Onderzoek en ontwikkeling van Android applicatie “Waar is mijn eigenaar?”

Scriptie – Bijlagen

Afdeling	Academie voor ICT en Media
Studie	Technische Informatica, Delft
Schooljaar	2010 – 2011
Afstudeerblok	2011-1.1
Startdatum	7 februari 2011
Einddatum	6 juni 2011
Begeleider/examinator	J.J. Visser
Expert/examinator	D.R. Stikkolorum
Versie	1.0
Datum	6 juni 2011
Student	M. van Kampen, 07004214

Inhoud

D – Documenten
Bijlage D1: Plan van Aanpak
Bijlage D2: Inception Fase Rapport
Bijlage D3: Elaboration Fase Rapport
Bijlage D4: Construction Fase Rapport
Bijlage D5: Trasition Fase Rapport

F – Figuren
Bijlage F1: Klassendiagram basisontwerp
Bijlage F2: Klassendiagram eindversie
Bijlage F3: Urenverantwoording

E – Extra Informatie
Bijlage E1: Interview Hans Abbink

Plan van aanpak

"Onderzoek en ontwikkeling van Android applicatie Waar is mijn eigenaar?"

Auteur: Marleine van Kampen
Datum: 10 februari 2011
Versie: 0.3

Versiebeheer

Versie	Datum	Auteur	Aanpassingen
0.1	7 februari 2011	M. van Kampen	Basis opzet
0.2	9 februari 2011	M. van Kampen	Verbeteringen
0.3	10 februari 2011	M. van Kampen	Verbeteringen na feedback

Reviewers

Versie	Datum	Auteur
0.3	10 februari 2011	J.J. Visser
0.3	10 februari 2011	D.R. Stikkolorum
0.2	9 februari 2011	S. Mulder
0.2	9 februari 2011	T. van As

Gerelateerde documenten

Naam	Versie	Datum	Auteur
Afstudeerscriptie	1.0	6 juni 2011	M. van Kampen

Distributielijst

Naam	Organisatie	Functie	Reden
S. Mulder	Sense	Technisch begeleider	Proces begeleider
T. van As	ASK	Technisch begeleider	Proces begeleider
J.J. Visser	Haagse Hogeschool	Docent	1e Examinator
D.R. Stikkolorum	Haagse Hogeschool	Docent	2e Examinator

Samenvatting

Marleine van Kampen, student van De Haagse Hogeschool Delft voert gedurende zeventien weken een afstudeeropdracht uit bij ASK Community Systems en Sense Observation Systems.

Sense en ASK zijn dochterbedrijven van Almende. Sense is gespecialiseerd in software die herkent wat gebruikers aan het doen zijn. Ze hebben hiervoor het Sense platform opgezet. ASK daarentegen houdt zich bezig met communicatiesystemen. Door ASK is het ASK Platform opgezet voor communicatie en planning tussen groepen.

De opdracht die uitgevoerd wordt door de student heeft betrekking op de twee platformen. Sense heeft namelijk in het Sense platform een applicatie zitten die draait op Android smartphones. De informatie die wordt ingewonnen via de sensoren van de smartphone worden doorgezonden naar een server. De ingewonnen informatie heeft betrekking op het gedrag van de eigenaar van de telefoon. Op het moment dat de eigenaar niet in de buurt is van de telefoon, klopt de informatie die doorgezonden wordt naar de server ook niet.

De opdracht die de student uitvoert is het onderzoeken en ontwikkelen van een Android applicatie die controleert of de eigenaar van de telefoon en de telefoon samen zijn, zo niet worden er door de applicatie acties ondernomen.

Om een goede planning op te stellen voor het project is een ontwikkelmethode gekozen. Hierbij is gekozen voor de iteratieve methode RUP. Van deze methode is ook een open source versie beschikbaar, OpenUP, deze methode is meer gericht op kleine projecten.

De kosten en baten van het project zijn niet uit te drukken in geld maar in tijd in verhouding met voordelen voor het bedrijf en de student. Zo stoppen ASK en Sense energie in de begeleiding van de student, maar levert het een extra uitbreiding op de platformen op.

Inhoudsopgave

I. INLEIDING....6

I.I Aanleiding....6
I.II Doel....6
I.III Scope....6

1. ACHTERGROND....7

1.1 Almende....7
1.2 Sense....7
1.3 ASK....7
1.4 Deal....8
1.5 Samenwerking....8

2. OPDRACHTOMSCHRIJVING....9

2.1 Opdrachtgever en opdrachtnemer....9
2.2 Probleemstelling....9
2.3 Doelstelling opdracht....10
2.4 Opdrachtbeschrijving....10
2.5 Op te leveren producten....10
2.6 Projectgrenzen....11

3. AANPAK....12

3.1 Methode en technieken....12
3.2 Werkzaamheden....14
3.3 Standaarden....15
3.4 Planning....16

4. PROJECTINRICHTING....18

4.1 Projectorganisatie....18
4.2 Informatievoorziening....18
4.3 Faciliteiten....18
4.4 Kwaliteitsborging....18

5. RISICOFACTOREN....19

6. KOSTEN EN BATEN....20

6.1 Student....20
6.2 Bedrijf....20

BRONVERMELDING....21

Ontwikkelmethodes....21
Risicoanalyse....21
Rapportage....21

Tabellen

TABEL 1 VERGELIJKING METHODES.....12

TABEL 2 PLANNINGSTABEL.....16

TABEL 3 RISICOANALYSE.....19

Figuren

FIGUUR 1 ORGANISATIE.....7

FIGUUR 2 SENSE APPLICATIE.....9

FIGUUR 3 RUP/OPENUP FASES.....14

FIGUUR 4 PLANNING.....17

I. Inleiding

I.I Aanleiding

Door Marleine van Kampen, afstudeerder van De Haagse Hogeschool Delft wordt gedurende zeventien weken gewerkt aan het project “Waar is mijn eigenaar?”. Het project is een onderzoek naar en ontwikkeling van een Android applicatie bovenop de bestaande platformen van de dochterbedrijven, Sense Observation Systems en ASK Community Systems, van het Rotterdamse onderzoeksinstituut Almende.

I.II Doel

Het doel van het plan van aanpak is het organiseren en definiëren van de werkzaamheden tijdens het afstuderen. Hierbij wordt de opdracht volledig beschreven met de daarbij komende kosten, baten en risico's.

Om het project binnen zeventien weken af te ronden is een onderzoeksmethode vastgesteld en aan de hand van de fases hiervan is een planning opgezet.

I.III Scope

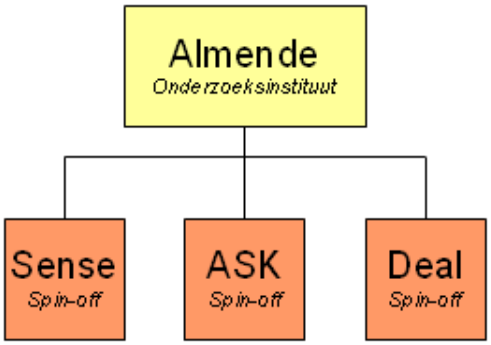
Het plan van aanpak bevat de volgende onderdelen:

1. Achtergrond: Beschrijft de achtergrond van het bedrijf en de aanleiding van de opdracht
2. Opdrachtbeschrijving: Beschrijft de opdracht van probleem tot resultaat
3. Aanpak: Beschrijft de methode die gehandhaafd wordt tijdens het project, compleet met planning
4. Projectinrichting: Beschrijft de verschillende voorzieningen van het project
5. Risicofactoren: Beschrijft de verschillende risico's waar de student mee te maken krijgt tijdens het project
6. Kosten en baten: Beschrijft de kosten en baten van het project voor de student en voor het bedrijf.

1. Achtergrond

1.1 Almende

Almende, een Rotterdams (non-profit) onderzoeksinstituut, houdt zich bezig met onderzoek naar zelf-organiserende systemen. Vanuit Almende zijn verschillende dochterbedrijven opgezet die zich specialiseren in de verschillende vormen van zelf-organiserende systemen. Drie van deze bedrijven zijn ASK, Sense en Deal. In de afbeelding rechts is de opzet te zien.



Figuur 1 Organisatie

1.2 Sense

De afstudeeropdracht wordt uitgevoerd bij Sense Observation Systems (Later benoemd als Sense). Sense is een (for profit) spinoff van Almende dat is gestart in 2009.

Sense maakt software die herkent wat de gebruikers aan het doen zijn. Hierbij wordt gebruik gemaakt van data uit de sensoren van smartphones en van draadloze sensornetwerken (WSNs), en machine learning algoritmes die deze data analyseren.

Het Sense platform kan worden gebruikt voor bijvoorbeeld:

- Geautomatiseerde planning en tijdregistratie
- Locatie-gebaseerde diensten om informatie te krijgen waar en wanneer het nodig is
- Monitoring voor ongebruikelijke patronen en het nemen van passende maatregelen, bijvoorbeeld in de gezondheidszorg en onderhoud van het gebouw, of bij calamiteiten

1.3 ASK

Voor de opdracht is er samenwerking met het bedrijf ASK Community Systems (verder benoemd als ASK). ASK is in 2003 gestart als een (for profit) spinoff van Almende.

ASK is gespecialiseerd in communicatie systemen. Zo heeft ASK het ASK platform opgezet. ASK is een intelligent communicatieplatform dat alle communicatiemiddelen (waaronder sms, internet en telecommunicatie) combineert om zo communicatie effectiever te maken bij bedrijven en groepen.

Het ASK platform werkt met het idee dat elke persoon een virtuele persoonlijkheid heeft. Deze virtuele persoonlijkheid wordt ook wel een 'agent' genoemd. De agent kan bijvoorbeeld aan de hand van de status en de agenda van de gebruiker bepalen of hij een afspraak in zijn agenda kan plaatsen.

De agent is lid van verschillende groepen, de groepen kunnen bedrijfsafdelingen zijn maar daarnaast ook bijvoorbeeld de hockeyvereniging of de voetbalclub.

Zo kan het systeem een bericht (bijv. sms of e-mail) sturen van een gebruiker naar alle agents binnen een groep of binnen meerdere groepen.

1.4 Deal

Deal services is net als ASK en Sense een spin-off van Almende. DEAL Services levert technologie die het mogelijk maakt om contactmomenten af te stemmen met een klant. Of het nu gaat om thuisbezorging van een online gekocht artikel, onderhoud aan een CV-ketel, of (medische) thuiszorg

1.5 Samenwerking

Waar ASK gespecialiseerd is in de communicatie tussen verschillende gebruikers en naar een specifieke gebruiker, is Sense gespecialiseerd in het inwinnen van informatie over de gebruiker.

Deze twee onderdelen moeten worden gekoppeld in de nieuwe applicatie; MyASK. MyASK moet aan de hand van de ingewonnen informatie van het Sense Platform berichten versturen of afspraken inplannen. Zo kan uit de locatiegegevens gehaald worden dat de gebruiker op het moment niet op kantoor is en bijvoorbeeld op een dusdanige afstand van het kantoor is dat deze de komende twee uur niet aanwezig zal zijn. Hieruit kan gehaald worden dat ASK geen afspraak op kantoor kan inplannen binnen die twee uur voor die agent/persoon.

Sense en ASK zijn op het moment bezig om de samenwerking tussen de twee platformen op te zetten. De opdracht die tijdens het afstuderen wordt gedaan wordt als onderdeel van deze nieuwe samenwerking gemaakt.

Het uiteindelijke idee van MyASK is het betrekken van Deal bij de applicatie. Hierdoor kan er aan de hand van de gegevens van Sense en de planning van ASK een duidelijke route gemaakt worden voor bijvoorbeeld pakketbezorging. Als een pakket in de buurt is van de koper wordt deze ingelicht dat deze over een kwartier wordt bezorgd. Aan de koper wordt gevraagd of dat uitkomt, komt dit niet uit rijdt de pakketbezorger een andere route en komt pas later aan bij de koper. Op deze manier wordt er met verschillende platformen een zo efficiënt mogelijke route uitgewerkt.

2. Opdrachtomschrijving

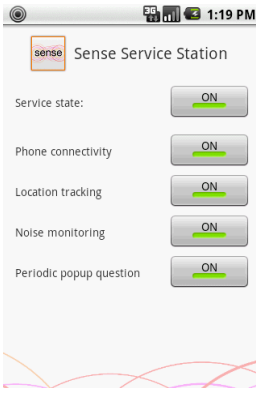
2.1 Opdrachtgever en opdrachtnemer

Sense en ASK zijn de opdrachtgevende organisaties voor de afstudeeropdracht van Marleine van Kampen. Binnen ASK is Thijs van As contactpersoon voor de afstudeerder. Binnen Sense is Steven Mulder de contactpersoon.

2.2 Probleemstelling

Door Sense is een applicatie voor Android smartphones gemaakt die gebruikt wordt om te monitoren wat de gebruiker aan het doen is. Het monitoren gebeurt in de huidige applicatie aan de hand van de volgende gegevens:

- Status van de telefoon (bijv. Stand-by)
- Connectie met netwerk
- Locatie (GPS)
- Wi-fi spots in de buurt van de telefoon
- Externe Bluetooth punten
- Versnelling
- Ambience (Geluid / Licht)
- Online Calendar



Figuur 2 Sense Applicatie

De verkregen data wordt verstuurd naar CommonSense, de server die alle data verwerkt en hier per gebruiker of per groep een resultaat uit geeft. De combinatie van de Android applicatie en de server wordt ook wel het Sense Platform genoemd. Daarnaast wordt er ook nog informatie ingewonnen van Wireless Sensor Nodes, maar deze informatie is niet van toepassing op de opdracht.

Daarnaast is er door ASK het ASK Platform ontwikkeld. Hiermee wordt er vanuit gegaan dat elke persoon ook een soort virtuele persoonlijkheid heeft, in ASK termen een agent. Deze agent kan binnen het ASK platform voor de gebruiker verschillende beslissingen nemen, bijvoorbeeld afspraken maken in de agenda van de gebruiker.

De samenwerking die op het moment plaatsvindt tussen ASK en Sense is het ontwikkelen van een nieuwe applicatie MyASK. Deze applicatie combineert de gegevens die het Sense platform inwint met de beslissingen en communicatie van het ASK platform.

In de gehele samenwerking ontstaat een probleem op het moment dat de gebruiker niet in de buurt is van zijn telefoon. De informatie die door de telefoon wordt verzonden naar de server over de status van de gebruiker is dan niet juist meer.

2.3 Doelstelling opdracht

Het onderzoeken en ontwikkelen van een applicatie die periodiek controleert of de eigenaar in de buurt is van de telefoon. Mocht dit niet het geval zijn, moet de applicatie acties ondernemen om de eigenaar en de telefoon weer te herenigen.

Hierbij worden de volgende doelen gesteld:

- Het definiëren van de tijd dat de applicatie controleert of de eigenaar in de buurt is (keuze maken aan de hand van voor- en nadelen van lang / kort tijdsbestek)
    - Bijvoorbeeld: Simpele vragen laten beantwoorden door de gebruiker of controleren hoelang de smartphone al in Stand-by stand is.
- Het definiëren hoe de eigenaar wordt ingelicht
    - Bijvoorbeeld: Eenmaal bellen als controle, vrienden/collega's van de eigenaar bellen of een bericht verzenden naar e-mail of andere berichtservice (hierbij kan gedacht worden aan het gebruik van het ASK Platform).
    - Werken met een zelflerend algoritme. De applicatie moet bijvoorbeeld leren welke mogelijkheid hij als eerste pakt om de eigenaar te vinden.
- Het ontwikkelen van een applicatie die bovenstaande taken uitvoert.
    - Hierbij kijken naar de smartphone applicatie maar ook naar de gegevens die worden verzonden naar de server en hoe deze worden verwerkt op de server op het moment dat hier nieuwe gegevens bijkomen.

2.4 Opdrachtbeschrijving

Bij het uitvoeren van de opdracht wordt geprobeerd bovenstaande doelen te behalen. Bij het halen van deze doelen wordt de volgende resultaten verwacht:

- Onderzoeksrapport naar aanleiding van de verkregen informatie en mogelijkheden
- Proof-of-concept systeem (Implementatie van de oplossing die gevonden is in het onderzoek) voor een Android smartphone die kan uitvinden of de eigenaar van de telefoon nog in de buurt is, en indien nodig acties onderneemt om de eigenaar te laten weten dat hij/zij de smartphone niet bij zich heeft.
- Uitgevoerde tests met de applicatie om te valideren of de applicaties volgens de wensen werkt.

2.5 Op te leveren producten

Aan het einde van het project worden de volgende producten opgeleverd:

- Plan van Aanpak
- Inception Fase Rapport
- Elaboration Fase Rapport
- Contruction Fase Rapport
- Implementatie (Code / Uitvoerbare applicatie)
- Transition Fase Rapport
- Adviesrapport
- Scriptie

2.6 Projectgrenzen

Door ASK, Sense en Deal wordt op het moment een grote samenwerking opgezet voor de MyASK applicatie. De student is enkel bezig met het maken van de uitbreiding “waar is mijn eigenaar?” en de daarbij behorende aanpassingen in het systeem.

De documenten worden alle in het Nederlands geschreven, enkel de documenten die van toepassing zijn op het bedrijf worden ook in het Engels geschreven. Het gaat hierbij om het Elaboration Fase Rapport en het Construction Fase Rapport. De scriptie wordt in het Nederlands opgeleverd.

3. Aanpak

3.1 Methode en technieken

Voor het kiezen van een juiste faseringsmethode is een vergelijkingstabel gemaakt:

Als basis is gekozen voor de methodes RUP (Rational Unified Process), Scrum, FDD (Feature Driven Development), AUP (Agile Unified Process)

Eigenschappen	Eisen	RUP	Scrum	FDD	AUP
Features Het opdelen van het systeem in verschillende blokken	~	-	-	+	-
Iteratief De methode is iteratief opgesteld. Oftewel opgedeeld in kleinere trajecten waarin alle of een deel van alle fases in plaats vinden	+	+	+	+	+
Duidelijke fases De werkzaamheden worden uitgevoerd in duidelijk aangegeven fases	+	+	+	~	+
Vroeg testen Bij de methode wordt rekening gehouden met het testen tijdens de gehele ontwikkeling (niet enkel aan het einde)	+	+	~	+	+
Gedetailleerde planning In de eerste week van de opdracht kan een duidelijke planning gemaakt worden	+	+	+	+	+
Duidelijke informatie over methode Is er voldoende informatie beschikbaar over de methode om deze goed te volgen	+	+	+	+	+
Bewaken vooruitgang Zijn er voldoende feedback punten in de planning waardoor op tijd fouten of problemen worden herkend	+	~	+	~	~
Individueel nuttig Is de methode individueel goed uit te voeren	+	~	–	+	+
Modellering Voldoende tijd om ontwerpen en modellen uit te werken	+	+	+	~	+

Tabel 1 Vergelijking methodes

Het gaat in alle bovenstaande gevallen om een iteratieve methode. Hier is van te voren al op geselecteerd. Het project gaat namelijk over het ontwikkelen van software waarbij in een vroeg stadium al getest wordt en gekeken wordt of het systeem nog voldoet aan de eisen. Om deze manier van ontwikkelen te hanteren is een iteratieve methode nodig. Het systeem wordt in kleine onderdelen verdeeld en al deze onderdelen worden onderzocht, ontworpen, geïmplementeerd en getest. Mocht er in de test iets nog niet kloppen dan wordt dit verbeterd.

Als gekeken wordt naar de verdere eisen die worden gesteld voor het ontwikkelen komen RUP en FDD het meest in de buurt van bruikbare methodes. Scrum is namelijk teveel ingesteld op een projectgroep, het vroegtijdig constateren van problemen met ontwikkelen binnen groepen. De verschillende feedback momenten worden door de student in ieder geval al ingepland waardoor deze niet perse via Scrum gepland hoeven te worden.

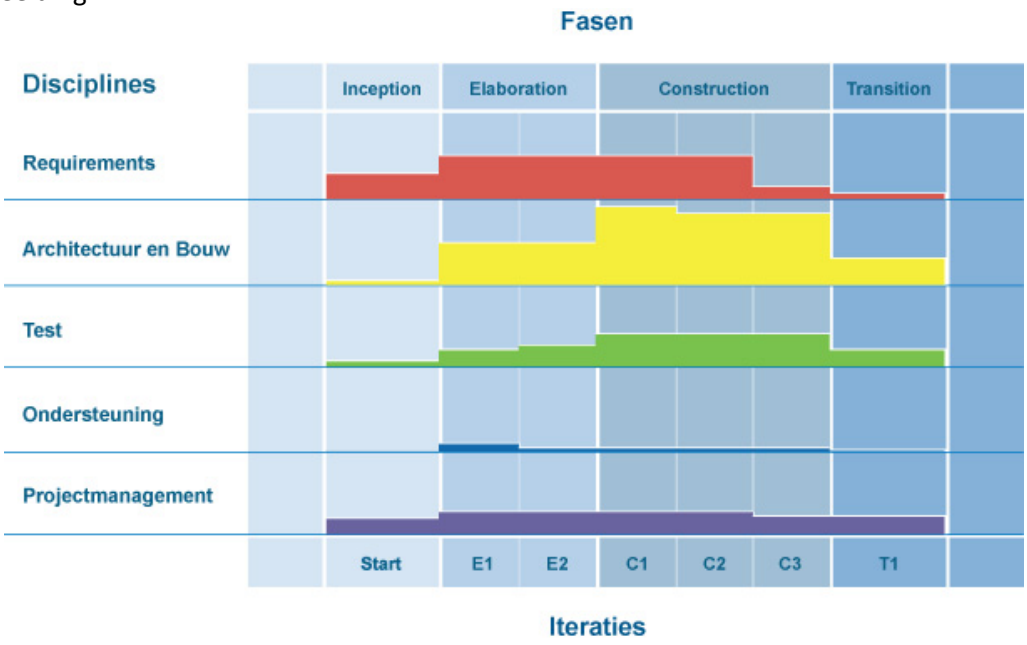
AUP richt zich vooral op het modelleren en ontwikkelen van het systeem maar besteedt weinig tijd aan de analysefase. Deze fase is in het afstudeertraject juist erg belangrijk omdat het om een bestaand systeem gaat waar de gebruiker nog onvoldoende kennis van heeft.

Feature Driven Development levert al meer voordelen voor het ontwikkeltraject. Enkel wordt er minder duidelijk gewerkt met iteraties maar meer met features. De features zijn in verhouding kleinere trajecten dan iteraties.

Voor het project is RUP het best van toepassing. Het project wordt opgedeeld in duidelijke fases, met duidelijke mijlpalen. Na meer onderzoek naar RUP is ook de methode OpenUP gevonden. Deze methode is gebaseerd op RUP maar is de Open Source versie hiervan. Hierbij is meer gericht op kleinere projecten en dus minder rollen binnen een project.

3.2 Werkzaamheden

De verschillende fases, disciplines en mijlpalen van RUP/OpenUp zijn te zien in onderstaande afbeelding



Figuur 3 RUP/OpenUP Fases

In de eerste fase, de inception phase, draait het vooral om het voorbereiden van het project, kennis op doen van de Sense en ASK platformen en het ontwikkelen voor Android. Daarnaast wordt er gewerkt aan het pakket van eisen.

De tweede fase, de elaboration phase, gaat verder op het vergaren van kennis over van de platformen en Android. Daarnaast wordt er aan de hand van de eisen een ontwerp voor het systeem opgezet. Als alle ontwerpen op papier staan wordt er gekeken hoe het systeem uiteindelijk getest gaat worden en kan er een basis opgezet worden voor de applicatie.

Tijdens de derde fase, de construction phase, wordt de applicatie verder ontwikkeld. Tijdens deze fase gaat het voornamelijk om het programmeren en het gelijktijdig testen. Op het moment dat er een dergelijk systeem is opgezet wordt er ook getest met andere gebruikers (medewerkers van ASK en Sense).

Tijdens de Transition Phase, de laatste fase van het project, ligt het accent vooral op het testen van de applicatie. De applicatie functioneert volledig en naast het testen wordt er ook gekeken naar de mogelijkheden voor uitbreidingen van de applicatie in de toekomst. Op deze manier is het voor ASK en Sense mogelijk om na de afstudeerperiode verder te ontwikkelen en te verbeteren aan de applicatie.

3.3 Standaarden

Voor het project zijn de volgende standaarden vastgesteld

Documentatie

- De documenten worden in .doc en PDF formaat opgeleverd
- Documenten worden in het Nederlands geschreven, enkel het Elaboration- en het Construction Fase Rapport worden ook in het Engels geschreven
- Documenten die van toepassing zijn voor het bedrijf worden ook in de Google Docs Repository geplaatst
- Voor documenten wordt de OpenUp documentatieformaat gebruikt waar mogelijk of toepasbaar.

Ontwerpen

- Modelleren gebeurt aan de hand van UML

Coderen

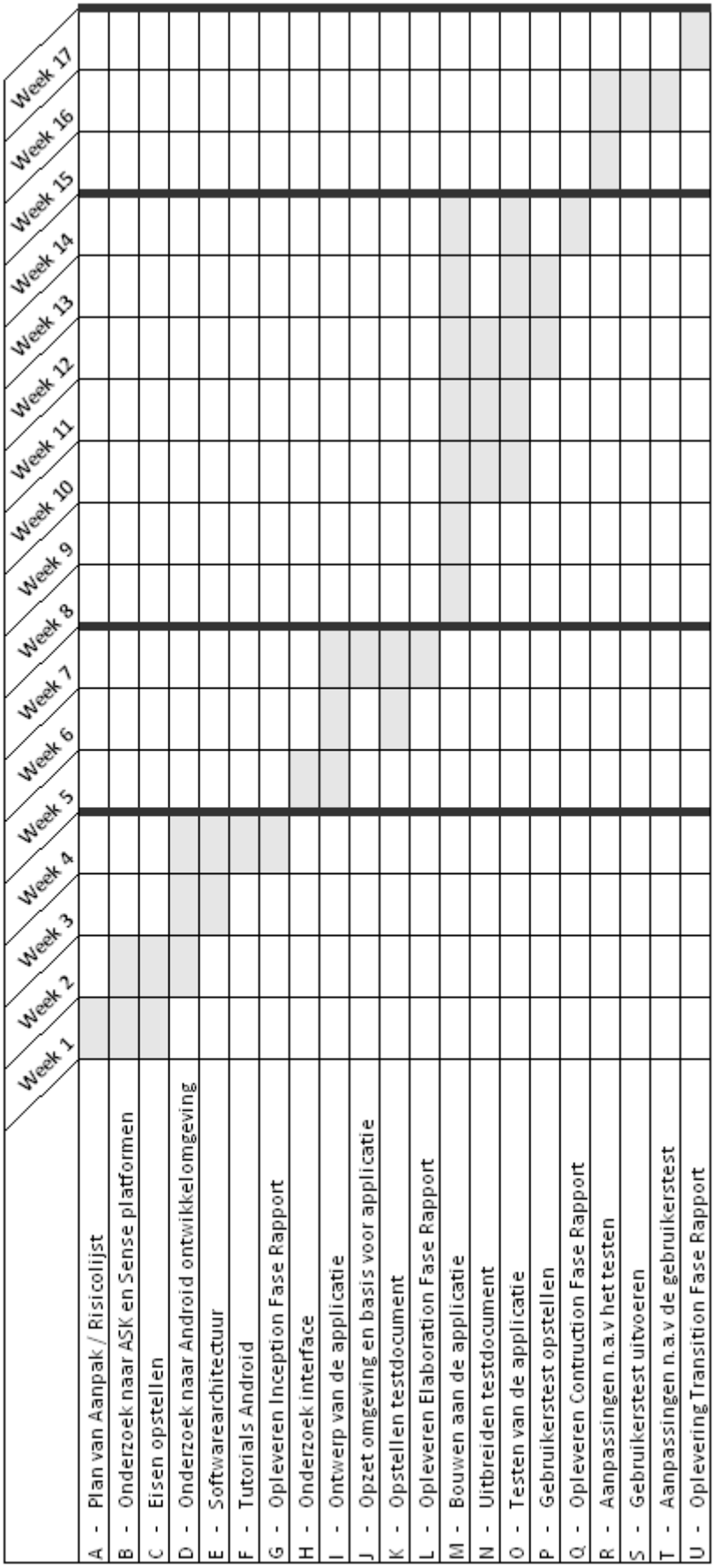
- Het coderen gebeurt in een Eclipse omgeving
- Het coderen gebeurt in Android-Java

3.4 Planning

Aan de hand van de ontwikkelmethode OpenUP is een planning opgesteld voor het afstudeertraject van zeventien weken.

Iteratie	Activiteiten	Geplande data en mijlpalen
Inception Fase		
A	Maken van het plan van aanpak en risicolijst	Week 1
B	Onderzoek naar ASK en Sense platformen	Week 1 / 2
C	Eisen opstellen aan de hand van gesprekken met medewerkers ASK en Sense	Week 1 / 2
D	Onderzoek naar Android ontwikkelomgeving	Week 2 / 3 / 4
E	Onderzoek naar softwarearchitectuur van Android, mogelijke toepassing van design patterns	Week 3 / 4
F	Ontwikkelen van proefapplicaties om Android-Java te leren kennen	Week 4
G	Opleveren Inception Fase Rapport	Einde week 4
Elaboration Fase		
H	Onderzoek en ontwerp van de interface voor de applicatie.	Week 5
I	Ontwerp van de applicatie	Week 5 / 6 / 7
J	Opzet omgeving en basis voor applicatie	Week 7
K	Opstellen testdocument	Week 6 / 7
L	Opleveren Elaboration Fase Rapport	Einde week 7
Contruction Fase		
M	Bouwen aan de applicatie	Week 8 / 9 / 10 / 11 / 12 / 13 / 14
N	Uitbreiden testdocument	Week 10 / 11 / 12
O	Testen van de applicatie	Week 10 / 11/ 12 / 13 / 14
P	Gebruikerstest opstellen	Week 12 / 13
Q	Opleveren Contruction Fase Rapport	Einde week 14
Transition Fase		
R	Aanpassingen n.a.v het testen	Week 15 / 16
S	Gebruikerstest uitvoeren	Week 16
T	Aanpassingen naar aanleiding van de gebruikerstest	Week 16
U	Oplevering Transition Fase Rapport	Einde week 17

Tabel 2 Planningstabel



Figuur 4 Planning

4. Projectinrichting

4.1 Projectorganisatie

Tijdens het afstudeertraject zijn er vier personen ter beschikking gesteld voor het begeleiden van de student. Deze begeleiders zijn onder te verdelen in twee verschillende groepen: de schoolbegeleiders en de opdrachtgevers/begeleiders.

De groep opdrachtbegeleiders bestaat uit twee personen. Ten eerste Steven Mulder, werknemer bij Sense en is tevens de opdrachtgever. Steven Mulder begeleidt de student met de opdracht door aan te geven wat de wensen zijn van de opdracht en tussentijds feedback te geven op de resultaten. Daarnaast is er contact en feedback vanuit Thijs van As, werknemer bij ASK.

Vanuit school zijn twee begeleiders toegewezen aan de student. John Visser is de procesbegeleider. Hij is het aanspreekpunt voor de student en komt op bezoek bij het bedrijf. Als tweede examinator is Dave Stikkolorum toegewezen. Deze docent zal op de achtergrond toezicht houden op het afstudeerproces en is bij de verdediging aanwezig zijn als examinator.

De student, Marleine van Kampen, is de uitvoerder van het project.

4.2 Informatievoorziening

Met de begeleiding vanuit het bedrijf wordt gecommuniceerd via telefoon, e-mail en mondeling. De documenten worden opgeleverd in Google Docs en de begeleiders worden op de hoogte gesteld bij oplevering

Communicatie met de begeleiding vanuit school gebeurt voornamelijk via e-mail. In week vier van het afstudeertraject komt de eerste examinator, John Visser, langs op het bedrijf.

4.3 Faciliteiten

Vanuit het bedrijf wordt een laptop met daarop de benodigde software geleverd. Hierbij te denken aan Eclipse en tekstverwerkingssoftware.

4.4 Kwaliteitsborging

Om de kwaliteit van het product hoog te houden wordt er veel tijd besteed aan het testen van de software. De stappen die hiervoor genomen worden zijn:

- Vroegtijdig opstellen van het testplan
- Vroegtijdige terugkoppeling van eisen en wensen
- Testen uitgevoerd door student, projectbegeleider, testgebruikers en personen die buiten het project staan
- Uitslagen van tests verwerken in het product

Om de documentatie van voldoende kwaliteit te laten zijn wordt er op meerdere momenten feedback gevraagd van de begeleiding.

5. Risicofactoren

Aan het project zitten verschillende risico´s verbonden. Aan de hand van de OpenUP risk list is de risicoanalyse opgezet. De risico´s hebben een bepaalde omvang. Deze wordt bepaald aan de hand van de impact en de kans van voordoen. Het risico met het ID=1 is het risico met de grootste omvang, het risico waar dus het meest aan gewerkt moet worden.

ID	Omschrijving	Type	Impact (1 – 10)	Kans van voordoen(%)	Omvang (x / 10)	Oplossing
1	De kennis van het project is op het moment van starten nog niet voldoende bij de student.	Technical	7	100%	7,0	Door in de planning voldoende tijd in te plannen voor het onderzoek kan de student zich inlezen in het project en in de benodigde stof hiervoor
2	Tijdens het project kan de tijd die beschikbaar is een risico opleveren.	Resource (Time)	7	50%	3,5	Door een goede planning op te zetten vooraf aan het project en door een constant te controleren of het project nog op planning ligt moet tijd minder een probleem zijn. Rekening moet wel gehouden worden met onverwachte uitval. Door de ruim opgezette planning moet dit deels op te vangen zijn.
3	De opdracht voor het project is geleverd door twee verschillende bedrijven. De communicatie tussen de bedrijven is vers en daardoor kan er op bepaalde punten onduidelijkheid ontstaan	Resource (Organization)	6	50%	3,0	Voldoende overleg met beide bedrijven samen inplannen moet zorgen voor voldoende duidelijkheid van alle partijen. Onduidelijkheid moet op tijd naar voren komen.
4	De student heeft twee werkplekken, waardoor contact met de twee begeleiders per dag kan verschillen.	Resource (Organization)	3	30%	0,9	Goede afspraken inplannen met begeleiders. Daarnaast zorgen dat het mailcontact makkelijk en duidelijk is waardoor werkplekken geen probleem moeten zijn voor het project.

