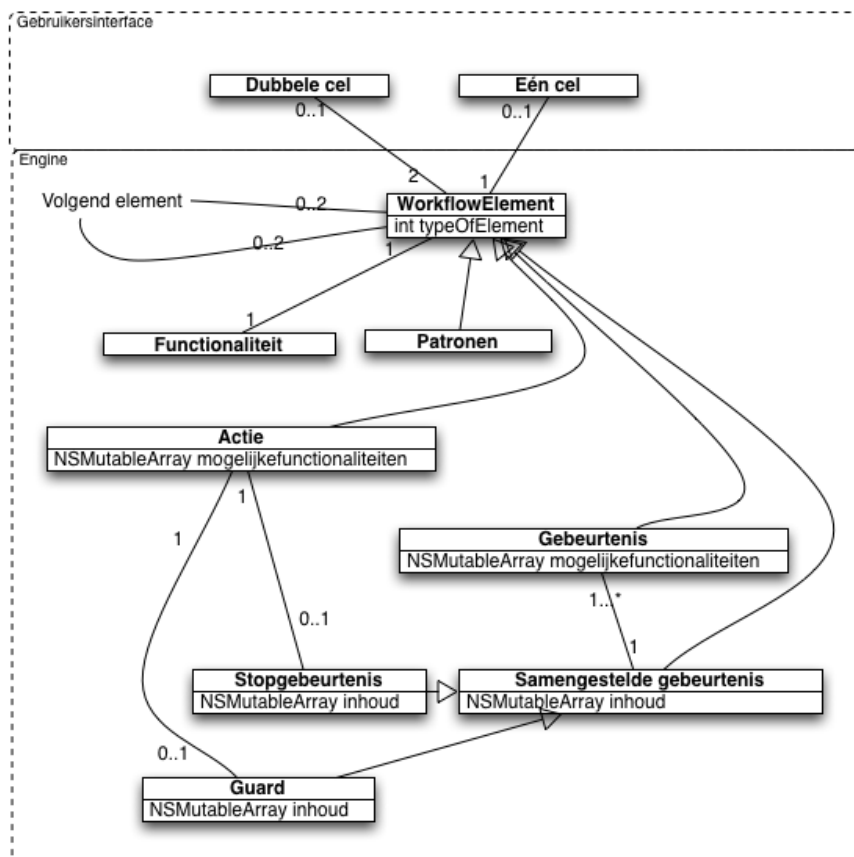
7.1.2 Actie en gebeurtenis

Op figuur 38 zien we dat gebeurtenissen een subklasse zijn van een workflow-element. Een actie en gebeurtenis bezitten een NSMutableArray met alle mogelijke functionaliteiten, die ingevuld kunnen worden in een actie of gebeurtenis. Zo kan de maker van de workflow een welbepaalde functionaliteit uit de array selecteren om aan een actie of gebeurtenis een functionaliteit te binden.

De geïmplementeerde functionaliteiten voor acties en gebeurtenissen zijn:

1. Acties:

- (a) **Speel muziek:** wanneer deze actie uitgevoerd wordt, zal er een muziekstuk afgespeeld worden. Eens het muziekstuk afgelopen is, is deze actie afgerond.
- (b) **Stuur SMS:** wanneer deze actie uitgevoerd wordt, zal er een SMS verzonden worden. Eens deze SMS verzonden is, is deze actie afgerond. Bij het maken van de workflow, kan de maker van de workflow aan deze actie een boodschap en telefoonnummer verbinden.
- (c) **Initialiseren van een variabele:** wanneer deze actie uitgevoerd wordt, zal er een variabele geïnitieerd worden. Eens deze variabele geïnitieerd is, is deze actie afgerond. Bij het maken van de workflow, kan de maker van de workflow aan deze actie een variabele naam en inhoud binden.
- (d) **Geef variabele weer:** wanneer deze actie uitgevoerd wordt, zal er een variabele getoond worden. Eens deze variabele getoond is



Figuur 38: UML klassendiagram van een workflow element

aan de gebruiker, wordt deze actie afgerond. Bij het maken van de workflow, kan de maker van de workflow bij deze actie een variabele selecteren die weergegeven moet worden.

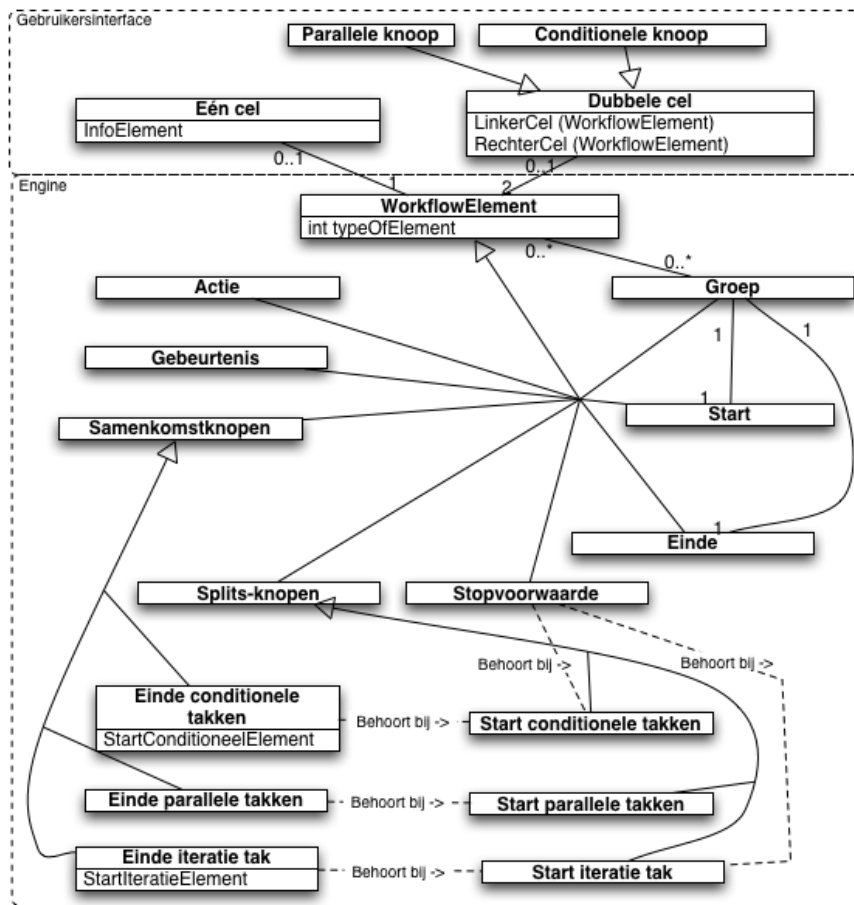
- (e) Variabele demo: wanneer deze actie uitgevoerd wordt, zal er een variabele foo geïnitieerd worden. Eens deze variabele geïnitieerd werd, is deze actie afgerond
- (f) Initialiseren van de demo variabele: wanneer deze actie uitgevoerd wordt, zal de variabele foo weergegeven worden. Eens deze variabele getoond werd aan de gebruiker, is deze actie afgerond.

2. Gebeurtenissen

- (a) Wacht tot een aantal seconden voorbij is: wanneer deze gebeurtenis uitgevoerd wordt, wordt er gewacht tot een aantal seconden voorbij is; eens deze tijdsduur verlopen is, vindt deze gebeurtenis plaats. Bij het maken van de workflow kan de maker van de workflow de tijdsduur instellen.
- (b) De smartphone ondergaat een hevige beweging: wanneer deze gebeurtenis uitgevoerd wordt, zal er gewacht worden tot door de smartphone een beweging gevoeld wordt; eens deze beweging gevoeld wordt, zal deze gebeurtenis plaatsvinden.
- (c) Duw op de knop die op het scherm getoond wordt: wanneer deze gebeurtenis uitgevoerd wordt, zal deze er een knop op het scherm getoond worden; eens de gebruiker het scherm aanraakt op de positie van deze knop, vindt deze gebeurtenis plaats.

7.1.3 Patronen en knopen

Naast acties en gebeurtenissen, bezit de taal ook een aantal patronen; deze bestaan uit meerdere knopen, zoals, bijvoorbeeld, een groep bestaat uit een start- en eindknoop. Al deze verbanden, bestaand op programmeerniveau, zijn uitgetekend in het UML schema op figuur 39.



Figuur 39: UML klassendiagram weergave van de verschillende patronen

7.1.4 Functionaliteit van een workflow bouwblok

Zoals al reeds hoger vermeld, bezit elk element een welbepaalde functionaliteit. Zo gaat het start blok van parallelle takken, de volgende takken concurrent startten, of gaat de actie, “speel muziek” een liedje afspelen. De UML voorstelling van de functionaliteit van een workflow bouwblok, wordt getoond in figuur 40.

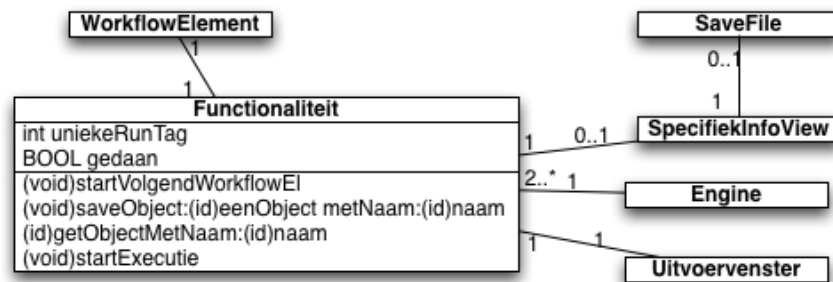
De functionaliteiten van de patronen zijn vast gedefinieerd. De functionaliteit van samengestelde gebeurtenissen, acties en gebeurtenissen kan bij het maken van de workflow veranderen. De maker van de workflow kan (onder andere) aan een actie de functionaliteit “speel muziek” of “speel video” hangen.

De functionaliteit van een bouwblok of knoop wordt bij het uitvoeren van een workflow uitgevoerd.

Opdat getest zou kunnen worden of de workflow doet wat gevraagd wordt, heeft elke functionaliteit een venster. Dit venster wordt getoond bij het uitvoeren van een bouwblok of knoop.

Ook is het mogelijk dat bij het maken van de workflow, de maker van de workflow sommige variabelen van het bouwblok wil aanpassen, zoals, instellen hoe lang de gebeurtenis moet wachten vooraleer deze plaatsvindt. Voor deze functionaliteit is er ook een grafisch venster nodig, de SpecifiekInfoView.

Tenslotte, om het mogelijk te maken om een workflow op te slaan, wordt elke functionaliteit voorzien van een opslagfunctie, die de parameters die door de maker van de workflow ingegeven kunnen worden, gaat opslaan.



Figuur 40: UML klassendiagram weergave van de functionaliteit van een workflow bouwblok

7.1.5 Voeg nieuw workflow bouwblok toe

Er werd gekozen om de workflow uitbreidbaar te maken met nieuwe acties en gebeurtenissen. Hiervoor moeten, voor elke nieuwe actie of gebeurtenis, enkele stappen ondernomen worden

1. Een nieuwe functionaliteitsklasse moet toegevoegd worden als subklasse van de functionaliteitsklasse; deze nieuwe functionaliteitsklasse beschrijft hoe het nieuwe bouwblok uitgevoerd gaat worden.
2. Een functie “start uitvoering”, moet geïmplementeerd worden; deze functie beschrijft hoe de gegeven actie of gebeurtenis in werking gezet wordt tijdens het uitvoeren ervan.
3. Een venster moet verbonden worden aan de functionaliteit, voor het weergeven van de uitvoering van dit bouwblok
4. Eens de uitvoering van dit bouwblok afgelopen is, moet de functie `startVolgendWorkflowEl` uit de superklasse opgeroepen worden. Deze functie gaat ervoor zorgen dat de rest van de workflow gestart wordt.
5. Een optionele stap is dat de maker van de workflow sommige parameters wil openstellen. Deze parameters kunnen het verloop van het bouwblok bijsturen. Om dit mogelijk te maken, moet een `Specifiek-InfoView` gecreëerd worden, waar de maker van de workflow de parameters van de functionaliteit kan instellen
6. Het nieuwe blok toevoegen in de array van alle mogelijke acties of gebeurtenissen; dit laat toe dat de maker van een workflow dit nieuwe blok kan selecteren bij het definiëren van een actie of gebeurtenis.

Voor het toevoegen van een nieuw patroon zijn de stappen iets complexer, vermits er veel meer nieuwe dingen geïntroduceerd moeten worden, zoals bijvoorbeeld, de multitouch gesture die ervoor zorgt dat een patroon correct in de workflow toegevoegd wordt.

Anderzijds, als de specifieke werking van een patroon veranderd moet worden, moet enkel de functionaliteit van het taalelement aangepast worden.