Tabel 3 Risicoanalyse

6. Kosten en baten

6.1 Student

Omdat het voor de student gaat om een schoolproject is er geen sprake van kosten en baten in de vorm van geld. De kosten en baten hieronder beschreven verhouden zich in tijd en studieresultaten.

Voor de student heeft het project de volgende kosten tot gevolg:

- 17 weken, 5 dagen per week werken aan het project. Dit komt neer op 85 dagen van ca. 8 uur, oftewel ca. 680 uur werk.

Voor de student levert het project de volgende baten:

- 30 studiepunten

6.2 Bedrijf

Voor het bedrijf is niet in geld te bepalen wat de kosten en baten zijn. De onderstaande kosten en baten verhouden zich in tijd en voordelen uit de nieuwe applicatie

Voor het bedrijf heeft het project de volgende kosten tot gevolg:

- Begeleiding vanuit Sense en ASK
- Een stagair in dienst voor 17 weken

Voor het bedrijf heeft het project de volgende baten:

- Een extra in de samenwerkingsapplicatie MyASK
- Door de mogelijkheid de eigenaar van de telefoon terug te vinden is de data die wordt verzameld door Sense van betere kwaliteit. De telefoon heeft op tijd door dat de eigenaar niet in de buurt is van de telefoon waardoor deze geen onjuiste informatie gaat doorsturen naar de server.

Bronvermelding

Ontwikkelmethodes

http://www.rupopmaat.nl/
http://www.scrumalliance.org/
http://www.featuredrivendevelopment.com/
http://www.ambysoft.com/unifiedprocess/agileUP.html

Risicoanalyse

risico analyse.pdf – J.J. van der Hoek
http://epf.eclipse.org/wikis/openup/

Rapportage

Rapportagetechniek Digitaal Handboek 4.2

Inception Fase Rapport

"Onderzoek en ontwikkeling van Android applicatie – Waar is mijn eigenaar?"

Auteur: Marleine van Kampen
Datum: 6 juni 2011
Versie: 1.0

Versiebeheer

Versie	Datum	Auteur	Aanpassingen
0.1	9 februari 2011	M. van Kampen	Basis opzet
0.2	16 februari 2011	M. van Kampen	Feedback verwerkt en uitgebreide eisen
0.3	21 februari 2011	M. van Kampen	Laatste feedback en samenvatting
1.0	23 februari 2011	M. van Kampen	Oplevering

Reviewers

Versie	Datum	Auteur
0.1	14 februari 2011	S. Mulder
0.1	14 februari 2011	T. van As
0.3	21 februari 2011	S. Mulder
0.3	21 februari 2011	T. van As

Gerelateerde documenten

Naam	Versie	Datum	Auteur
Afstudeerscriptie	1.0	6 juni 2011	M. van Kampen
Elaboration Fase Rapport	1.0	6 juni 2011	M. van Kampen
Construction Fase Rapport	1.0	6 juni 2011	M. van Kampen
Transistion Fase Rapport	1.0	6 juni 2011	M. van Kampen

Distributielijst

Naam	Organisatie	Functie	Reden
S. Mulder	Sense	Technisch begeleider	Proces begeleider
T. van As	ASK	Technisch begeleider	Proces begeleider
J.J. Visser	Haagse Hogeschool	Docent	1e Examinator
D.R. Stikkolorum	Haagse Hogeschool	Docent	2e Examinator

Samenvatting

Marleine van Kampen, student van De Haagse Hogeschool Delft voert gedurende zeventien weken een afstudeeropdracht uit bij ASK Community Systems en Sense Observation Systems. De opdracht is genaamd “Waar is mijn eigenaar?”, een Android applicatie die opzoek gaat naar de eigenaar van de telefoon.

Om de opdracht goed uit te werken is in de inception fase onderzoek gedaan naar de onderdelen waar de applicatie mee te maken heeft. Daarnaast is het pakket met eisen opgezet met bij behorende Use Case diagrammen.

Om voor de applicatie te weten wat de locatie van de telefoon is, wordt gebruik gemaakt van het Sense Platform. Sense heeft het platform ontwikkeld dat alle sensoren uit een smartphone uitleest en deze op verschillende manieren kan verwerken in CommonSense. Daarnaast kan er ook nog sensordata worden uitgelezen uit een draadloos sensor netwerk. Met CommonSense kan bijvoorbeeld uit de waardes worden gehaald dat iemand als hij gedurende een bepaalde tijd veel geluid om zich heen had en met een snelheid van 20 km/h bewoog, hij waarschijnlijk aan het fietsen was.

ASK heeft het ASK Platform opgezet. Het platform wordt gebruikt voor communicatie tussen groepen. Het platform gaat er vanuit dat elke persoon een virtuele persoonlijkheid heeft. Deze persoon wordt in het ASK platform vertegenwoordigd als een *agent.* ASK bestaat uit een dynamisch netwerk van agents die worden gestuurd om acties uit te voeren. Zo kan er voor een groep mensen een afspraak of planning worden gemaakt zonder dat een persoon hiermee bezig hoeft te zijn. Het systeem gaat contact opnemen met mensen om te kijken of zij beschikbaar zijn op bepaalde momenten.

De applicatie wordt geschreven als een losse applicatie op een Android smartphone. De applicatie moet wel al gegevens ophalen uit de Sense en ASK platformen. De uitbreiding van het systeem wordt dat deze in het MyASK systeem wordt gehangen. MyASK is een nieuwe applicatie die werkt als een combinatie van de Sense en ASK platformen.

De basisfunctionaliteiten van de te maken applicatie zijn het herkennen, vinden en het herenigen van de eigenaar. Oftewel herkennen of de eigenaar in de buurt is van de telefoon, als de eigenaar niet in de buurt van de telefoon is gaat de applicatie de eigenaar proberen te vinden en deze proberen te herenigen met zijn smartphone.

Vanuit de gebruiker is er weinig input nodig voor de applicatie. De applicatie doet het grootste deel van het werk zelf. Om de gebruiker wel te laten weten wat de applicatie aan het doen is, worden meldingen gegeven aan de gebruiker.

Inhoudsopgave

I. INLEIDING 6

1. ASK PLATFORM 7

1.1 INLEIDING 7
1.2 ONDERDELEN VAN HET ASK PLATFORM 7
1.3 ASK WEBINTERFACE 11

2. SENSE PLATFORM 12

2.1 INLEIDING 12
2.2 ONDERDELEN VAN HET SENSE PLATFORM 12

3. MYASK 16

3.1 INLEIDING 16
3.2 DOEL 16
3.3 OPBOUW SAMENWERKING ASK, SENSE EN DEAL 17
3.4 MULTIPLATFORM EN PLATFORM SPECIFIEK 18

4. ANDROID 19

4.1 INLEIDING 19
4.2 SPECIFICATIES ANDROID 19

5. PAKKET VAN EISEN 21

5.1 FUNCTIONELE EISEN 22
5.2 USABILITY (BRUIKBAARHEID) 23
5.3 RELIABILITY (BETROUWBAARHEID) 23
5.4 PERFORMANCE (PRESTATIE) 23
5.5 SUPPORTABILITY (ONDERSTEUNING) 24
5.6 IMPLEMENTATIE EISEN 24
5.7 INTERFACE EISEN 24
5.8 SYSTEEM LICENTIE 25

6. USE CASE DIAGRAMMEN 26

6.1 GEBRUIKER 26
6.2 APPLICATIE 28

VERKLARENDE WOORDENLIJST 33

BRONVERMELDING 34

Tabellen

Tabel 1: Basiseis F1 22
Tabel 2: Basiseis F2 22
Tabel 3: Usability eisen 23
Tabel 4: Reliability eisen 23
Tabel 5: Performance eisen 23
Tabel 6: Supportability eisen 24
Tabel 7: Implementatie eisen 24
Tabel 8: Interface eisen 24
Tabel 9: Systeem licentie 25
Tabel 10: Use Case Scenario - Templates instellen 27
Tabel 11: Use Case Scenario - Aangeven gevonden 27
Tabel 12: Use Case Scenario - Eigenaar aanwezig of niet 29
Tabel 13: Use Case Scenario - Eigenaar terugvinden 30
Tabel 14: Use Case Scenario - Templates bouwen 31
Tabel 15: Meldingen weergeven bij ondernemen actie 32
Tabel 16: Verklarende woordenlijst 33

Figuren

Figuur 1: Overzicht ASK 7
Figuur 2: Voorbeeld ASK Desktop 9
Figuur 3: Webinterface ASK 11
Figuur 4: Sense platform 12
Figuur 5: Screenshot: Keuze welke sensoren uitgelezen mogen worden 13
Figuur 6: Screenshot: Pop-up vraag; Hoe voel je je? 13
Figuur 7: Screenshot: Bij de meldingen wordt aangegeven dat Sense actief is 13
Figuur 8: deel MyriaNed 14
Figuur 9: Webinterface CommonSense 15
Figuur 10: Samenwerking ASK, Sense en DEAL 17
Figuur 11: Android 19
Figuur 12: Use Case - Gebruiker 26
Figuur 13: Use Case - Applicatie totaal 28
Figuur 14: Use Case - Eigenaar herkennen 29
Figuur 15: Use Case – Eigenaar terugvinden 30
Figuur 16: Use Case – Templates bouwen 31
Figuur 17: Use Case – Meldingen weergeven bij ondernemen actie 32

I. Inleiding

I.I Aanleiding

Door Marleine van Kampen, afstudeerder van De Haagse Hogeschool Delft wordt gedurende zeventien weken gewerkt aan het project “Waar is mijn eigenaar?”. Het project is een onderzoek naar en ontwikkeling van een Android applicatie bovenop de bestaande platformen van de dochterbedrijven, Sense en ASK, van het Rotterdamse onderzoeksinstituut Almende.

I.II Doel

Het doel van het inception fase rapport is het documenteren van de bevindingen van de inception fase. In de inception fase zijn de verschillende componenten, die van belang zijn tijdens het ontwikkelen van de Android applicatie, onderzocht.

Aan de hand van de analyse worden de eisen van de opdrachtgevers opgesteld. De eisen worden volgens FURPS+ onderverdeeld en worden daarnaast SMART beschreven.

Aan de hand van de eisen worden de Use Case diagrammen opgezet om een inzicht te krijgen in de rollen van alle componenten in het systeem.

I.III Scope

1. ASK Platform: Beschrijving van het platform dat ontwikkeld is door ASK en waar de Android applicatie uiteindelijk mee moet samenwerken
2. Sense Platform: Beschrijving van het platform dat ontwikkeld is door Sense en waar de Android applicatie uiteindelijk mee moet samenwerken
3. MyASK: Beschrijving van de applicatie die momenteel wordt ontwikkeld door Sense en ASK samen
4. Android: Een basis over Android, wat het is en welke specificaties de omgeving heeft
5. Pakket van eisen: Alle eisen die gesteld zijn door het bedrijf opgedeeld in FURPS+
6. Use Case diagrammen: Aan de hand van de eisen verschillende Use Case diagrammen die de rollen van de componenten in het systeem beschrijven
7. Bronvermelding: De informatie uit de analysefase komt van verschillende bronnen, deze worden genoemd in de bronvermelding. Daarnaast zijn er bij het opstellen van de eisen en het onderzoek naar de systemen ook mondelinge gesprekken gehouden.

1. ASK platform

1.1 Inleiding

ASK (Access Society's Knowledge) is een intelligent en adaptief agent-gebaseerd communicatie platform ontwikkeld door ASK-CS. Het platform kan gebruikt worden om de communicatie te coördineren in dynamische omgevingen tussen meerdere personen of systemen, met behulp van verschillende media.

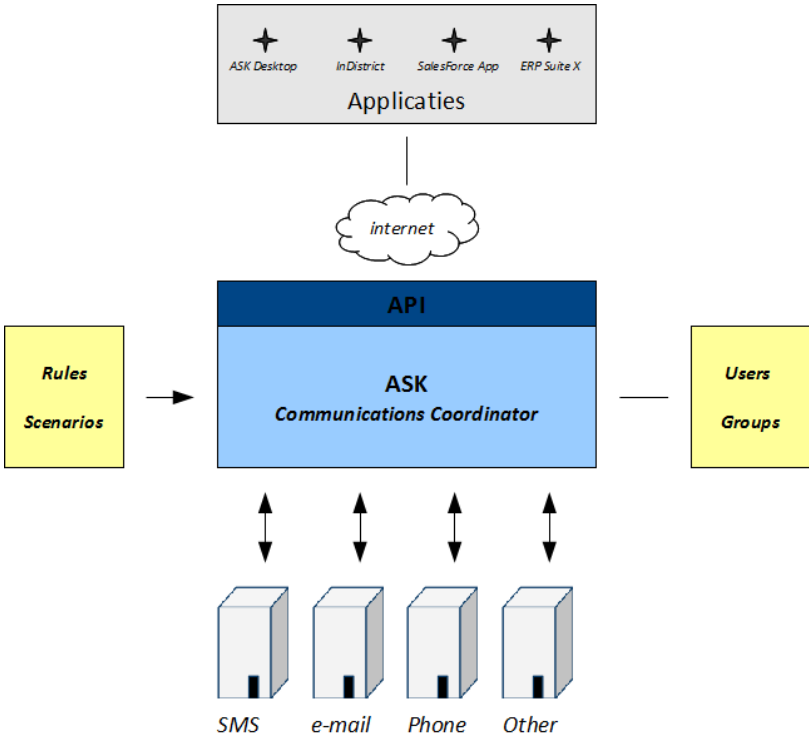
ASK gaat er vanuit dat elke persoon een virtuele persoonlijkheid heeft. Deze persoon wordt in het ASK platform vertegenwoordigd als een *agent.* ASK bestaat uit een dynamisch netwerk van agents die reageren op veranderingen van processen waar zij verantwoordelijk voor zijn.

De agents zijn zo opgezet dat ze altijd een bepaalde toestand handhaven binnen het systeem. Op het moment dat er een situatieverandering is kunnen de agenten acties doorspelen naar andere agents of mensen aan de hand van verschillende soorten communicatie. De meest voorkomende manieren van communicatie zijn op het moment telefoon(IVR menu's), SMS en e-mail.

Om te kijken met welke persoon contact mee opgenomen moet worden, wordt er gekeken naar de beschikbaarheid (user state), bereikbaarheid (medium), feedback van de gebruikers (heuristiek) en aangepaste matching algoritmen (bijvoorbeeld op basis van locatie).

1.2 Onderdelen van het ASK platform

In figuur 1 is de opbouw van het ASK platform te zien.



Figuur 1: Overzicht ASK

Users and groups

Het systeem is opgedeeld in gebruikers en groepen. De gebruikers zijn losstaande personen met eigen agenda's.

De gebruikers kunnen lid zijn van één of meerdere groepen. Deze groepen hoeven niet van bijvoorbeeld één bedrijf te zijn. Een gebruiker kan lid zijn van de hockeyvereniging waar hij af en toe opgeroepen wordt als scheidsrechter en daarbij ook lid zijn van één of meerdere afdelingen van zijn werk.

Een manager kan in het systeem berichten sturen naar één of meerdere groepen om afspraken te maken of om bijvoorbeeld de personen op de hoogte te brengen van een vergadering.

Communicatiemiddelen

Voor het communiceren naar de gebruikers zijn er verschillende methodes mogelijk. Als een manager bijvoorbeeld aangeeft dat er de volgende dag een vergadering om 13:00 is en de vraag heeft of een persoon hierbij aanwezig is, kan het systeem de gebruiker op verschillende manieren bereiken.

De eerste mogelijkheid is telefoon. De persoon wordt gebeld aan de hand van een IVR menu. Bij een IVR (Interactive voice response) krijgt de gebruiker een telefoontje met daarin één of meer vragen en moet de gebruiker antwoorden met toetsen die gerelateerd zijn aan de opties die in het bericht worden gegeven. Zo kan de gebruiker bijvoorbeeld aangeven met “1” dat hij bij de vergadering aanwezig is en met “2” dat hij niet aanwezig is. De keuze wordt geregistreerd door het systeem bij de vertegenwoordigende agent van de persoon.
Een deel van de IVR menu's wordt van te voren al ingesproken. Als het om een speciale vraag of tekst gaat kan de manager ook nog een persoonlijke tekst inspreken.

De volgende mogelijkheid is e-mail. Naar alle genodigden voor de vergadering wordt een e-mail gestuurd met de vraag of zij aanwezig zijn bij de vergadering. Via een link naar een webpagina kunnen de personen aangeven of zij aanwezig zijn bij de vergadering.

De derde mogelijkheid is sms. Hierbij krijgt de persoon een sms met het verhaal over de vergadering en moet deze antwoorden met een van de mogelijkheden, bijvoorbeeld “beschikbaar” of “niet beschikbaar”. Het probleem dat hier momenteel nog mee optreedt is dat het herkennen van sms teksten lastig is. Als de gebruiker bijvoorbeeld “beschkbaar” intypt in plaats van “beschikbaar” wordt de reactie niet goed herkend. Daarnaast kunnen er in een sms meerdere vragen tegelijk worden gesteld en is er bij het antwoord onduidelijk welk antwoord op welke vraag slaat.

Andere methodes zijn bijvoorbeeld sociale netwerk sites (denk aan Facebook en Twitter). Door de uitbreiding met de Android applicatie wordt het ook mogelijk om gebruikers een pop-up bericht op hun smartphone te laten verschijnen. Doordat ASK nog steeds uitgebreid wordt komen er steeds meer communicatie methodes in.

API

Via de API wordt het ASK platform aangesproken. De API biedt de schil om ASK heen om aanspreekbaar te zijn voor de applicaties.

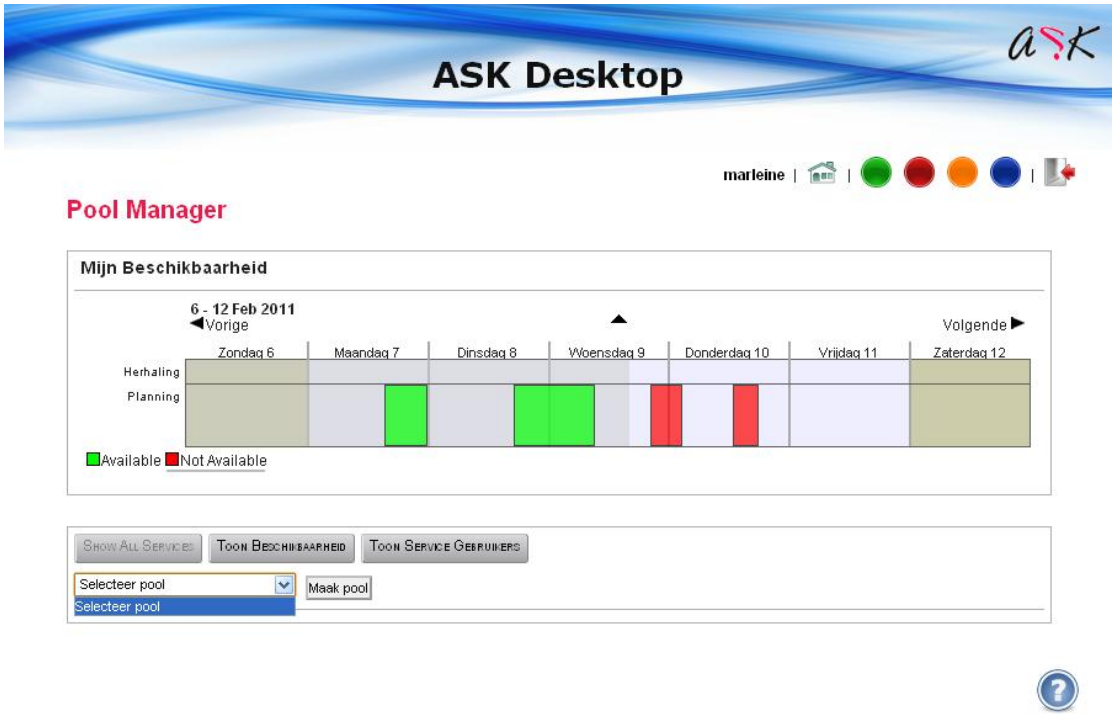
Applicaties

Om ASK goed te kunnen gebruiken is zijn er verschillende applicaties opgezet voor de gebruikers.

Een van deze applicaties is ASK Desktop. ASK desktop is een webinterface waarin de gebruiker een naar zijn wens ingerichte omgeving heeft. ASK zelf is ook een webinterface maar is dusdanig uitgebreid en ingewikkeld voor de gebruiker dat er een applicatie voor zit.

Bij uitlevering van ASK aan de klant wordt door een van de medewerkers van ASK de desktop ingericht met alle benodigde functionaliteiten maar ook met de juiste IVR menu's. De keuzes die de beller moet maken worden vooraf gedefinieerd.

In onderstaande afbeelding is ASK Desktop te zien:



Figuur 2: Voorbeeld ASK Desktop

Een van de functies van ASK Desktop is de pool manager. Een pool is ook wel een groep, bijvoorbeeld een groep van de reddingbrigade. De manager kan onderaan de pool selecteren en van de hele pool de agenda zien.

Mijn beschikbaarheid geeft de beschikbaarheid van de persoon zelf aan. Door in de kalender te klikken kan de persoon zijn beschikbaarheid en niet beschikbaarheid doorgeven. Leeg laten betekent voor het systeem “niet ingevuld”, oftewel het systeem kan contact met je opnemen als er een brigadierstekort is.

Een andere pagina binnen ASK Desktop is “Voer een bel of internetactie uit”. Hierin kan daadwerkelijk een actie worden uitgevoerd. Het hoofd van de brigade wil bijvoorbeeld meer brigadiers op een bepaald moment. Hij kan dan een belactie uitvoeren naar alle brigadiers die nog niets in hun kalender hebben aangegeven. Hij hoeft ze dan zelf niet allemaal na te bellen maar het systeem doet dat voor hem.

Rules and scenarios

Voor het verzenden van berichten of het inplannen van afspraken worden scenario's opgezet. Via een scenario kan worden aangegeven wanneer en hoe vaak een actie plaats moet vinden.

Als het hoofd van een reddingsbrigade zegt dat een agent in het systeem elk uur moet controleren of er voor de komende 24 uur genoeg reddingsbrigadiers beschikbaar zijn, wordt er een scenario opgezet. Er wordt vastgezet wat en in welke regelmaat gecontroleerd moet worden. Op het moment dat aan de hand van een controle een actie ondernomen moet worden(bijvoorbeeld belactie), wordt er bepaald welke stappen deze actie inhoudt.

Deze scenario's worden ingericht met behulp van een aantal agents in het systeem die een speciale taak toegewezen krijgen. Er is bijvoorbeeld een agent verantwoordelijk voor het controleren van de beschikbaarheid van reddingsbrigadiers in een groep.

Elke agent heeft een aantal rules volgens welke hij handelt. Indien er een tekort is in de beschikbaarheid van de brigadiers, kan deze agent bijvoorbeeld een aantal brigadiers bellen met een ingesteld IVR menu, en hen vragen of ze beschikbaar zijn.

ASK Communication Coordinator

De ASK Communication Coordinator is te zien als het regelcentrum van ASK. Vanuit hier wordt alles verwerkt. Dit is het hart van het platform, waar alle agenten 'leven'.

Op het moment dat er een bericht wordt verzonden door een manager vanuit bijvoorbeeld ASK Desktop wordt via de API ASK aangesproken, wordt het scenario gemaakt en wordt er aangegeven welke gebruikers / groepen van toepassing zijn op het bericht. Via de gewenste communicatielijn wordt het bericht verstuurd.

1.3 ASK webinterface

Zoals al eerder besproken is er een webinterface als basis voor ASK. Dit deel wordt vaak ASK genoemd. Als het om de uitbreidingen, zoals applicaties als ASK Desktop, gaat dan worden deze bij naam benoemd.

Figuur 3: Webinterface ASK

In onderstaande afbeelding is een voorbeeld van de webinterface te zien:

In de afbeelding is links het totale menu aan mogelijkheden in ASK te zien. Verschillende scenario's en users kunnen worden beheerd.

Het onderdeel dat openstaat is het instellen van een IVR menu. Hierin kan aangegeven worden om wat voor menu het gaat en welke keuzes er gemaakt kunnen worden.

De pop-up die openstaat laat één van de keuzes zien. In dit geval de keuze “2” waarbij de persoon aangeeft dat hij niet beschikbaar is. In de pop-up wordt aangegeven wat de volgende stap is. Komt hij in een volgend IVR menu of zit hij aan het einde en wordt er opgehangen.

Deze omgeving wordt ingesteld door de medewerkers van ASK, naar wens van de klant. De klant krijgt vervolgens enkel te maken met een applicatie die ASK gebruikt, bijvoorbeeld ASK Desktop.

2. Sense platform

2.1 Inleiding

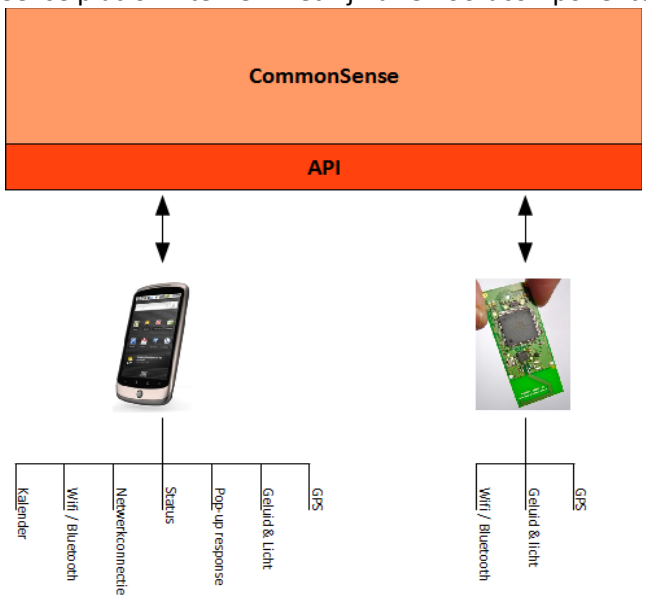
Vandaag de dag is de wereld gevuld met sensoren die allerlei eigenschappen meten. Mensen zelf zijn van nature uitgerust met een breed scala aan sensoren, voor onder andere zicht, evenwicht en aanraken. Hetzelfde geldt voor computers, smartphones en huishoudelijke apparaten, zoals koelkasten en thermostaten, die sensoren voor oriëntatie, temperatuur, locatie, connectiviteit en/of voeding hebben.

Het zou mooi zijn als een systeem de verschillende gegevens begrijpt en kan combineren om te bepalen wat een gebruiker aan het doen is, zonder dat de gebruiker het systeem zijn activiteiten moet vertellen. Aan de hand van de gegevens kan de computer of smartphone weer actie ondernemen of keuzes voor de gebruiker maken.

Het Sense platform speelt in op dit idee. Aan de hand van een smartphone en draadloos sensor netwerk worden verschillende sensoren uitgelezen en deze worden verwerkt om de activiteiten en keuzes van een persoon te analyseren, zodat andere systemen hier gebruik van kunnen maken.

2.2 Onderdelen van het Sense platform

Het Sense platform is gebaseerd op drie hoofdcomponenten, namelijk de smartphone, de MyriaNed Node en een server genaamd CommonSense. In het onderstaande figuur is de opbouw van het Sense platform te zien met zijn drie hoofdcomponenten.



Figuur 4: Sense platform

Smartphone

Tegenwoordig is er een opmars in smartphones. Steeds meer mensen hebben zo'n kleine computer bij zich. Het zal niet lang meer duren voordat iedereen de hele dag verbonden is met internet. Sense speelt in op deze trend door gebruik te maken van de vele sensor data die de smartphones kunnen verzamelen. De meeste mensen hebben hun telefoon de hele dag bij zich waardoor er uit de data te bepalen is wat de gebruiker aan het doen is.

Sensor data die de telefoon kan verzamelen en waar het Sense platform gebruikt van maakt:

- Locatie: Aan de hand van GPS of 3G/Wi-Fi-netwerkinfo kan de locatie van de telefoon bepaald worden
- Naburige personen of apparaten: aan de hand van Wi-Fi of Bluetooth apparaten in de buurt van de telefoon is te zien wat het lokale netwerk is, zowel wat betreft apparatuur als wat betreft personen, via hun smartphones.
- Omgeving: Door de microfoon te gebruiken en de lichtintensiteit te meten is te herkennen in wat voor omgeving een persoon zich bevindt. Dit is een lastig te meten sensor als de telefoon de gehele dag in de tas zit, maar dit kan op zich ook al waardevolle informatie zijn.
- Beweging: de oriëntatie en versnelling van de telefoon geven een goede indicatie of de telefoon wordt meegedragen door de gebruiker.
- Pop-up response: Niet alles is te meten met sensoren, soms werkt het het best om de gebruiker direct een vraag te stellen. Zo weet het systeem hoe een persoon zich bijvoorbeeld voelt.
- Telefoongebruik: Aan de hand van de status van de telefoon kan bepaald worden of de persoon deze veel gebruikt. Er kan bijvoorbeeld gekeken worden naar de stand-by tijd, het batterijniveau of de inkomende gesprekken en sms-berichten.

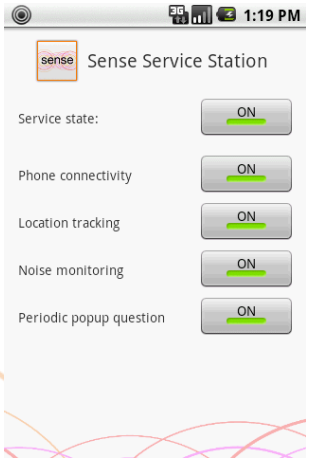
Een uitbreidingsoptie voor het Sense platform is:

- Agenda: Door het kijken in de agenda van de gebruiker kan er bepaald worden wat deze op dat moment aan het doen is.

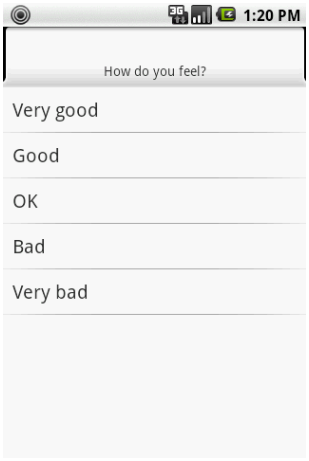
Door verschillende sensordata te koppelen kan er nog duidelijker herkend worden wat een gebruiker aan het doen is. Zo kan er aan de hand van geluid en tijd en locatie gedefinieerd worden dat de persoon op dat moment thuis aan het slapen is.

Een ander voorbeeld is de locatie binnen een gebouw definiëren. Door te kijken naar de wireless devices en de locatie van een persoon kan er bepaald worden dat deze zich op de tweede verdieping bevindt.

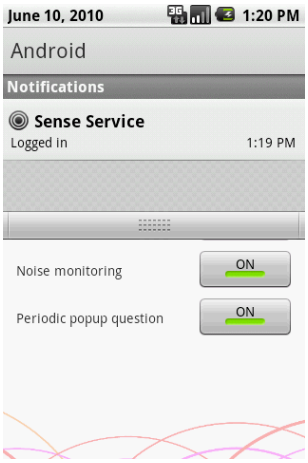
In onderstaande afbeeldingen zijn de screenshots van het Sense platform op een Android smartphone te zien:



Figuur 5: Screenshot: Keuze welke sensoren uitgelezen mogen worden

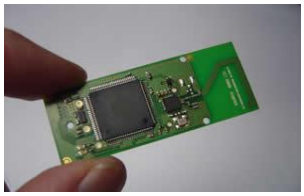


Figuur 6: Screenshot: Pop-up vraag; Hoe voel je je?



Figuur 7: Screenshot: Bij de meldingen wordt aangegeven dat Sense actief is

MyriaNed



MyriaNed is een draadloos sensor netwerk dat ontwikkeld is door DevLab, een samenwerkingsorganisatie tussen kleine en middelgrote bedrijven in Nederland.

Het netwerk bestaat uit allemaal kleine nodes (ook wel Wireless Sensor Node genoemd). Deze nodes communiceren via een broadcast protocol met elkaar, waardoor alles wat ze verzenden, wordt ontvangen door alle nodes die bij ze in de buurt zijn.

De nodes zelf bevatten weinig sensoren, maar ze zijn zo gebouwd dat er sensoren opgezet kunnen worden en dat deze uitgelezen kunnen worden. Voorbeelden van dit soort sensoren zijn:

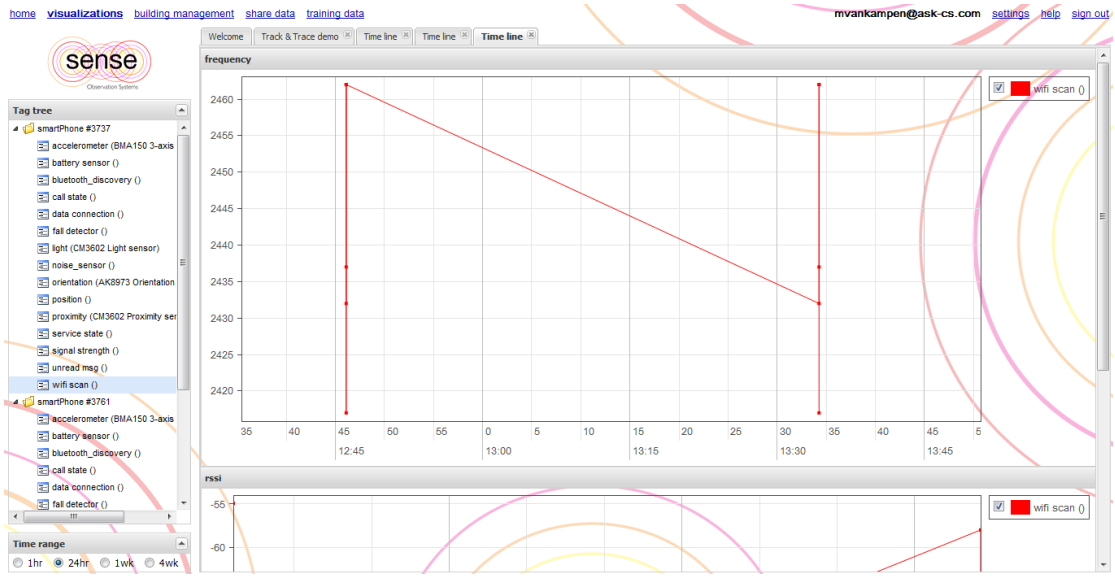
- GPS: Locatiebepaling
- Wifi: Omliggende Wireless devices kunnen de locatie van de node aangeven
- Bluetooth: communicatie met andere devices in de omgeving kan de locatie aangeven van de node
- Licht: Door lichtintensiteit te meten kan het dagritme van een persoon bepaald worden (door bijvoorbeeld een node buiten op de kleding te dragen)

Voor het Sense platform kan een persoon maar ook een object een node meekrijgen. Dit bijvoorbeeld om te bepalen waar een pakket is bij pakketbezorging.

CommonSense

CommonSense is een intelligente datastore voor de meetgegevens van de smartphone en de MyriaNed Node. De gegevens worden niet alleen opgeslagen, CommonSense beschikt over geavanceerde algoritmes. De algoritmes kunnen de binnenkomende data analyseren, patronen herkennen en de data verrijken. Daarnaast is het mogelijk om gegevens van groepen gebruikers te delen.

Via de online interface (te zien in onderstaande afbeelding) is data gemakkelijk te visualiseren in grafieken en tabellen. Ook is het mogelijk om de data weer te geven op landkaarten of plattegronden van gebouwen. Door data van verschillende sensoren te combineren is het mogelijk snel verbanden te ontdekken en problemen op te sporen.



Figuur 9: Webinterface CommonSense

De data die op de CommonSense wordt geplaatst is realtime in te zien en te beheren. Zo wordt er altijd een actueel beeld van de situatie gegeven en kan snel worden gereageerd op incidenten. CommonSense kan ook zo worden ingesteld dat deze e-mails of sms-berichten stuurt bij bepaalde gebeurtenissen.

CommonSense maakt gebruik van verschillende algoritmes om de status van de gebruikers van de applicatie te herkennen. Hiervoor worden geavanceerde “machine learning” algoritmes gebruikt die voor alle gebruikers van CommonSense beschikbaar zijn. Daarnaast wordt er gebruik gemaakt van een bibliotheek met standaardfuncties, om bijvoorbeeld gemiddeldes te berekenen.

Naast de applicatie die is gemaakt is het ook mogelijk om zelf een applicatie te maken en gebruik te maken van CommonSense. CommonSense heeft een open API voor communicatie met de buitenwereld. De gebruiker kan via een eigen applicatie sensor data (ook ander soort sensoren) doorsturen naar CommonSense en deze verwerken.

3. MyASK

3.1 Inleiding

MyASK is een van de applicaties die gemaakt wordt, gebruik makende van en binnen het ASK platform. Ook ASK gaat mee met de opmars van de smartphones en maakt MyASK,, een multiplatform applicatie voor bijvoorbeeld onder andere iPhone en Android™ smartphones. Naast de MyASK webinterface kan er per platform een applicatie worden opgezet.

Op mobiele apparaten zal MyASK bestaan uit een 'native' gedeelte dat op de telefoon zelf draait, en een user interface die bestaat uit een web view. Het native gedeelte zal op de achtergrond blijven draaien, en maakt het mogelijk om actief de gebruiker te notificeren, en sensoren uit te lezen. De web view is een web interface die de gebruiker mogelijkheden tot interactie biedt met zijn agent.

3.2 Doel

De bedoeling van MyASK is de gebruikers van ASK inzicht te geven in hun persoonlijke agent. Via de applicatie is het mogelijk om te bekijken wat de agent voor de gebruiker geregeld heeft, alsmede het opgeven van nieuwe opdrachten. Ook via een applicatie te notificeren of afspraken in te laten plannen. Oftewel een dergelijk systeem als ASK Desktop maar dan voor de smartphone en computer. Het lijkt conceptueel op de ASK Desktop applicatie, maar MyASK is gericht op het individu, waarbij de ASK Desktop voor beheerders was.

Onderdelen die voor MyASK ontwikkeld zijn of nog worden ontwikkeld:

- Notificaties:
  - Gebruiker notificeren: via de applicatie een bericht weergeven aan de gebruiker
  - Gebruiker vragen om feedback: Bij het inplannen van een afspraak via een bericht vragen of de gebruiker aanwezig kan zijn.
  - Gebruiker meerdere vragen stellen om volgende taken te bepalen (in samenwerking met het dochterbedrijf Regas, gericht op project management)
- Status:
  - Status van de gebruiker opvragen en periodiek controleren
  - Ontdekken of de gebruiker in de buurt is van de telefoon (als applicatie op de telefoon)
  - Voorgedefinieerde statussen waar de gebruiker bepaalde actie aan toe kan voegen en speciale statussen toevoegen
- Inkomende berichten / oproepen
  - Door de status van de gebruiker kan de telefoon bepaalde acties ondernemen wat betreft inkomende berichten/oproepen. Zo kan de gebruiker zijn status op "in vergadering" hebben staan. Dan zal de applicatie ervoor zorgen dat de telefoon niet doorgeschakeld wordt door de gebruiker, maar dat hij achteraf een bericht krijgt dat hij is gebeld. Dus eigenlijk worden alle inkomende oproepen, SMS en e-mail notificaties uitgeschakeld.
  - Mocht het gesprek dusdanig belangrijk zijn kan er wel een "emergency call" label aan gegeven worden om het gesprek alsnog door te laten voeren.

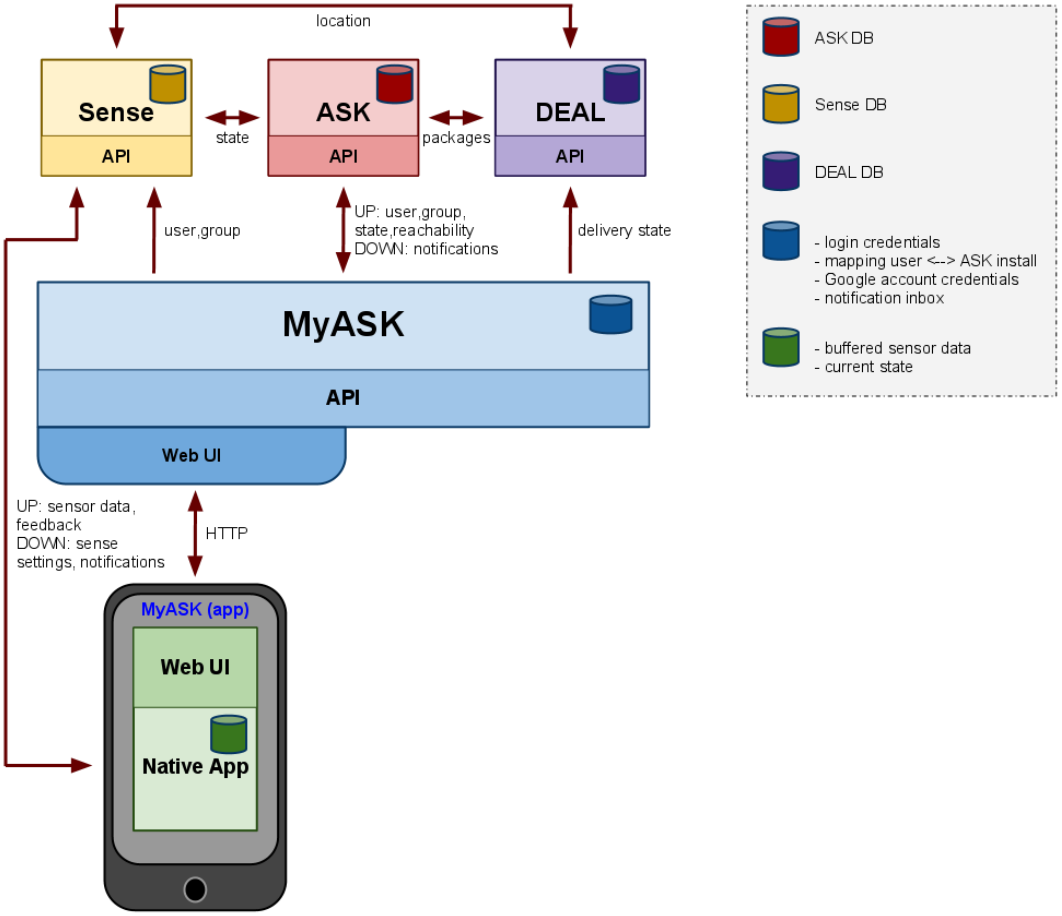
- Teams
  - Groepen die binnen ASK bekend zijn als groep kunnen communiceren via MyASK
  - Speciale notificaties sturen naar personen binnen de groep
- Agenda
  - Mogelijk om agenda items van meerdere personen te verplaatsen om deze op een beter moment plaats te laten vinden. Hier hoeft de gebruiker niet bewust mee bezig te zijn, het wordt voor hem gedaan. Het systeem vergelijkt de agenda's en kiest het beste tijdstip uit aan de hand van de eisen.
- Locatie
  - Het tonen van nabijheid van collega's of mensen uit een groep
- Locatie

Het tonen van nabijheid van collega's of mensen uit een groep

3.3 Opbouw samenwerking ASK, Sense en DEAL

Eén applicatie die specifiek wordt gebouwd aan de hand van MyASK is een logistieke applicatie voor het dochterbedrijf DEAL in samenwerking met ASK en Sense.

In de onderstaande afbeelding is de samenwerking weergegeven.



Figuur 10: Samenwerking ASK, Sense en DEAL

Om een pakketbezorger zo efficiënt mogelijk te laten werken is een goede route nodig. De MyASK applicatie maakt gebruik van zoveel mogelijk gegevens om deze route zo goed mogelijk te bepalen.

Op het moment dat een aantal pakketten in de wagen liggen van de bezorger wordt een route opgesteld met alle adressen waar de pakketjes naartoe moeten. De ontvangers die een MyASK applicatie op hun telefoon hebben staan kunnen kijken hoe laat hun pakket arriveert. De applicatie stelt dan de vraag aan de gebruiker of deze op dat tijdstip ook echt beschikbaar is. De gebruiker kan bijvoorbeeld aangeven dat deze op dat moment niet aanwezig is en het pakket liever later op de dag of op een andere dag bezorgd wil hebben. Ook kan de gebruiker aangeven dat hij het pakket bezorgd wil hebben op een ander adres (bijvoorbeeld bij de buren of op zijn werk), de applicatie kijkt dan of dit haalbaar is binnen de route.

Op het moment dat het pakket binnen bepaalde afstand van de ontvanger is krijgt hij ook een notificatie om te laten weten dat het pakket eraan komt en vraagt daarbij of de ontvanger het pakket ook echt in ontvangst kan nemen op dat moment.

In het bovenstaande verhaal wordt DEAL gebruikt voor het organiseren van de pakketten en informatie op te vragen van de pakketten. Vanuit MyASK wordt richting DEAL aangegeven dat een pakket wel of niet is afgeleverd.

ASK wordt gebruikt om berichten te versturen naar de ontvanger en de responses te verwerken.

Sense wordt gebruikt voor de status en locatie van de ontvanger. Stel de ontvanger is op twee uur afstand van zijn huis en de pakketbezorger op een kwartier afstand van het bezorgadres; dan wordt er niet meer aan de gebruiker gevraagd of hij het gaat redden maar worden er andere opties voor bezorging gegeven en wordt de route aangepast.

3.4 Multiplatform en platform specifiek

De MyASK Deal applicatie is echt een applicatie die wordt gebouwd als applicatie op de smartphone en is dus platform specifiek.

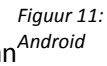
Het idee van de “waar is mijn eigenaar”-applicatie is ook dat deze als applicatie op een Android smartphone draait en kan samenwerken met de MyASK webinterface.

4. Android

4.1 Inleiding

Google Android is een open source platform voor mobiele telefoons en is gebaseerd op de Linux-kernel en het Java-programmeerplatform.

4.2 Specificaties Android



Figuur 11: Android

Het Android heeft verschillende specificaties waar ook met het ontwikkelen van nieuwe applicaties rekening mee gehouden moet worden of juist voordelen uitgehaald kunnen worden.

Specificatie	Uitleg
Instelbare layout	Het platform is aanpasbaar naar grotere, VGA, 2D graphics library, 3D graphics library specificaties en traditionele smartphone layouts
Opslag	SQLite (lightweight relation database) wordt gebruikt voor data opslag
Connectiviteit	Android support als connectie technologieën: • GSM / EDGE • IDEN • CDMA • EV-DO • UMTS • Bluetooth • Wi-Fi • LTE • WiMAX
Berichten	Android support als bericht technologieën: • SMS • MMS • Threaded text messaging • Android cloud to Device Messaging Framework (C2DM)
Web browser	The web browser die beschikbaar is voor Android is gebaseerd op de open-source WebKit layout engine gekoppeld met Chrome V8 JavaScript engine
Media support	Android support de volgende audio/video/still media formaten: • WebM • H.263 • H.264 • MPEG-4 SP • AMR-WB • AAC • HE-AAC • MP3 • MIDI • Ogg Vorbis • WAV • JPEG • PNG • GIF • BMP[45]
Streaming media support	roid support de volgende streaming media formaten: • RTP/RTSP (3GPP PSS, ISMA) • HTML progressive download (HTML5 <video> tag • Adobe Flash Streaming (RTMP)

	• HTTP Dynamic Streaming met Flash 10.1 plugin • Apple HTTP Live streaming met RealPlayer for mobile, in 3.0 zonder ook beschikbaar • Verwacht is Microsoft Smooth Streaming in een nieuwere versie Android
Gekoppelde hardware support	Android kan gebruik maken van: • Camera's • Touchscreens • GPS • Accelerometers • Gyroscopes • Magnetometers • Nabijheids- en druksensoren • Thermometers • Accelerated 2D bit blits (met hardware orientatie, schaling, pixel formaat conversie) • Acceleratie 3D graphics
Ontwikkelomgeving	Voor het ontwikkelen van Android applicaties is voor de Eclipse IDE een Android Development Tools (ADT) beschikbaar. Er kan geprogrammeerd worden in de object-georiënteerde programmeertalen Java en C/C++
Market	De Android Market is een catalogus van applicaties die geinstalleerd is op alle Android systemen.
Multi-touch	Voor Android is multi-touch beschikbaar
Bluetooth	Android support de volgende mogelijkheden van Bluetooth: • A2DP • AVRCP • OPP (Verzenden van bestanden) • PBAP (Toegang tot contacten) • Spraakberichten / tekstberichten tussen telefoons • Functioneren als extern toetsenbord, muis of joystick aan de hand van applicaties van derden
Video bellen	Android support standaard geen video bellen. Sommige speciale telefoons beschikken binnen het systeem wel over mogelijkheden, bijvoorbeeld via UMTS of over IP.
Spraakherkenningsfuncties	Binnen Android zijn de volgende spraakherkenningsfuncties beschikbaar: • Google search • Bellen • Berichten versturen aan de hand van spraakherkenning • Navigatie • Applicaties van derden die gebruik maken van spraak
Tethering	Met Android is het mogelijk om een Android apparaat te gebruiken als Wi-Fi hotspot.

5. Pakket van eisen

De eisen zijn verdeeld in verschillende groepen volgens het zogenaamde FURPS+ systeem. FURPS+ verdeelt de eisen in de volgende groepen:

- Functional (Functioneel)
- Usability (Bruikbaarheid)
- Reliability (Betrouwbaarheid)
- Performance (Werking)
- Supportability (Onderhoudbaarheid)
- Implementation (Implementatie)
- Interface (Hoe ziet de applicatie eruit)
- Systeem licentie (Wat betreft de licentie / eigenaar van het op te leveren product)

De eisen worden gekwalificeerd aan de hand van de volgende MOSCOW-kenmerken:

- Must have
  Het uiteindelijke product moet deze eis ondersteunen.
- Should have
  Het zou mooi zijn als het uiteindelijke product deze functionaliteit heeft, maar het hoeft niet (niet perse)
- Could have
  Deze functionaliteit staat niet bij de basiseisen, maar als het wordt toegevoegd is het een mooi extra
- Won't have
  Deze eis wordt niet meegenomen in het uiteindelijke product

Aan de eisen wordt ook een prioriteit gehangen, waarbij één de hoogste prioriteit is. Aan de hand van de prioriteit kan een duidelijke planning voor het ontwikkelen van het systeem worden opgezet.

5.1 Functionele eisen

Het systeem heeft twee basiseisen deze zijn hieronder uitgewerkt. Aan de basiseisen zitten verschillende oplossingseisen. Deze eisen zijn in de basiseisen verwerkt.

Nr	Eis	Klassificatie	Prioriteit
F1	De applicatie moet herkennen of de eigenaar in de buurt van de smartphone is of niet	Must	
Oplossingseisen			
F1-1	Het systeem kan kijken naar de agenda van de eigenaar om te weten waar deze op het moment is	Won't	
F1-2	Het systeem moet een pop-up vraag kunnen geven om te controleren of de eigenaar in de buurt is	Should	
F1-3	Het systeem moet even de microfoon kunnen aanzetten van de telefoon om te herkennen of de stem van de eigenaar in de buurt is	Won't	

Tabel 1: Basiseis F1

Nr	Eis	Klassificatie	Prioriteit
F2	De applicatie moet actie ondernemen als de eigenaar niet in de buurt is van de smartphone	Must	
Oplossingseisen			
F2-1	De applicatie moet een e-mail kunnen sturen naar de eigenaar met daarin de locatie van de telefoon (met in de e-mail een adres aan de hand van GPS locatie en een link naar de kaart)	Should	
F2-2	De applicatie kan de eigenaar bellen op andere telefoonnummers die bij hem bekend zijn	Could	
F2-3	Het systeem moet andere personen kunnen bellen bij afwezigheid met als doel de eigenaar te wijzen op zijn afwezigheid en deze aan te sturen zich te melden bij zijn telefoon.	Should	
F2-4	Het systeem moet andere personen kunnen notificeren bij afwezigheid met als doel de eigenaar te wijzen op zijn afwezigheid en deze aan te sturen zich te melden bij zijn telefoon.	Should	
F2-5	Het systeem moet andere personen kunnen e-mailen bij afwezigheid met als doel de eigenaar te wijzen op zijn afwezigheid en deze aan te sturen zich te melden bij zijn telefoon.	Should	
F2-6	De eigenaar zou zich moeten melden bij de eigenaar door middel van een antwoord op de notificatie van de telefoon, waarmee aangeeft dat hij “gevonden” is.	Should	
F2-7	De eigenaar zou zich kunnen melden bij de telefoon door middel van een webinterface waar hij zich aanmeldt en aangeeft dat hij “gevonden” is.	Could	
F2-7	Het systeem kan kijken naar de agenda van de eigenaar om te weten waar deze op het moment is en welke personen hij vervolgens contact mee op kan nemen	Won't	
F2-8	Het systeem kan aan de hand van eerdere keuzes bepalen welke persoon hij belt / een bericht stuurt (zelf-lerend algoritme)	Could	

Tabel 2: Basiseis F2

5.2 Usability (bruikbaarheid)

Nr	Eis	Klassificatie	Prioriteit
U1	De applicatie moet een duidelijke en simpele Graphical User Interface(GUI) hebben. De applicatie draait op de achtergrond, de gebruiker hoeft er dus niet veel mee te doen	Must	
U2	De gebruiker moet duidelijke meldingen krijgen op het moment dat de applicatie acties gaat ondernemen.	Must	
U3	Vanuit de gebruiker moet een constante interrupt mogelijk zijn. De gebruiker kan weer bij zijn telefoon aanwezig zijn en moet dan aangeven dat de applicatie geen actie meer moet ondernemen om hem te zoeken	Must	

Tabel 3: Usability eisen

5.3 Reliability (betrouwbaarheid)

Nr	Eis	Klassificatie	Prioriteit
R1	De applicatie moet uiterlijk iedere 15 minuten door hebben of zijn eigenaar in de buurt is of niet	Must	
R2 (P1)	De applicatie moet op geen enkel moment andere applicaties verstoren.	Must	
R3	De applicatie moet binnen 30 minuten zijn eigenaar vinden	Must	
R4	De applicatie moet bij onbereikbaarheid de status “onvindbaar” geven.	Must	
R5	Na 60 minuten na status verandering naar “onvindbaar” en de eigenaar nog steeds afwezig is (dus status nog steeds “onvindbaar”) gaat de telefoon opnieuw proberen om de eigenaar te vinden.	Should	
R6	De applicatie moet minimaal 3 manieren hebben om de eigenaar terug te vinden	Must	
R7	Om zeker te kunnen weten dat de echte eigenaar de telefoon oppakt, zou de applicatie een authenticatie mogelijkheid kunnen gebruiken (bijvoorbeeld patroon invoer)	Could	

Tabel 4: Reliability eisen

5.4 Performance (prestatie)

Nr	Eis	Klassificatie	Prioriteit
P1 (R2)	De applicatie moet op geen enkel moment andere applicaties verstoren.	Must	
P2	Bij een fout moet de applicatie een duidelijke melding naar de gebruiker geven	Must	
P3	De batterij van de telefoon moet minstens 12 uur meegaan. Hierbij wordt er vanuit gegaan dat enkel het Sense platform en de applicatie draaien (oftewel geen intensief gebruik van andere applicaties die het batterijniveau verlagen. De batterijduur geldt voor een nieuwe telefoon	Must	

Tabel 5: Performance eisen

5.5 Supportability (ondersteuning)

Nr	Eis	Klassificatie	Prioriteit
S1	De applicatie kan na oplevering beheerd worden door Sense Observation Systems en ASK Community Systems	Could	
S2	Voor een duidelijke documentatie moet Javadoc worden gebruikt voor alle functies	Must	
S3	De applicatie moet uitbreidbaar zijn voor samenwerking of inbouw in het Sense platform	Must	
S4	De applicatie moet uitbreidbaar zijn voor samenwerking of inbouw in MyASK	Must	
S5	Sense en ASK geven ondersteuning bij ontwikkeling als de code op de SVN server wordt geplaatst	Must	

Tabel 6: Supportability eisen

5.6 Implementatie eisen

Nr	Eis	Klassificatie	Prioriteit
IMP1	De applicatie moet worden geschreven voor Android	Must	
IMP2	De applicatie moet werken vanaf versie 2.1 van Android	Must	
IMP3	De applicatie zou moeten werken met versie 1.6 van Android	Should	
IMP4	De code van de applicatie moet samenwerken met de code van het Sense platform (versie februari 2011)	Must	
IMP5	De code van de applicatie moet samenwerken met de code van het ASK platform (versie februari 2011)	Must	
IMP6	De e-mails en berichten die worden verzonden vanuit de applicatie moeten als hoofdtaal in het Engels zijn	Must	
IMP7	De applicatie moet als hoofdtaal Engels hebben	Must	
IMP8	De applicatie zou meerdere taalpakketten kunnen ondersteunen	Could	
IMP9	De e-mails en berichten die worden verzonden vanuit de applicatie zouden in meer talen geschreven kunnen worden (instelling aan de hand van systeem taal)	Could	

Tabel 7: Implementatie eisen

5.7 Interface eisen

Nr	Eis	Klassificatie	Prioriteit
GUI1	Voor de gebruiker zou de interface (GUI) simpel en duidelijk moeten zijn	Should	
GUI2	In de GUI zou de gebruiker kunnen aangeven of hij zijn telefoon laat liggen (door uitschakelen services)	Could	
GUI3	Bij de eerste keer gebruik van de applicatie wordt de gebruiker gevraagd een stemband op te nemen voor belacties naar collega's / vrienden (De gebruiker moet een template voor gesprekken kunnen opzetten)	Should	
GUI4	Bij de eerste keer gebruik van de applicatie wordt de gebruiker gevraagd of hij een standaard e-mail of zelf geschreven e-mail wil verzenden (De gebruiker moet een template voor e-mails kunnen opzetten)	Should	

Tabel 8: Interface eisen

5.8 Systeem licentie

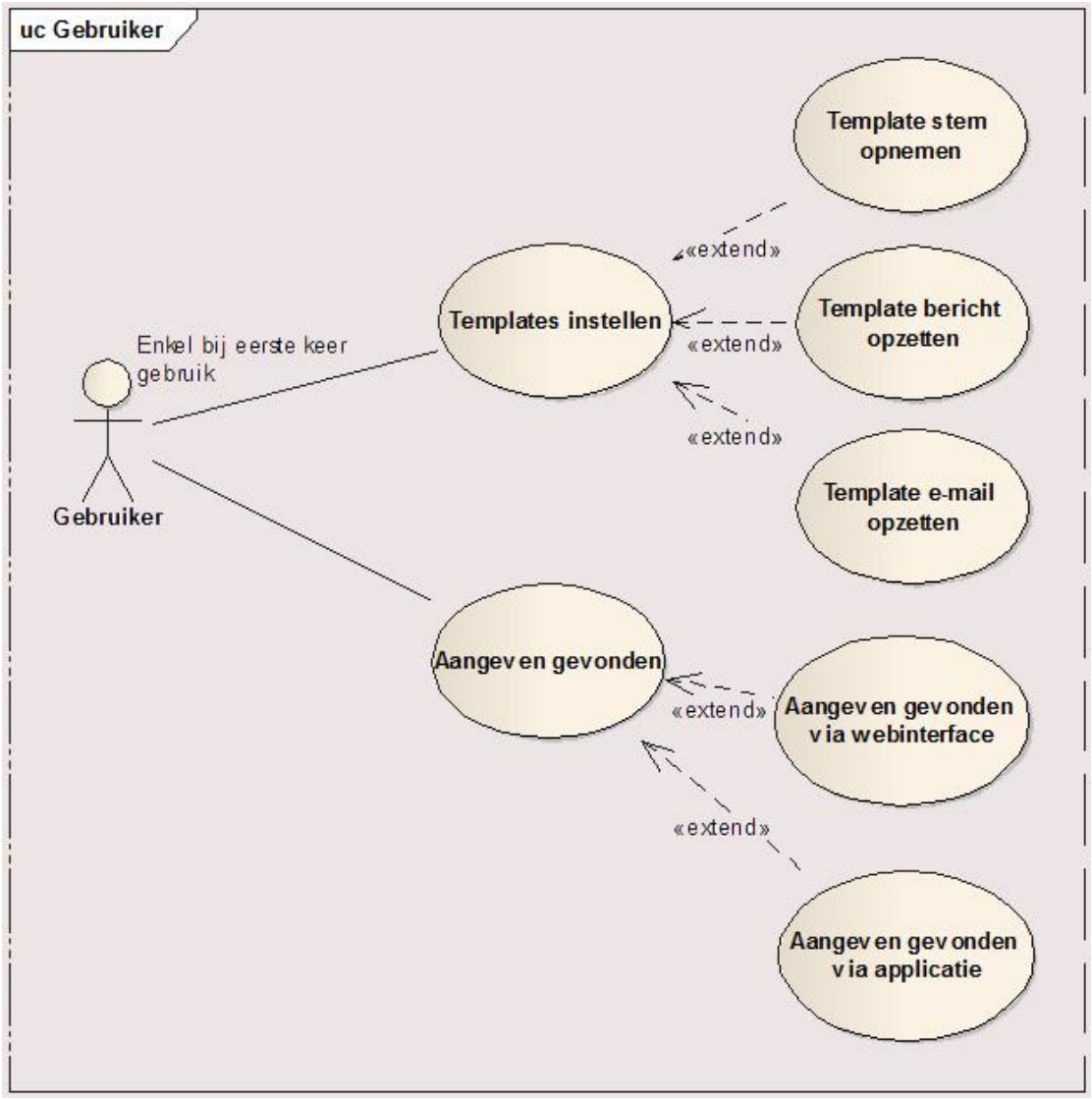
Nr	Eis	Klassificatie	Prioriteit
LIC1	De applicatie is eigendom van de bedrijven ASK Community Systems en Sense Observation Systems	Must	

Tabel 9: Systeem licentie

6. Use case diagrammen

6.1 Gebruiker

In onderstaande afbeelding is de Use Case vanuit de gebruiker weergegeven. Het zijn de acties waar de gebruiker bewust mee bezig is. De acties die door de applicatie worden uitgevoerd zijn te vinden in de paragraaf over de applicatie (6.2).



Figuur 12: Use Case - Gebruiker

Templates instellen	
Actoren	Gebruiker
Beschrijving	*Template stem opnemen*: Voor het IVR menu is een persoonlijk bericht nodig. Bij de eerste keer opstarten wordt gevraagd aan de gebruiker zijn persoonlijke bericht in te spreken. *Template bericht opzetten*: Voor het bericht dat naar een ander persoon wordt verzonden is een tekst nodig. De applicatie vraagt bij de eerste keer gebruik van de applicatie om de tekst goed te keuren of om een eigen tekst in te voeren. *Template e-mail opzetten*: Voor de e-mail dat naar een ander persoon wordt verzonden is een tekst nodig. De applicatie vraagt bij de eerste keer gebruik van de applicatie om de tekst goed te keuren of om een eigen tekst in te voeren.
Resultaat	Bij het zoeken naar de eigenaar van de telefoon is een vraag ingesproken of opgeschreven door de gebruiker zelf. Hij weet dan zelf ook wat er wordt gevraagd aan andere personen en is het hiermee eens.
Uitzonderingen	–
Uitbreidingen	Het wijzigen van de berichten door middel van het instellingen

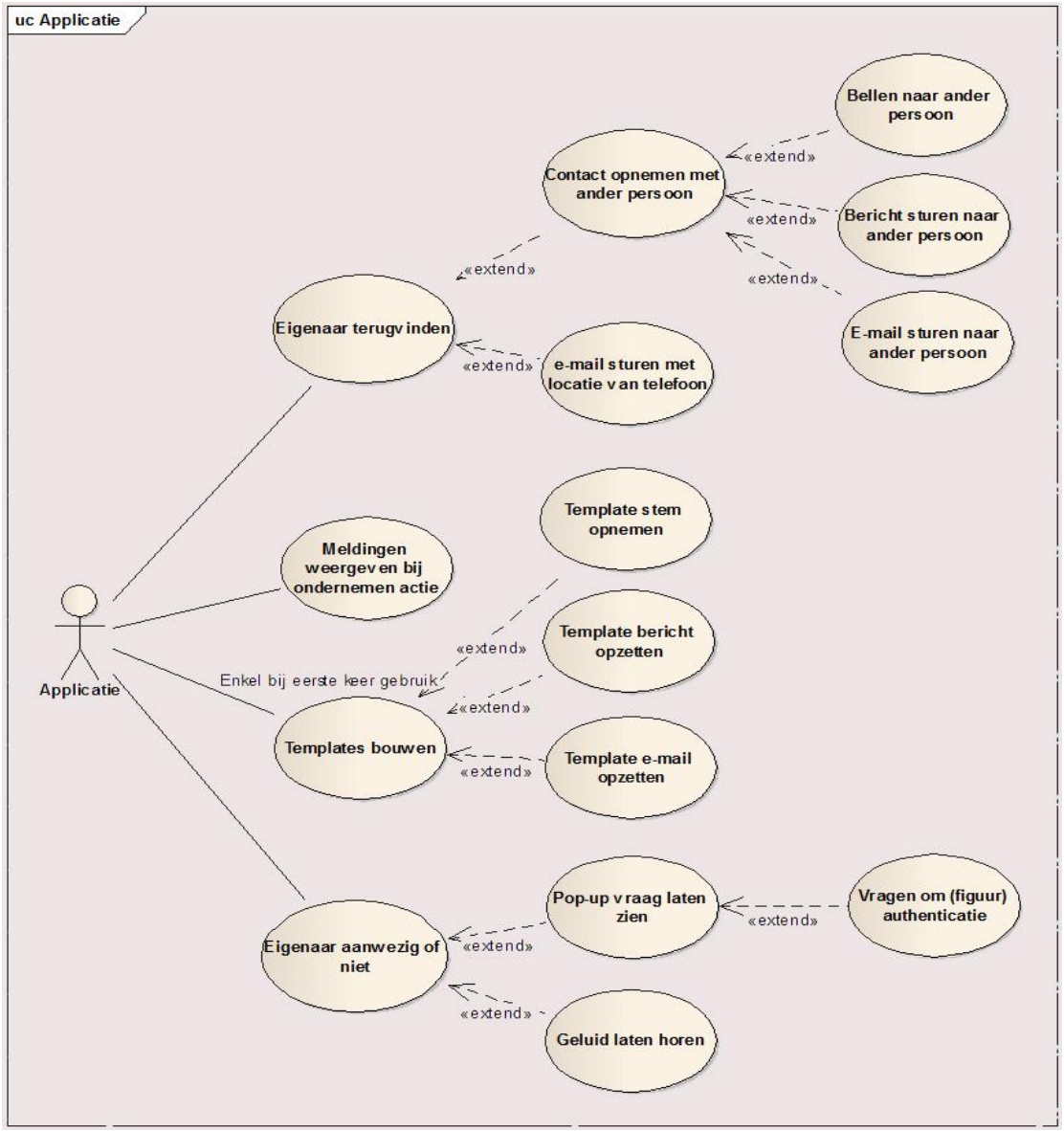
Tabel 10: Use Case Scenario - Templates instellen

Aangeven gevonden	
Actoren	Gebruiker
Beschrijving	*Aangeven gevonden via webinterface*: Als de eigenaar van een ander persoon te horen krijgt dat hij zich moet melden bij zijn telefoon, kan dit ook via een webinterface. De eigenaar kan namelijk niet bij de telefoon in de buurt zijn (op kantoor en de telefoon thuis). Door via de webinterface aan te geven waar hij is, wordt de status van de eigenaar op “gevonden, maar niet aanwezig” gezet. Op het moment dat de eigenaar weer bij de telefoon is, wordt de status op “aanwezig” gezet. *Aangeven gevonden via telefoon:* Door de telefoon weer aan te zetten, en mogelijk te authenticeren, wordt de status van de gebruiker op “gevonden” gezzet. Dit speelt ook als de telefoon nog acties aan het uitvoeren is en de eigenaar de telefoon weer gaat gebruiken. Deze blijkt dan wel degelijk in de buurt te zijn.
Resultaat	De eigenaar is weer herenigd met de telefoon of voor de telefoon is bekend waar de eigenaar zich bevindt.
Uitzonderingen	–
Uitbreidingen	De gebruiker kan aangeven tot wanneer hij niet aanwezig kan zijn bij zijn telefoon. Hierdoor gaat het systeem niet elk kwartier zoeken waar zijn eigenaar is.