7.2 Implementatie van samengestelde gebeurtenissen

Bij samengestelde gebeurtenissen, is de implementatie gelijkaardig aan de implementatie van de workflow. De semantische betekenis van een gebeurtenis in een samengestelde gebeurtenis en een gebeurtenis uit de workflow is volledig dezelfde. Er wordt een onderscheid gemaakt op programmeerniveau om beide gebruikersinterfaces te ondersteunen.

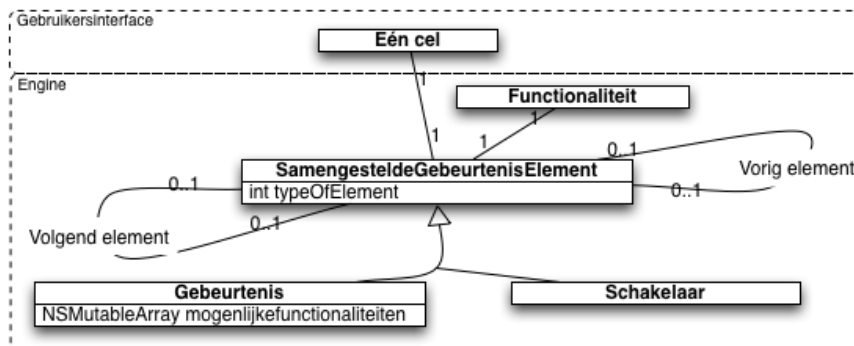
Dit betekent dat we, net zoals bij een workflow, de samengestelde gebeurtenissen kunnen verdelen in verschillende bouwblokken, knopen en patronen. Het onderscheid tussen gebeurtenissen in de workflow en de gebeurtenissen in een samengestelde gebeurtenis, is de weergave ervan. De interface aangeboden bij het uitvoeren van een samengestelde gebeurtenis, is anders dan deze aangeboden bij een gebeurtenis in een workflow (zie sectie 7.3.2).

Om deze reden, bestaan er gebeurtenissen, waar een specifiek venster voor het uitvoeren van een samengestelde gebeurtenis, nodig is.

7.2.1 Grafische weergave

Voor de grafische weergave van een samengestelde gebeurtenis, zijn er slechts enkele cellen. Dit wordt uitgebeeld in figuur 41.

Elk element heeft zijn eigen functionaliteit; deze functionaliteit bepaalt hoe het element zal worden uitgevoerd.



Figuur 41: UML klassendiagram weergave van een samengestelde gebeurtenis element

7.2.2 Gebeurtenissen

Een gebeurtenis uit een samengestelde gebeurtenis, is een subklasse van een enkele cel.

Net zoals bij acties en gebeurtenissen, kan bij gebeurtenissen van samengestelde gebeurtenissen uit verschillende functionaliteiten gekozen worden. Deze verschillende functionaliteiten die geïmplementeerd werden voor samengestelde gebeurtenissen zijn:

1. Teller: wanneer deze gebeurtenis uitgevoerd wordt, zal deze wachten tot een aantal seconden voorbij is; eens deze tijdsduur verlopen is, vindt deze gebeurtenis plaats. Bij het maken van de workflow kan de maker deze tijdsduur instellen.
2. Duw op de knop die op het scherm getoond wordt: wanneer deze gebeurtenis uitgevoerd wordt, zal het venster bij het uitvoeren van deze functionaliteit een knop op het scherm tonen; eens de gebruiker het scherm aanraakt op de positie van deze knop, vindt deze gebeurtenis plaats.

Deze gebeurtenissen kunnen in samengestelde gebeurtenissen gecombineerd worden aan de hand van schakelaars.

7.2.3 EN en OF schakelaars

Een EN schakelaar in een samengestelde gebeurtenis maakt het mogelijk om de verschillende gebeurtenissen te combineren. De verschillende EN schakelaars kunnen naderhand gecombineerd worden aan de hand van een OF schakelaar. Voor meer informatie over het samenstellen van samengestelde gebeurtenissen, zie sectie 6.3.

De verschillende knopen die de maker van de workflow kan toevoegen zijn: een gebeurtenis, start- en eindknoop.

Een speciaal geval is de EN schakelaar. Deze schakelaar wordt automatisch toegevoegd aan de gebruikersinterface wanneer 2 gebeurtenissen onder elkaar staan zonder OF schakelaar ertussen. Dit wil dus semantisch zeggen dat deze 2 gebeurtenissen moeten gebeuren alvorens deze tak afgerond is.

Bij het toevoegen van een nieuwe gebeurtenis, wordt gebruik gemaakt van de referentie naar het volgend en vorig element om na te gaan of de gebeurtenis zelf onder een andere gebeurtenis staat. Zo ja, wordt de EN schakelaar geplaatst.

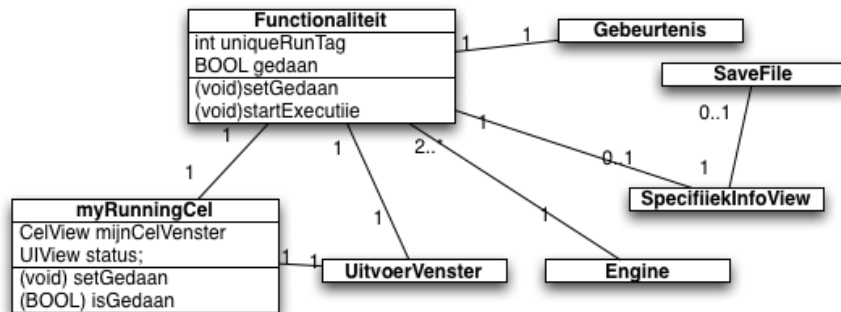
Indien later een OF schakelaar tussen deze 2 gebeurtenissen geplaatst wordt, wordt de EN schakelaar automatisch verwijderd.

7.2.4 Functionaliteit van een gebeurtenis in een samengestelde gebeurtenis

De functionaliteit van een gebeurtenis in een samengestelde gebeurtenis is gelijkaardig aan de functionaliteit van een workflow element, op enkele verschillen na, namelijk:

1. Het venster, dat gebruikt wordt bij het uitvoeren van een gebeurtenis, bestaat uit 2 delen:
 - (a) een cel die als indicator dient of de gebeurtenis plaatsgevonden heeft of niet
 - (b) de weergave van de gebeurtenis gedurende de uitvoering van de samengestelde gebeurtenis.
2. De setGedaan functie wordt opgeroepen wanneer de gebeurtenis plaatsgevonden heeft; de setGedaan functie zal dan ook de grafische indicatie geven, na het plaatsvinden van deze gebeurtenis, dat deze afgelopen is (zie sectie 7.3).

Een UML weergave van de functionaliteit van een gebeurtenis, in een samengestelde gebeurtenis, wordt weergegeven op figuur 42.



Figuur 42: Een UML klassendiagram weergave van de functionaliteit van een gebeurtenis

7.3 Controleren van de uitvoering van een workflow

Eens de workflow gemaakt is, kan deze uitgevoerd worden. De maker van de workflow kan als controle de gemaakte workflow uitvoeren. De maker van de workflow kan zo tijdens het uitvoeren van de workflow het verloop volgen. Er werd een interface gerealiseerd voor het uitvoeren van de workflow en samengestelde gebeurtenissen. De gebruikersinterface voor het uitvoeren van de workflow laat toe om verschillende deelworkflows tegelijkertijd weer te geven.

De gebruikersinterface voor samengestelde gebeurtenissen werd zodanig gebouwd dat nagegaan kan worden welke gebeurtenissen al voorbij zijn en welke nog hoeven plaats te vinden. In deze sectie worden deze gebruikersinterfaces geïntroduceerd.

7.3.1 Verschillende bouwblokken die tegelijkertijd uitgevoerd worden

Bij het uitvoeren van een sequentie, heeft elk bouwblok één venster. Het huidige uit te voeren bouwblok venster wordt weergegeven aan de gebruiker; hier is dan ook geen probleem, omdat, eens het bouwblok afgelopen is, dit venster vervangen wordt door het uitvoerscherm van het volgende bouwblok. Het probleem stelt zich echter wanneer er 2 (of meer) bouwblokken tegelijkertijd uitgevoerd worden. Op dit ogenblik zijn er minimaal 2 uitvoerschermen, die tegelijkertijd getoond moeten worden. Vermits de smartphone slechts een klein scherm heeft, is het niet mogelijk om de verschillende schermen van de bouwblokken tegelijkertijd weer te geven. Om deze reden is er gekozen om het mogelijk te maken om tussen de verschillende schermen (van de verschillende bouwblokken) te wisselen. Op deze manier kan de gebruiker naar alle schermen van de bouwblokken, die uitgevoerd worden, kijken.

Het wisselen tussen de uitvoerschermen van de verschillende bouwblokken gebeurt door op de 2 tap zones (een tap zone is een plaats waarop de gebruiker kan duwen) te duwen die afgebeeld worden op figuur 43.

Elk bolletje in het midden van de afbeelding staat voor één scherm. Op deze figuur worden er 2 verschillende bouwblokken tegelijkertijd uitgevoerd.

Nieuwe bouwblokken worden altijd aan de meest rechtse kant van de rij toegevoegd. Indien een blok uitgevoerd is, wordt dit automatisch verwijderd. Elke scherm dat bij een bouwblok hoort, is specifiek aan het bouwblok; zo zal er bij de gebeurtenis, “er wordt op een knop gedrukt” op het scherm één knop zijn. Bij de gebeurtenis, “wacht tot er 10 seconden voorbij zijn”, zal er een teller lopen van 10 tot 0.

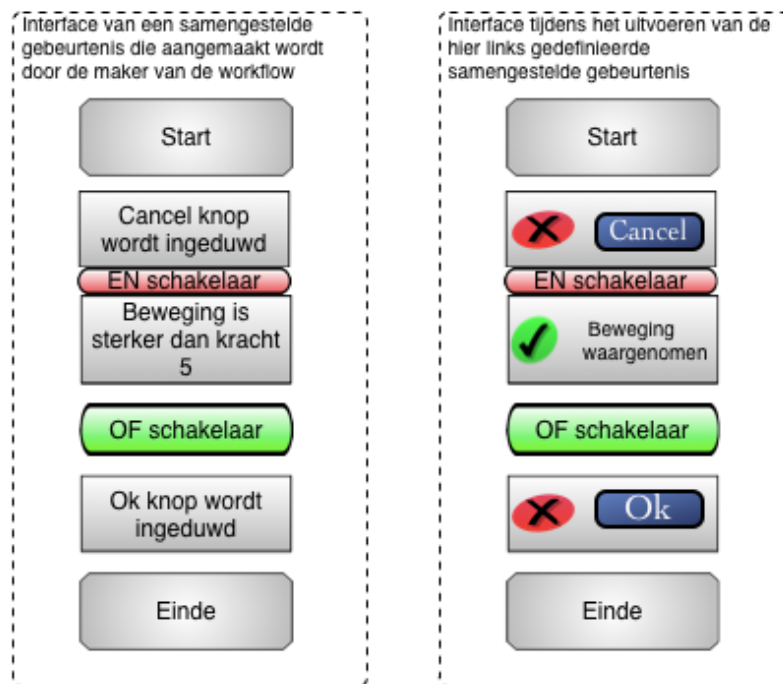


Figuur 43: Grafische weergave van de verschillende tapzones tijdens het uitvoeren van een workflow

7.3.2 Uitvoeren van samengestelde gebeurtenissen

Bij het uitvoeren van een samengestelde gebeurtenis, wordt het belangrijk geacht dat, op elk ogenblik, geweten is welke gebeurtenissen al verlopen zijn en welke nog hoeven te gebeuren. Op deze manier wordt er een overzicht over de gebeurtenissen uit de samengestelde gebeurtenissen gegeven.

Om deze reden wordt er een speciale gebruikersinterface aangeboden, die een overzicht biedt van de gebeurtenissen en hoe deze verbonden zijn door de EN en OF schakelaars. Er werd gekozen om de interface bij het uitvoeren heel gelijkend te houden aan de interface bij het maken van een samengestelde gebeurtenis; dit wordt getoond in figuur 44. Dit wil zeggen dat dezelfde structuur tussen de EN en OF schakelaars getoond wordt, enkel de gebeurtenissen zelf worden nu uitgevoerd en worden vervangen door hun uitvoervenster. Eens een gebeurtenis afgelopen is, wordt dit aangeduid aan de hand van een symbool: deze symbolen worden getoond op figuur 44.



Symbool dat weergeeft dat de gebeurtenis plaatsgevonden heeft



Symbool dat weergeeft dat de gebeurtenis nog niet plaatsgevonden heeft

Figuur 44: Grafische weergave van het uitvoeren van een samengestelde gebeurtenis

7.4 Design keuzes

In de voorbije secties werden alle bouwblokken, patronen en schakelaars besproken. In deze sectie zal beschreven worden hoe deze aan elkaar gelinkt zijn in de implementatie van de nieuwe programmeertaal en -omgeving. Ook zal de engine, die ervoor zorgt dat de workflow uitgevoerd wordt, besproken worden.

7.4.1 Graf

Voor het verbinden van de verschillende elementen in de workflow, werd er gekozen om een gelinkte lijst te gebruiken [18]. Dit betekent dat elk element een pointer bezit naar het volgend bouwblok of knoop.