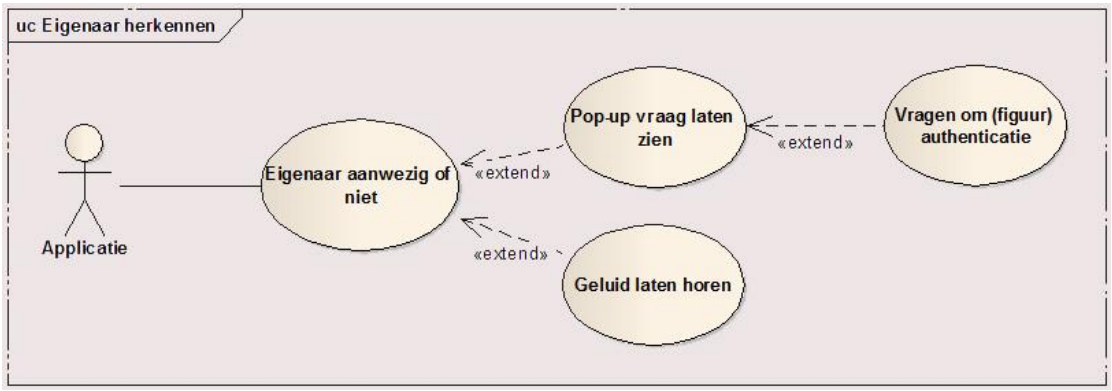
Tabel 11: Use Case Scenario - Aangeven gevonden

6.2 Applicatie

In onderstaande afbeelding is de volledige Use Case te zien over wat de applicatie moet doen aan functies. Als hoofdfuncties is gegeven dat de applicatie moet herkennen of de eigenaar in de buurt is of niet en als de eigenaar niet bij de telefoon in de buurt is, de eigenaar terug te vinden. Daarnaast is het van belang dat de gebruiker duidelijke feedback krijgt, oftewel duidelijke meldingen bij acties. Om de applicaties te personaliseren is het mogelijk om templates op te zetten.



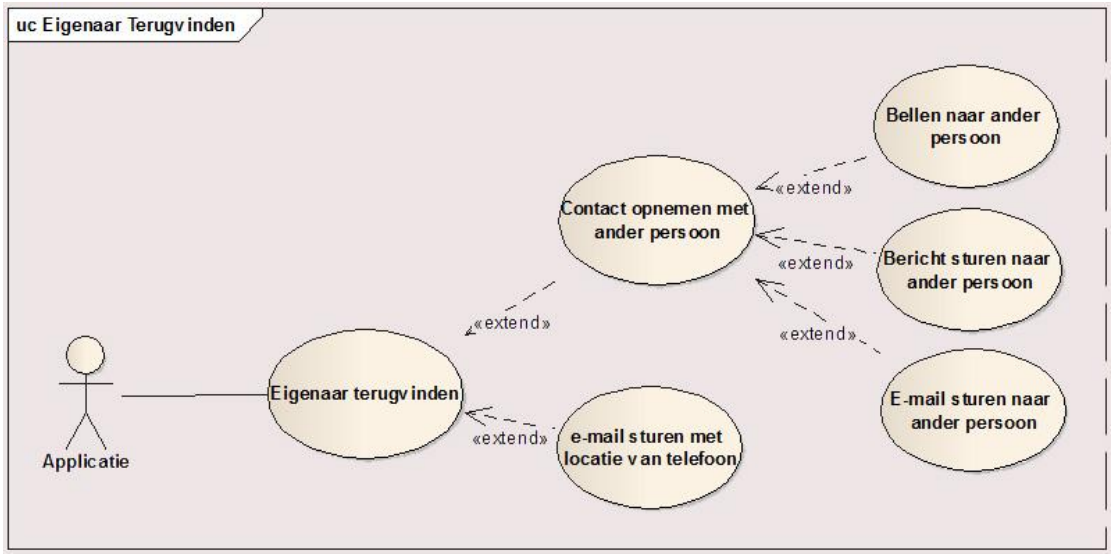
Figuur 13: Use Case - Applicatie totaal



Figuur 14: Use Case - Eigenaar herkennen

Eigenaar aanwezig of niet	
Actoren	Applicatie
Beschrijving	De applicatie gaat uitzoeken of de eigenaar in de buurt van de telefoon is *Pop-up vraag laten zien*: De gebruiker krijgt een een bericht van de telefoon waarop hij moet antwoorden. De applicatie kan vragen om *figuur authenticatie.* Hierbij moet de gebruiker een patroon invoeren. Het patroon is in de meeste gevallen alleen bekend bij de eigenaar. Het kan hierdoor minder snel voorkomen dat iemand anders dan de eigenaar de telefoon oppakt en het bericht beantwoordt. *Geluid laten horen*: De applicatie speelt een simpel geluid af, door de telefoon uit stand-by te halen of door op een knop te drukken wordt het geluid uitgezet en is de eigenaar gevonden.
Resultaat	Als de eigenaar in de buurt van de telefoon is, dus berichten van de telefoon beantwoordt, wacht de telefoon 15 minuten met opnieuw herkennen van de eigenaar Als de eigenaar niet in de buurt is van de telefoon, hij reageert niet op de berichten van de telefoon, dan gaat de telefoon opzoek naar de eigenaar.
Uitzonderingen	–
Uitbreidingen	–

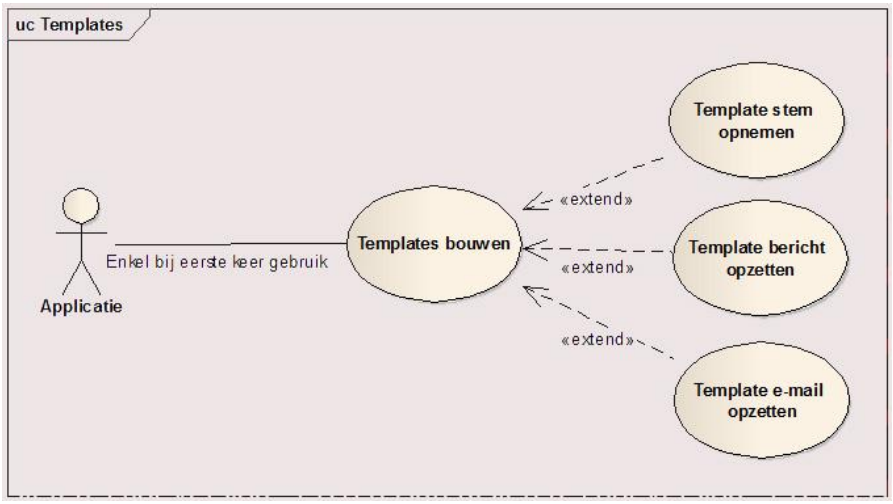
Tabel 12: Use Case Scenario - Eigenaar aanwezig of niet



Figuur 15: Use Case – Eigenaar terugvinden

Eigenaar terugvinden	
Actoren	Applicatie
Beschrijving	De applicatie gaat proberen de eigenaar terug te vinden *Contact opnemen met ander persoon*: De applicatie kiest personen uit een groep van de eigenaar (bijvoorbeeld collega's) en gaat deze proberen te *bellen*, *sms'en* of *e-mailen*. De persoon waar contact mee opgenomen wordt krijgt de vraag of hij weet waar de eigenaar zich bevindt. *E-mail sturen met locatie van de telefoon*: De eigenaar krijgt een e-mail met daarin de locatie van de telefoon. De locatie wordt gegeven aan de hand van een adres en een kaart.
Resultaat	Als een collega reageert op een bericht van de applicatie en deze geeft aan dat hij weet waar de eigenaar is (bijvoorbeeld bij hem op kantoor) dan wordt deze persoon gevraagd de eigenaar in te lichten en te vragen of hij zich wil melden bij de telefoon (zie Use Case *melden bij telefoon* van gebruiker). Als de eigenaar de e-mail krijgt met de locatie erin, kan de gebruiker op zoek gaan naar zijn telefoon en melden dat hij aanwezig is.
Uitzonderingen	Niemand van de personen die gebeld wordt weet waar de eigenaar is. Op dat moment wordt de status van de eigenaar op “onvindbaar” gezet. Heeft de eigenaar zich binnen een uur nog niet gemeld, dan gaat de applicatie opnieuw opzoek naar zijn eigenaar. De eigenaar is eerder aanwezig dan de telefoon iemand heeft gevonden die weet waar hij is. De eigenaar kan ten allen tijden aangeven dat hij weer bij de telefoon aanwezig is. Dit kan zijn door enkel de telefoon uit stand-by te halen en te ontgrendelen of ook door de applicatie te starten.
Uitbreidingen	–

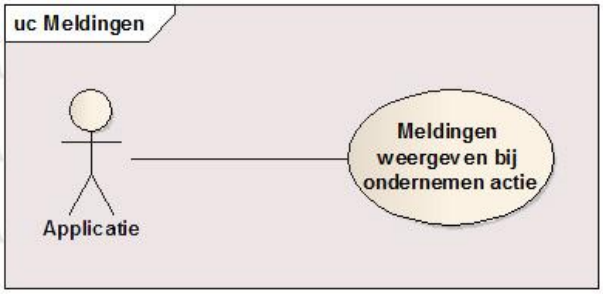
Tabel 13: Use Case Scenario - Eigenaar terugvinden



Figuur 16: Use Case – Templates bouwen

Templates bouwen	
Actoren	Applicatie
Beschrijving	*Template stem opnemen*: Voor het IVR menu is een persoonlijk bericht nodig. Bij de eerste keer gebruik van de applicatie wordt gevraagd aan de gebruiker zijn persoonlijke bericht in te spreken. *Template bericht opzetten*: Voor het bericht dat naar een ander persoon wordt verzonden is een tekst nodig. De applicatie vraagt bij de eerste keer gebruik van de applicatie om de tekst goed te keuren of om een eigen tekst in te voeren. *Template e-mail opzetten*: Voor de e-mail dat naar een ander persoon wordt verzonden is een tekst nodig. De applicatie vraagt bij de eerste keer gebruik van de applicatie om de tekst goed te keuren of om een eigen tekst in te voeren.
Resultaat	Bij het zoeken naar de eigenaar van de telefoon is een vraag ingesproken of opgeschreven door de gebruiker zelf. Hij weet dan zelf ook wat er wordt gevraagd aan andere personen en is het hiermee eens.
Uitzonderingen	–
Uitbreidingen	Het wijzigen van de berichten door middel van het instellingen

Tabel 14: Use Case Scenario - Templates bouwen



Figuur 17: Use Case – Meldingen weergeven bij ondernemen actie

Meldingen weergeven bij ondernemen actie	
Actoren	Applicatie
Beschrijving	Door de applicatie wordt van alle acties die hij onderneemt ook een melding gemaakt
Resultaat	De gebruiker weet altijd waar de applicatie mee bezig is.
Uitzonderingen	–
Uitbreidingen	–

Tabel 15: Meldingen weergeven bij ondernemen actie

Verklarende woordenlijst

Woord, afkorting	Verklaring
FURPS+	Methode om eisen te beschrijven
SMART	Beschrijving van eisen: Specifiek, Meetbaar, Acceptabel, Reëel, Tijdgebonden
Android	Mobiel platform ontwikkeld door Google
ASK, ASK Community Systems	Dochterbedrijf van Almende dat het ASK Platform heeft opgezet
ASK, ASK webinterface	De basis van ASK, hier wordt alles gecombineerd. Dit deel wordt door ASK CS ingesteld voor een klant
ASK Desktop	Een applicatie waar ASK wordt gebruikt op de achtergrond
ASK Platform	Het ASK systeem met alle onderdelen (Communication Coordinator, API, Rules and scenario's, Users and groups, applicatiies en communicatiemiddelen)
IVR Menu	Interactive voice response, een telefonisch keuze menu. Ook wel bekend van klantenservice telefoongesprekken
Agent	Een virtuele persoonlijkheid binnen het ASK platform die één of meerdere verantwoordelijkheden binnen het systeem heeft.

Tabel 16: Verklarende woordenlijst

Bronvermelding

1. Rien Elling, Bas Andeweg, Jaap de Jong, (2005). *Digitaal Handboek Rapportagetechniek 4.2.* Noordhoff Uitgevers (cd-rom)
2. The Eclipse Foundation, (-2011). *OpenUP.* Via http://epf.eclipse.org/wikis/openup/
3. The Eclipse Foundation, (-2011). *Systemwide Requirement Specificaties.*
4. Peter Eeles, (01-07-2004). *What, no supplementary specification?* IBM Software Group
5. ASK-CS, (-2011). *About ASK.* Via http://www.ASK-CS.com
6. T. van As en L. Stellingwerff, (2011). *ASK Reference Guide.* ASK Community Systems
7. T. van As, (2011). *ASK Architecture.* ASK Community Systems
8. T. van As en S. Mulder, (2011). *MyASK Architecture*. ASK Community Systems
9. T. van As, (2011). *MyASK Features, Applications & Roadmap.* ASK Community Systems
10. Sense-OS, (-2011). *About Sense.* Via http://www.Sense-OS.nl
11. S. Mulder, (2011). *CommonSense Flyer*. Sense Observation Systems
12. Mark L. Murphy, (2010). *Android Programming Tutorials.* CommonsWare
13. Mark L. Murphy, (2010). *The Busy Coder's Guide To Android Development.* CommonsWare
14. Mark L. Murphy, (2009-2010). *The Busy Coder's Guide To Advanced Android Development.* CommonsWare
15. Android, (-2011). *Android Developers.* Via http://developer.android.com/guide/index.html
16. Nederlandse taalunie, (-2011). *Literatuurlijsten (algemeen).* Via http://taaladvies.net/taal/advies/tekst/87/#7

Elaboration Fase Rapport

"Onderzoek en ontwikkeling van Android applicatie – Waar is mijn eigenaar?"

Auteur: Marleine van Kampen
Datum: 28 april 2011
Versie: 1.0

Versiebeheer

Versie	Datum	Auteur	Aanpassingen
0.1	18 februari 2011	M. van Kampen	Basis opzet
0.2	22 maart 2011	M. van Kampen	Grammatica, tekstuele aanpassingen
0.3	9 mei 2011	M. van Kampen	Uitbreiding teksten en afronding document
1.0	6 juni 2011	M. van Kampen	Eindversie

Reviewers

Versie	Datum	Auteur
0.1	22 maart 2011	M. van Kampen
0.2	28 april 2011	M. van Kampen
0.3	9 mei 2010	M. van Kampen

Gerelateerde documenten

Naam	Versie	Datum	Auteur
Afstudeerscriptie	1.0	6 juni 2011	M. van Kampen
Inception Fase Rapport	1.0	6 juni 2011	M. van Kampen
Construction Fase Rapport	1.0	6 juni 2011	M. van Kampen
Transition Fase Rapport	1.0	6 juni 2011	M. van Kampen

Distributielijst

Naam	Organisatie	Functie	Reden
S. Mulder	Sense	Technisch begeleider	Proces begeleider
T. van As	ASK	Technisch begeleider	Proces begeleider
J.J. Visser	Haagse Hogeschool	Docent	1e Examinator
D.R. Stikkolorum	Haagse Hogeschool	Docent	2e Examinator

Samenvatting

Marleine van Kampen, student van De Haagse Hogeschool Delft voert gedurende zeventien weken een afstudeeropdracht uit bij ASK Community Systems en Sense Oberservation Systems. De opdracht is genaamd “Waar is mijn eigenaar”?, een Android applicatie die opzoek gaat naar de eigenaar van de telefoon.

Om gestructureerd te programmeren is het handig om, per onderdeel en van de gehele applicatie, ontwerpen op te zetten. Het gaat hierbij om ontwerpen over het verloop van statussen, de ordening van klassen en het verloop van functies. De ontwerpen zijn gemaakt aan de hand van UML 2.0.

Als basis voor alle ontwerpen en de ontwikkeling van de applicatie is een iteratie plan opgesteld. Hierin is per iteratie beschreven welke onderdelen er ontwikkeld worden. Onder de iteraties vallen de elaboration, construction en transition fase, oftewel het ontwerpen, implementeren en testen van de applicatie. Op het moment dat een onderdeel of een versie van de applicatie wordt opgeleverd, wordt het onderdeel getest. Deze tests worden uitgevoerd aan de hand van de ontwerpen en de implementatie.

De ontwerpen van de werking van het systeem zijn opgezet per onderdeel van de applicatie. Hierbij gaat het om de hoofdonderdelen FindAction (het vinden van de eigenaar), de verbinding met de platformen van Sense en ASK en de webinterface.

Naast de ontwerpen van de werking van het systeem zijn er ook ontwerpen gemaakt van de Graphical User Interface (GUI). De GUI is het onderdeel waar de gebruiker direct mee te maken heeft. De belangrijkste eis voor gui is dat deze overzichtelijk en gebruiksvriendelijk moet zijn. De gebruiker moet enkele acties kunnen uitvoeren en moet hiervoor niet te veel stappen te hoeven uitvoeren. Het ontwerp van de GUI daarnaast opgebouwd volgens de Android componenten en de weergave hiervan.

Naast de applicatie op de telefoon is er een simpele webinterface nodig voor het afwezig melden van de eigenaar bij de telefoon. Ook hiervoor is een simpele GUI opgebouwd. Deze heeft dezelfde opbouw als de Android applicatie en gebruiksvriendelijk in een PC browser of op een smartphone.

Door het maken van ontwerpen is de ontwikkeling van de applicatie sneller verlopen. Onderdelen die gebouwd moesten worden zijn van te voren uitgedacht waardoor het enkel om de uitwerking in java ging en niet om de logica tussen klassen en de volgorde van acties.

Bij de ontwerpen is rekening gehouden met de uitbreidbaarheid van de applicatie. De applicatie moet namelijk gebruikt kunnen worden in andere applicaties van Sense en ASK.

Inhoudsopgave

I. INLEIDING....8

1. ITERATIEPLAN....9

1.1 ONTWERP VAN HET SYSTEEM....10
1.2 ITERATIEPLAN....11

2. TOTALE APPLICATIE....12

2.1 KLASSENDIAGRAM TOTAAL....12
2.2 VERSIES VAN OPLEVERING....12
2.3 ACTIES VAN ALLE ACTOREN BINNEN HET SYSTEEM....13
2.3 ACTIVITEITENDIAGRAM APPLICATIE....16
2.4 TOESTANDSDIAGRAM APPLICATIE....18

3. GUI....19

3.1 ONTWERPEN OP PAPIER....19
3.2 STAPPENPLAN GUI....20

4. FINDACTION PACKAGE: ALLE VINDACTIES IN ÉÉN PACKAGE....21

4.1 OVERZICHT SAMENWERKING TUSSEN KLASSES BINNEN DE FINDACTION PACKAGE....21
4.2 THREADS BINNEN HET SYSTEEM....22
4.3 VERLOOP VAN ACTIES BIJ HET HERKENNEN VAN DE EIGENAAR....23
4.4 VERLOOP VAN ACTIES BIJ HET VINDEN VAN DE EIGENAAR....24
4.6 FOUNDOWNER....25

5. VERBINDING MET COMMONSENSE....26

5.1 DE INFORMATIE DIE VERZONDEN MOET WORDEN NAAR COMMONSENSE....26
5.2 DE INFORMATIE DIE NIEUW IS IN COMMONSENSE....27
5.3 SEQUENTIE DIAGRAM VERBINDINGSOPZET....28

6. VERBINDING MET ASK....29

6.1 INLEIDING....29
6.2 WERKING VAN ASK....29
6.3 INLOGGEN....30
6.4 WEBSERVICES....30

7. WEBINTERFACE....32

7.1 COMMUNICATIE MET DE WEBINTERFACE EN COMMONSENSE....32
7.2 GUI VAN DE WEBINTERFACE: EEN SIMPEL ONTWERP....34

CONCLUSIE....35

VERKLARENDE WOORDENLIJST....36

BRONVERMELDING....37

BIJLAGEN....38

I. Inleiding

I.I Aanleiding

Door Marleine van Kampen, afstudeerder van De Haagse Hogeschool Delft wordt gedurende zeventien weken gewerkt aan het project “Waar is mijn eigenaar?”. Het project is een onderzoek naar en ontwikkeling van een Android applicatie bovenop de bestaande platformen van de dochterbedrijven, Sense en ASK, van het Rotterdamse onderzoeksinstituut Almende.

I.II Doel

Het doel van het elaboration fase rapport is het bespreken van alle resultaten uit de elaboration fase, hierbij gaat het voornamelijk over de ontwerpen van de applicatie.

I.III Scope

Door de verschillende iteraties waar de ontwikkelmethode uit bestaat is de scope van het document iets anders dan een ontwerpdocument waarin eerst alle ontwerpen worden gemaakt en er dan pas geïmplementeerd wordt.

Het document is opgebouwd uit ontwerpen per onderdeel van de applicatie. Tijdens het schrijven van het elaboration fase rapport zijn de implementatie en testresultaten verwerkt in het construction fase rapport en het transistion fase rapport.

Het elaboration fase rapport bevat naast de ontwerpen ook het iteratieplan, met daarin de werkzaamheden in de verschillende iteraties.

1. Iteratieplan

Voor het ontwikkelen van de "*Find My Owner*" applicatie is gekozen om de iteratieve methode OpenUP te gebruiken. Deze methode verdeelt de verschillende te ontwikkelen onderdelen van het systeem in zogenaamde iteraties. Om een goed overzicht te krijgen van de iteraties en wanneer de iteraties worden uitgevoerd, wordt er een iteratieplan opgezet.

Om goed te weten welke onderdelen er in het systeem zitten is er aan de hand van de Use Case Diagrammen een Klassendiagram gemaakt. In de afbeelding op de volgende pagina is het volledige klassendiagram te zien.

Bij deze eerste versie van het klassendiagram(figuur 1[1]) is rekening gehouden met de hoofdfuncties die het systeem moet uitvoeren. De specifieke functies worden in de eerste iteratie toegevoegd aan het klassendiagram.

Daarnaast is er in de eerste versie rekening gehouden met de mogelijke design patterns. Simpele design patterns zijn al gebruikt zoals Abstract Factory. Design patterns waar aan gedacht kan worden om toe te passen in de uiteindelijke versie van het klassendiagram zijn:

- Factory Method
- Abstract Factory
- Singleton (Found, Recognize en Find)
- Decorator (MEDIA)
- Bridge (MEDIA)
- Model View Controller

Toelichting over welke design patterns er uiteindelijk gebruikt zijn is te vinden in het construction fase rapport.

1 *Figuur is uit vergroot te vinden in bijlage A*

1.1 Ontwerp van het systeem

OWN

OWN Stelt de basisklasse van het systeem voor, ook wel de *main*. Vanuit OWN worden de GUI en de achterliggende code aangeroepen.

GUI

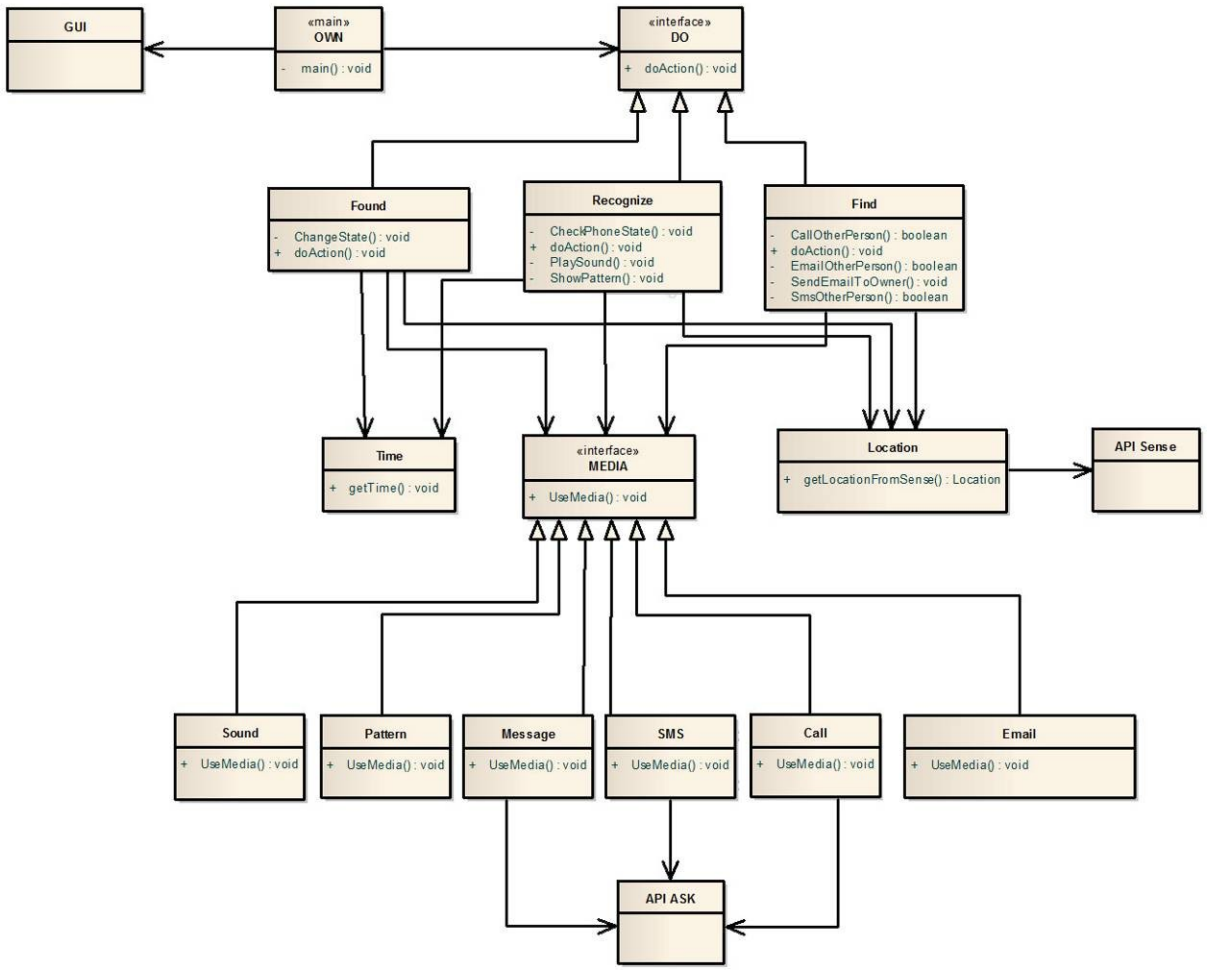
De GUI is de grafische user interface van het systeem. Dit is het deel wat de gebruiker ziet.

DO

Het uitvoerende deel van het systeem. Hier zijn de hoofdfunctionaliteiten in verschillende klassen verdeeld. Found is de klasse die wordt gebruikt op het moment dat de eigenaar is gevonden. Recognize wordt vooral als eerste gebruikt om te kijken of de eigenaar in de buurt van de telefoon is. Find is de klasse die de eigenaar gaat zoeken.

MEDIA

Voor het verzenden van bijvoorbeeld sms is een aparte klasse sms nodig. Als deze klasses voor contact opnemen vallen onder de klasse MEDIA.



Figuur 1: Klassediagram versie 1

1.2 Iteratieplan

In onderstaande afbeelding is een onderdeel van het iteratieplan te zien, in bijlage B is het volledige iteratieplan te vinden.

Naam/ uitleg	Prioriteit	Verwachte grootte	Status	Iteratie	Verantwoordelijk	Maximaal aantal uren	Uren aan gewerkt
Iteratie 1						***26***	
Klassendiagram totaal	*1*	*2*	*Open*	*1*	*Marleine*	*6*	
Activiteitendiagrammen totaal / Toestandsdiagram totaal	*2*	*1*	*Open*	*1*	*Marleine*	*5*	
Bouwen FindMyOwner Android applicatie	*3*	*4*	*Open*	*1*	*Marleine*	*15*	
Iteratie 2						***36***	
Ontwerpen GUI	*1*	*1*	*Open*	*2*	*Marleine*	*3*	
Stappenplan GUI	*2*	*1*	*Open*	*2*	*Marleine*	*3*	
Bouwen GUI	*3*	*4*	*Open*	*2*	*Marleine*	*15*	
Testplan opzetten	*4*	*4*	*Open*	*2*	*Marleine*	*15*	

Tabel 1: Iteratieplan - Iteratie 1 en 2

Het iteratieplan bestaat uit:

- De verschillende onderdelen die ontwikkeld[2] moeten worden. Voor het gemak zijn de onderdelen al per iteratie verdeeld. Uiteindelijk kan er per iteratie ook nog een iteratieplan worden opgezet.
- De prioriteit[3] die elke onderdeel heeft. Naast de verdeling per iteratie is de prioriteitverdeling ook al gesorteerd. In bovenstaande afbeelding is niet te zien dat meerdere onderdelen dezelfde prioriteit hebben. In de bijlage is te zien dat bijvoorbeeld het ontwikkelen van Find, Found en Recognize dezelfde prioriteit hebben.
- Verwachtte grootte / Maximaal aantal uren. Deze twee waardes zijn met elkaar in verhouding. Naarmate het onderdeel groter is wordt het aantal uren dat er maximaal aan besteed mag worden ook groter.
- In de loop van het ontwikkelproces wordt het iteratieplan bijgewerkt. Dan is er te zien dat de status op afgerond wordt gezet en dat het aantal uur aan gewerkt wordt ingevuld. Daarnaast kunnen onderdelen in kortere / langere tijd worden ontwikkeld, hier wordt in de loop van de tijd de planning op aangepast.
- De lichtgrijze onderdelen(rechter kolommen) in de tabel zijn onderdelen niet verplicht in het iteratieplan ingevuld hoeven te worden. Voor het gemak zijn ze in dit plan wel ingevuld.

2 *Onder ontwikkelen wordt de volledige cyclus van analyse, ontwerp, implementatie, testen en oplevering verstaan*
3 *Met één als hoogste prioriteit*

2. Totale applicatie

2.1 Klassendiagram totaal

Omdat er vooraf aan het project bij de student nog niet voldoende kennis is van de opbouw van Android/Java, is ervoor gekozen geen uitgebreid klassendiagram op te zetten.

Aan de hand van het domein klassendiagram is bepaald welke onderdelen het systeem moet bevatten en welke hoofdfunctionaliteiten hier een rol bij spelen.

Het systeem wordt in verschillende versies opgeleverd. Bij elke versie wordt een klassendiagram gemaakt wat de opbouw van de betreffende versie weergeeft. Hiermee wordt een zo reëel mogelijk beeld gegeven van de applicatie.

In de volgende hoofdstukken en paragrafen worden ontwerpen van onderdelen van de applicatie beschreven. Deze bevatten wel delen van het klassendiagram, omdat deze later in de ontwikkeling van de applicatie gemaakt zijn en er meer kennis over Android Java is opgedaan.

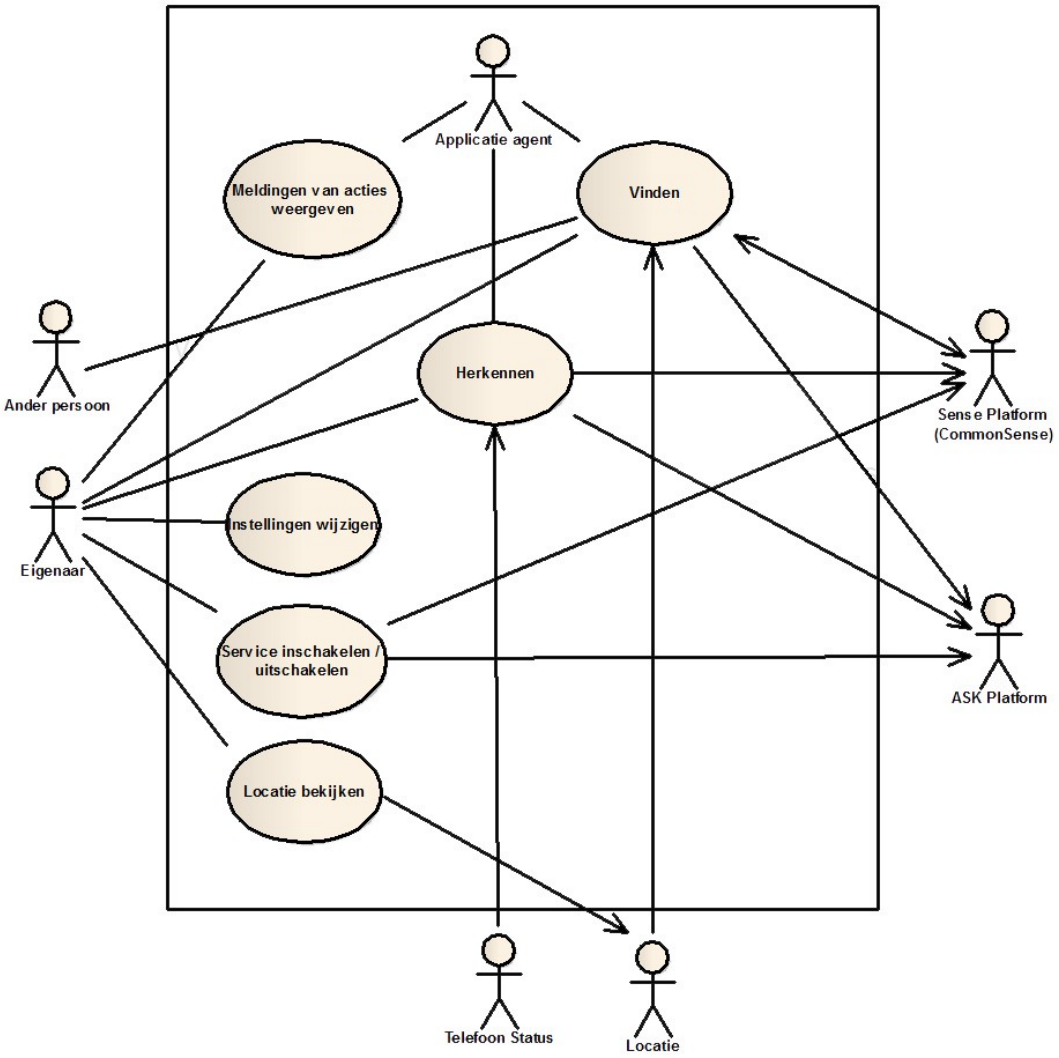
2.2 Versies van oplevering

De applicatie wordt in verschillende versies opgeleverd. Los van iteraties zijn er een aantal hoofdonderdelen die per versie worden toegevoegd. In onderstaande tabel is te zien wat er per versie aan de applicatie wordt toegevoegd en wat er dan in die versie dus getest moet worden.

Versie	Toevoegingen	Oplevering
1	– Opzet van de applicatie – Herkennen van de eigenaar	Week 6
2	– Communicatie met CommonSense – Communicatie met ASK – Eerste vindactie via ASK	Week 10
3	– Alle vindacties via ASK	Week 12
4	– Gebruik van webinterface – Resultaten uit gebruikerstesten	Week 14

Tabel 2: Versies van oplevering

2.3 Acties van alle actoren binnen het systeem



Figuur 2: Use Case - Totale applicatie

In figuur 2 is de Use Case te zien met daarin de verschillende actoren van het systeem. In onderstaande tabellen zijn de Use Case scenario's te vinden. Deze scenario's geven meer uitleg over elke Use Case.

Meldingen van acties weergeven	
Actoren	Applicatie agent, Eigenaar
Beschrijving	Door de applicatie wordt van alle acties die hij onderneemt ook een melding gemaakt. Deze verschijnt in de status bar en is te bekijken door de eigenaar.
Resultaat	De eigenaar weet altijd waar de applicatie mee bezig is en kan door de melding aan te klikken de actie annuleren.
Uitzonderingen	–
Uitbreidingen	–

Tabel 3: Use Case Scenario - Meldingen van acties weergeven

Herkennen	
Actoren	Applicatie agent, Eigenaar, Telefoon status, Sense Platform(CommonSense)
Beschrijving	Op het moment dat de telefoon langere tijd niet meer gebruikt wordt (stand-by stand) gaat de applicatie actie ondernemen. Hierbij wordt de toestand van de applicatie op herkennen gezet (in de app) en wordt deze toestand doorgezonden naar Sense. De eigenaar moet reageren op de verschillende acties zodat de applicatie weet of de eigenaar aanwezig is of niet.
Resultaat	Als de eigenaar de telefoon heeft gebruikt, geeft hij aan dat hij wel aanwezig is. Als de eigenaar niet reageert op de signalen is deze hoogstwaarschijnlijk niet aanwezig, dan gaat de toestand over naar vinden en wordt ook deze toestand doorgegeven aan Sense.
Uitzonderingen	–
Uitbreidingen	–

Tabel 4: Use Case Scenario - Herkennen

Vinden	
Actoren	Applicatie agent, Eigenaar, Sense Platform(CommonSense), ASK Platform, Ander persoon, Locatie
Beschrijving	Als de eigenaar niet reageert op de herkenningsacties gaat de applicatie verdere acties ondernemen. Hij zet zijn status op vinden en stuurt dit door aan CommonSense. Één van de acties is het contact opnemen (via ASK Platform) met een ander persoon. Hij geeft aan de persoon de locatie van de telefoon door en vraagt aan de persoon of hij weet waar de eigenaar is.
Resultaat	Als een vindactie de locatie van de eigenaar oplevert wordt de status op beschikbaar gezet. Mocht de eigenaar via de webinterface aangeven dat de telefoon niet meer hoeft te zoeken maar dat hij op dat moment niet in de buurt van de telefoon is, wordt de status hierop aangepast. Als de vindacties geen geven, wordt de status op niet beschikbaar gezet. Een uur later wordt er opnieuw geprobeerd om de eigenaar te vinden.
Uitzonderingen	–
Uitbreidingen	–

Tabel 5: Use Case Scenario - Vinden

Instellingen wijzigen	
Actoren	Eigenaar
Beschrijving	De gebruiker kan verschillende instellingen wijzigen wat betreft de applicatie
Resultaat	De nieuwe instellingen worden direct gebruikt
Uitzonderingen	–
Uitbreidingen	–

Tabel 6: Use Case Scenario - Instellingen wijzigen

Service inschakelen / uitschakelen	
Actoren	Gebruiker, ASK Platform, Sense Platform (CommonSense)
Beschrijving	De gebruiker kan ervoor kiezen of hij de applicatie wel of niet gebruikt.
Resultaat	Bij het inschakelen en uitschakelen wordt er contact gelegd met het ASK Platform en het Sense platform voor inloggen en uitloggen.
Uitzonderingen	–
Uitbreidingen	–

Tabel 7: Use Case Scenario - Service inschakelen / uitschakelen

Locatie bekijken	
Actoren	Gebruiker
Beschrijving	De locatie van de telefoon wordt bijgehouden door de applicatie. De eigenaar kan deze bekijken via de applicatie of via de webinterface.
Resultaat	De eigenaar weet de locatie van de telefoon.
Uitzonderingen	–
Uitbreidingen	–

Tabel 8: Use Case Scenario - Locatie bekijken

2.3 Activiteitendiagram applicatie

In figuur 3 is het activiteitendiagram te zien van de totale applicatie.

Als eerste wordt er gecontroleerd of de eigenaar aanwezig is door te kijken of de telefoon in stand-by stand is. Als deze niet in stand-by stand is, is de eigenaar hem aan gebruiken en kan de status op *Happy*.

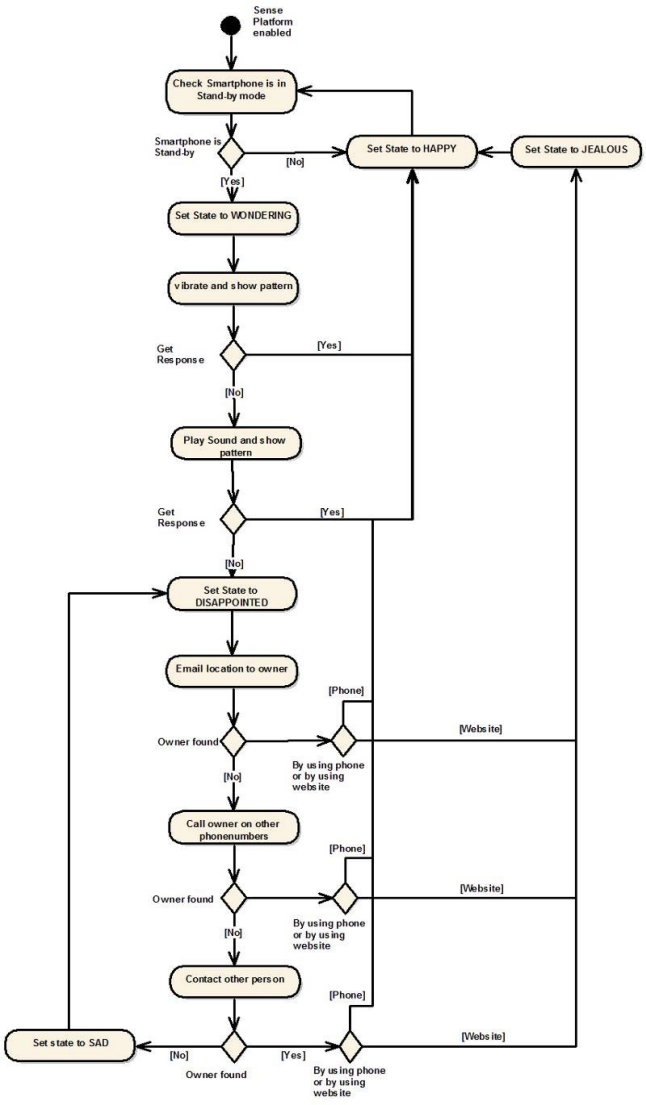
Als de eigenaar niet met zijn telefoon bezig is, gaat de applicatie kijken of de eigenaar dan misschien wel in de buurt (van de telefoon) is door de telefoon te laten trillen en de eigenaar een patroon te laten tekenen.

Mocht de eigenaar hier niet op reageren laat de telefoon ook nog een geluid laten horen.

Reageert de eigenaar echt niet, dan gaat de applicatie opzoek naar de eigenaar. Als eerst wordt de locatie van de telefoon naar de eigenaar gestuurd via e-mail. De eigenaar kan dan niet in de buurt zijn van de telefoon maar kan dan via een webpagina aangeven dat hij weet dat hij niet bij zijn telefoon is. De telefoon gaat dan niet verder met zoeken, maar wacht totdat de eigenaar weer aanwezig is.

Als de eigenaar niet op de e-mail reageert gaat hij de eigenaar bellen via andere nummers die hij weet (bijvoorbeeld thuis of werk telefoonnummer).

Als de eigenaar zelf niet reageert gaat de applicatie opzoek naar personen bij hem in de buurt die de eigenaar kunnen aansporen de telefoon te gebruiken of zich te melden bij de telefoon.

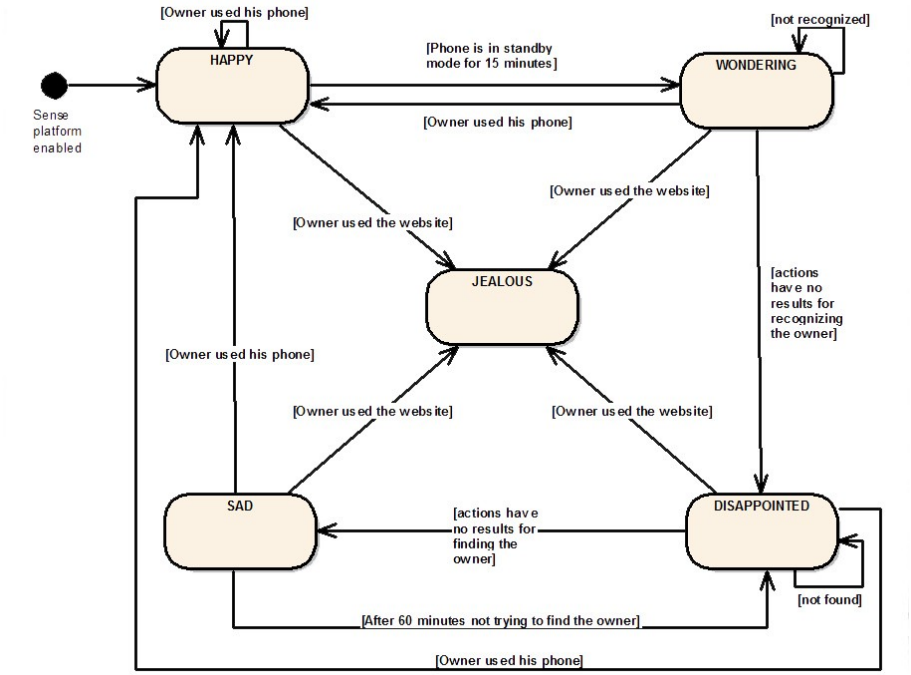


Figuur 3: Activiteitendiagram - Totale applicatie

Als de eigenaar wel wordt gevonden, moet hij zich aanmelden bij de telefoon. Het aanmelden kan door de telefoon zelf te gebruiken of door op een website aan te geven dat hij weet dat hij niet bij zijn telefoon in de buurt is. Bij de laatste optie zal de telefoon niet meer gaan zoeken naar de eigenaar totdat deze hem weer gebruikt.

Mocht de applicatie de eigenaar echt niet vinden dan gaat deze uur later weer zoeken naar de eigenaar.

2.4 Toestandsdiagram applicatie



Figuur 4: Toestandsdiagram - Applicatie

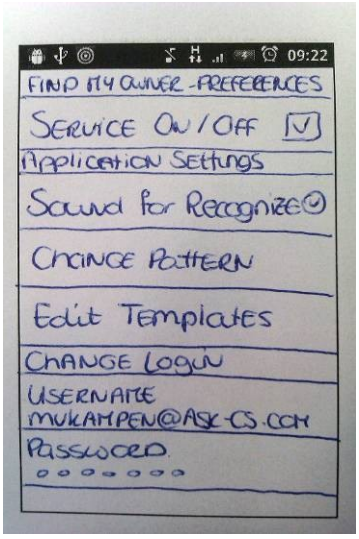
	Eigenaar gebruikt / reageert op zijn telefoon	Eigenaar gebruikt niet / reageert niet op zijn telefoon	Eigenaar meldt zich via website
Happy De applicatie controleert elke 15 minuten op aanwezigheid van eigenaar.	*Happy*	*Wondering*	*Jealous*
Wondering De applicatie voert verschillende herkenningsacties uit voor het herkennen van de eigenaar	*Happy*	*Disappointed*	*Jealous*
Disappointed De applicatie voert verschillende vindacties uit voor het vinden van de eigenaar	*Happy*	*Sad*	*Jealous*
Sad De applicatie komt in wachtstand omdat hij de eigenaar niet kan vinden. Na 60 minuten gaat de applicatie weer opzoek naar de eigenaar	*Happy*	*Finding*	*Jealous*
Jealous De applicatie weet dat de eigenaar even niet in de buurt is, de eigenaar geeft dit aan via de webinterface	*Happy*	*Can't happen*	*Jealous*

Tabel 9: State Transistion Table - Applicatie

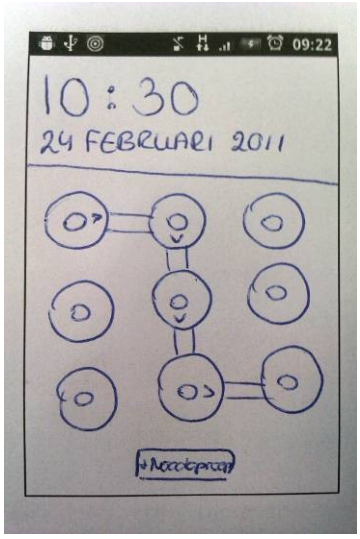
3. GUI

3.1 Ontwerpen op papier

Figuur 5: GUI - Hoofdscherm

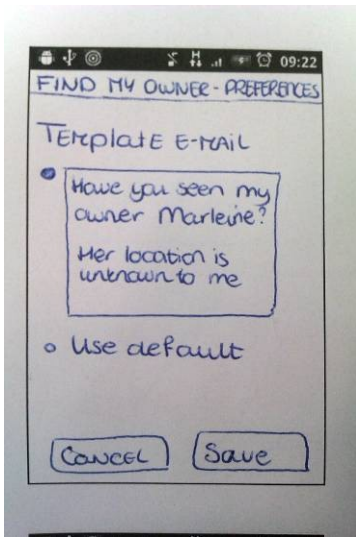


Figuur 6: GUI - Preferences



Figuur 7: GUI – Patroon voor herkenning

Figuur 8: GUI - Status



Figuur 9: GUI - Aanpassen e-mail template

In bovenstaande figuren zijn de schetsen van de GUI te zien. Het idee van de GUI is dat de gebruiker er weinig mee te maken heeft omdat de applicatie de vind acties op de achtergrond uitvoert op het moment dat de eigenaar de telefoon niet gebruikt. Hierdoor hoeft de GUI ook niet veel informatie te bevatten

In figuur 5 is het scherm te zien wat de gebruiker als eerste ziet. Hier kan informatie komen over de status van de eigenaar en de locatie. Mogelijk ook informatie over wanneer de applicatie opzoek is geweest naar de eigenaar.

In figuur 6 is het instellingen menu te zien. De gebruiker kan hier simpele instellingen voor de applicatie wijzigen

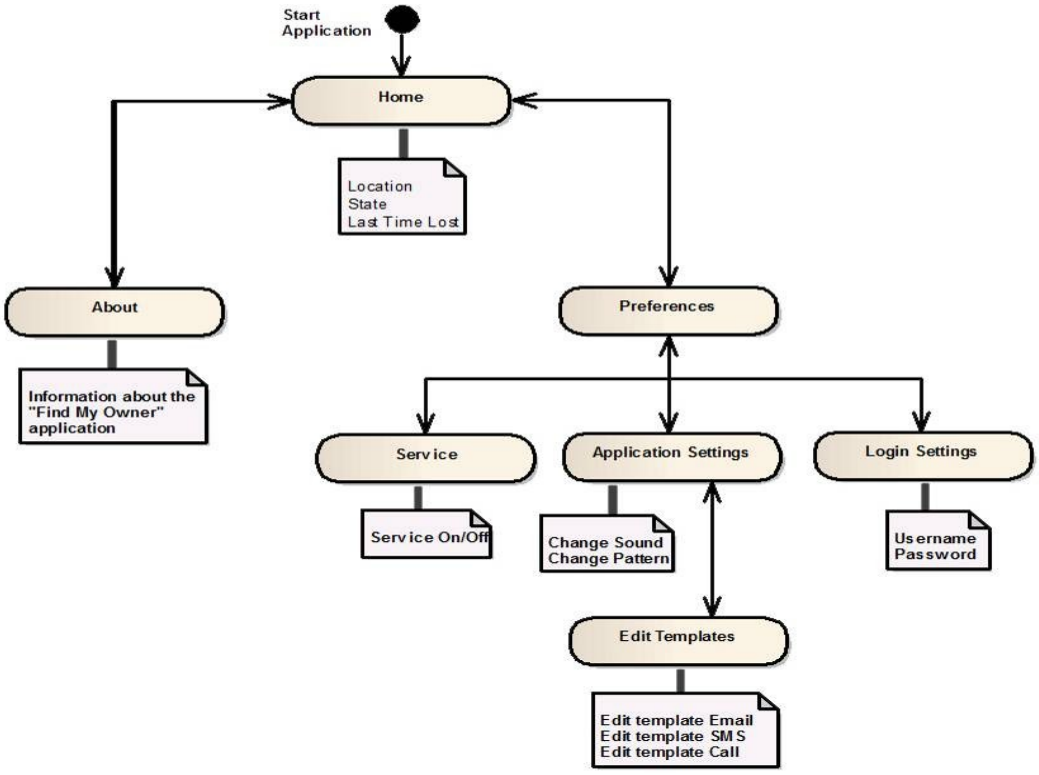
In figuur 7 Is het tekenen van een patroon te zien voor het herkennen van de eigenaar. De eigenaar stelt zelf zijn patroon in. Het patroon wordt gebruikt in combinatie met een trilling. Als er enkel gebruik gemaakt wordt van een trilling en een bericht waar de eigenaar op moet reageren, is de kan groter dat een ander persoon te telefoon aanzet en hem “tevreden stelt”.

In figuur 8 is de *notification bar* te zien. Hier verschijnen meldingen van applicaties. De *Find My Owner* applicatie geeft aan dat hij actief is. En geeft mogelijke acties aan zoals dat hij bezig is met het bellen naar andere personen. Om de applicatie te laten stoppen met zijn actie is het mogelijk om op de notificatie te klikken. De acties worden dan door de applicatie gestopt.

In figuur 9 is een voorbeeld van een instelling te zien. Hierin kan de gebruiker de template voor de e-mail die naar andere personen wordt gestuurd aanpassen. Na aanpassing wordt deze template ook bij ASK opgeslagen.

3.2 Stappenplan GUI

Het idee van de GUI is dat de gebruiker in verschillende schermen terecht komt. In onderstaand figuur is het *stappenplan* van de GUI te zien. Het is het idee van een activiteitendiagram maar meer gebaseerd op de schermen waar de gebruiker mee te maken heeft.



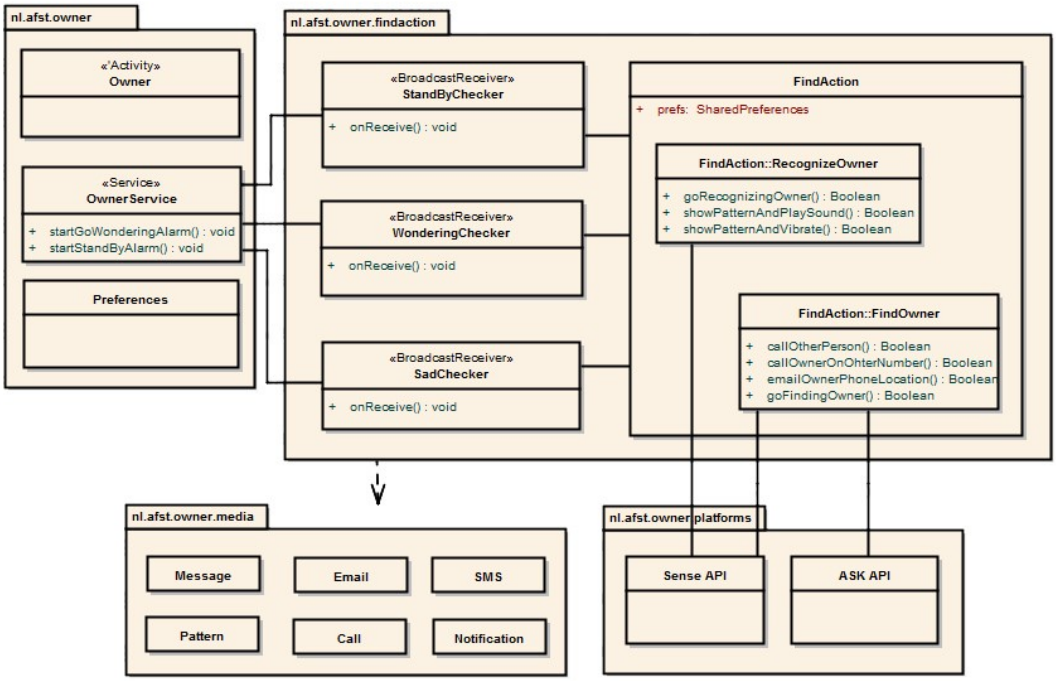
Figuur 10: Stappenplan GUI

4. FindAction package: Alle vindacties in één package

Het uiteindelijke doel is dat de applicatie acties gaat uitvoeren om de eigenaar te herkennen en te vinden. Om deze acties uit te voeren is een apart package opgezet in de applicatie, namelijk de *FindAction package.* FindAction moet uiteindelijk de verschillende stappen doorlopen voor het herkennen en vinden van de eigenaar.

Aan de hand van een sequentie diagram en een klassendiagram wordt weergegeven hoe de klassen binnen de package samenwerken.

4.1 Overzicht samenwerking tussen klasses binnen de FindAction package



Figuur 11: Klassendiagram - FindAction

In bovenstaand diagram (figuur 11) is het klassediagram van *FindAction* te zien. Het diagram is gericht op de samenwerking tussen de *findaction* package en de rest van de packages. In het volledige klassendiagram is ook de samenwerking tussen de andere packages te zien.

Het *findaction* package bevat de klasses voor het herkennen en vinden van de eigenaar. Op het moment dat de eigenaar zijn scherm uit zet gaat de telefoon in stand-by stand en gaat het StandByChecker lopen. Als na 15 minuten de telefoon nog niet gebruikt is door de eigenaar wordt de *findAction* klasse gestart. Deze klasse voert de acties uit om de eigenaar te herkennen of te vinden.

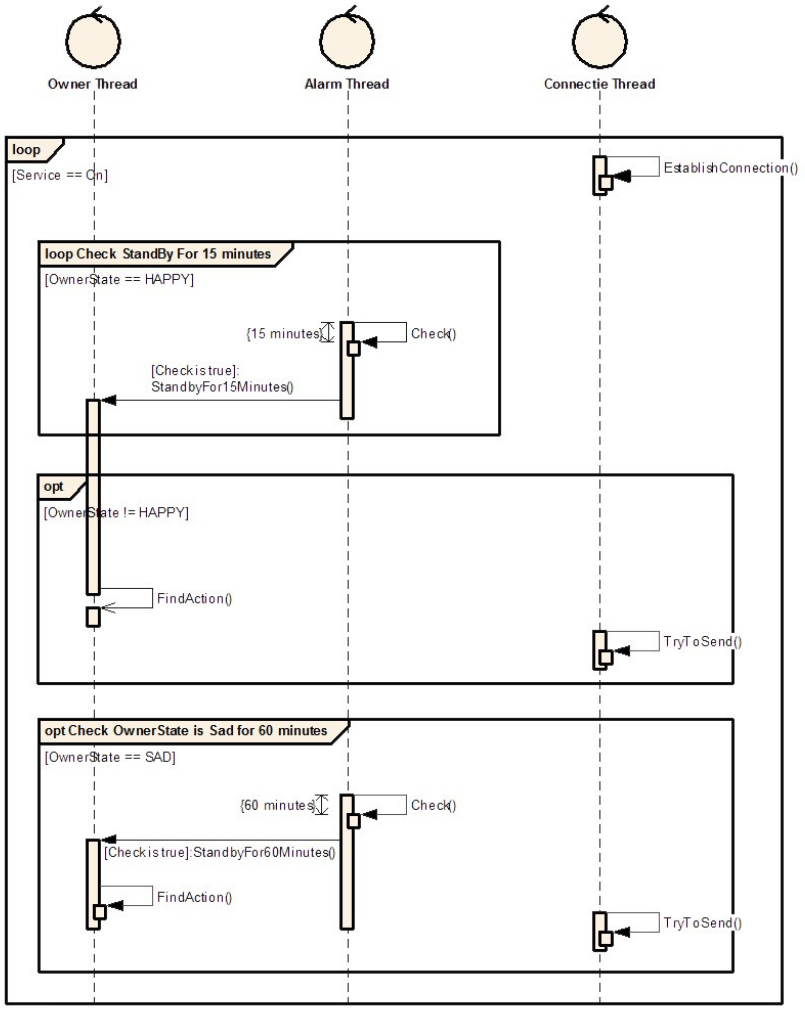
Het *media* package wordt gebruikt voor alle vormen van output. Zo wordt er bij herkenning een patroon laten zien voor het ontgrendelen en een notificatie gegeven van de herkenningsactie.

Het *platform* package wordt gebruikt voor de verbinding met de Sense en ASK platformen.

4.2 Threads binnen het systeem

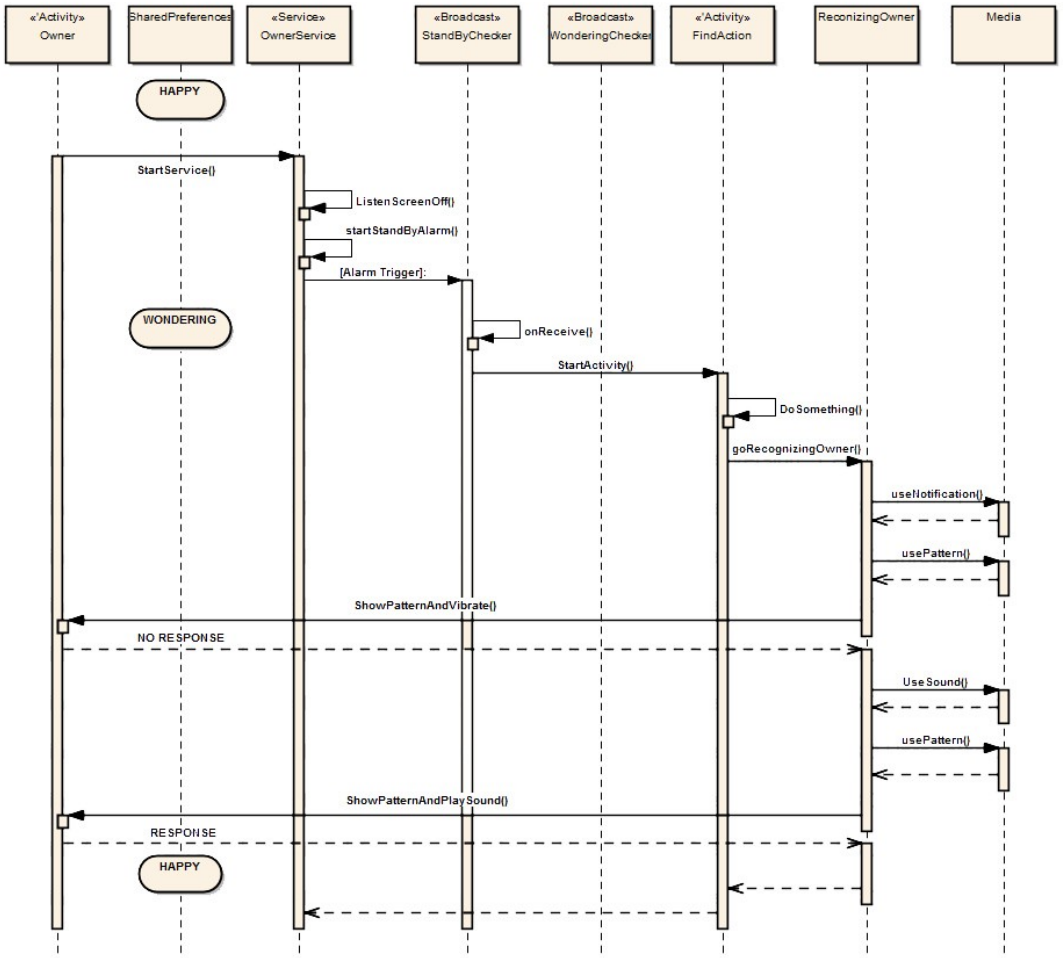
In het systeem worden drie *threads* gebruikt om mogelijke interrupts direct te verwerken.

- De *hoofdthread* zorgt voor alle acties binnen het systeem. Oftewel als de eigenaar afwezig is, is de thread bezig met het vinden van de eigenaar. Op het moment dat de eigenaar bezig is met zijn telefoon / met de applicatie, zorgt de thread voor de goede verwerking hiervan.
- De alarm*thread* houdt constant de tijd bij. Hierbij wordt gekeken hoelang de telefoon in stand-by stand is. Als deze vijftien minuten in stand-by stand is, geeft deze een trigger. De *hoofdthread* gaat dan acties ondernemen om de eigenaar te vinden. Op het moment dat de eigenaar de telefoon weer gaat gebruiker moet de timer weer op nul gezet worden.
- De *connectiethread* zorgt voor de communicatie met ASK en Sense. Deze controleert periodiek of de connectie met ASK en met Sense nog bestaat. Ook zorgt deze thread voor het verzenden van data naar Sense en ASK. De thread coördineert wanneer pakketten worden verzonden.



Figuur 12: Sequentie Diagram - Samenwerking Threads

4.3 Verloop van acties bij het herkennen van de eigenaar



Figuur 13: Sequentie diagram – Herkennen eigenaar

In figuur 13 is het sequentie diagram van de herkenningsactie te zien.

Bij het inschakelen van de *Service,* ook wel het inschakelen van het zoeken naar de eigenaar, wordt de *OwnerService* gestart. Deze *service* wordt gebruikt voor het starten van de verschillende alarmen en het inloggen bij Sense en ASK.

De *Service* voert enkel een *BroadcastReceiver* uit voor het luisteren naar het gebruik van aan/uit knop. Op het moment dat de knop wordt ingedrukt voor het uitschakelen van het scherm gaat de *StandByChecker* lopen. Dit alarm zorgt dat de *FindAction* klasse aangeroepen wordt als de telefoon gedurende vijftien minuten niet in stand-by stand is geweest.