In sommige gevallen voldoet een gelinkte lijst niet meer en werden er uitbreidingen op gebouwd; deze gevallen zijn:

1. Bij een conditioneel, iteratie of parallel patroon: het start element van deze patronen bezit 2 pointers, één naar elk beginelement van de deelworkflows.
2. Referentie naar het groepelement: de eindknoop van een groep heeft een referentie nodig om terug naar het groep bouwblok te kunnen terugkeren. Op deze manier kan, eens het uitvoeren van een groep voltooid is, het volgende element van de groep oproepen worden.
3. Referentie naar het start element van een iteratief of conditioneel patroon: bij het bereiken van het einde van een conditionele stop tak, is het noodzakelijk om een referentie te bezitten naar de startknoop van het patroon. Op deze manier kan de andere tak gevonden worden en de uitvoering ervan stopgezet.

Het plaatsen van deze speciale referenties gebeurt bij het maken van de patronen. Voor een groep wordt bijvoorbeeld automatisch het start- en eindelement geplaatst. Bij het aanmaken van de groep kan dan ook de referentie in het eind element geplaatst worden naar het groep element.

7.4.2 Dubbel gelinkte lijsten

Dubbel gelinkte lijsten [32] worden gebruikt om samengestelde gebeurtenissen voor te stellen; hier is het noodzakelijk om zowel naar het vorig element te kunnen kijken als naar het volgend element; dit, onder andere, om efficiënt en automatisch de EN schakelaar te kunnen plaatsen.

7.4.3 Dubbele dispatch

Elk gebruikt element (bouwblok, patroon en knoop) heeft zijn eigen functionaliteit en manier van interageren op een handeling van de maker van de workflow. Het toevoegen van een nieuw element in een sequentie is anders dan in een parallelle tak.

Er werd gekozen om de functionaliteit van hoe een element wordt toegevoegd, verwijderd... te definiëren in het element zelf. Zo kan een element, indien nodig, de gewenste bijkomende pointers automatisch correct plaatsen. Dit laat ook toe om, bij het aanpassen van een element, enkel in het element te hoeven werken en niet in de hele code van het project.

Voor elk nieuw element moet de functionaliteit echter niet voorzien worden, zoals, bij het toevoegen van een nieuw element. Deze functionaliteit kan overgeërfd worden van de *vader klasse*, maar indien specifieke stappen ondernomen moeten worden, kunnen ze de functie overschrijven die ze overerven.

Indien een nieuw element toegevoegd wordt in de workflow, wordt de functie, die zorgt om het object toe te voegen, opgeroepen in het bouwblok. Het bouwblok zelf gaat zichzelf vervolgens toevoegen aan de workflow, dit is dubbele dispatching [12].

Enkele voorbeelden waar dubbele dispatching gebruikt wordt zijn:

1. Wanneer de maker van de workflow bij het maken van de workflow op een bouwblok duwt.
 - (a) Een eindelement: deze aanraking wordt genegeerd
 - (b) Een startelement: een venster moet geopend worden waar de maker van de workflow een nieuwe naam kan ingeven voor deze workflow
 - (c) Een groepelement: de inhoud van de groep wordt weergeven
 - (d) Een actie/gebeurtenis: een venster wordt geopend waar de maker van de workflow de functionaliteit van de actie of gebeurtenis kan selecteren
 - (e) Een samengestelde gebeurtenis: een venster wordt geopend waar de maker van de workflow de samengestelde gebeurtenis kan definiëren of wijzigen.
2. Wanneer de maker van de workflow, bij het maken van de workflow, een bouwblok verwijdert.
 - (a) Bij het verwijderen van een enkele cel, moet de pointer van het vorig element geplaatst worden naar het volgende element dat volgde op het verwijderde element

- (b) Bij het verwijderen van een *start parallel* knoop is de semantiek dat het hele parallelle patroon verwijderd wordt; de pointer van het element voor de splitsing moet geplaatst worden naar het eerstvolgend element na het parallelle patroon.

7.4.4 Enkel één gebeurtenis wordt getriggerd, observer patroon

Bij het parallel uitvoeren van verschillende gebeurtenissen, ontstond het probleem dat bijvoorbeeld 2 gebeurtenissen tegelijkertijd aan het wachten waren op een hevige beweging waaraan de smartphone zou worden onderworpen. Deze 2 gebeurtenissen verwachten gegevens van dezelfde sensor.

Het probleem dat zich stelde, is dat enkel de laatst toegevoegde gebeurtenis de waarden kreeg van de sensor en de andere gebeurtenis niet en deze gebeurtenis dus nooit meer kon plaatsvinden. Deze laatste zit vast voor altijd.

Als oplossing voor dit probleem, werd er gekozen om het observer patroon [17] te gebruiken. De observer is de enigste die de sensorwaarden krijgt. Deze sensorwaarden worden dan doorgestuurd naar alle elementen die zich hebben ingeschreven bij de observer.

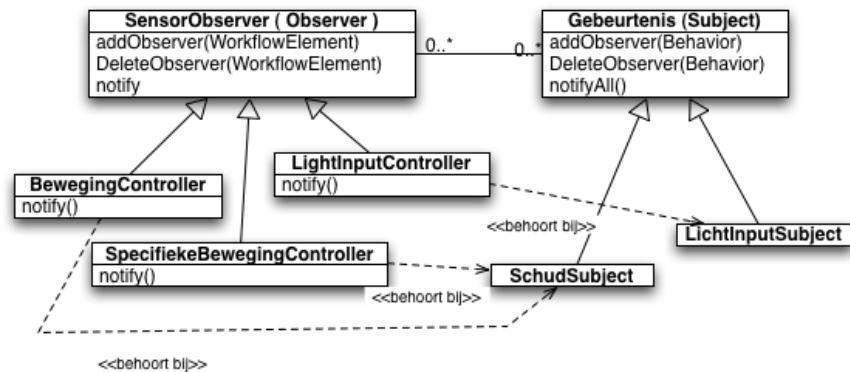
De gebeurtenissen krijgen dan data van de sensor binnen via de observer.

Indien, bij het uitvoeren van een workflow, een gebeurtenis uitgevoerd wordt, moet deze zich inschrijven bij de observer die op de correcte sensor staat te luisteren. Wanneer de observer nieuwe data van de sensor binnenkrijgt, worden alle ingeschreven klassen verwittigd; hiervoor wordt de *notificeer functie* gebruikt. Eens een gebeurtenis plaatsgevonden heeft, moet de gebeurtenis uitgeschreven worden uit de observer.

Op figuur 45 is uitgebeeld hoe het observer patroon geïntegreerd is in het project.

Indien bijvoorbeeld een nieuwe observer toegevoegd wordt, die de lichtintensiteit gaat meten, moet er een nieuwe subklasse van de observer klasse gemaakt worden. Hierop kunnen nieuwe gebeurtenissen, die de lichtintensiteitsensor gebruiken, zich inschrijven aan de hand van de *addObserver functie*.

Om een gebeurtenis uit te schrijven uit een observer, wordt de functie *deleteObserver* gebruikt.



Figuur 45: UML klassendiagram weergave van het observer patroon

7.4.5 Meerdere looks en feels, abstract factory patroon

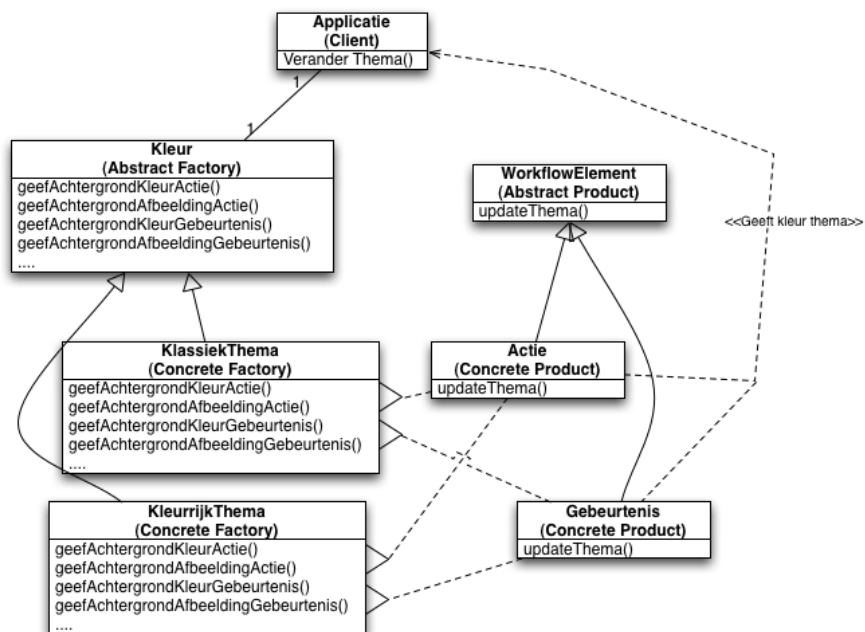
In sectie 6.4.1 wordt geïntroduceerd hoe de gebruikersinterface gemakkelijk kan aangepast worden aan de hand van verschillende thema's. Zo kan de kleur en vorm aangepast worden. Om dit mogelijk te maken, is er gebruik gemaakt van het abstract factory patroon [20].

Aan de hand van dit patroon kunnen nieuwe looks en feels toegevoegd worden.

Een kleine verandering is aangebracht op dit patroon; in de implementatie, zullen de concrete factories de concrete producten niet maken, maar voorzien van alle nodige informatie voor de gebruikersinterface.

Op figuur 46 wordt getoond dat, bij het maken van (bijvoorbeeld) acties en gebeurtenissen aan de applicatie gevraagd wordt welk kleurenthema (de abstract factory) gebruikt wordt. Hierna zal een actiebouwblok aan de abstract factory vragen om de kleur en vorm van een actie te geven. Deze kleur en vorm worden dan ingevuld in het actiebouwblok.

De reden van deze aanpassing op het standaard patroon, is dat we de look and feel willen aanpassen bij het uitvoeren van de applicatie. Op deze manier kan, bij het maken van een workflow, de kleur en vorm van de bouwblokken aangepast worden, zonder dat de bouwblokken, knopen en patronen opnieuw geïnitieerd moeten worden.



Figuur 46: UML klassendiagram weergave van het abstract factory patroon

7.4.6 Workflow elementen, composite patroon

Een composite patroon [43] beschrijft een manier om een object op dezelfde manier als een groep te behandelen; door groepen en objecten te combineren, kunnen we een boomstructuur bouwen. Het uniform gebruiken van de groepen en objecten, wordt gedaan door deze dezelfde interface te geven. Een composite patroon wordt aan de hand van 3 klassen beschreven, namelijk:

1. Component: stelt de abstracte klasse voor van alle componenten
2. Leafs: stelt de objecten voor
3. Composite: stelt groepen voor, die bestaan uit verschillende composities en/of leafs

Op figuur 39 (op pagina 99) wordt dit patroon gebruikt voor het voorstellen van de structuur van de workflow. Groepen stellen de composite elementen voor, waar de andere bouwblokken en knopen de leafs voorstellen. Tenslotte is een workflow element de component van het composite patroon.

7.4.7 Factory method patroon om functionaliteit aan een actie, gebeurtenis toe te voegen

Het factory method patroon [21] is een aanmaakpatroon. Dit patroon maakt objecten aan, zonder aan te geven welke functionaliteit hieraan gebonden wordt.

Dit is nodig om bij het maken van een bouwblok, dat een actie of gebeurtenis is, de functionaliteit open te laten. Voor dit bouwblok (actie of gebeurtenis) moet vervolgens nog een specifieke functionaliteit gedefinieerd worden. De functionaliteit, die later gebonden wordt, steunt op het bouwblok; zo bezit het bouwblok de referentie naar het volgende bouwblok.

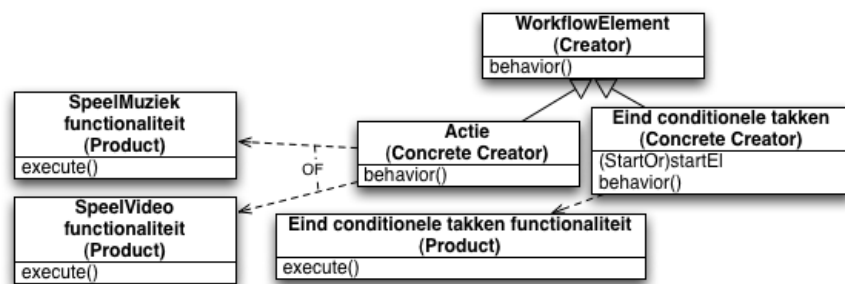
Door separation of concerns tussen de creator en functionaliteit, is het mogelijk om de maker van de workflow te laten kiezen welke functionaliteit gebonden moet worden aan een actie of gebeurtenis. De actie of gebeurtenis zal dan de geselecteerde functionaliteit aanmaken.