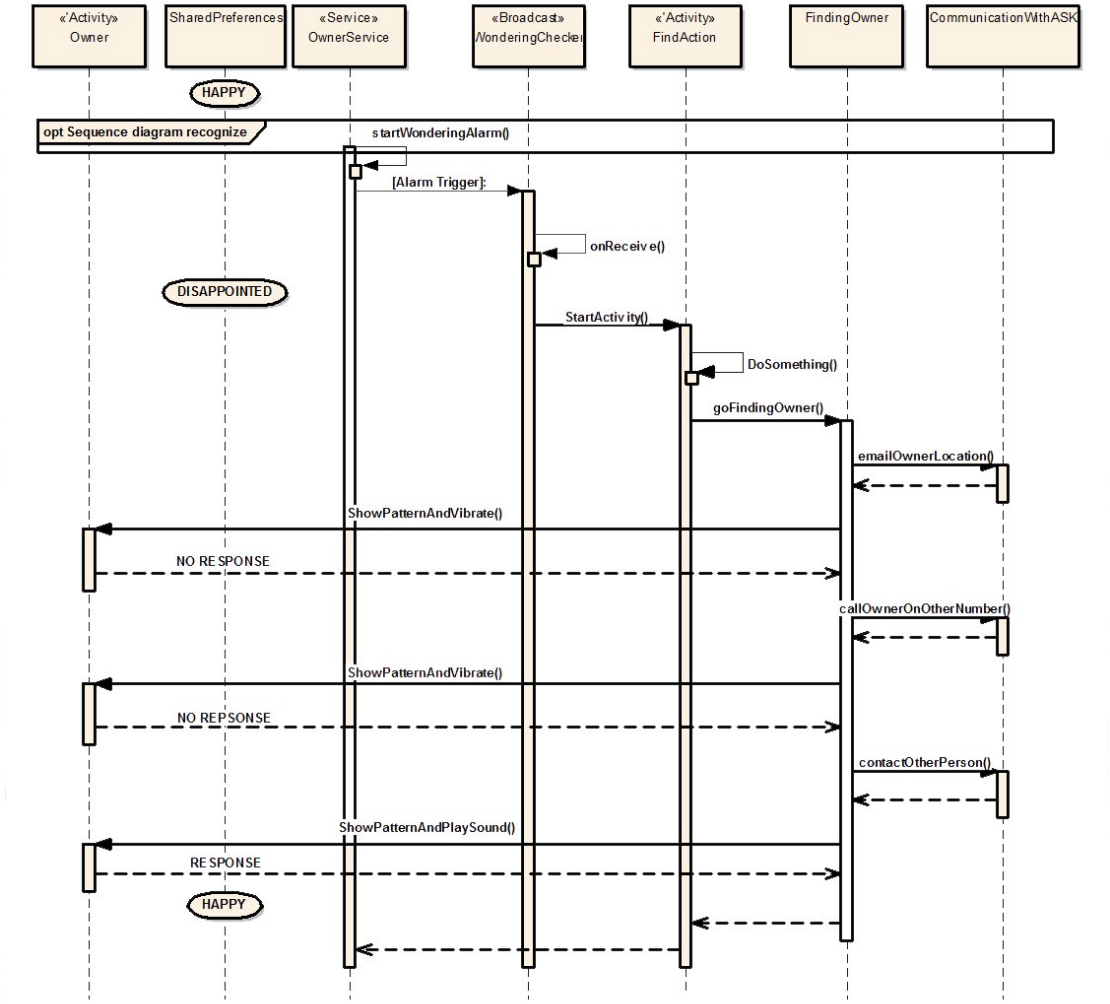
Op het moment dat de eigenaar zijn telefoon vijftien minuten niet heeft gebruikt, komt deze in de W*ondering* toestand. Afhankelijk hiervan wordt de herkenningsactie uitgevoerd.

De gebruiker krijgt een notificatie in de statusbalk. De gebruiker moet op deze notificatie reageren door een patroon in te voeren. Op het moment dat de persoon niet reageert wordt de notificatie nogmaals gegeven maar dan mét geluid.

Op het moment dat de eigenaar het juiste patroon invoert wordt de status van de applicatie weer veranderd in *Happy*.

4.4 Verloop van acties bij het vinden van de eigenaar

Op het moment dat de eigenaar gedurende een kwartier niet reageert op de herkenningsacties, wordt de status veranderd in *Disappointed* en worden de zoekacties uitgevoerd. De verloop van de vindacties is te zien in de onderstaande afbeelding:



Figuur 14: Sequentie diagram - Verloop vindacties

Bovenin het diagram wordt verwezen naar het sequentie diagram van herkennen. Enkel reageert in dit geval de eigenaar niet op de herkenningsacties en wordt de toestand op *Disappointed* gezet. Doordat de toestand Disappointed is, gaat de FindAction klasse vindacties uitvoeren. Als eerst moet er een e-mail gestuurd worden naar de eigenaar met daarin zijn locatie.

Als de eigenaar hier niet op reageert wordt hij gebeld op een ander nummer, bijvoorbeeld werk of thuis telefoon.

Elke keer wordt er bij een vindactie weer een patroon weergegeven. Hierdoor kan de eigenaar op zijn telefoon aangeven dat hij weer aanwezig is. Door het patroon wordt de status weer op Happy gezet.

Op het moment dat de eigenaar niet reageert op het telefoongesprek naar een ander nummer van hem, wordt er contact opgenomen met een van de contactpersonen van de eigenaar.

4.6 FoundOwner

In het klassendiagram (figuur 11) maar ook in het sequentie diagram(figuur 13) is te zien dat de klasse *FoundOwner* is weggelaten. Origineel was deze klasse bedacht als klasse die ervoor zorgde dat de applicatie weet dat de eigenaar is gevonden.

De keuze is uiteindelijk gevallen om de toestand van de applicatie in *SharedPreferences* te integreren. Hierdoor kan de toestand vanuit elke klasse benaderd en aangepast worden. Ook kan de opdracht worden gegeven om op het moment dat de toestand veranderd, de toestand door te sturen naar Sense.

De klasse *FoundOwner* heeft verder geen toepassing en is daaromweggelaten uit de verdere applicatie.

5. Verbinding met CommonSense

CommonSense wordt gebruikt voor het verzamelen van alle sensordata van de MyriaNode en de Smartphone[4]. Uit de sensordata worden in CommonSense conclusies getrokken over wat de eigenaar voor activiteiten heeft uitgevoerd. Zo kan er uit verschillende sensoren bepaald worden dat de eigenaar aan het slapen was of dat hij aan het fietsen was.

Om deze data goed up to date te houden is het handig om te weten wanneer de eigenaar daadwerkelijk in de buurt van de telefoon is geweest. Aan de hand van de *Find My Owner* applicatie wordt herkend of de eigenaar in de buurt is en anders wordt de eigenaar zo snel mogelijk gezocht.

Voor CommonSense is het van belang dat de alle data juist is. Er moet in CommonSense een lijst met alle toestanden die de applicatie heeft verzameld worden opgeslagen. Het kan voorkomen dat de toestand niet juist is met de situatie die de eigenaar op dat moment had. Hiervoor moet het uiteindelijk mogelijk zijn voor de gebruiker om deze waarde aan te passen.

Vanuit de *Find My Owner* applicatie moet verbinding gemaakt worden met CommonSense en moeten er verschillende pakketjes verzonden worden met data. Welke informatie dit is en hoe deze informatie verzonden wordt, is uitgelegd in dit hoofdstuk.

5.1 De informatie die verzonden moet worden naar CommonSense

Voordat een pakket überhaupt verzonden kan worden naar CommonSense moet de gebruiker bekend zijn bij Sense. Hiervoor moet de gebruiker zich registreren bij Sense. Omdat de applicatie als voorwaarde stelt dat ook de Sense applicatie op de telefoon draait is het niet van toepassing dat vanuit de *Find My Owner* applicatie een inlogsysteem wordt opgezet. Een apart inlog systeem is namelijk uitgebreid en zal precies hetzelfde werken als in de Sense applicatie . Binnen Android is het mogelijk om componenten van andere applicaties aan te roepen, zoals in dit geval het inloggen bij CommonSense van de Sense applicatie.

Op het moment dat een pakket naar Sense wordt gestuurd wordt er gebruik gemaakt van de *MessageHandler* van de Sense applicatie. Dit is een *receiver* die alle *Intents* “opvangt” die een actie wat betreft verzenden bevatten. In de *Find My Owner* applicatie wordt dus ook geen klasse opgezet die de volledige verbinding met CommonSense opzet. Dit gebeurt allemaal via de Sense Applicatie. Aan de *MessageHandler* hoeven geen inloggegevens meegegeven te worden, omdat de Sense Applicatie de inloggegevens al heeft.

De toestand in van de applicatie wordt voor CommonSense als een Sensor gezien. De applicatie geeft een bepaalde waarde af over zijn toestand. De waarde kan gezien worden als een sensor. Hierdoor hoeven er geen speciale constructies binnen CommonSense opgezet te worden.

Binnen CommonSense is het mogelijk om sensordata weer te geven in een lijngrafiek of een tabel. In de toekomst komt hier nog een blockdiagram bij wat voor bijvoorbeeld de toestand van de applicatie duidelijker weergeeft welke statussen er op elk tijdstip waren. De bestaande weergaves zijn voldoende voor het project.
De informatie die aan de *MessageHandler Intent* meegegeven moet worden is weergegeven in tabel 9.

4 *Uitleg over het gebruik van de smartphone en de MyriaNode is te vinden in het inception fase rapport*

Data	Toelichting
`MsgHandler.ACTION_NEW_MSG`	Geeft aan dat het om een MessageHandler Intent gaat. Hierdoor wordt hij opgevangen door de MessageHandler van Sense
`MsgHandler.KEY_SENSOR_NAME`	De naam van de Sensor. Voor deze naam moet een standaard komen ander worden er allemaal verschillende sensoren aangemaakt in CommonSense met vergelijkbare namen. De naam komt hoogstwaarschijnlijk neer op "*Find My Owner*"
`MsgHandler.KEY_VALUE`	De waarde van de sensor. In het geval van de applicatie, de toestand van de applicatie
`MsgHandler.KEY_DATA_TYPE`	Het soort data dat verzonden wordt. Waardes kunnen zijn: Float, Bool, String en JSON
`MsgHandler.KEY_TIMESTAMP`	Het tijdstip waarop de data wordt verzonden

Tabel 10: Gegevens naar CommonSense in MessageHandler

5.2 De informatie die nieuw is in CommonSense

Aan de lijst met sensoren wordt een sensor toegevoegd met de volgende data:

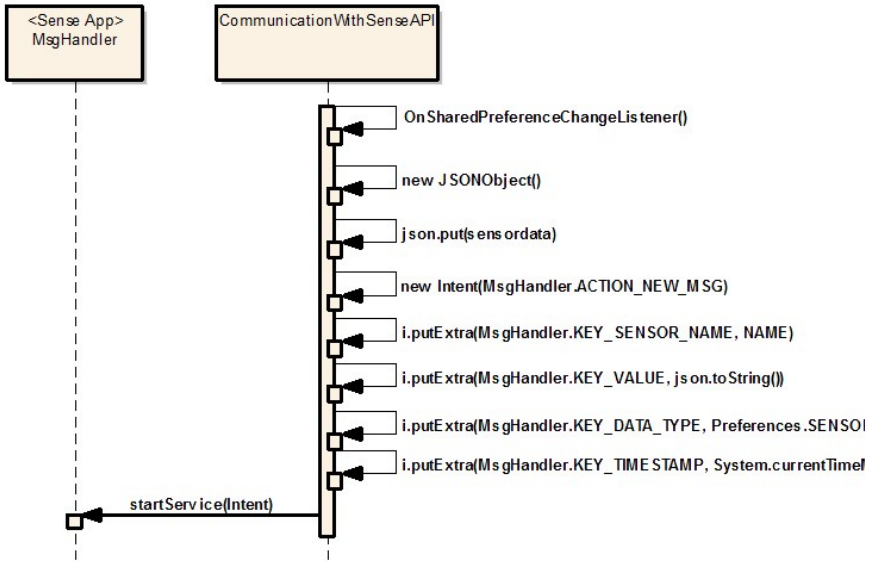
- De toestand van de applicatie
- De actie die door de applicatie als laatst is toegevoegd
- Het nummer van de actie die als laatste is toegevoegd
- De tijd waarop het bericht is verzonden

Door de laatst uitgevoerde actie ook in CommonSense te zetten, kunnen bij uitval van de telefoon Sense en ASK de acties overnemen[5].

Daarnaast is het handig om te zien wanneer de toestand verandering heeft plaatsgevonden. Zo is voor de eigenaar te zien op welke momenten de telefoon dacht dat de eigenaar niet in de buurt was.

5 *In het proof-of-concept wordt deze optie niet geïmplementeerd, maar is als uitbreiding van de applicatie te zien*

5.3 Sequentie diagram verbindingsopzet



Figuur 15: Verbindingsopzet met de MessageHandler van Sense

In figuur 15 is het sequentie diagram van de verbinding te zien. Hier zijn de stappen binnen de klasse *CommuncationWithSense* uitgewerkt. De klasse heeft een *OnSharedPreference-ChangeListener* om te controleren of de toestand, of een andere instelling die van toepassing is op CommonSense, is veranderd.

Als een van de gevraagde instellingen is veranderd moet deze informatie ook worden doorgestuurd naar CommonSense. Binnen de communicatie klasse wordt een *JSONObject* opgezet met daarin de sensordata.

Daarna wordt een *Intent*[6] voor de *MsgHandler* aangemaakt en daarbij de gegevens uit tabel 9. Als de hele *Intent* is opgebouwd wordt de *MsgHandler* Service gestart met de *Intent.*

6 *Abstracte beschrijving van een actie die moet worden uitgevoerd. Onderdeel is uitgebreid beschreven in het Construction fase rapport*

6. Verbinding met ASK

6.1 Inleiding

Om de eigenaar te vinden wordt er in de applicatie gebruik gemaakt van ASK. Vanuit ASK kan er namelijk contact opgenomen worden met de eigenaar of met andere personen. Het contact kan gelegd worden via e-mail, sms en bellen.

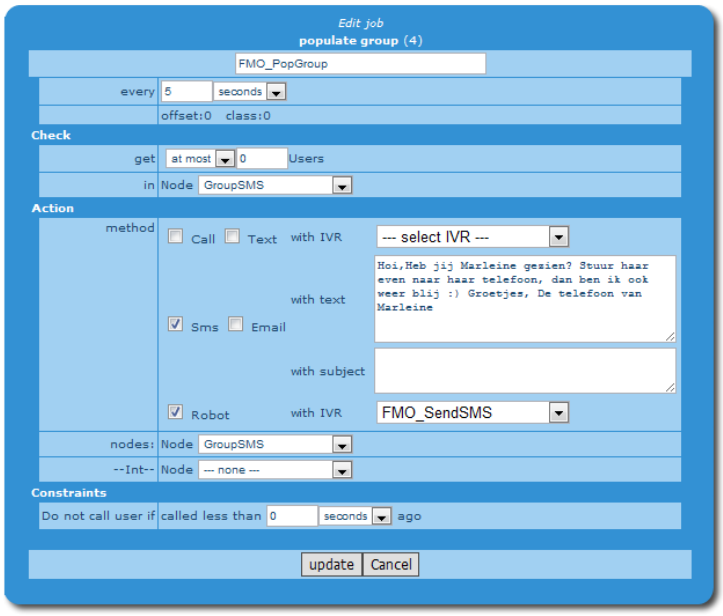
6.2 Werking van ASK

Om contact op te nemen met een persoon moeten er in ASK een paar onderdelen worden ingesteld. Er zijn veel verschillende mogelijkheden in ASK, bijvoorbeeld het bellen van een groep mensen aan de hand van een IVR[7] menu of het sturen van een SMS naar een persoon.

Niet alle instellingen van het systeem zijn uit te leggen in de documentatie van het project. Er worden namelijk meerdere sets van instellingen gemaakt voor de opdracht. Maar om een idee te geven hoe het systeem in elkaar zit, is volgend voorbeeld met het sms'en naar een gebruiker opgezet. Deze set van instellingen wordt in de tweede versie van de applicatie gebruikt om de verbinding met ASK op te zetten en testen.

Binnen ASK moeten de volgende onderdelen worden aangemaakt / ingesteld:

- Nodes
  In het systeem moeten voor de test twee verschillende nodes worden aangemaakt. Hierbij gaat het om de node van de eigenaar. Deze node bevat de naam, e-mailadres en het telefoonnummer van de eigenaar.
  Om een actie uit te kunnen voeren wordt er een zogenaamde “groepnode” aangemaakt. Aan deze node worden dan andere personen gehangen, waardoor een actie uitgevoerd kan worden. In het voorbeeld is de GroupSMS gekoppeld.
- PopulateGroup



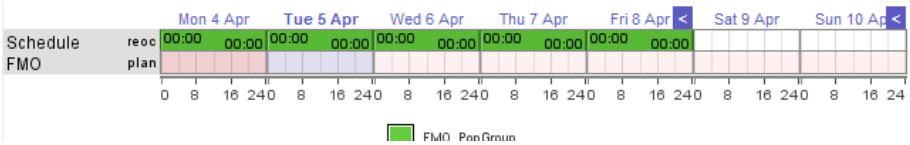
Figuur 16: Populate group in ASK

Op het moment dat de *populateGroup* wordt aangeroepen vanuit de *Scheduler* wordt een actie op een groep uitgevoerd. In bovenstaande afbeelding zijn de instellingen van de *populateGroup* te zien.

7 *Interactie Voice Response, telefoon gesprek waarin verschillende keuzes gemaakt kunnen worden*

Aangegeven wordt om de hoeveel tijd (5 sec) de agent moet controleren of de groep meer dan 0 nodes bevat. Op het moment dat de groep meer dan 0 gebruikers bevat wordt er een sms gestuurd naar de personen in die groep en wordt er een IVR menu gestart.

- IVR Menu
  Het IVR menu (FMO_SendSMS) bevat één regel, deze actie zorgt ervoor dat de node die gesmst is ook verwijderd wordt uit de groep (GroupSMS).
- Scheduler
  Om aan te geven wanneer er gekeken moet worden naar de grootte van de groep, buiten elke 5 seconde, moet er een scheduler worden ingesteld. In onderstaande afbeelding is hier een voorbeeld van te zien.



Figuur 17: Scheduler ASK

Per datum kan ingesteld worden wanneer de agent moet gaan controleren. Hier staat de onderste rij voor. Daarnaast kan er ook per dag worden aangegeven. In het voorbeeld is er voor elke maandag tot vrijdag ingesteld dat de agent gaat controleren.

6.3 Inloggen

Bij ASK hoeft er niet ingelogd te worden om gebruik te maken van de services. Enkel de moet er een sessie worden gestart met een *private authentication key*. Bij ASK gaat het namelijk niet om een directe gebruiker maar om het gehele systeem dat wordt beheerd. Aan de hand van UUID's kunnen kenmerken van een specifiek onderdeel worden aangesproken.

Een UUID is een Unique User ID. Hiermee wordt aan elke node en object in het ASK systeem een unieke waarde gegeven.

6.4 Webservices

Om gebruik te maken van de functies van het ASK Platform wordt er gewerkt met webservices. Vanuit de applicatie moet er een *SOAPClient* worden aangemaakt die de functies aanroept van ASK.

SOAP is een protocol voor het communiceren tussen twee componenten. Bijvoorbeeld een website en een applicatie. Binnen de website kunnen functies worden aangemaakt die van buitenaf gebruikt kunnen worden. Aan de hand van webservices zijn deze functies aan te spreken. Aan de kant van de applicatie moet de communicatie met de website worden opgezet. Voor het opzetten van de communicatie kan in veel programmeertalen gebruik gemaakt worden van een *SOAPClient*.

6.5 Opzet

Om het geheel soepel werkend te krijgen wordt de verbinding met ASK in meerdere versie opgezet.

Data	Toelichting
v2.1	– Opbouw in alleen ASK – Door ASK-CS wordt een alleenstaande versie van ASK gemaakt waar de afstudeerder mee kan werken. – Er wordt een opzet gemaakt waarbij een SMS gestuurd kan worden naar één of meerdere personen in een groep.
v2.2	– Aanmaken van de webservices in een losstaande applicatie – Testen door middel van een SMS te sturen naar eigenaar
v2.3	– Code losse applicatie toevoegen aan *Find My Owner* applicatie – Functies voor aanroepen van ASK
v2.4	– Vindacties uitwerken met de webservices – Inrichten ASK met juiste groepen en instellingen

Tabel 11: Opzet verbinding met ASK

7. Webinterface

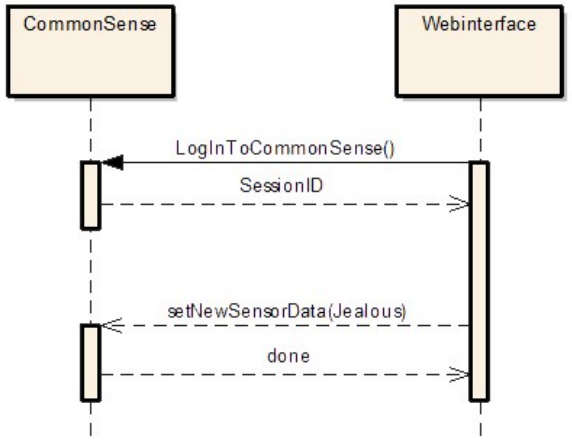
Om vanaf buitenaf aan te geven dat de telefoon niet in de buurt is van de eigenaar moet er een simpele webinterface worden opgezet. De webinterface moet ervoor zorgen dat de toestand van de applicatie *Jealous* wordt. Doordat de toestand *Jealous* is, gaat de telefoon niet meer opzoek naar de eigenaar en wacht deze totdat de eigenaar de telefoon weer gebruikt.

7.1 Communicatie met de webinterface en CommonSense

De verschillende systeemcomponenten die in de webinterface meespelen zijn:

- Webinterface
  Een simpele GUI waarbij de gebruiker moet inloggen en kan aangeven dat hij niet in de buurt is van de telefoon.
- CommonSense
  De gewijzigde toestand, oftewel *Jealous*, wordt gestuurd naar CommonSense. In CommonSense wordt de toestand toegevoegd aan de juiste sensor.
- *Find My Owner* applicatie
  Vanuit de applicatie moet regelmatig gecheckt worden of de toestand *Jealous* is. Deze toestand moet daarvoor opgehaald worden bij CommonSense.

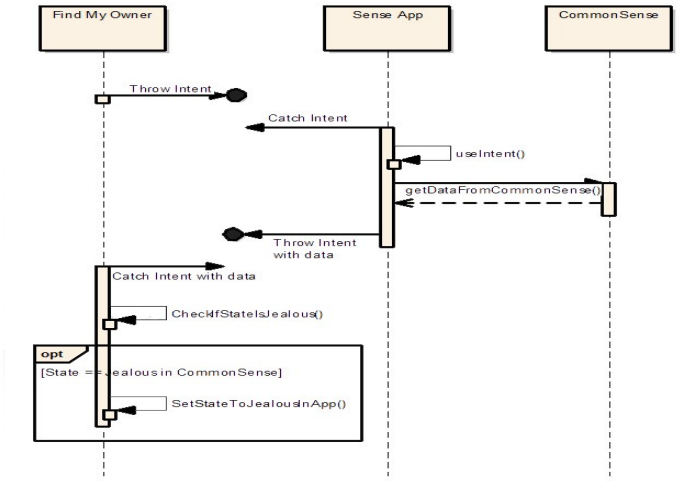
In het onderstaande diagrammen is de communicatie tussen de verschillende componenten in het systeem te zien. De gebruiker heeft enkel te maken met de GUI. De status verandering naar *Happy* gebeurt in de applicatie, maar valt buiten het onderdeel webinterface. Het is namelijk een algemeen onderdeel van de gehele applicatie.



Figuur 18: Verbinding tussen de webinterface en CommonSense

In het eerste diagram(figuur 18) is de communicatie tussen de webinterface en CommonSense te zien. De gebruiker logt in bij de webinterface en krijgt een scherm met keuzes te zien. Hierin moet de gebruiker kunnen kiezen voor het veranderen van de toestand naar *Jealous*. Zodra de eigenaar deze optie kiest, wordt er naar CommonSense een bericht gestuurd met de nieuwe toestand *Jealous* en de tijd.

In het tweede diagram is de communicatie met de applicatie te zien. Deze twee diagrammen zijn los van elkaar getekend omdat deze twee acties niet direct met elkaar te maken hebben. De acties kunnen namelijk ook tegelijkertijd plaats vinden.



Figuur 19: Verbinding tussen de applicatie en CommonSense

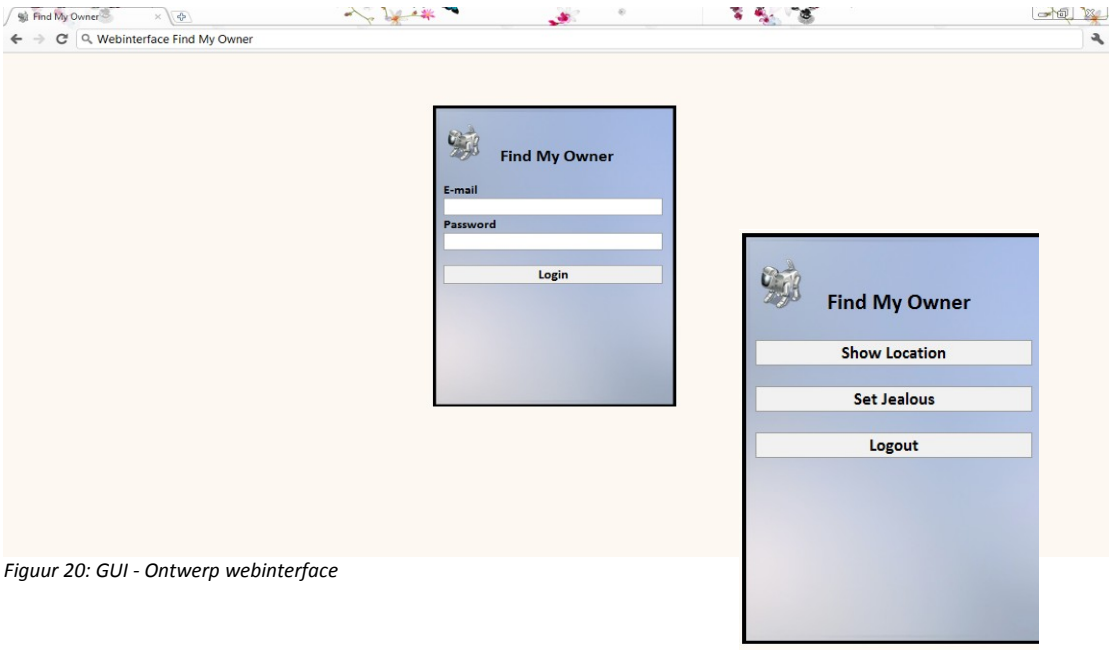
De applicatie controleert periodiek of de toestand is veranderd en tegelijkertijd kan de eigenaar zijn toestand wijzigen in de webinterface. Om te voorkomen dat hierdoor wel een zoekactie plaats vindt en de status eigenlijk al veranderd is naar *Jealous*, wordt in ieder geval voor elke zoekacties gecontroleerd of de toestand niet anders is dan bij de zoekacties hoort (*Wondering* of *Disappointed*). In het diagram is de controle bij CommonSense te zien. Hierbij wordt een response gegeven. Op het moment dat de response data bevat dat de toestand *Jealous* is bij CommonSense, wordt de toestand van de applicatie ook hierin veranderd[8].

8 *In het construction fase rapport wordt meer uitleg gegeven over het gooien en opvangen van de intents, zoals te zien in figuur 19*

7.2 GUI van de webinterface: een simpel ontwerp

Voor de webinterface geldt ook dat er maar een simpele GUI nodig is. De eigenaar moet kunnen inloggen met dezelfde gegevens als bij CommonSense en de toestand van de applicatie moet kunnen worden veranderd. Daarnaast is het bruikbaar voor de eigenaar om te zien waar de telefoon zich bevindt. Als dit op dezelfde locatie is als de eigenaar kan deze doorgaans beter naar de telefoon lopen in plaats van de webinterface te gebruiken.

Door deze simpele onderdelen te combineren is het ontwerp van de GUI als volgt:



Figuur 20: GUI - Ontwerp webinterface

Bij het ontwerp is rekening gehouden met het gebruik van een mobiele browser. Het kan namelijk zijn dat de eigenaar een tweede smartphone heeft of de smartphone van een collega/vriend gebruikt om naar de webinterface te gaan. Voor deze toepassing is de GUI ook gemakkelijk te gebruiken.

Ook over de implementatie van de webinterface is meer te lezen in het construction fase rapport

Conclusie

Aan de hand van de ontwerpen die gemaakt zijn tijdens de elaboration fase is de ontwikkeling van de applicatie soepeler verlopen. Tijdens het programmeren is er weinig tijd gestoken te worden in de logica achter het verloop van functies.

Doordat er geen voorkennis was van Android is vooral het begin van de ontwerpen gemaakt voor de logica van de acties en de basisopbouw van het systeem. Door het gebruik van Android veranderen de acties en specifieke opbouw van klasses aanzienlijk.

Het gebruik van iteraties is een slimme aanpak om van te voren een indruk te krijgen van de onderdelen in het project en wat hiervoor allemaal voor ontworpen, geïmplementeerd en getest moet worden. In de loop van het project wordt duidelijk dat onderdelen groter uitpakken of juist snel af zijn. Hierdoor is halverwege het iteratie plan aangepast, maar ook deze is niet volledig gevolgd.

Verklarende woordenlijst

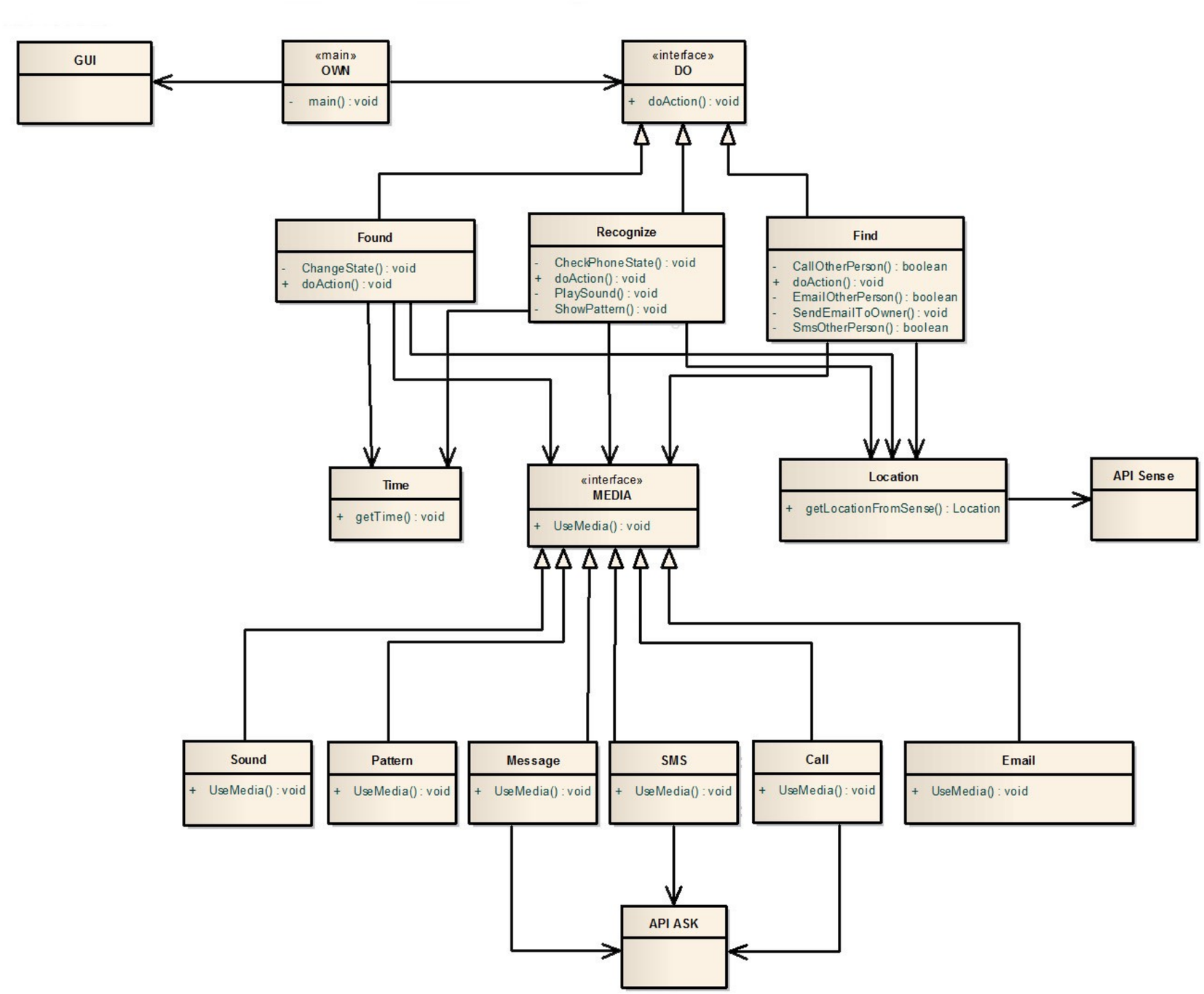
Woord, afkorting	Verklaring
ASK, ASK Community Systems	Dochterbedrijf van Almende dat het ASK Platform heeft opgezet
ASK, ASK webinterface	De basis van ASK, hier wordt alles gecombineerd. Dit deel wordt door ASK CS ingesteld voor een klant
ASK Desktop	Een applicatie waar ASK wordt gebruikt op de achtergrond
ASK Platform	Het ASK systeem met alle onderdelen (Communication Coordinator, API, Rules and scenario's, Users and groups, applicatiies en communicatiemiddelen)
IVR Menu	Interactive voice response, een telefonisch keuze menu. Ook wel bekend van klantenservice telefoongesprekken
Agent	Een virtuele persoonlijkheid binnen het ASK platform die één of meerdere verantwoordelijkheden binnen het systeem heeft.
Thread	Het hoofdproces verdelen in meerdere processen om verschillende taken tegelijkertijd uit te kunnen voeren
Herkennen, Recognizing, Wondering	Het herkennen van de eigenaar. Hierbij wordt er gekeken of de eigenaar in de buurt van de telefoon is. De toestand die hieraan gegeven wordt is Wondering.
Vinden, Find, Disappointed	Het vinden van de eigenaar. Hierbij wordt gekeken waar de eigenaar is en wordt er geprobeerd deze herenigen met de telefoon. De toestand die hieraan gegeven wordt is Disappointed
Toestand, State, Status (van de eigenaar / applicatie)	De toestanden van de applicatie zijn de toestanden waarin de applicatie verschillende acties gaat uitvoeren. De toestanden zijn uitgedrukt in verschillende emoties
Package	De groepering van klasses in verschillende namespaces.
Iteratie	De verschillende stappen binnen het iteratief ontwikkelen. Elke iteratie bevat weer kleinere stappen.
BroadcastReceiver	Interrupt van bijvoorbeeld een alarm afvangen en vervolgens bijbehorende acties uitvoeren (bijvoorbeeld de findAction klasse aanroepen)
Service	Een service is een actie die op de achtergrond uitgevoerd wordt zonder dat de gebruiker input hoeft te leveren
Activity	Een activity is een activiteit die op de voorgrond wordt uitgevoerd. De activiteit is doorgaans gekoppeld aan een UI.
Notification bar / Status bar / Notificatie balk	De balk die bij Android “altijd” bovenaan in beeld te zien is. Bij aanraken kan de balk worden uitgeschoven en worden actieve applicaties en meldingen weergegeven
GUI, Gebruikersinterface, UI	Graphical User Interface. Het deel dat de gebruiker zit van de applicatie
Webinterface	In het deel van de applicatie gaat het bij de webinterface om de website waar de eigenaar zich kan aanmelden om de toestand van de applicatie te veranderen in *Jealous*. In het verhaal van ASK is de webinterface de plek waar grafisch aanpassingen gedaan kunnen worden in de instellingen van ASK in plaats van via webservice calls.