Het factory method patroon bestaat uit 2 delen, namelijk:

1. Creator: de creator is onze taal het bouwblok (actie of gebeurtenis) die de functionaliteit gaat aanmaken
2. Producten: zijn de verschillende functionaliteiten die gemaakt kunnen worden door een creator.

In het voorbeeld op figuur 47 zien we een subset van de implementatie. Er wordt getoond hoe een actie (een creator) 2 verschillende functionaliteiten (producten) kan aanmaken.

De *eind conditionele takken* knoop bouwt automatisch de enige functionaliteit die gebonden is. De reden hiervoor is dat het niet mogelijk is voor de maker van de workflow om verschillende functionaliteiten te hebben voor één *eind conditionele takken* knoop.



Figuur 47: UML klassendiagram weergave van het factory method patroon

7.4.8 State en visitor patroon voor de uitvoering van een workflow

De engine is verantwoordelijk voor het uitvoeren van een workflow. De implementatie van de engine is een combinatie van 2 verschillende patronen, het state en het visitor patroon [2].

1. Het *state patroon* staat in voor het correct doorgeven van de controle van de uitvoering naar de volgende elementen (bouwblok, knoop of patroon). Het volgende element wordt bepaald aan de hand van het huidige element, wat betekent dat elk element een referentie heeft naar een volgend bouwblok; dit voorkomt lange conditionele takken voor het bepalen van een volgend element.
2. Het *visitor patroon* is een manier om een algoritme te splitsen van een objecten structuur. In de engine, die instaat voor het uitvoeren van de workflow, wordt dit algoritme gebruikt voor het weergeven van de verschillende vensters, die door de verschillende elementen bij het uitvoeren van de workflow weergegeven worden.

Het algoritme wordt, aan de hand van een visitor klasse, aan alle klassen doorgegeven. De klassen worden aangesproken aan de hand van een dubbele dispatch. Elke bezochte klasse moet een functie voorzien om op deze visitor klasse in te kunnen spelen. Een visitor kan door de hiërarchie gestuurd worden. In de voorgestelde taal in dit onderzoek kan een hiërarchie opgebouwd worden aan de hand van groepen.

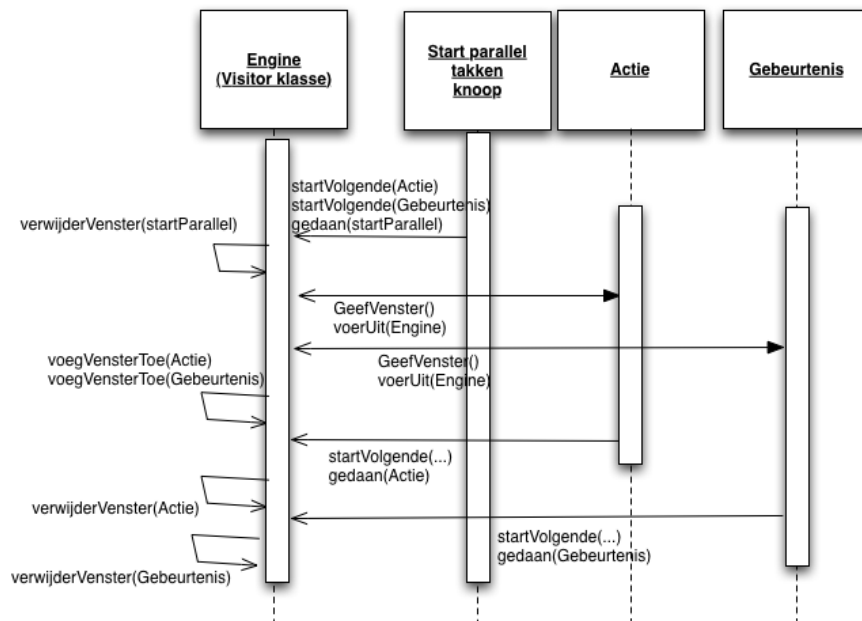
Dit patroon kan ook verschillende bouwblokken tegelijkertijd bezoeken; dit betekent dat een visitor tegelijkertijd in verschillende takken van de workflow gepropageerd kan worden, wat nodig is voor het uitvoeren van parallelle takken.

In onze implementatie werd er gekozen om state transitie tussen de verschillende bouwblokken te gebruiken om de visitor klasse door te geven. Eens een visitor in een bouwblok binnenkomt, wordt het bouwblok uitgevoerd en geeft deze het gewenste uitvoerscherm aan de visitor. De visitor zorgt ervoor dat het venster van het bouwblok correct weergegeven wordt bij het uitvoeren van de workflow. Een subset van de klassen, die betrokken zijn bij het visitor en state patroon, wordt weergegeven in figuur 49.

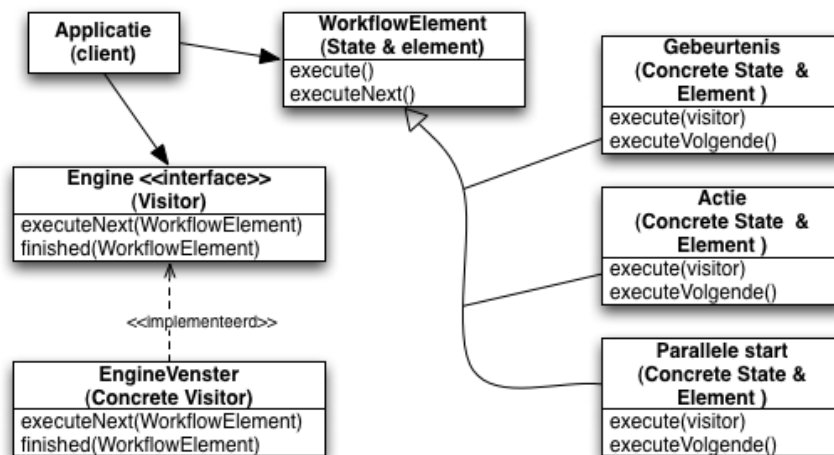
Het gebruiken van het visitor patroon, voor het implementeren van de engine die de workflow uitvoert, wordt ook in BPEL gebruikt [19, 15]. *The visitor design pattern meets these requirements as it separates the data structures and the semantics. The behavior of each BPEL element is represented as a visit method and the set of these methods contained in a class (the visitor). As the engine code will be modular, it will be easy to understand, maintain, extend by inheritance, and modify by visit method overriding.*

Aan de hand van het sequentie diagram op figuur 48, wordt getoond hoe beide patronen samenwerken.

1. Het eerste element, het starten van 2 parallel takken, heeft 2 volgende bouwblokken. Hierdoor worden deze 2 volgende bouwblokken gestart. Er wordt nu gebruik gemaakt van het state patroon, om de volgende bouwblokken te bepalen. Eens de volgende bouwblokken bepaald werden aan de hand van de referentie aanwezig in het state patroon, wordt de visitor klasse doorgegeven aan beide elementen.
2. Voor het starten van de volgende bouwblokken, wordt het uitvoervenster van de *start parallel takken* gesloten.
3. De volgende 2 bouwblokken worden gestart, op dit moment vraagt de visitor klasse de vensters, die worden weergegeven bij het uitvoeren van de workflow, aan beide bouwblokken (in dit geval een actie en gebeurtenis). Beide vensters worden aan het iPhone scherm toegevoegd.
4. De actie geeft weer dat zij klaar is en bepaalt het volgende bouwblok aan de hand van het state patroon. Ondertussen zorgt het visitor patroon ervoor dat het uitvoervenster van de actie correct verwijderd wordt.
5. Even later geeft de gebeurtenis weer dat deze plaatsgevonden heeft en start het volgende bouwblok. Ondertussen zorgt het visitor patroon dat het uitvoervenster van de gebeurtenis correct verwijderd wordt.



Figuur 48: UML sequentie diagram voor een state en visitor patroon



Figuur 49: UML klassendiagram weergave van een state en visitor patroon

7.5 Multitouch gestures

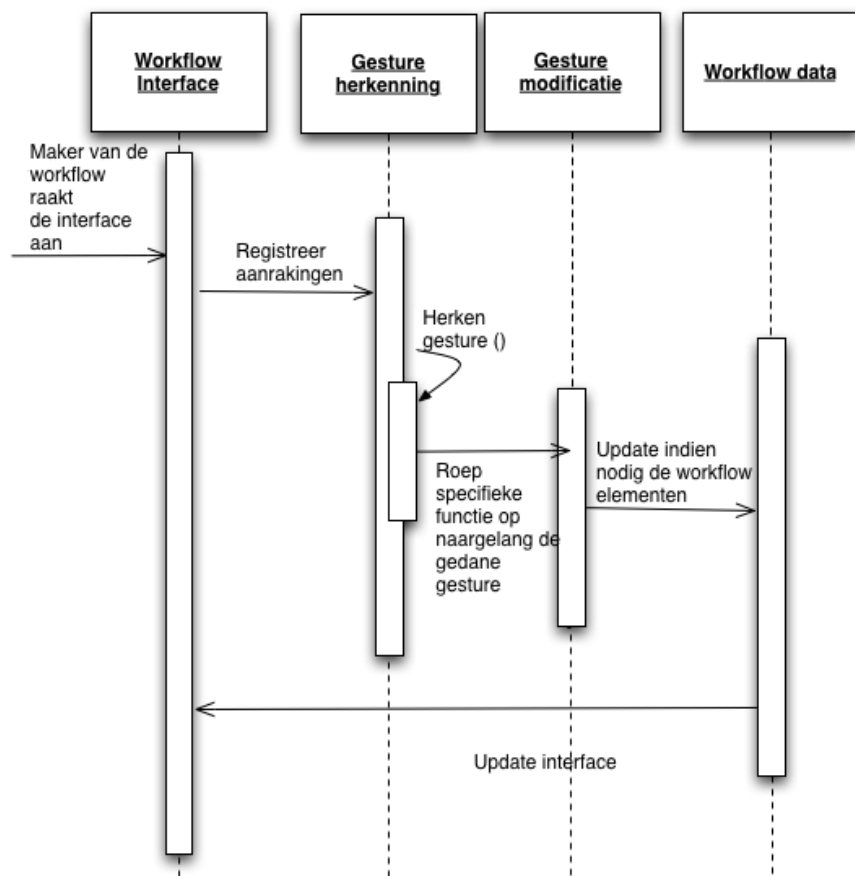
Na onderzoek [5, 33, 45] bleek dat een multitouch gesture uit 2 delen bestaat, namelijk enerzijds het herkennen van de gesture en anderzijds de gevolgen van deze gestures op de workflow. In de implementatie van dit onderzoek, is er gekozen om deze 2 delen in afzonderlijke klassen te definiëren. Dit laat toe om, indien gewenst, dezelfde gestures te gebruiken en er een totaal ander gevolg aan te binden of omgekeerd.

In onze programmeeromgeving zijn er 2 verschillende soorten gebruikersinterfaces, namelijk die van de workflow en die van de samengestelde gebeurtenissen. De gestures van beide interfaces zijn dezelfde, enkel de modificaties, die deze teweegbrengen, verschillen.

Aan de hand van deze implementatie, is het nu gemakkelijk om dezelfde herkenning van gestures aan de hand van dezelfde klasse te gebruiken voor beide interfaces. De herkenning van gestures klasse wordt hierna gekoppeld aan een andere modificatie klasse, die specifiek is voor de interface. Bijvoorbeeld voegt de *voegtoe gesture* een actie of gebeurtenis toe in een workflow, wanneer in een samengestelde gebeurtenis deze zelfde gesture een gebeurtenis toevoegt.

In figuur 50 wordt er getoond, aan de hand van een sequentiediagram, hoe multitouch gestures doorheen de applicatie gebruikt worden.

1. De maker van de workflow drukt op het scherm; op dit moment, zal de interface de aangeraakte coördinaten aan de gesture herkenning klasse doorgeven. De punten die worden doorgegeven, zijn de begin- en eindpositie van elke vinger.
2. De gesture herkenningsklasse gaat nu automatisch, aan de hand van de aangeraakte posities, nagaan indien de maker van de workflow een zekere gesture heeft uitgevoerd.
3. Indien de maker van de workflow geen gesture heeft uitgevoerd, wordt deze schermaanraking genegeerd (dit staat niet op het sequentiediagram uitgebeeld)
4. Indien de maker van de workflow een gesture heeft uitgevoerd, wordt de corresponderende modificatie opgeroepen in de modificatie klasse.
5. De opgeroepen modificatie functie gaat de workflow dataelementen aanpassen, naargelang de semantiek van de gesture.
6. De gebruikersinterface wordt nu geüpdatet.