Tabel 12: Verklarende woordenlijst

Bronvermelding

1. Rien Elling, Bas Andeweg, Jaap de Jong, (2005). *Digitaal Handboek Rapportagetechniek 4.2.* Noordhoff Uitgevers (cd-rom)
2. The Eclipse Foundation, (-2011). *OpenUP.* Via http://epf.eclipse.org/wikis/openup/
3. The Eclipse Foundation, (-2011). *Iteration Plan.*
4. The Eclipse Foundation, (-2011). *Example Iteration Plan*.
5. T. van As, (2011). *ASK Architecture.* ASK Community Systems
6. T. van As en S. Mulder, (2011). *MyASK Architecture*. ASK Community Systems
7. T. van As, (2011). *MyASK Features, Applications & Roadmap.* ASK Community Systems
8. Mark L. Murphy, (2010). *Android Programming Tutorials.* CommonsWare
9. Mark L. Murphy, (2010). *The Busy Coder's Guide To Android Development.* CommonsWare
10. Mark L. Murphy, (2009-2010). *The Busy Coder's Guide To Advanced Android Development.* CommonsWare
11. George Huling, (1999). *State Transistion Table*. Via http://sunset.usc.edu/classes/cs665s99/class11/statetables.pdf
12. IBM, (-2011). *Real Time Use case.* Via http://www.ibm.com/developerworks/rational/library/5272.html
13. Android, (-2011). *Android Developers.* Via http://developer.Android.com/guide/index.html
14. Nederlandse taalunie, (-2011). *Literatuurlijsten (algemeen).* Via http://taaladvies.net/taal/advies/tekst/87/#7
15. Jos Warner & Anneke Kleppe, (2007). *Praktisch UML.* Pearson Education Benelux
16. Bruce Powel Douglas, (2009). *Real Time UML: Advances In The UML For Real-Time Systems*. Addison -Wesley. Via Google Books

Bijlage A: Klassediagram eerste versie



Bijlage B: Iteratieplan eerste versie

Naam/ uitleg	Priori-teit	Verwachte grootte	Status	Iteratie	Verantwoordelijk	Maximaal aantal uren	Uren aan gewerkt	Referentie materiaal
Iteratie 1						***36***	***0***	
Klassendiagram totaal	*1*	*2*	*Open*	*1*	*Marleine*	*6*	*0*	
Sequencediagram main	*2*	*1*	*Open*	*1*	*Marleine*	*5*	*0*	
Bouwen FindMyOwner Android application	*3*	*4*	*Open*	*1*	*Marleine*	*15*	*0*	
Testplan opzetten	*4*	*3*	*Open*	*1*	*Marleine*	*10*	*0*	
Iteratie 2						***47***	***0***	
Ontwerpen GUI	*1*	*2*	*Open*	*2*	*Marleine*	*7*	*0*	
Toestandsdiagram GUI	*2*	*3*	*Open*	*2*	*Marleine*	*10*	*0*	
Bouwen GUI	*3*	*7*	*Open*	*2*	*Marleine*	*25*	*0*	
Testscenario opzetten	*4*	*1*	*Open*	*2*	*Marleine*	*5*	*0*	
Iteratie 3						***22***	***0***	
Klassendiagram DO	*1*	*1*	*Open*	*3*	*Marleine*	*4*	*0*	
Sequencediagram DO	*2*	*2*	*Open*	*3*	*Marleine*	*6*	*0*	
Implementatie DO	*3*	*3*	*Open*	*3*	*Marleine*	*9*	*0*	
Testscenario opzetten	*4*	*1*	*Open*	*3*	*Marleine*	*3*	*0*	
Iteratie 4						***56***	***0***	
Klassendiagram MEDIA	*1*	*1*	*Open*	*4*	*Marleine*	*5*	*0*	
Sequencediagram MEDIA	*2*	*2*	*Open*	*4*	*Marleine*	*8*	*0*	
Implementatie MEDIA	*3*	*4*	*Open*	*4*	*Marleine*	*15*	*0*	
Sequencediagram Location	*4*	*2*	*Open*	*4*	*Marleine*	*6*	*0*	
Implementatie Location	*5*	*5*	*Open*	*4*	*Marleine*	*18*	*0*	
Testscenario opzetten	*6*	*1*	*Open*	*4*	*Marleine*	*4*	*0*	
Iteratie 5						***54***	***0***	
Activiteitendiagrammen Sound,Pattern, Email en Message	*1*	*3*	*Open*	*5*	*Marleine*	*10*	*0*	
Implementatie Sound	*2*	*2*	*Open*	*5*	*Marleine*	*6*	*0*	
Implementatie Pattern	*3*	*2*	*Open*	*5*	*Marleine*	*6*	*0*	
Implementatie Message	*4*	*2*	*Open*	*5*	*Marleine*	*8*	*0*	
Implementatie Email	*5*	*5*	*Open*	*5*	*Marleine*	*16*	*0*	
Testscenario opzetten	*6*	*1*	*Open*	*5*	*Marleine*	*4*	*0*	
Testen	*7*	*1*	*Open*	*5*	*Marleine*	*4*	*0*	
Iteratie 6						***60***	***0***	
Activiteitendiagrammen Find, Found en Recognize	*1*	*3*	*Open*	*6*	*Marleine*	*10*	*0*	
Toestandsdiagram Found	*2*	*1*	*Open*	*6*	*Marleine*	*3*	*0*	
Implementatie Find	*3*	*3*	*Open*	*6*	*Marleine*	*10*	*0*	
Implementatie Found	*4*	*4*	*Open*	*6*	*Marleine*	*15*	*0*	
Implementatie Recognize	*5*	*4*	*Open*	*6*	*Marleine*	*15*	*0*	
Testscenario opzetten	*6*	*1*	*Open*	*6*	*Marleine*	*5*	*0*	
Testen	*7*	*1*	*Open*	*6*	*Marleine*	*2*	*0*	
Iteratie 7						***50***	***0***	
Activiteitendiagrammen Call, SMS, ASK API	*1*	*1*	*Open*	*7*	*Marleine*	*5*	*0*	
Implementatie Call	*2*	*4*	*Open*	*7*	*Marleine*	*15*	*0*	
Implementatie SMS	*3*	*3*	*Open*	*7*	*Marleine*	*10*	*0*	
Implementatie ASK met Call, SMS en E-mail	*4*	*4*	*Open*	*7*	*Marleine*	*15*	*0*	
Testen	*5*	*1*	*Open*	*7*	*Marleine*	*5*	*0*	
Iteratie 8						***23***	***0***	
Toestandsdiagram Time	*1*	*1*	*Open*	*8*	*Marleine*	*5*	*0*	
Implementatie Time	*2*	*2*	*Open*	*8*	*Marleine*	*8*	*0*	
Testen	*3*	*3*	*Open*	*8*	*Marleine*	*10*	*0*	
2 uur per week iteraties, 8 uur documenteren, 12 weken lang				***Totaal aantal uur***		*348*		

Construction Fase Rapport

"Onderzoek en ontwikkeling van Android applicatie – Waar is mijn eigenaar?"

Auteur: Marleine van Kampen
Datum: 12 mei 2011
Versie: 1.0

Versiebeheer

Versie	Datum	Auteur	Aanpassingen
0.1	23 februari 2011	M. van Kampen	Basis opzet
0.2	19 maart 2011	M. van Kampen	Verbeteren en aanvullen
1.0	12 mei 2011	M. van Kampen	Aanvullen informatie en controleren voor oplevering

Reviewers

Versie	Datum	Auteur
0.2	18 maart 2011	S. Mulder
0.2	18 maart 2011	T. van As
0.3	22 april 2011	M. van Kampen

Gerelateerde documenten

Naam	Versie	Datum	Auteur
Afstudeerscriptie	1.0	6 juni 2011	M. van Kampen
Inception Fase Rapport	1.0	6 juni 2011	M. van Kampen
Elaboration Fase Rapport	1.0	6 juni 2011	M. van Kampen
Transition Fase Rapport	1.0	6 juni 2011	M. van Kampen

Distributielijst

Naam	Organisatie	Functie	Reden
S. Mulder	Sense	Technisch begeleider	Proces begeleider
T. van As	ASK	Technisch begeleider	Proces begeleider
L. Stellingwerff	ASK	Technisch begeleider	Proces begeleider
J.J. Visser	Haagse Hogeschool	Docent	1e Examinator
D.R. Stikkolorum	Haagse Hogeschool	Docent	2e Examinator

Samenvatting

Marleine van Kampen, student van De Haagse Hogeschool Delft voert gedurende zeventien weken een afstudeeropdracht uit bij ASK Community Systems en Sense Observation Systems. De opdracht genaamd “Waar is mijn eigenaar?”, is een Andriod applicatie die opzoek gaat naar de eigenaar van de telefoon, bij afwezigheid van de eigenaar.

Het project is opgezet in vier verschillende fases volgens de OpenUP methode. In de contruction fase wordt gewerkt aan de implementatie van de applicatie.

De applicatie is ontwikkeld op het mobiele platform Android en als ontwikkelomgeving is gebruik gemaakt van Eclipse. Eclipse biedt de gemakkelijkste manier om een Android applicatie op te zetten en te testen.

Het project is opgezet in vier opleveringen. Per oplevering zijn onderdelen toegevoegd aan de applicatie.

Het systeem bestaat uit een aantal packages namelijk een package dat opzoek gaat naar de eigenaar, een package dat communiceert met de platformen van ASK en Sense, een package dat de activiteiten / GUI beheert en een package dat de instellingen en standaardwaardes beheert.

De applicatie werkt met één sensor om te herkennen of de eigenaar in de buurt is. De applicatie is zo opgezet dat er gemakkelijk meer sensoren toegevoegd kunnen worden.

Voor de gebruiker is een simpele GUI opgezet die de eigenaar de mogelijkheid geeft zijn toestand en locatie te bekijken en de *Find My Owner* service in- en uit te schakelen.

Om de gebruiker de mogelijkheid te geven te applicatie stop te zetten terwijl deze niet bij de telefoon in de buurt is, is een webinterface opgezet.

Inhoudsopgave

I.I Aanleiding....5
I.II Doel....5
I.III Scope....5

2. OPZETTEN ONTWIKKELOMGEVING....6

2.1 Opzetten ontwikkelomgeving....6
2.2 Opzetten project....7

3. BASIS VERSIE....10

3.1 De opzet van de GUI....10
3.2 Code van de GUI....11

4. EERSTE OPLEVERING....14

4.1 Samenhang van Ipackages en klassen in de tweede oplevering....14
4.2 GUI....15
4.3 Implementatie....16
4.3.1 Activity....16
4.3.2 Shared Preferences....18
4.3.3 Intent....19
4.3.4 Context....20
4.3.5 Service....20
4.3.6 AlarmManager....21
4.3.7 BroadcastReceiver....22

5. TWEEDE OPLEVERING....23

5.1 Samenhang van packages en klassen in de tweede oplevering....23
5.2 Implementatie....24
5.2.1 Verbinding met CommonSense....24
5.2.2 Verbinding met ASK....26

6. DERDE OPLEVERING....32

6.1 Samenhang en verantwoordelijkheden van de klassen....32
6.2 Implementatie....33
6.2.1 Vind-acties door middel van ASK en Android....33
6.2.2 Locatie uit Google Maps....40

7. VIERDE OPLEVERING – EINDVERSIE....44

7.1 Samenwerking tussen de klassen in eindversie....44
7.2 Webinterface – Toestand van de applicatie veranderen naar Jealous....48
7.2.1 Inloggen bij CommonSense....48
7.3 De applicatie – Toestandsverandering naar Jealous herkennen. 49
7.3 Eindversie van de GUI....52

VERKLARENDE WOORDENLIJST....54

BRONVERMELDING....55

I. Inleiding

I.I Aanleiding

Door Marleine van Kampen, afstudeerder van De Haagse Hogeschool Delft wordt gedurende zeventien weken gewerkt aan het project “Waar is mijn eigenaar?”. Het project is een onderzoek naar en ontwikkeling van een Android applicatie bovenop de bestaande platformen van de dochterbedrijven, Sense en ASK, van het Rotterdamse onderzoeksinstituut Almende.

I.II Doel

Het doel van het Construction Fase Rapport is het weergeven van de implementatie van de applicatie.

I.III Scope

Het Construction fase rapport is opgebouwd uit de verschillende versies die opgeleverd zijn. Per versie worden een aantal codefragmenten gegeven. Bijgeleverd wordt de volledige code en de uitvoerbare versies van de applicatie. Daarnaast is er gebruik gemaakt van Javadoc. Via www.tiny.cc/fmo_javadoc is de volledige documentatie van de functies te vinden.

Per versie wordt ook een nieuw klassendiagram en mogelijke extra verbeterde diagrammen gegeven om te verduidelijken hoe het systeem opgebouwd is.

Daarnaast is er informatie toegevoegd over de opzet van een Android applicatie in de Eclipse omgeving.

2. Opzetten ontwikkelomgeving

Het eerste deel van de implementatie van de Find My Owner applicatie gaat over het opzetten van een applicatie in de ontwikkelomgeving

2.1 Opzetten ontwikkelomgeving

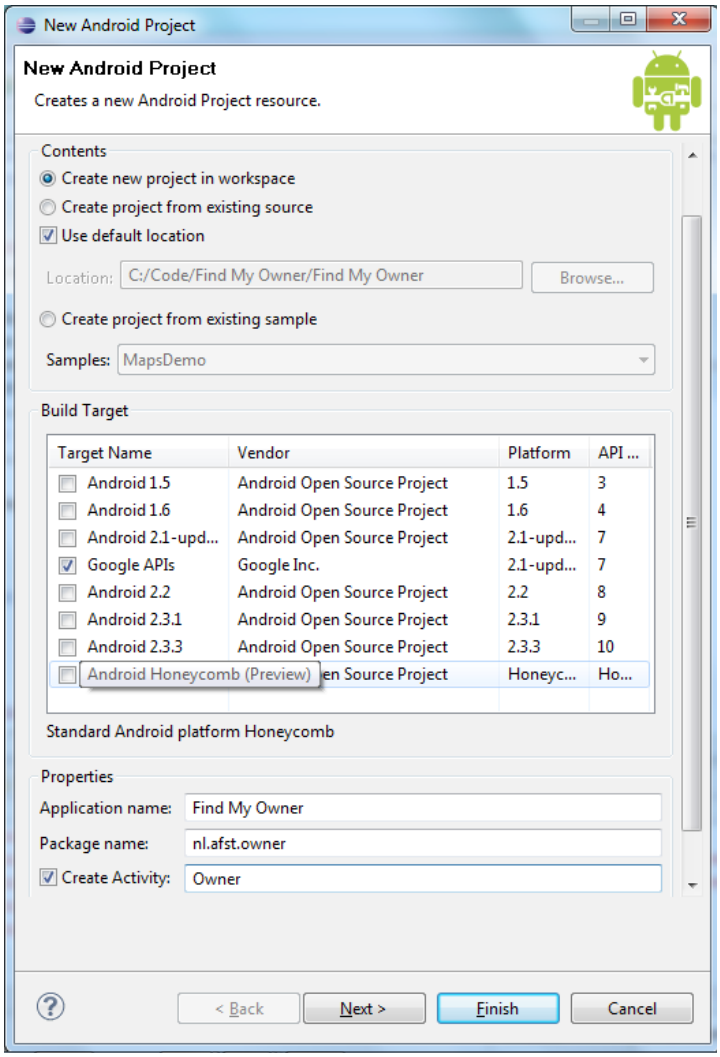
Voor het ontwikkelen van de Find My Owner applicatie wordt de Eclipse ontwikkelomgeving gebruikt. De Eclipse omgeving wordt zeer aangeraden omdat deze goed samenwerkt met de Android SDK. Andere ontwikkelomgevingen zijn wel mogelijk maar vergen extra kennis en hebben vaak nog veel handmatige instellingen nodig.

Voor het opzetten van de ontwikkelomgeving zijn de volgende onderdelen geïnstalleerd:

- Eclipse Classic Helios 3.6.1
  Standaard versie van Eclipse. Voor het ontwikkelen voor Android wordt minimaal versie 3.4 geëist.
- Java JDK
  Voor het ontwikkel van Android wordt Java gebruikt. De Java SDK is vereist voor het ontwikkelen voor Android.
- Android SDK
  Laatste versie van de Android Software Development Kit.
- SDK Platform Android 2.1
  Platform gericht op functionaliteiten als toevoeging op de SDK.
- SDK Platform Android 1.6
  Platform gericht op functionaliteiten als toevoeging op de SDK. De applicatie wordt ontwikkeld voor versie 2.1 maar moet mogelijk ook kunnen werken met 1.6. Voor het testen kan ook de SDK van versie 1.6 worden gebruikt.
- ADT for Eclipse
  Android Development Tools; De schil voor het ontwikkelen van Android in Eclipse. Het tool pakket bevat onder andere het gemakkelijk opzetten van projecten en het grafisch ontwerpen van de user interface(UI).

2.2 Opzetten project

Voor het opzetten van een nieuw project is de wizard “New Android Project”. Via deze wizard wordt het gehele project opgezet met basisgegevens van het project. In onderstaande afbeelding is de wizard te zien.



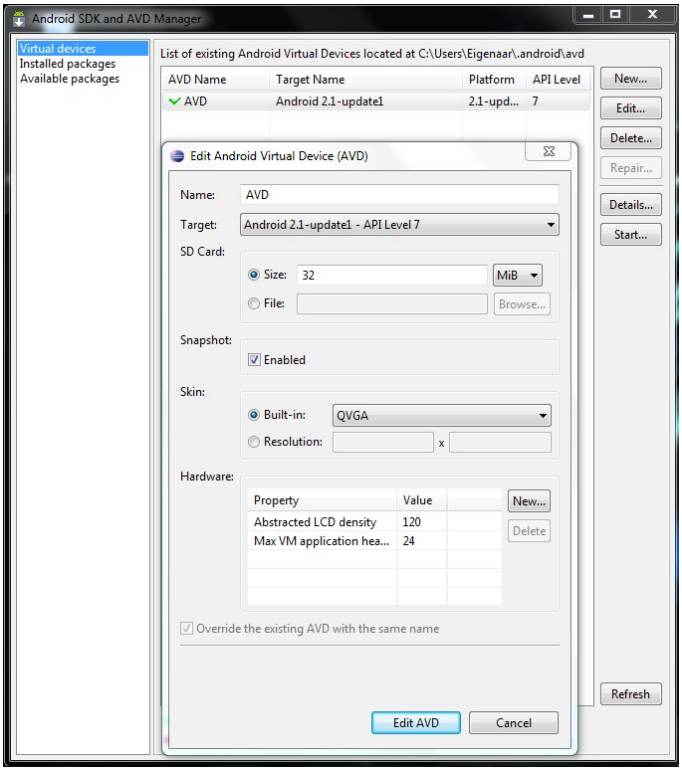
Figuur 1: Projectwizard

Voor het project wordt de naam gegeven en de bijbehorende locatie voor opslag. Bij het aanmaken wordt aangegeven voor welk platform de applicatie wordt geschreven.

Binnen het project zijn er enkele eigenschappen wat betreft de naam van de basis*package* en de naam van de applicatie. Daarnaast kan er voor worden gekozen de hoofdactiviteit(de main) aan te maken.

De extra klassen die toegevoegd worden kunnen vallen binnen het *nl.afst..owner package* maar kunnen ook onder verdeeld worden in andere *package*s. Gezien het klassendiagram kan de keuze gemaakt worden om de klassen die onder MEDIA vallen onder te verdelen in een losse *package*.

Om de applicatie uit te testen is een Android Virtual Device (AVD) toe te voegen aan het project. De instellingen hiervoor verlopen via *"*Android SDK en AVD manager*"*, te zien in onderstaande afbeelding.



Figuur 2: Android SDK en AVD manager

De instellingen betreffen de versie van het Android platform en mogelijk geheugen dat toegevoegd wordt. Bijvoorbeeld als voor de applicatie de SD kaart gebruikt wordt. Daarnaast is er een aanpasbaar beeld formaat, afhankelijk van het device. In de volgende afbeelding is te zien hoe het virtuele device wordt weergegeven.

Figuur 3: Menu en basisscherm in Android Virtual Device

In onderstaand fragment is de code te zien die wordt gegenereerd bij het aanmaken van een project.

```
package nl.afst.owner;                                                    [1]

import android.app.Activity;                                            [2]
import                                                                   android.os.Bundle;


public class Owner extends Activity {                                     [3]

        @Override
        public void onCreate(Bundle savedInstanceState) {               [4]
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);                          [5]
        }
}
```

Codefragment 1: Basiscode Owner.java

[1] De hoofdactiviteit *(main) FindMyOwner* valt onder het *package nl.afst.owner*

[2] Voor elk class die gebruikt wordt, moet een import worden toegevoegd

[3] De klasse *Owner* is van het type *Activity.* Bij het implementeren van de klasse wordt ook gebruik gemaakt van bestaande functies van *Activity*

[4] De o*nCreate()* functie is een voorbeeld van een bestaande functie van *Activity.* De *@Override* geeft aan dat het een *overrided* functie is. Bij het creëren van de *Owner Activity* wordt de functie o*nCreate()* aangeroepen en kunnen waardes en functies worden aangemaakt voor de klasse.

[5] *setContentView()* wordt gebruikt voor het koppelen van de lay-out aan de code. In dit geval wordt *main.xml* (wat zich bevindt in *resources (R) - layout*) gekoppeld aan de klasse *Owner.*

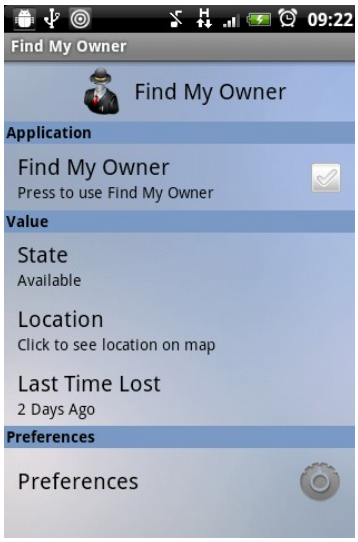
3. Basis versie

Als toevoeging aan de basiscode is de Graphical User Interface (GUI) opgezet. In het Elaboration Fase Rapport is het ontwerp hiervan te vinden.

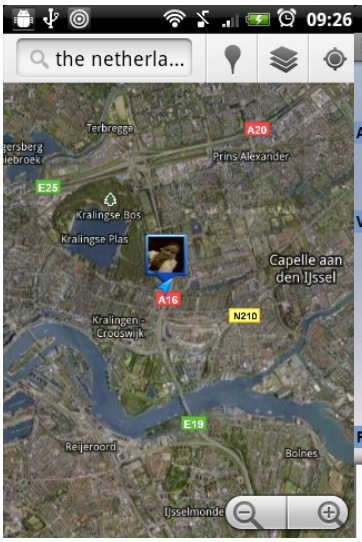
De GUI is gebouwd met de basisfunctionaliteiten en mogelijke opties. In de uiteindelijke versie van GUI kunnen wat onderdelen verschillen met dit ontwerp GUI. Deze verschillende worden in latere hoofdstukken besproken.

3.1 De opzet van de GUI

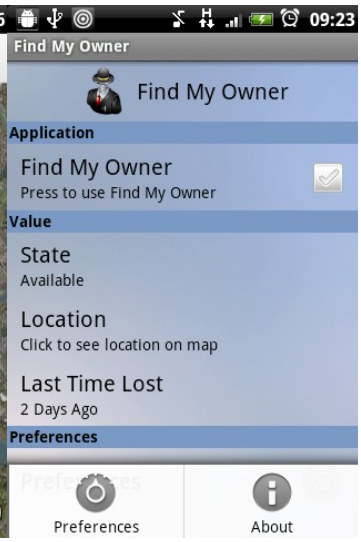
Hieronder de screenshots te zien van de eerste opzet van de GUI.



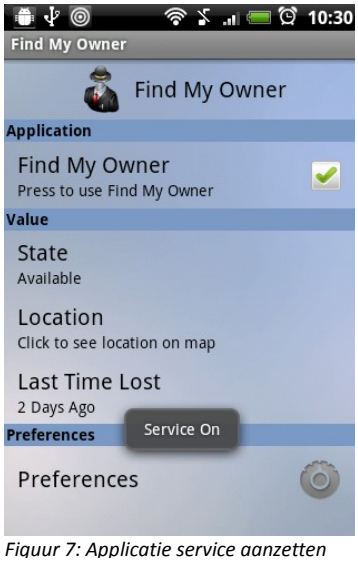
Figuur 4: Startscherm Find My Owner applicatie



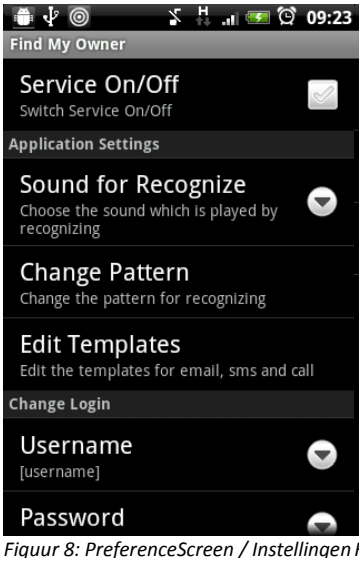
Figuur 5: Locatie van de telefoon



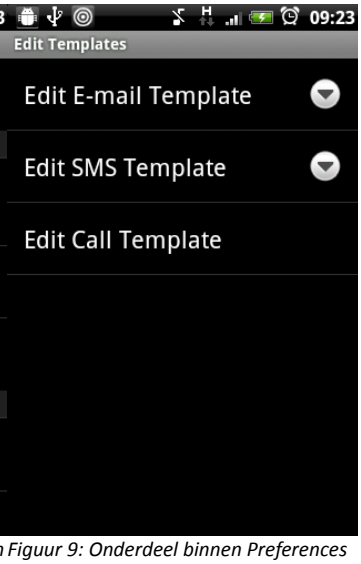
Figuur 6: Menu starten voor instellingen en meer informatie



Figuur 7: Applicatie service aanzetten



Figuur 8: PreferenceScreen / Instellingen



Figuur 9: Onderdeel binnen Preferences voor het veranderen van templates

In figuur 4 is de het hoofdscherm van de applicatie te zien. Op het moment dat de gebruiker de applicatie opstart wordt dit scherm weergegeven.

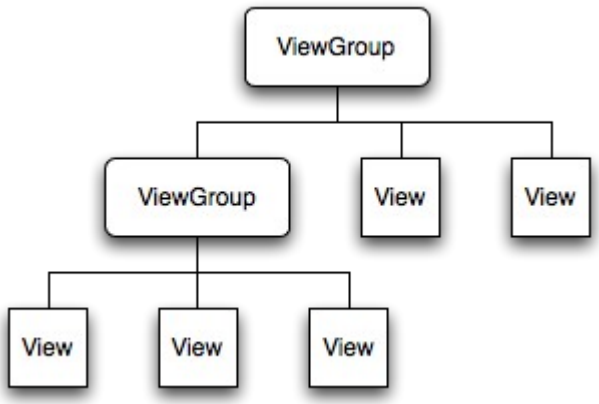
- De eerste keuze die de gebruiker heeft is het in- of uitschakelen van de applicatie-service. In de uiteindelijke versie van Sense en ASK wordt de service automatisch ingeschakeld met de inschakeling van de Sense service
- Als het goed is ziet de gebruiker zijn status altijd op *Happy* staan. Mocht deze status anders zijn betekent het dat de eigenaar zich moet aanmelden bij de applicatie. De status van de gebruiker wordt meer onderwater gebruikt om voor de applicatie te bepalen wat er moet gebeuren.
- Als de telefoon de eigenaar niet kan 'herkennen' wordt de locatie doorgestuurd naar de eigenaar per e-mail. Als extra is de locatie van de telefoon / eigenaar ook via de applicatie te zien (figuur 5)
- De gebruiker kan kijken wanneer de telefoon voor het laatst naar hem opzoek is geweest. Mocht hier iets niet kloppen dan kan in de uiteindelijke versie aangegeven worden dat de eigenaar tussen bepaalde tijdstippen wel aanwezig was bij zijn telefoon.
- Aan de hand van het menu (figuur 6) en via de knop Preferences (figuur 4) worden de instellingen gestart (figuur 8 en 9). In de instellingen kan de gebruiker enkele voorkeursinstellingen van de applicatie wijzigen.

3.2 Code van de GUI

Binnen het project is standaard een *main.xml* toegevoegd. Deze bevindt zich in de resources map *layout*. Deze bevat de lay-out van het basisscherm. De lay-out kan worden opgebouwd in XML maar ook via een graphische “drag&drop” mogelijkheid.

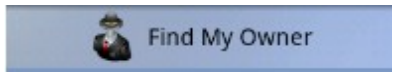
main.xml valt onder de *views* van een applicatie. In figuur 10 is de boom van views te zien. Deze kan per applicatie verschillen van grootte. Met de views worden de verschillende getoonde schermen bedoeld.

In het geval van Find My Owner kan de bovenste *ViewGroup* gezien worden als *main.xml*. De tweede *ViewGroup* is *Preferences* met bijvoorbeeld het *Edit Templates* scherm als extra *view.*



Figuur 10: ViewGroups en views binnen een Android applicatie

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:gravity="center_horizontal"
    android:background="@drawable/bg4"
    android:cacheColorHint="#00000000" >

<!-------------------------------------------------------------------------------------->

    <!-- Top application banner -->
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="6dp"
        android:paddingBottom="6dp"
        android:gravity="center_vertical" >
        <ImageView
            android:layout_width="wrap_content"

            android:layout_height="wrap_content"
            android:src="@drawable/iconsmall" />
        <TextView
            android:text="Find My Owner"
            android:textColor="@color/black"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:layout_marginLeft="8dp" />
    </LinearLayout>
<!-------------------------------------------------------------------------------------->

    <!-- Header for "Application" section -->

    <TextView
        android:text="Application"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:paddingLeft="2dp"
        android:background="@color/bar"
        android:textColor="@android:color/black"
        android:textStyle="bold" />
```



Codefragment 2: main.xml

Bovenstaande code weergeeft een deel van main.xml. Gebruik wordt gemaakt van een *LinearLayout*, een van de mogelijke layouts. Aan de lay-out worden enkele basiswaardes meegegeven zoals de standaard oriëntatie van het scherm (horizontaal of verticaal) en hoe het scherm uitgevuld moet worden.

Als eerste wordt de *header* met het logo en de applicatie naam weergegeven. Hiervoor wordt een extra LinearLayout gebruikt met daarin een afbeelding en de tekst. De uitlijning wordt gezet op *center_vertical*, hierdoor wordt komt de tekst precies in het midden van het scherm te staan.

Om het scherm op te delen in verschillende categorieën wordt een Textview tussen de velden gezet.

Voor het preferences scherm is ook een xml gemaakt. Hierin wordt het speciale element *PreferenceScreen* gebruikt.

Voor het menu wordt de XML tag *menu* gebruikt. Hierdoor wordt automatisch een menu

opgezet. In onderstaand codefragment is te zien hoe items aan het menu toegevoegd worden.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
        <item   android:enabled="true"
                android:checked="false"
                android:title="Preferences"
                android:visible="true"
                android:id="@+id/Preferences"
                android:icon="@android:drawable/ic_menu_preferences">
        </item>
        <item   android:enabled="true"
                android:title="About"
                android:visible="true"
                android:id="@+id/About"
                android:icon="@android:drawable/ic_menu_info_details">
        </item>
</menu>
```

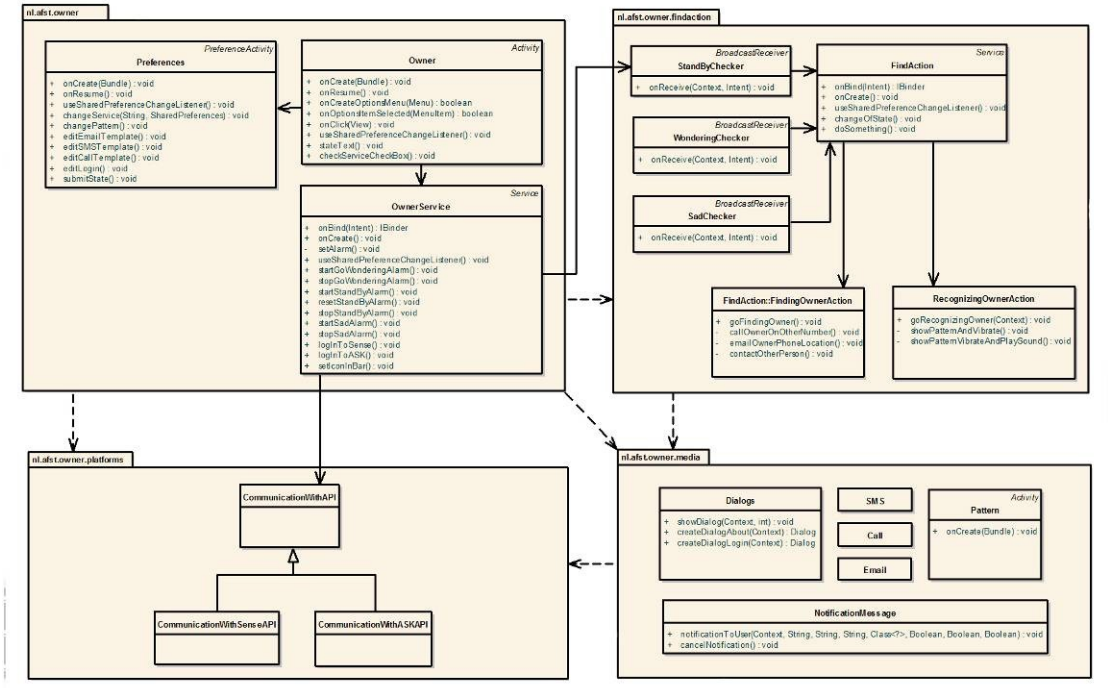
Codefragment 3: menu.xml

4. Eerste oplevering

Voor de eerste oplevering is gekozen de volgende onderdelen toe te voegen:

- Basiscode
- Werking achter de GUI
- Alle herkenningsfuncties zonder samenwerking met Sense en ASK

4.1 Samenhang van I*packages* en klassen in de tweede oplevering



Figuur 11: Klassendiagram - Eerste versie (klein formaat)

In figuur 11 is het klassendiagram van de eerste oplevering te zien. In bijlage A is een grotere weergave van het diagram te vinden.