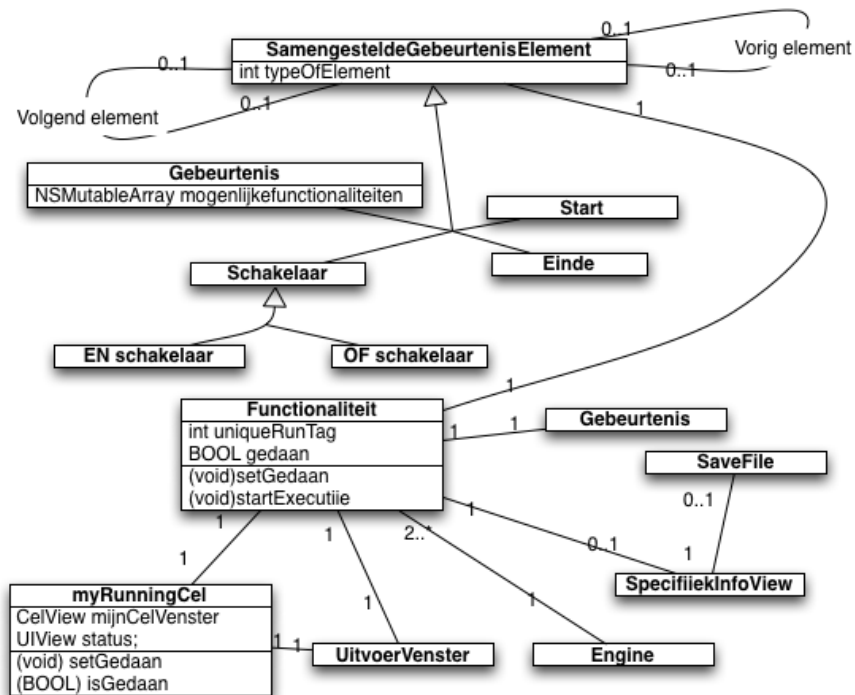


Figuur 50: UML sequentie diagram voor het opvangen en verwerken van multitouch gestures

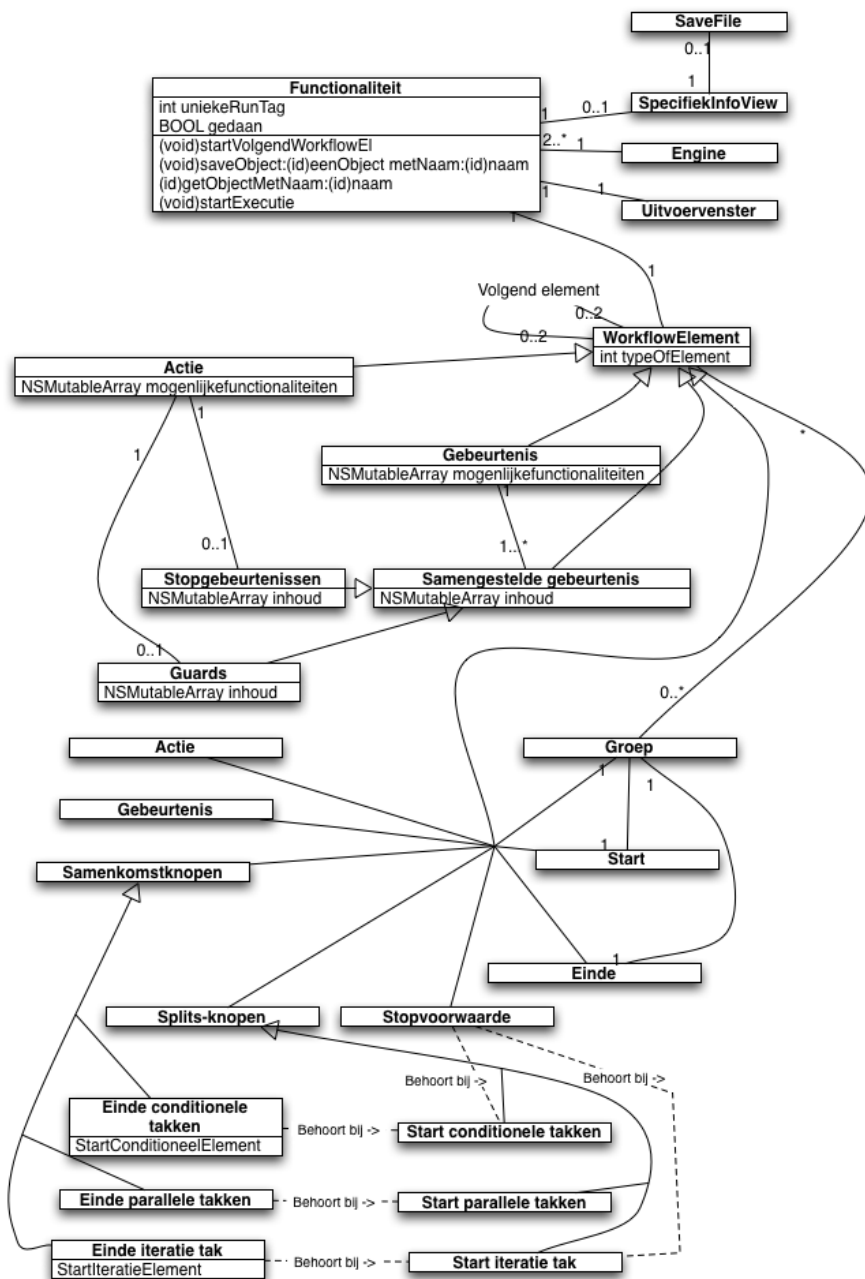
7.6 Meta model

Als overzicht van de implementatie sectie, wordt een vereenvoudigde versie van het meta model gegeven, met alle belangrijke aspecten van de hierboven besproken onderwerpen. Het meta model van de workflow wordt weergegeven in figuur 52.

Het meta model van samengestelde gebeurtenissen wordt weergegeven in figuur 51.



Figuur 51: UML klassendiagram weergave van een vereenvoudigd meta model van samengestelde gebeurtenissen.



Figuur 52: UML klassendiagram weergave van een vereenvoudigd meta model van de workflow.

8 Nagaan op het correct gebruik van variabelen

Het nadeel om gebruik te maken van een omgeving om variabelen op te slaan en te gebruiken, is dat het mogelijk is om een variabele te gebruiken die voorheen nooit aangemaakt werd. Om dergelijke fout na te gaan, is er een statisch controle mechanisme, dat nagaat, vooraleer een workflow opgeslagen of uitgevoerd wordt, of alle variabelen correct geïnitieerd zijn. Dit laat toe om fouten te detecteren en de gebruiker te helpen bij het bouwen van de workflow.

Het basisidee van het controle mechanisme is om doorheen alle elementen te lopen en een container aan te maken van alle variabelen die geïnitieerd worden. Indien een variabele gebruikt wordt in een bouwblok, wordt er nagegaan of deze aanwezig is in de omgeving. In deze sectie wordt het gemaakte algoritme geïntroduceerd.

8.1 In een sequentieel patroon

Het controleren van de gebruikte en gemaakte variabelen in een sequentieel patroon verloopt op de volgende manier:

1. Indien een nieuwe variabele geïnitieerd wordt in een bouwblok, wordt deze in een container bijgehouden.
2. Indien een variabele gebruikt wordt, kunnen zich 2 scenario's voordoen:
 - (a) De gebruikte variabele in een bouwblok is al een deel van de in de container geïnitieerde elementen. In dit geval wordt er niets gedaan.
 - (b) De gebruikte variabele in een bouwblok is nog geen deel van de in de container geïnitieerde elementen. In dit geval zal het element toegevoegd worden in de *error container*.

Nadat alle bouwblokken uit het sequentieel proces achtereenvolgens overlopen werden, zal de *error container* aan de gebruiker worden weergegeven. Op figuur 53 zien we een sequentiële workflow, waar een testvariabele gebruikt wordt, maar die nooit geïnitieerd werd.

Wanneer de gebruiker hierna de workflow probeert op te slaan, wordt er weergegeven dat de testvariabele niet correct geïnitieerd werd.



Figuur 53: Grafische weergave van het nagaan van variabelen in een sequentieel patroon en weergave van slecht geïnitieerde variabelen bij het opslaan van de workflow.

8.2 In een parallel patroon

Het nagaan of variabelen in een sequentieel patroon correct geïnitieerd zijn, is tamelijk triviaal.

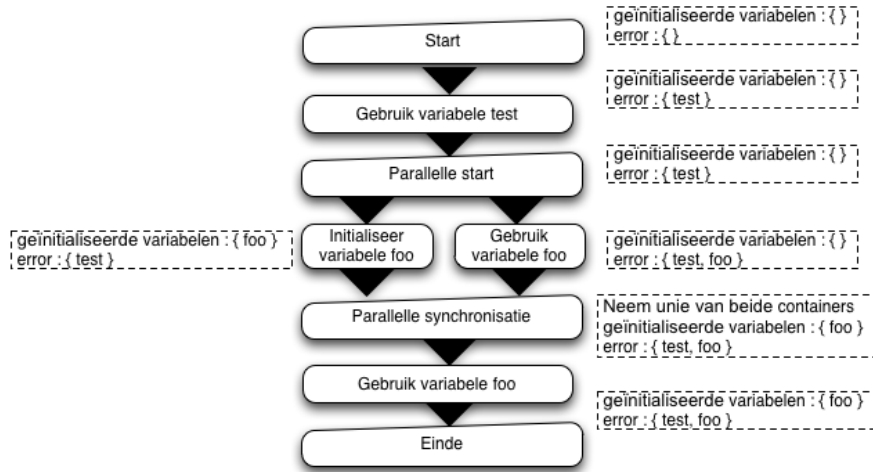
Het probleem is echter dat in het parallelle patroon variabelen in beide takken geïnitieerd kunnen worden en dat er geen controle is over welke tak als eerste uitgevoerd gaat zijn. Wat wel zeker is, is dat eens beide takken samenkomen zijn, in het vervolg van de workflow gebruik kan gemaakt worden van de geïnitieerde variabelen in de parallelle takken.

Bij het uitvoeren van verschillende concurrente takken is het niet mogelijk om te voorspellen of tak A voor tak B, afgelopen zal zijn. Indien er een variabele in tak A geïnitieerd wordt en deze in tak B gebruikt wordt, zal dit als een fout van initialiseren van variabelen aanzien worden.

Het controleren van de gebruikte en gemaakte variabelen in een parallel patroon verloopt op de volgende manier:

1. Bij het starten van 2 parallelle takken, worden de gebruikte containers gedupliceerd, één voor elke tak. De containers overlopen beiden takken concurrent van elkaar na de *parallelle start* knoop.
2. Elke tak wordt overlopen en de error en geïnitieerde containers worden voor elke tak onafhankelijk ingevuld.
3. Bij het synchroniseren van beide parallel takken komen beide takken aan met verschillende containers; deze moeten ook in de synchronisatie knoop samengevoegd worden zodat de rest van de workflow kan afgetoetst worden met 1 container. We zijn zeker dat beide takken, na het uitvoeren van het parallel patroon, uitgevoerd zijn, vooraleer de rest van de workflow na het synchronisatie punt, gestart wordt. Om deze reden wordt de unie genomen van beide containers uit beide takken.
4. Met de unie van de containers wordt de rest van de workflow afgetoetst.

Op het einde van het toetsen van de workflow, wordt de *error container* getoond aan de maker van de workflow. Als voorbeeld wordt er een parallelle tak met bijhorende containers volledig uitgewerkt in figuur 54.



Figuur 54: Grafische weergave van het nagaan van variabelen in een parallel patroon

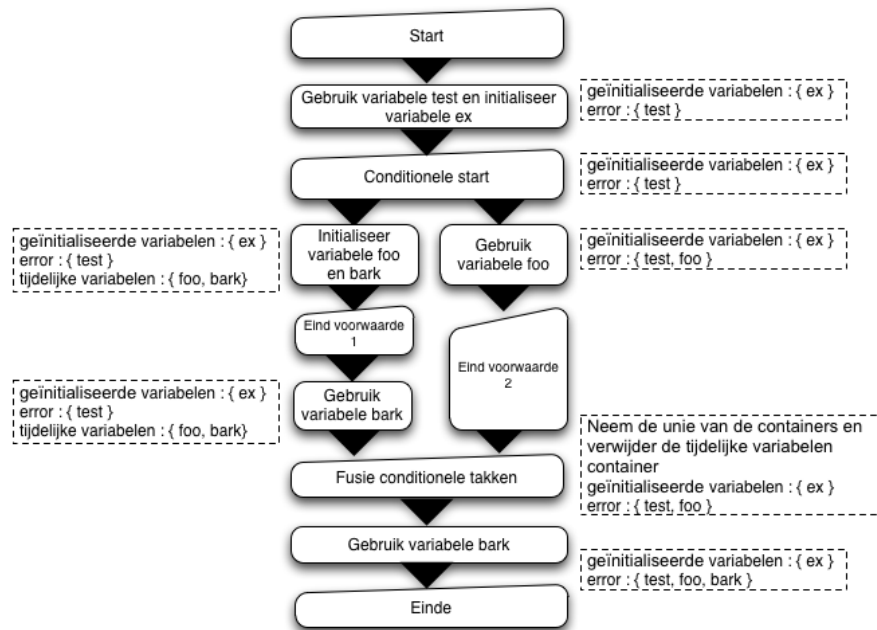
8.3 In een conditioneel patroon

In een conditioneel patroon worden ook verschillende takken concurrent van elkaar uitgevoerd. Het verschil met het parallel patroon is dat, in een conditioneel patroon, niet alle takken uitgevoerd worden. We kunnen bij het nagaan van de variabelen niet voorspellen welke conditionele tak uitgevoerd zal worden. Om deze reden wordt een tijdelijke container geïntroduceerd, die de geïnitieerde variabelen gaat bijhouden voor de verschillende takken. Het controleren van de gebruikte en gemaakte variabelen, in een conditioneel patroon, verloopt op de volgende manier:

1. Bij het starten van de conditionele takken worden de gebruikte containers gedupliceerd, één voor elke tak. De containers overlopen de conditionele takken concurrent van elkaar. Op dit moment wordt er een tijdelijke container aangemaakt, die de geïnitieerde variabelen voor elke tak gaat bijhouden.
2. Indien er in een tak een variabele gebruikt wordt, wordt er zowel in de tijdelijke als in de geïnitieerde variabelen container gekeken.
 - (a) De variabele wordt in de *error container* geplaatst, indien deze variabele niet in de geïnitieerde of tijdelijke container zit.
 - (b) Indien de variabele aanwezig is in één van beide containers, wordt er niets gedaan.

3. Bij het einde van de conditionele takken kunnen we niet voorspellen welke conditionele tak bij het uitvoeren van de workflow genomen zal worden. Enkel de variabelen die in beide takken geïnitieerd zijn, zijn zeker om geïnitieerd te worden na het uitvoeren van dit patroon. Voor deze reden wordt, na het uitvoeren van het einde van de conditionele takken, de intersectie van de tijdelijke containers toegevoegd aan de geïnitieerde variabelen. De unie van de error container wordt ook genomen om verder de workflow te analyseren.
4. Hierna worden de tijdelijke containers verwijderd

Op het einde van het toetsen van de workflow, wordt de fout container getoond aan de maker van de workflow. Als voorbeeld wordt er een conditionele tak met bijhorende containers volledig uitgewerkt in figuur 55.



Figuur 55: Grafische weergave van het nagaan van variabelen in een conditioneel patroon

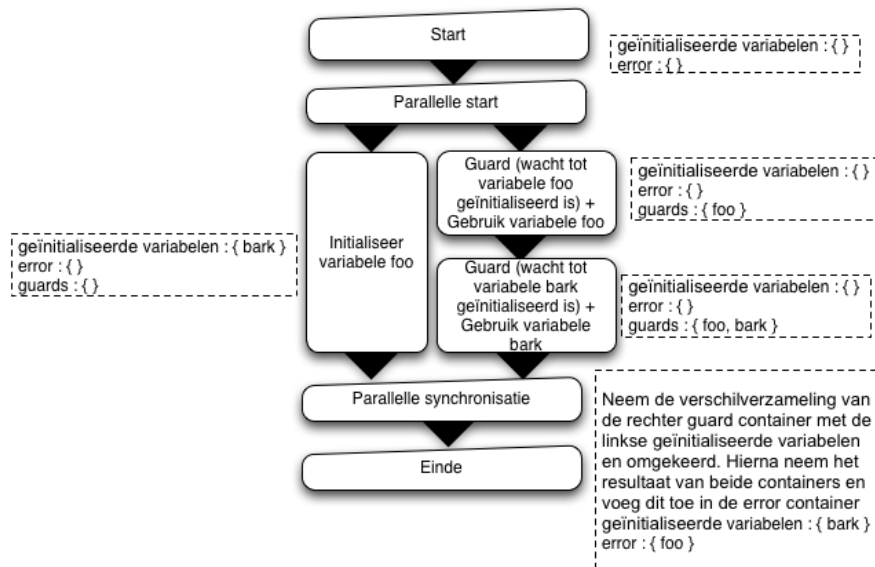
8.4 In een parallel patroon met guards

Tenslotte, kunnen guards ervoor zorgen dat er gewacht wordt tot een zekere variabele geïnitieerd is, onder andere in een andere tak van de workflow, vooraleer een actie uitgevoerd wordt. Dit betekent dat, indien de variabele in een andere concurrente tak geïnitieerd wordt en een guard aan de actie, die de variabele gebruikt, gekoppeld wordt, wordt deze variabele correct gebruikt. Bij een conditionele tak, is dit algoritme maar geldig tot aan de stopblokken van de voorwaarden. Hierna is er maar één tak meer actief en kan er niet gewacht worden op de andere tak vermits deze niet meer bestaat. Om rekening te houden met guards in concurrente takken, wordt er gebruik gemaakt van de guard container.

Het controleren van de gebruikte en gemaakte variabelen in een concurrent proces met guards, verloopt op de volgende manier:

1. Bij het starten van 2 parallelle takken, worden de gebruikte containers gedupliceerd, één voor elke tak. De containers overlopen beiden takken concurrent van elkaar. Er wordt er een guard container aangemaakt, die variabelen, waarop een guard staat te wachten, gaat bijhouden.
2. Bij elke guard, die wacht het initialiseren van een variabele, wordt de variabele in de guard container toegevoegd. Ook worden de parallelle takken overlopen en de error en geïnitieerde containers ingevuld voor elke tak afzonderlijk.
3. Eens de parallelle takken synchroniseren, zal de verschilverzameling van de guard container met de geïnitieerde variabele van de andere tak genomen worden. Dit wordt voor beide guard containers gedaan. De verschilverzameling van deze verzamelingen zijn de variabelen waarop gewacht wordt aan de hand van een guard, maar die nooit geïnitieerd werden. De verschilverzameling van deze verzamelingen wordt hierna aan de error container toegevoegd.
4. Met de unie van de error containers en geïnitieerde containers wordt de rest van de workflow afgetoetst.

Op het einde van het toetsen van de workflow, wordt de fout container getoond aan de maker van de workflow. Als voorbeeld wordt er een parallelle patroon met guards met bijhorende containers volledig uitgewerkt in figuur 56.



Figuur 56: Grafische weergave van het nagaan van variabelen in een parallel patroon met guards

8.5 Visitor patroon.

De implementatie van dit algoritme werd gedaan aan de hand van het visitor patroon (zoals uitgelegd in sectie 7.4.8). Elk element in de workflow zal overlopen moeten worden aan de hand van de visitor klasse. De visitor klasse houdt de verschillende containers bij en zorgt voor een correcte samensmelting en duplicatie bij de verschillende patronen.

9 Validatie

Als validatie is er gekozen om de ontwikkelde programmeertaal en -omgeving te laten uitproberen door verschillende niet-programmeurs, met een verschillende achtergrond en leeftijd.

Dit experiment vormt geen volledige evaluatie. Enkel een grootschalig experiment kan voldoende resultaten bieden om algemene conclusies te trekken en daarvoor was in de context van deze thesis geen tijd. Wat deze testen wel aanduiden, is een eerste indicatie of de aangeboden taal en omgeving bruikbaar zijn door niet-programmeurs.

9.1 Hoe

Er is gekozen om aan de hand van field studies [30] te werken. Zodat de programmeertaal en -omgeving getest kan worden in verschillende omgevingen zoals thuis in de zetel of op de trein.

Bij het testen van de programmeertaal en -omgeving, werd er gevraagd om luidop te redeneren en de stappen uit te leggen die uitgevoerd werden. Op deze manier kan er, indien er fouten gemaakt worden, nagegaan worden indien er consistent, bij verschillende personen op dezelfde plaats in de programmeertaal en -omgeving, een redeneerfout gemaakt wordt. Het uitproberen van de programmeertaal en -omgeving door de testpersonen verliep op de volgende manier:

- Er wordt een inleiding gegeven over de verschillende taalelementen en hoe deze te gebruiken
- Samen met de testpersoon wordt er een voorbeeld uitgewerkt en uitgevoerd.
- Hierna wordt er aan de persoon gevraagd om zelfstandig enkele scenario's uit te werken. Terwijl de persoon de scenario's zelfstandig implementeert, worden er 2 zaken gemeten, primo of de persoon in de test slaagt en, secundo, in welke tijdsspanne de verschillende scenario's opgelost geraken. Aan iedereen wordt gevraagd de verschillende scenario's (uit volgende sectie) te implementeren.
- Na het implementeren van de scenario's, wordt er gevraagd aan de persoon om punten (tussen 0 en 10) te geven op verschillende aspecten van de programmeertaal en -omgeving en hierna wordt er een klein interview afgenomen om de algemene indruk van betreffende testpersoon te kunnen bevragen.

9.2 De gevraagde scenario's

Er is gekozen om verschillende scenario's op te stellen, die elk verschillende taalelementen introduceren. Zo kan er gezien worden bij welk taalelement de personen de meeste problemen tegenkomen. De eerste scenario's zijn tamelijk gemakkelijk gehouden probleemstellingen, zodat de testpersonen zich niet blindstaren op het probleem, maar zich kunnen concentreren op hoe de workflow voor te stellen in de voorgestelde taal en omgeving. In het laatst gevraagde scenario, komen verschillende taalelementen aan bod.

Scenario 1 : Maak een applicatie die wacht tot de gebruiker op een knop op het scherm drukt. Eens de gebruiker op de knop gedrukt heeft, moet er een muziekje afgespeeld worden gedurende 10 seconden of tot de gebruiker op de fysieke homeknop duwt.

Scenario 2 : Maak een applicatie die een muziekstuk speelt gedurende 10 seconden en indien de gebruiker gedurende die tijd op de homeknop duwt, wordt het muziekje gestopt en wordt een ander muziekstuk gespeeld. Indien de gebruiker niet binnen de 10 seconden op de homeknop duwt, wordt het afspelen van het liedje gestopt. Hierna wordt een teller, die tijdens 10 seconden aftelt, gestart.

Scenario 3 : Maak een applicatie die een muziekje steeds opnieuw speelt, zolang de gebruiker, binnen de 10 seconden na het afspelen van het liedje, op de knop op het scherm duwt.

Scenario 4 : Maak een applicatie die een stuk muziek speelt en ondertussen, als de smartphone aan een hevige beweging onderworpen wordt, wordt er een knop beschikbaar gesteld op het scherm. Als de gebruiker op de knop duwt en het muziekstuk afgelopen is, betekent dit het einde van de applicatie.

Scenario 5 : Maak een applicatie die, indien er binnen de 10 seconden, na het starten van de applicatie, op een knop op het scherm gedrukt wordt, start er een muziekje. Indien deze 10 seconden verlopen zijn en er niet op de knop gedrukt wordt, moet een SMS verzonden worden naar een nummer naar keuze. In beide gevallen, dus indien de gebruiker op de knop duwt of niet, moet er een SMS verzonden worden naar het zelfde nummer als het nummer hier zoniet ingegeven. Om het nummer van het laatste sms bouwblok in te geven, moet er gebruikt gemaakt worden van het browsen van variabelen.

9.3 Resultaten

De testen werden afgelegd door een aantal testpersonen van verschillende leeftijd en met een andere achtergrond; deze worden verdeeld in de volgende leeftijdscategorieën:

1. 2 kinderen van 10 en 11 jaar
2. 2 leerlingen middelbaar onderwijs, van 14 en 17 jaar
3. 3 personen met universitaire opleiding, van 21 tot 29 jaar, zonder programmeer achtergrond
4. 2 personen met universitaire opleiding, van 22 jaar met programmeer achtergrond
5. 4 volwassenen met verschillende achtergrond, ouder dan 50 jaar, die niet opgegroeid zijn met computers en smartphones.

Het eerste resultaat dat wordt weergegeven, is de tijdsduur nodig om de verschillende scenario's uit te voeren; de tijd wordt weergegeven, naargelang hier boven aangeduide categorieën. De tijd wordt uitgedrukt in minuten

Leeftijds- categorieën	1	2	3	4	5
scenario 1	2-3 min	1 min	≤ 1 min	≤ 1 min	2-3 min
scenario 2	!	1-2 min	1-2 min	± 1 min	2-3 min
scenario 3	!	1-2 min	1-2 min	± 1 min	2-3 min
scenario 4	± 5 min.	1-2 min	1-2 min	± 1 min	2-3 min
scenario 5	!	2-3 min	2-3 min	1-2 min	5-6 min

De kinderen van 10 en 11 jaar hadden problemen bij het begrijpen van de conditionele takken. Zij begrepen niet dat als één tak aan de eindvoorwaarden (van een conditioneel patroon) geraakte, dat de andere tak gestopt werd en de gebruiker dus (bijvoorbeeld) niet meer op de knop op het scherm kan drukken. De proefpersonen vanaf 14 jaar blijken hiermee dan helemaal geen probleem te hebben en implementeerden de applicaties zelfstandig en correct. Ook werden de stopvoorwaarden correct geplaatst in de voorbeelden. Sommige testpersonen maakten eerst een actie aan om een tijdsduur uit te drukken. Na even nagedacht te hebben, werd deze verwijderd om er een gebeurtenis van te maken.

Het gebruiken van de gestures en browsen van variabelen werd aangenaam gevonden door de proefpersonen. Ook blijkt het gebruik van verschillende kleuren en vormen doorheen de workflow een pluspunt te bieden. Tenslotte bleek het voor de gebruikers gemakkelijker om te werken met sequenties en parallelle takken dan met conditionele takken en iteraties. De algemene indruk van de voorgestelde taal en omgeving is positief; dit volgt trouwens ook uit het onderstaande gemiddelde van de punten, die de gebruikers gaven na het uitvoeren van de scenario's, op de volgende aspecten van de programmeertaal en omgeving.

Vraag	Gemiddelde punten gegeven door de testpersonen op een schaal van 0 tot 10
Bruikbaarheid van gestures	8,5
Intuïtiviteit van het gebruik van de gestures	8
Gemak van het browsen van variabelen	8,5
Gebruikersinterface	8
Gebruik van vormen en kleuren	8,5
Acties en activiteiten	8
Gemak van een sequentiële tak	9
Gemak van een parallele tak	8,5
Gemak van een conditionele tak	7,5
Gemak van een iteratie	7

10 Besluit & toekomstig werk

Als besluit zal er eerst een samenvatting gegeven worden van het geleverd werk gevolgd door een discussie en een kritische blik omtrent het gedane onderzoek. Om deze sectie te beëindigen, zullen er mogelijke toekomstige uitbreidingen op de huidige programmeertaal en -omgeving besproken worden.

10.1 Samenvatting

In dit onderzoek is een nieuwe programmeertaal en -omgeving ontworpen, die niet programmeurs toelaat om, aan de hand van workflows, applicaties te bouwen op de eigen smartphone.

De gemaakte programmeertaal bestaat uit verschillende bouwstenen, namelijk :

- Gebeurtenissen: een gebeurtenis is iets dat zich voordoet in tijd en ruimte.
- Samengestelde gebeurtenissen : een samengestelde gebeurtenis is een combinatie van verschillende gebeurtenissen, gecombineerd aan de hand van de EN en OF schakelaar.
- Acties: een actie wordt zelfstandig uitgevoerd door de smartphone; deze kunnen zelfstandig uitgevoerd worden zonder gebruikersinteractie. Acties kunnen gecombineerd worden met guards en stopgebeurtenissen om een nieuwe bouwblok te maken.
 - Stopgebeurtenis: een stopgebeurtenis is een samengestelde of eenvoudige gebeurtenis, die vastlegt hoe een actie vroegtijdig afgerond kan worden.
 - Guards: een guard is een samengestelde of eenvoudige gebeurtenis, die beschrijft welke gebeurtenissen moeten plaatsgevonden hebben, vooraleer een actie uitgevoerd wordt.

Deze verschillende bouwblokken kunnen we vervolgens combineren aan de hand van verschillende patronen om een applicatie te bouwen. De ondersteunde patronen zijn:

- Sequentieel patroon: dit patroon gaat in een workflow verschillende bouwblokken verbinden en een volgorde aanbrengen.
- Parallel patroon: dit patroon laat toe om verschillende deelworkflows concurrent van elkaar te laten uitvoeren

- Conditioneel patroon: een conditioneel patroon bestaat uit verschillende takken. Voor elke tak wordt er een voorwaarde vermeld, die een deelworkflow kan zijn. Een tak in een conditioneel patroon wordt gekozen indien de voorwaarde van die tak als eerste afgelopen is.
- Iteratie patroon: een iteratie patroon heeft een loop voorwaarde en stop voorwaarde. Het patroon zal een deelworkflow herhaaldelijk uitvoeren zolang de loopvoorwaarde voor de stopvoorwaarde afgelopen is.

Om data tussen de verschillende bouwblokken uit te wisselen, wordt er gebruik gemaakt van een globale omgeving. Om zeker te zijn dat de maker van de workflow alle nodige data beschikbaar heeft in de globale omgeving, wordt er een algoritme voorgesteld om foutief gebruik van deze omgeving te detecteren.

Naast het maken van de nieuwe programmeertaal is een programmeeromgeving ontwikkeld om, aan de hand van onze programmeertaal, workflows te kunnen maken op een multitouch smartphone. Door het kleine scherm werden restricties op patronen geplaatst; zo is het bijvoorbeeld enkel mogelijk om 2 conditionele of parallelle taken te maken.

Voor het bouwen van de workflow, werd er gekozen om aan de hand van verschillende multitouch gestures te werken. De multitouch gestures maken het mogelijk om verschillende bouwblokken en patronen in de workflow toe te voegen, in te vullen en te bewerken.

Bouwblokken kunnen geconfigureerd worden, hier is zoveel mogelijk gekozen om via selectiemenu's te werken. Voor het invullen van verschillende identieke configuratie parameters van verschillende acties of gebeurtenissen of het leggen van dynamische referenties tussen verschillende bouwblokken, kan de maker van de workflow doorheen de workflow browsen; hierdoor moet de maker van de workflow, bijvoorbeeld, niet meer in 2 verschillende bouwblokken hetzelfde telefoonnummer invullen.

Een prototype van de voorgestelde programmeertaal en -omgeving is geïmplementeerd voor iPhones in Objective-C. Een eerste reeks scenario's is dan geïmplementeerd op de iPhone om het geheel te demonstreren.

Tenslotte is de voorgestelde programmeertaal en -omgeving uitgetest door verschillende testpersonen met een verschillende achtergrond. Jongeren vanaf 14 jaar tot oudere mensen bleken geen probleem te hebben om de gevraagde scenario's te implementeren in de aangeboden workflowtaal en omgeving.

10.2 Discussie

In de scope van dit onderzoek, is er gekozen om de programmeertaal en -omgeving te implementeren voor één specifiek platform, namelijk de iPhone. Het zou mogelijk zijn om de beschreven programmeertaal en -omgeving op een ander, gelijkaardig, platform te implementeren, bijvoorbeeld op android-toestellen. Op deze laatste toestellen zouden nieuwe functionaliteiten aanwezig kunnen zijn (bijvoorbeeld andere sensoren). Deze nieuwe functionaliteiten voor deze toestellen zouden dan, aan de hand van bouwblokken (acties en gebeurtenissen), toegevoegd moeten worden aan de workflowtaal. De huidige implementatie van de voorgestelde programmeertaal kan altijd verrijkt worden met nieuwe acties en gebeurtenissen. Ook zou het mogelijk kunnen zijn om een bestaande workflows als een bouwblok in een workflow toe te voegen.

Ook zou kunnen bekeken worden hoe de voorgestelde programmeertaal en -omgeving op andere mobiele toestellen bruikbaar zou zijn. Bijvoorbeeld zou voor een tablet een aangepaste programmeeromgeving kunnen uitgewerkt worden.

Aan de hand van een grootschalig experiment zou er kunnen nagegaan worden of de voorgestelde programmeertaal en -omgeving gebruiksvriendelijk is voor niet programmeurs. Uit deze testen zou kunnen blijken dat er nood is aan nieuwe workflow patronen of aanpassingen in de workflowtaal en/of -omgeving.

De uitbreiding en verandering van de programmeertaal en -omgeving worden best altijd gevolgd door een experiment om na te gaan dat veranderingen toegevoerde waarde bieden. Het uitbreiden, gevolgd door experimenten met testpersonen, zou dus best iteratief gebeuren.

10.3 Kritische blik

De voorgestelde programmeertaal en -omgeving bezit geen volledige statische analyse die nagaat of de workflow foutloos uitgevoerd kan worden. Het is bijvoorbeeld niet mogelijk om uit te sluiten dat de maker van de workflow scenario's maakt die tot een deadlock leiden.

De maker van de workflow kan contradicties in de workflow inbouwen, bijvoorbeeld samengestelde gebeurtenissen, die nooit plaats kunnen vinden omdat de verschillende gebeurtenissen mekaar tegenspreken.

Een andere problematische situatie is dat, terwijl in één parallelle tak muziek afgespeeld wordt, er ondertussen een video gestart wordt in een andere parallelle tak; dit is problematisch omdat beide acties de muzikspeler nodig hebben; de mediaspeler kan of muziek of een video afspelen maar niet tegelijkertijd. Om deze probleemsituaties te voorkomen, kunnen guards ingeschakeld worden.

10.3.1 Voordeel van guards

Wanneer verschillende bouwblokken in parallel geplaatst worden, is er geen mogelijkheid om te voorspellen of bouwblokken uit de linkertak voor de bouwblokken uit de rechtertak voltooid zullen zijn.

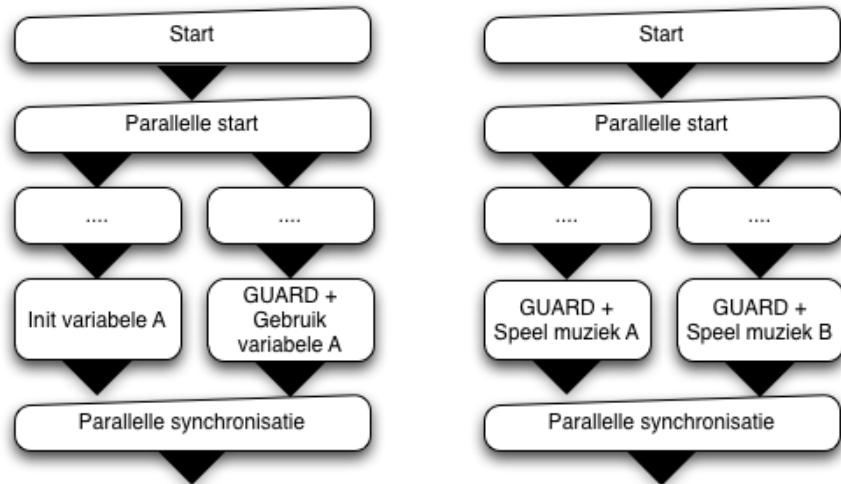
Indien, bij het ontwikkelen van een applicatie, de maker van de workflow zeker wil zijn dat een zeker element uit een andere tak eerder uitgevoerd wordt, kan hij hiervoor een guard inschakelen. Dit kan handig zijn voor het initialiseren van variabelen, zoals uitgebeeld in het volgende voorbeeld.

In het linkse voorbeeld in figuur 57, merken we op dat er 2 parallelle takken aanwezig zijn; op het einde van een lang sequentieel proces initialiseert de ene tak een variabele waar de andere tak deze zelfde variabele gebruikt.

Door voor de actie die de variabele A gebruikt, een guard te definiëren, kunnen we verzekeren dat het initialiseren van variabele A gebeurd is, vooraleer de actie uitgevoerd wordt. Hiervoor wordt er een gebeurtenis gedefinieerd, die plaatsvindt wanneer variabele A geïnitieerd is. Eens variabele A geïnitieerd is, kan de actie die de variabele gebruikt, uitgevoerd worden. Op deze manier kunnen we tussen verschillende parallelle takken een zekere extra semantiek invoeren en verzekeren dat een ander bouwblok voltooid is. Het rechter voorbeeld op figuur 57 toont dat er in 2 parallelle takken tegelijkertijd geluid afgespeeld kan worden; het zou kunnen zijn dat tijdens het uitvoeren van de applicatie het muziekstuk uit de linkse tak gestart wordt en enkele seconden later dan uit de rechtste tak. Deze twee muziekstukken kunnen niet samen afgespeeld worden omdat de mediaspeler enkel één muziekstuk tegelijkertijd kan afspelen. De maker van de workflow wil zeker zijn dat beide muziekstukken, na elkaar, gespeeld kunnen worden.

De oplossing voor dit probleem is het definiëren van guard gebeurtenissen, die plaatsvinden wanneer er geen geluid door het toestel afgespeeld wordt. Veronderstel dat deze applicatie wordt uitgevoerd en dat in de linker tak het muziekstuk als eerste wordt afgespeeld en dat er in de rechtse tak ondertussen de actie gestart wordt om een muziekstuk te spelen. De guard op deze laatste actie zal ervoor zorgen dat het muziekstuk A voltooid is, vooraleer de actie, die muziekstuk B speelt, uitgevoerd mag worden.

Nu worden beide muziekstukken afzonderlijk gespeeld.



Figuur 57: Voorbeeld van het voordeel van guards

Om deze probleemsituaties in verschillende parallelle takken op te lossen, zou er een algoritme geïntroduceerd kunnen worden dat guards suggereert aan de maker van de workflow en dit op de nodige plaatsen.

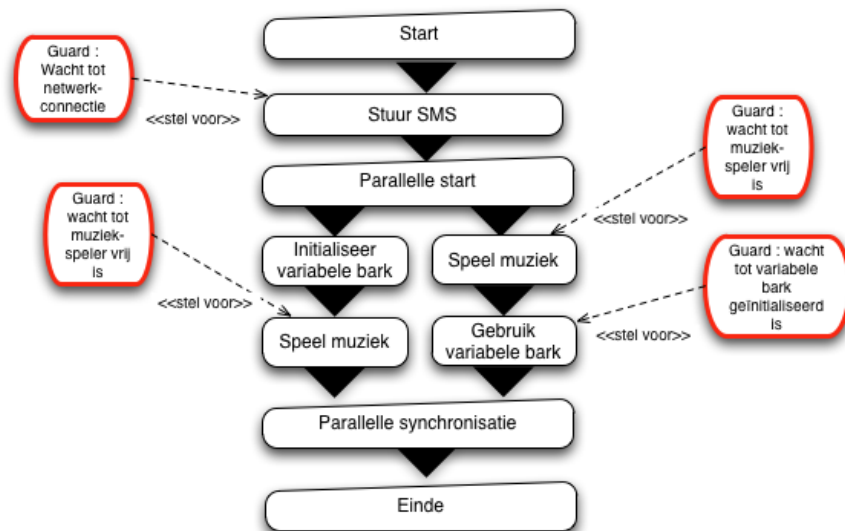
10.4 Toekomstig werk

In deze sectie zal toekomstig werk geïntroduceerd worden dat kan toegevoegd worden in de volgende iteraties van de voorgestelde programmeertaal en -omgeving.

10.4.1 Automatisch suggereren van guards

Bij het ontwikkelen van een workflow, zou een algoritme toegevoegd kunnen worden, dat automatisch guards voorstelt aan de maker van de workflow. Bijvoorbeeld, indien in verschillende concurrente takken muziek afgespeeld wordt, zou dit algoritme kunnen voorstellen om een guard te plaatsen voor het spelen van muziek zodat de mediaspeler beide muziekstukken, op een correcte wijze, na elkaar, kan afspelen. Er wordt in figuur 58 getoond dat verschillende guards op verschillende plaatsen in de workflow voorgesteld zouden kunnen worden.

Als toekomstig werk, kan onderzocht worden op welke plaatsen guards verplicht zouden moeten zijn, zodat de workflow uitgevoerd kan worden zonder enig probleem of ongewild gedrag.



Figuur 58: Grafische weergave van het automatisch voorstellen van guards in de workflow

10.5 Tegelijkertijd uitvoeren van verschillende workflows

In de huidige engine is het slechts mogelijk om één workflow tegelijkertijd uit te voeren. Het zou moeten mogelijk gemaakt worden om verschillende workflows naast elkaar op te starten. Dit zou nuttig zijn indien de gebruiker verschillende workflows tegelijkertijd wil uitvoeren, bijvoorbeeld dat de smartphone van Kim, na hevige beweging gevoeld te hebben, naar haar man belt. Op hetzelfde ogenblik worden geluidswaarden naar de NoiseTube server gestuurd worden (voorbeeldscenario's uit sectie 1.2). Er worden dus dan 2 verschillende workflows gelijktijdig uitgevoerd.

Het probleem van deze extra functionaliteit is niet het uitbreiden van de engine maar het gebruik van dezelfde functionaliteiten door verschillende workflows. De smartphone bezit functionaliteiten die enkel door één bouwblok tegelijkertijd gebruikt kunnen worden (zoals de mediaspeler, belfunctie...). Dit betekent dat we de guards moeten complexer maken, zodat deze ook rekening houden met andere mogelijke workflows die uitgevoerd worden op hetzelfde ogenblik. Het zou mogelijk zijn dat meerdere workflows tegelijkertijd de mediaplayer nodig hebben. Indien twee of meer workflows aan het wachten zijn op dezelfde functionaliteit, zou een prioriteit tussen de verschillende workflows handig zijn en geïmplementeerd worden door de maker van de workflow.

Hiernaast zou een systeem toegevoegd kunnen worden, waar de gebruiker kan definiëren of een workflow, bij het opstarten, enkel één keer moet uitgevoerd worden, of in een oneindige lus, of een aantal keer.

Het zou ook mogelijk kunnen gemaakt worden om een workflow uit te voeren bij elk plaatsvinden van een specifieke samengestelde gebeurtenis. Het is denkbaar dat tijdens het uitvoeren van de workflow deze zelfde samengestelde gebeurtenis getriggerd wordt en dus een tweede instantie van de workflow gestart wordt. Op dit ogenblik worden er tegelijkertijd meerdere instanties (multiple instances) van dezelfde workflow uitgevoerd.

Op deze manier, zouden we de event based programmeer methodologie [35] introduceren in ons systeem waar, bij elk voorkomen van een samengestelde gebeurtenis, een welbepaalde workflow opgestart wordt.

Referenties

- [1] Touch screens now offer compelling uses. *IEEE Software*, 8:93–94, 107, 1991.
- [2] *A Pattern Language for Workflow Systems*, Aug 1997.
- [3] *A comparison of object oriented scripting languages: Python and Ruby*, December 2001.
- [4] *Using Legos and RoboLab (LabVIEW) with elementary school chi*, 2001. Frontiers in Education Conference, 2001. 31st Annual.
- [5] *A Multi-Touch and Tangible User Interface Rapid Prototyping Toolkit for Tabletop Interaction*, 2010. Sensyble Workshop 2010 manuscript.
- [6] M. Addison. Lego mindstorms nxt - kids can build robots too. Article Source: <http://ezinearticles.com/?LEGO-Mindstorms-NXT---Kids-Can-Build-Robots-Too&id=1523193>.
- [7] J. J. Alferes, F. Banti, and A. Brogi. An event-condition-action logic programming language. In *in the proceedings of the 10 th European Conference on Logics in Artificial Intelligence (JELIA Ó06*, pages 29–42. Springer, 2006.
- [8] A. Alves, A. Arkin, and C. Arkin. *Web Services Business Process*. 2007.
- [9] J. Bailey, A. Poulouvassilis, and P. T. Wood. An event-condition-action language for xml. In *Proceedings of the 11th international conference on World Wide Web, WWW '02*, pages 486–495, New York, NY, USA, 2002. ACM.
- [10] B. Ballard. *Designing the mobile user experience*. Little Springs Design, Inc., 2007.
- [11] L. Bass, P. Clements, and R. Kazman. *Software architecture in practice*. Addison-Wesley, Boston ; Munich [u.a.], 2005.
- [12] L. Bettini, S. Capecchi, and B. Venneri. Double dispatch in c++. *Softw. Pract. Exper.*, 36:581–613, May 2006.
- [13] B. Boehm. A spiral model of software development and enhancement. *SIGSOFT Softw. Eng. Notes*, 11:14–24, August 1986.
- [14] c4 engine. Script editor. website = http://www.terathon.com/wiki/index.php?title=Graphical_Scripting_Language.

- [15] C. Courbis and A. Finkelstein. Towards an Aspect Weaving BPEL engine. In Y. Coady and D. H. Lorenz, editors, *the Third AOSD Workshop on Aspects, Components, and Patterns for Infrastructure Software*, Lancaster, United Kingdom, Mar. 2004.
- [16] J. S. David Braun. State diagrams, 2000. http://atlas.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/state.htm.
- [17] D. Deugo, F. Oppacher, J. Kuester, and I. von Otte. Patterns as a means for intelligent software engineering. In *IC-AI'99*, pages 605–611, 1999.
- [18] P. F. Dietz. Maintaining order in a linked list. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, STOC '82, pages 122–127, New York, NY, USA, 1982. ACM.
- [19] L. Ehrler, M. Fleurke, M. Purvis, and B. Savarimuthu. Agent-based workflow management systems (WfMSs). *Information Systems and E-Business Management*, 4(1):5–23, Jan. 2006.
- [20] B. Ellis, J. Stylos, and B. Myers. The factory pattern in api design: A usability evaluation. In *Proceedings of the 29th international conference on Software Engineering*, ICSE '07, pages 302–312, Washington, DC, USA, 2007. IEEE Computer Society.
- [21] B. Ellis, J. Stylos, and B. Myers. The factory pattern in api design: A usability evaluation. In *Proceedings of the 29th international conference on Software Engineering*, ICSE '07, pages 302–312, Washington, DC, USA, 2007. IEEE Computer Society.
- [22] I. Epic Games. Unreal kismet, the visual scripting system, 2008 until 2011. website = <http://www.unrealengine.com/features/kismet/>.
- [23] J. Fuller. Mindstorm vs robolab. website = <http://www.teachers.ash.org.au/dbrown/robolab/rcx.htm>.
- [24] J. Gong and P. Tarasewich. Guidelines for handheld mobile device interface design. In *In Proceedings of the 2004 DSI Annual Meeting*, 2004.
- [25] Google. App inventor for android, 2011. website : <http://appinventor.googlelabs.com/about/>.
- [26] Google. App inventor for android statement, 2011. website : <http://appinventor.googlelabs.com/about/moreinfo/>.
- [27] B. image source: <http://en.wikipedia.org/wiki/File:BPMN-AProcesswithNormalFlow.jpg>.

- [28] Y. Inc. Yahoo pipes. website = <http://pipes.yahoo.com/pipes/>.
- [29] W. P. Initiative. Workflow data patterns, 2010. website : <http://www.workflowpatterns.com/patterns/data/>.
- [30] R. Jeffries, J. R. Miller, C. Wharton, and K. Uyeda. User interface evaluation in the real world: a comparison of four techniques. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, CHI '91, pages 119–124, New York, NY, USA, 1991. ACM.
- [31] P. Johnstone. Conditions related to de morgan's law. In M. Fourman, C. Mulvey, and D. Scott, editors, *Applications of Sheaves*, volume 753 of *Lecture Notes in Mathematics*, pages 479–491. Springer Berlin / Heidelberg, 1979. 10.1007/BFb0061829.
- [32] R. M. Karp, S. Shenker, and C. H. Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. *ACM Trans. Database Syst.*, 28:51–55, March 2003.
- [33] S. H. Khandkar and F. Maurer. A language to define multi-touch interactions. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, pages 269–270, New York, NY, USA, 2010. ACM.
- [34] A. J. Ko, B. A. Myers, and H. H. Aung. Six learning barriers in end-user programming systems. In *IEEE SYMP. ON VLHCC*, pages 199–206, 2004.
- [35] D. C. Luckham and J. Vera. An event-based architecture definition language. *IEEE Trans. Softw. Eng.*, 21:717–734, September 1995.
- [36] K. S. Martin Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Booch Jacobson Aumbaugh, 2000.
- [37] M. R. Minsky. Manipulating simulated objects with real-world gestures using a force and position sensitive screen. *SIGGRAPH Comput. Graph.*, 18:195–203, January 1984.
- [38] P. Norvig. Teach yourself programming in ten years, 2001. Website : <http://norvig.com/21-days.html>.
- [39] G. Papamarkos, A. Poulouvasilis, R. Poulouvasilis, and P. T. Wood. Rdftl: An event-condition-action language for rdf. In *In Proc. 3rd Int. Workshop on Web Dynamics (in conjunction with WWW2004)*, pages 223–248, 2004.
- [40] I. Poupyrev and S. Maruyama. Tactile interfaces for small touch screens. In *Proceedings of the 16th annual ACM symposium on User interface*

software and technology, UIST '03, pages 217–220, New York, NY, USA, 2003. ACM.

- [41] I. Poupyrev, S. Maruyama, and J. Rekimoto. Ambient touch: designing tactile interfaces for handheld devices. In *Proceedings of the 15th annual ACM symposium on User interface software and technology*, UIST '02, pages 51–60, New York, NY, USA, 2002. ACM.
- [42] M. Priestley and M. G. Hill. *Practical Object-Oriented Design With UML, Second Edition*. 2003.
- [43] D. Riehle. Composite design patterns. In *Proceedings of the 12th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, OOPSLA '97, pages 218–228, New York, NY, USA, 1997. ACM.
- [44] R. Roque. Open blocks. website : <http://education.mit.edu/openblocks>.
- [45] D. Rubine. Specifying gestures by example. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '91, pages 329–337, New York, NY, USA, 1991. ACM.
- [46] N. Russell, A. H. M. T. Hofstede, and D. Edmond. Workflow data patterns. Technical report, 2004.
- [47] N. Russell, A. H. M. T. Hofstede, and N. Mulyar. Workflow controlflow patterns: A revised view. Technical report, 2006.
- [48] B. Shneiderman, C. Plaisand, M. Cohen, and S. Jacobs. *Designing the user interface, 5 edition*. Pearson, 2009.
- [49] W. M. P. Van Der Aalst, A. H. M. Ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distrib. Parallel Databases*, 14:5–51, July 2003.
- [50] W3schools. Javascript tutorial. website : <http://www.w3schools.com/js/default.asp>.
- [51] W3schools. xml tutorial. website : <http://www.w3schools.com/xml/default.asp>.
- [52] A. website : <http://www.apple.com/macosx/what-is-macosx/apps-and-utilities.html#scripteditor>.
- [53] A. website. <http://www.apple.com/downloads/macosx/automator/>.
- [54] L. M. website. <http://www.lego.com/nl-nl/ms/default.aspx?site=mindStorms2>.

- [55] WfMC. Wfmc: Terminology and glossary. Online PDF, Feb. 1999.