Het systeem is opgedeeld in vier *package*s. De *package*s zijn gekozen op werking en raakvlak van de klassen. De *nl.afst.owner package* is de basis*package* van het systeem. Op het moment dat de applicatie wordt opstart wordt de *Owner Activity* gestart.

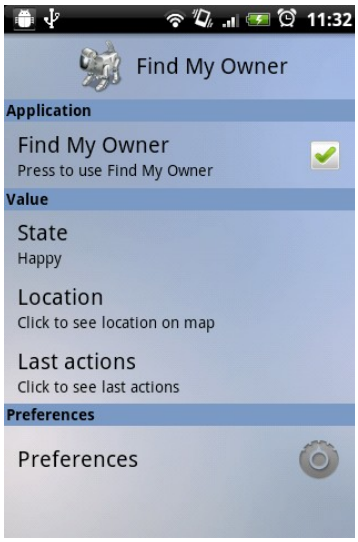
De belangrijkste actie van het systeem is het vinden van de eigenaar van de telefoon. De acties hiervoor worden uitgevoerd in het *nl.afst.owner.findaction package*.

Voor de acties wordt gebruik gemaakt van verschillende faciliteiten zoals een weergeven van een patroon of het verzenden van een e-mail. Deze klassen zijn verwerkt in het *nl.afst.owner.media package*.

Om connectie te maken met de Sense en ASK platforms zijn er klassen aangemaakt in het *nl.afst.owner.platforms package*.

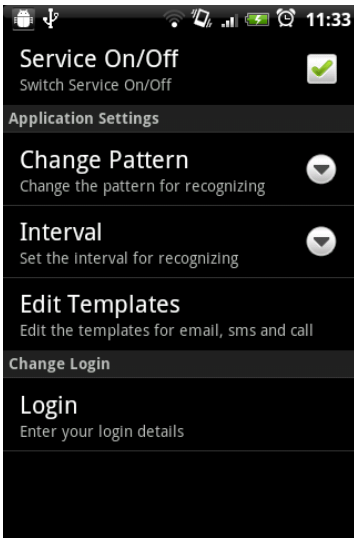
4.2 GUI

In onderstaande afbeeldingen zijn de screenshots van de eerste oplevering van de applicatie te zien. Aan de hand van deze afbeeldingen wordt het verschil besproken met het originele ontwerp.



Figuur 12: Basisscherm

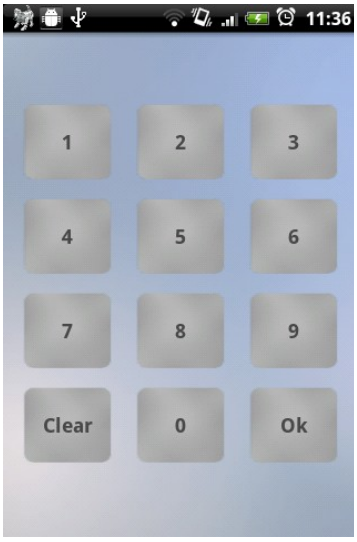
Figuur 13: Laatste acties



Figuur 14: Preferencescreen

Figuur 15: Notificatie voor herkennen

Figuur 16: Notificatie van herkennen



Figuur 17: Patroon ontgrendelen

Aan het hoofdscherm(figuur 12) is weinig veranderd ten opzichte van de basisversie. Enkel is het *Last Time Lost* veld veranderd in *Last Actions* (figuur 13). Hieraan is een lijst met laatste acties aan toegevoegd. Deze lijst is in deze versie enkel nog een weergave en wordt in latere versies reëel uitgewerkt.

Aan het Preferencescreen (figuur 14) zijn enkele instellingen toegevoegd en verwijderd omdat deze niet meer van toepassing waren of omdat er opties bij gekomen zijn. Zo is het in deze versie in te stellen om de hoeveel tijd de applicatie de stand-by stand moet controleren.

In figuur 15, 16 en 17 is de actie van herkennen te zien. De eigenaar krijgt een melding in de statusbalk en kan deze meldingen openen door de balk omlaag te schuiven. Op het moment dat de eigenaar op de melding klikt wordt het patroon weergegeven (figuur 17).

Origineel was het idee gebruik te maken van het Android patroon (figuur 18). Enkel is deze niet beschikbaar om ook in andere applicaties toe te passen. De mogelijkheid om de gebruiker te verplichten het patroon te gebruiken is er ook niet. In de Android instellingen op de smartphone is het vanaf versie 2.2 van Android niet mogelijk om de beveiligingsinstellingen te wijzigen. Dat wil zeggen instellingen als achtergrond en beltoon kunnen via code worden gewijzigd, maar instellingen voor pincode of ontgrendelingsmethode zijn niet te wijzigen.

De applicatie wordt ontwikkeld voor versie 2.1 van Android maar moet uiteindelijk ook kunnen draaien op 2.2 waardoor de keuze is gemaakt het patroon op een andere manier weer te geven.

Figuur 18: Android Patroon

De gebruiker kan in zijn instellingen het patroon instellen (een pincode met enkel cijfers). Bij het herkennen wordt dan een numeriek patroon weergegeven waarop de eigenaar zijn code invult(figuur 17). Als het patroon goedgekeurd wordt keert de telefoon terug naar het vorige scherm (bijvoorbeeld een andere applicatie of het hoofdscherm.

In code is de knop *home* niet af te vangen. Deze knop moet er altijd voor zorgen dat de gebruiker het hoofdscherm van de telefoon weer ziet. Daarnaast kan het zijn dat gebruiker de eerste keer de notificatie niet heeft gehoord. Om deze redenen is er gekozen een herhaling in de herkenningsactie te stoppen. Als de eigenaar na 5 minuten nog niet gereageerd heeft wordt de herkenningsactie opnieuw aangeroepen. Deze herhaling overschrijft de notificatie. Hierdoor wordt er maar één notificatie in de statusbalk gegeven.

4.3 Implementatie

De belangrijkste Android componenten die gebruikt zijn voor de eerste versie zijn

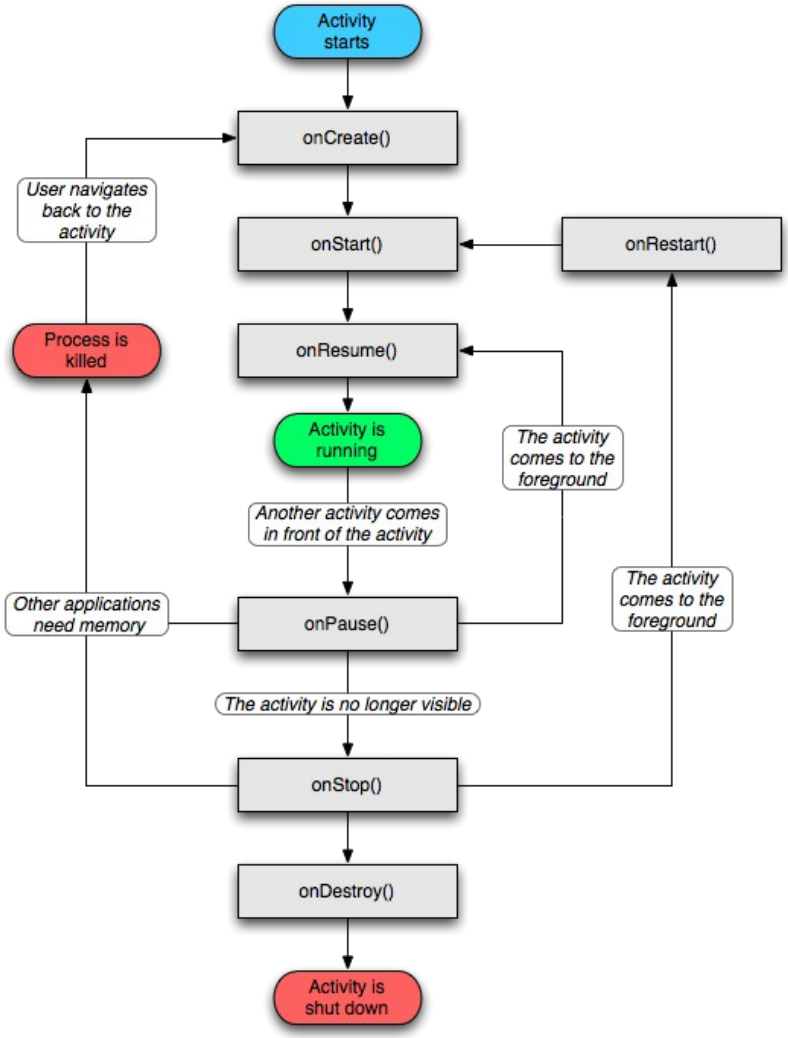
- Activity
- SharedPreferences
- Intent
- Context
- Service
- AlarmManager
- BroadcastReceiver

4.3.1 Activity

Processen binnen Android worden gepresenteerd als *Acitivities.* De business logica bevind zich in een *Activity* en is (meestal) gekoppeld aan een *view.* Hiermee kan de *Activity* reageren op handelingen van een gebruiker. Een *Activity* heeft een life-cycle, met een duidelijk begin en eind.

In de onderstaande afbeelding is de *life-cycle* van de *Activity* te zien. Een *Acitivity* heeft een aantal standaardfuncties. Deze worden aangeroepen op het moment dat er een status verandering plaatsvind zoals dat de activiteit op de achtergrond verdwijnt. De functies kunnen in de code van de applicatie worden overerfd om extra functionaliteiten aan de functie toe te voegen.

De *onCreate()* functie is een uitzondering. Deze wordt standaard aan de code van de applicatie toegevoegd. In deze functie wordt dan ook de view aan de *Activity* gekoppeld (te zien in codefragment 1).



Figuur 19: Activity Life-cycle

In de code van de "Find My Owner"-applicatie wordt *Activity* gebruikt bij de Owner klasse en de *lastActions* klasse. Voor het instellingenscherm is een speciale *Activity* gebruikt, namelijk de *PreferenceActivity.* De *PreferenceActivity* zorgt dat er een standaard instellingenscherm wordt weergegeven.

4.3.2 Shared Preferences

De *PreferenceActivity* bevat een key-value methode om instellingen op te slaan. Elke instelling krijgt een bepaalde *key* waaronder de instelling (*value*) wordt opgeslagen. Daarnaast bevat de PreferenceActivity een standaard opmaak voor het instellingenscherm waardoor hier een eenheid is bij elke applicatie.

De instellingen zijn op deze manier vanuit alle klassen op te vragen en aan te passen. Het opvragen is in codefragment 4 te zien. Hierin wordt in bij het creëren van de klasse een *SharedPreferences* variabele aangemaakt. De juiste *PreferenceManager* wordt hier aan toegekend. Vervolgens kan in *doSomething()* elke instelling worden opgevraagd door de *key* mee te geven waaronder de instelling is opgeslagen en een *default value*. Als een instelling niet gevonden wordt, wordt er in plaats van *null* de meegegeven waarde als resultaat gegeven.

```
@Override
public void onCreate() {
        shPrefs = PreferenceManager
                .getDefaultSharedPreferences(getBaseContext());

        useSharedPreferenceChangeListener();
        shPrefs.registerOnSharedPreferenceChangeListener(prefListener);

        doSomething();
}


public void doSomething() {
        if (shPrefs.getInt(Preferences.OWNER_STATE, Preferences.WONDERING) ==
                Preferences.WONDERING){
               RecOwn.goRecognizingOwner(this);
        }
}
```

Codefragment 4: Opvragen instelling (FindAction.java)

Om een instelling te wijzigen wordt er gebruik gemaakt van de *Editor.* De *Editor* wordt gekoppeld aan de *SharedPreferences*, waarna de instelling aangepast kan worden met de *key* en de nieuwe *value.* Als afsluiting moet altijd *commit()* worden aangeroepen om de instelling door te voeren.

```
Editor edit = shPrefs.edit();
edit.putInt(Preferences.OWNER_STATE, Preferences.WONDERING).commit();
```

Codefragment 5: Aanpassen instelling (FindAction.java)

In fragment 4 wordt ook de *OnSharePreferenceChangeListener* aangemaakt. Deze *Listener* kijkt of een instelling is gewijzigd. Als deze is gewijzigd wordt een actie uitgevoerd die hierbij van toepassing is. In de applicatie wordt de functie gebruikt om te kijken of de status veranderd is.

Er is voor gekozen de status op te slaan als instelling om deze vanaf elk punt in de applicatie opvraagbaar en aanpasbaar te maken. De basisopzet was een aparte klasse die deze status bij zou moeten houden, maar dan zou deze klasse vanuit elke andere klasse aangemaakt moeten worden of steeds meegegeven moeten worden. Het laatste geeft een onoverzichtelijke code en maakt de kans op fouten in afhandeling groter.

Doordat de status nu als een *SharedPreference* is opgeslagen kan er gebruik worden gemaakt van de *Listener*. Op het moment dat de status veranderd worden er acties die bij een bepaalde status horen gestopt en worden nieuwe acties gestart.

Zo wordt bij de status *Happy* een *StandByAlarm* aangeroepen om te kijken of de telefoon al langere tijd stand-by is. Als deze langere tijd stand-by is wordt de status op *Wondering* gezet en is het alarm niet meer nodig. Doordat de *Listener* een status verandering ziet, wordt er een actie uitgevoerd die zorgt dat het alarm stopt (codefragment 6).

```
public void useSharedPreferenceChangeListener() {
        prefListener = new OnSharedPreferenceChangeListener() {
        public void onSharedPreferenceChanged(
        SharedPreferences sharedPrefs, String key) {
                if (key.equals(Preferences.OWNER_STATE)) {
                                setAlarm();
                        }
                }
        };
}

private void setAlarm() {
        switch (shPrefs.getInt(Preferences.OWNER_STATE, Preferences.HAPPY)) {
                case Preferences.HAPPY:
                        ...
                        stopGoWonderingAlarm();
                        ...
                        break;
                case Preferences.WONDERING:
                        ...
                        stopStandByAlarm();
                        startGoWonderingAlarm();
                        ...
                        break;
                ...
                }
}
```

Codefragment 6: useSharedPreferenceChangeListener()

4.3.3 Intent

Een *Intent* is een abstracte beschrijving van een operatie die moet worden uitgevoerd. De *Intent* kan gebruikt worden:

- Een *Activity* aan te roepen samen met *startActivity()*
- Een *Service* te starten samen met *startService()*
- Een *BroadcastReceiver* Intent te sturen naar een belanghebbende *BroadcastReceiver*

Een *Intent* verzorgd de verbinding tussen de code en verschillende toepassingen. De belangrijkste toepassing is bij de start van *Activities*, waar de *Intent* kan worden gezien als lijm tussen de verschillende activiteiten. Een *Intent* is eigenlijk een passieve datastructuur die een abstracte beschrijving van een uit te voeren actie beschrijft. De standaardwaardes die worden meegegeven aan een *Intent* zijn:

- *Action:* De actie die uitgevoerd wordt
- *Data:* De data waarmee gewerkt wordt

```
String location = "geo:" + latitude + "," + longitude;
Uri geoUri = Uri.parse(location);
Intent mapCall = new Intent(Intent.ACTION_VIEW, geoUri);
startActivity(mapCall);
```

Codefragment 7: Intent (Owner.java)

In fragment 7 is de aanroep van een Intent te zien. Dit voorbeeld komt uit het aanroepen van Google Maps met de huidige locatie van de eigenaar. Als eerst wordt van de locatie een *Uri* gemaakt, deze wordt meegegeven als data aan de nieuwe *Intent*. Daarbij gaat het om data die moet worden weergegeven, als actie wordt ACTION_VIEW meegegeven. Met *startActivity()* wordt de activiteit gestart met het aangemaakte *Intent.*

4.3.4 Context

De *Context* is een interface die Informatie geeft over een applicatie-omgeving. De *Context* klasse is een abstracte klasse waarvan de uitvoer wordt verzorgd door Android. De *Context* geeft toegang tot applicatie specifieke klassen en middelen, maar bijvoorbeeld ook tot aanroepen van het starten van *Activities* en het uitzenden en ontvangen van *Intents.*

4.3.5 Service

Binnen een applicatie kan het voorkomen dat een langdurige actie op de achtergrond uitgevoerd moet worden, een actie zonder gebruikers interactie. Voor zo'n operatie kan een Service worden gebruikt. Het *Service* component kan gestart worden vanuit de applicatie en gestopt worden wanneer deze niet meer nodig is.

Een *Service* klasse werkt als een achtergrond proces maar valt wel onder één *Thread* met de andere processen, tenzij dit anders is gedefinieerd in de code. Op het moment dat een *Service* een zware operatie aan het uitvoeren is (bijvoorbeeld veel data verzenden) kunnen voorgrond processen hieronder lijden.

Binnen de applicatie is het *Service* component gebruikt voor de *OwnerService* en de *FindAction* klasse. Beide klassen zijn namelijk operaties die langere tijd en op de achtergrond moeten draaien.

Een *Service* wordt gestart met de *startService*() functie. Hieraan wordt de *Context* en de *Service* klasse meegegeven.

```
c.startService(new Intent(c, FindAction.class));
```

Codefragment 8: Service

Een *Service* component kan door zichzelf worden gestopt (fragment 9) of door een andere klasse (fragment 10).

```
stopSelf();
```

Codefragment 9: Stop Service klasse vanuit Service klasse zelf (OwnerService.java)

```
stopService(i);
```

Codefragment 10: Stop Service klasse vanuit andere klasse (Owner.java)

4.3.6 AlarmManager

De *AlarmManager* is simpel gezegd een wekker. De wekker wordt op een bepaalde tijd afgesteld en op dat tijdstip gaat het alarm af. Dat het alarm afgaat betekent dat een actie uitgevoerd moet worden.

In de code in fragment 11 wordt de *AlarmManager* gebruikt voor het checken van de stand-by stand van de telefoon. Op het moment dat het scherm uit gaat wordt het alarm gestart. Na vijftien minuten gaat het alarm "af". Op dat moment ontvangt de BroadcastReceiver voor een signaal (*onReceive()*).

```
//Aangeven welke actie er uitgevoerd moet worden op het moment dat het alarm afgaat

@Override
public void onCreate() {
...
mAlarmStandBy = PendingIntent.getBroadcast(this,
                StandByChecker.REQ_CHECK_STB, new Intent(this,
                StandByChecker.class), 0);
...
}

//Het alarm starten op het moment dat het scherm uit gaat (ACTION_SCREEN_OF)
public BroadcastReceiver bcrScreenOff = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            ...
            if (intent.getAction().equals(Intent.ACTION_SCREEN_OFF)) {
               Log.d(TAG, "ScreenOff");
                startStandByAlarm();
            }
            ...
            }
        }
    };

//Het starten van het alarm
public final long STB_INTERVAL = 1000 * 60 * 15;
public static long STB_FIRST_TIME;

public void startStandByAlarm() {
        STB_FIRST_TIME = SystemClock.elapsedRealtime() + STB_INTERVAL;
        AlarmManager am = (AlarmManager) getSystemService(ALARM_SERVICE);
        am.setRepeating(AlarmManager.ELAPSED_REALTIME_WAKEUP, STB_FIRST_TIME,
                STB_INTERVAL, mAlarmStandBy);

}
```

Codefragment 11: Alarm Stand-by (OwnerService.java)

4.3.7 BroadcastReceiver

Een *BroadcastReceiver* component ontvangt rondgestuurde signalen van bijvoorbeeld een alarm of actie. In fragment 11 is al een voorbeeld te zien van zo een *BroadcastReceiver.* Hierbij wordt de actie “scherm uit” afgevangen en een operatie uitgevoerd op het moment dat het scherm uit gaat.

In onderstaand fragment(12) is de *BroadcastReceiver* van *StandByChecker* te zien. Het alarm voor vijftien minuten stand-by is afgegaan, dat betekent dat de applicatie moet gaan twijfelen of de eigenaar aanwezig is. De status wordt op *Wondering* gezet en de applicatie gaat proberen de eigenaar te herkennen.

```
public class StandByChecker extends BroadcastReceiver {
        public static final String TAG = "StandByChecker";
        public static final int REQ_CHECK_STB = 1;

        SharedPreferences shPrefs;

        @Override
        public void onReceive(Context c, Intent i) {
                shPrefs = PreferenceManager.getDefaultSharedPreferences(c);

                Editor edit = shPrefs.edit();
                edit.putInt(Preferences.OWNER_STATE, Preferences.WONDERING).commit();
                c.startService(new Intent(c, FindAction.class));
        }
}
```

Codefragment 12: BroadcastReceiver StandByChecker

5. Tweede Oplevering

In de tweede oplevering is de connectie met ASK en Sense opgezet. Hierbij worden berichten verstuurd naar CommonSense die de toestand en de uitvoerende actie van de applicatie bevatten.

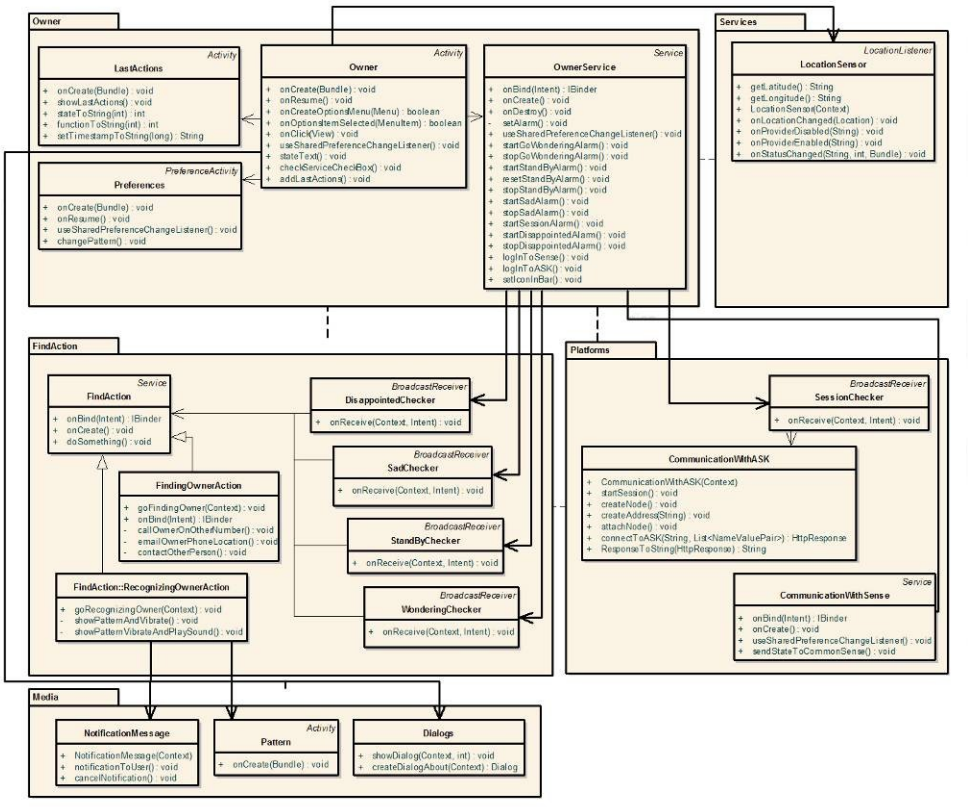
Via ASK wordt er contact opgenomen met iemand anders via een SMS. De werking hiervan is geheel in de applicatie gebouwd. Aan de hand van deze basis wordt in de derde versie de reële werking van de vind-acties uitgewerkt.

Na de tweede versie kan de applicatie opzoek naar de eigenaar aan de hand van een SMS en wordt de status doorgestuurd naar CommonSense. Zo kan de eigenaar zien welke acties er uitgevoerd zijn.

De werking en het ontwerp van de twee onderdelen wordt in de volgende paragrafen besproken.

In de GUI is in de tweede versie weinig veranderd. Enkel is de *LastActions* lijst beter uitgewerkt waardoor de acties die naar CommonSense worden gestuurd ook worden weergegeven in de lijst. Deze lijst is nog niet optimaal en wordt in derde oplevering beter uitgewerkt en besproken.

5.1 Samenhang van *package*s en klassen in de tweede oplevering



Figuur 20: Klassendiagram - tweede oplevering

In figuur 20 is het klassendiagram van de tweede oplevering van de applicatie te zien[1]. Aan de het diagram zijn de klassen en functies toegevoegd die in de tweede oplevering zijn geïmplementeerd. Zo is in de tweede versie de verbinding met ASK en Sense opgezet. Hieraan toegevoegd is de SessionChecker die elke drie kwartier een nieuw sessionId aanvraagt bij ASK.

Uit het *Media package* zijn de klassen SMS, Call en Email gehaald. Deze worden namelijk in de applicatie niet direct gebruikt. Een email, sms of telefoontje wordt opgezet in de AKS webinterface zelf.

Aan het Owner *package* is de klasse lastActions toegevoegd. Deze klasse verzorgd de lijst met laatste acties die zijn uitgevoerd.

In het diagram bij de eerste versie is te zien dat de klasse FindAction de klassen FindingOwnerAction en RecognizingOwnerAction gebruikt. In de tweede versie is het zo omgebouwd dat de twee klassen afgeleide klassen zijn van de FindAction klasse.

5.2 Implementatie

5.2.1 Verbinding met CommonSense

In het elaboration fase rapport is een beschrijving te vinden van de verbinding met CommonSense. Binnen de applicatie is deze opbouw ook gebruikt om de verbinding te maken met CommonSense.

```
public static final String ACTION_NEW_MSG = "nl.sense_os.app.MsgHandler.NEW_MSG";
public static final String ACTION_NEW_FILE = "nl.sense_os.app.MsgHandler.NEW_FILE";
public static final String ACTION_SEND_DATA = "nl.sense_os.app.MsgHandler.SEND_DATA";
public static final String KEY_DATA_TYPE = "data_type";
public static final String KEY_SENSOR_DEVICE = "sensor_device";
public static final String KEY_SENSOR_NAME = "sensor_name";
public static final String KEY_TIMESTAMP = "timestamp";
public static final String KEY_VALUE = "value";
public static final String NAME = "find my owner";

public void sendStateToCommonSense() {
        Thread t = new Thread() {
                @Override
                public void run() {
                        JSONObject json = new JSONObject();
                        try {
                                switch (shPrefs.getInt(Preferences.OWNER_STATE, 0)) {
                                case Preferences.HAPPY:
                                        json.put("State", "Happy");
                                        break;
                                case Preferences.WONDERING:
                                        json.put("State", "Wondering");
                                        break;
                                ...
                                case Preferences.JEALOUS:
                                        json.put("State", "Jealous");
                                        break;
                                default:
                                        json.put("State", "Unknown state");
                                }

                                switch (shPrefs.getInt(Preferences.OWNER_STATE_ACTION,
                                                Preferences.FUNC_UNK)) {
                                case Preferences.FUNC_UNK:
                                        break;
                                case Preferences.REC_PATTERN_VIBRATE:
                                        json.put("Function", "Show pattern and
```

1 *De grote versie van het klassendiagram is te zien in bijlage B*

```
                                        vibrate");
                                json.put("Function id",
                                        Preferences.REC_PATTERN_VIBRATE);
                                break;
                        ...
                        case Preferences.REC_PATTERN_VIBRATE_SOUND:
                                json.put("Function", "Show pattern,
                                        vibrate and play sound");
                                json.put("Function id",
                                        Preferences.REC_PATTERN_VIBRATE_SOUND);
                                break;
                        case Preferences.FIND_EMAIL_OTHER_PERSONS:
                                json.put("Function", "E-mail other
                                        person");
                                json.put("Function id",
                                        Preferences.FIND_CALL_OTHER_PERSONS);
                                break;
                        ...
                        default:
                                break;
                        }

                } catch (JSONException e) {
                        Log.e(TAG, "JSONException in
                                CommunicationWithSenseAPI", e);
                        return;
                }

                Intent i = new Intent(ACTION_NEW_MSG);
                i.putExtra(KEY_SENSOR_NAME, NAME);
                i.putExtra(KEY_VALUE, json.toString());
                i.putExtra(KEY_DATA_TYPE, Preferences.SENSOR_DATA_TYPE_JSON);
                i.putExtra(KEY_TIMESTAMP, System.currentTimeMillis());
                tContext.startService(i);
            }
        };
        t.start();
}
```

Codefragment 13: Zenden van statusinformatie naar CommonSene (CommunicationWithSense.java)

1816284	find my owner	**State:** Wondering **Function:** Service On **Function id:** 8	2011 Mar 30 14:44:38

Figuur 21: State in CommonSense

Om de informatie in figuur 20 weer te geven in CommonSense wordt de bovenstaande code uitgevoerd. Omdat de Sense applicatie ook draait op de smartphone is het mogelijk hier de MsgHandler van aan te spreken en hoeft er geen losse afhandeling van berichten te worden opgesteld. Op het moment dat de ACTION_NEW_MSG wordt aangeroepen, luistert de MsgHandler van de Sense applicatie en verwerkt deze het bericht wat gestuurd wordt.

De *"Message"* die verzonden wordt moet eerst worden opgebouwd. De *Message* is een *JSON Object*, een soort pakketje met verschillende berichten erin. Allereerst wordt het JSON Object aangemaakt en aan de hand van de waardes van *state* en *action* worden verschillende berichten toegevoegd aan het object.

De *state* is de toestand waar het systeem zich op dat moment in bevindt. In elke toestand kunnen een aantal acties *(actions)* worden uitgevoerd. Op het moment dat het systeem een andere actie uit gaat voeren wordt deze actie opgeslagen en ook doorgestuurd naar CommonSense.

Afhankelijk van de *state* wordt er een bepaalde tekst gestuurd naar CommonSense. Aan de hand van de *json.put*(*String*, *String*) wordt het bericht aan het *JSON Object* toegevoegd.

Van de actie wordt een tekst gemaakt en het ID wordt meegegeven. Het ID kan namelijk later worden gebruikt als herkenning voor CommonSense met welke actie de applicatie als laatst bezig is geweest. Op het moment dat de telefoon uitvalt kan de actie dan overgenomen worden CommonSense en ASK.

De gehele taak wordt uitgevoerd in een aparte *Thread*, zo kan het versturen van het bericht zonder onderbroken te worden doorgaan.

Op het moment dat Sense niet draait op de telefoon wordt er nog geen melding gegeven aan de gebruiker. Deze melding of afhankelijkheid wordt in een latere versie van de applicatie wel toegevoegd.

5.2.2 Verbinding met ASK

Voor de vind-acties wordt er gebruik gemaakt van een ASK installatie. Voor de applicatie is een aparte ASK installatie opgezet waar allerlei informatie aan toegevoegd kan worden. De installatie is te bereiken via: http://ask-dev.ask-cs.nl/~marleine/ask/.

5.2.2.1 Onderzoek naar protocollen voor benadering webservices

Om de ASK installatie aan te kunnen spreken vanuit de *"Find My Owner"* applicatie wordt er gebruik gemaakt van de ASK API. Via de SOAP berichten is het mogelijk om de API aan te spreken. Om vanuit een Android applicatie een SOAP bericht te sturen zijn er meerdere mogelijkheden:

Android standaard *library*

Android biedt zelf geen *library* voor het gebruik van SOAP binnen een applicatie. Het is wel mogelijk enkel moet de gebruiker hiervoor zelf een *library* toevoegen aan het project.

kSOAP2

Een populaire library voor het gebruik van SOAP binnen JAVA/Android is kSOAP2. De library biedt een overzichtelijke manier van aanspreken van de webservices. Echter is het gebruik niet mogelijk bij de Find My Owner applicatie, of eerder gezegd de ASK installatie. De webservices worden namelijk aangesproken via een WSDL (Web Service Description Language). Voor kSOAP2 is het van belang dat deze WSDL is gegenereerd aan de hand van het dotNet (.Net) framework, en dus een .asmx extensie heeft. De WSDL van de ASK installatie is gegenereerd aan de hand van PHP en is als volgt te bereiken: http://ask-dev.ask-cs.nl/~marleine/ask/webservices/index.php?wsdl. Een .asmx extensie is hierbij dus niet aanwezig. Ondanks dat WSDL een standaard is voor het beschrijven van Webservices, is de kSOAP library niet generiek en niet te gebruiken voor de "*Find My Owner*" applicatie.

WSClient++

WSClient biedt een generator voor het genereren van code aan de hand van een WSDL link. De code is te genereren in verschillende programmeertalen, ook voor Android Java. Echter is de WSClient++ een betaalde applicatie en wordt hier liever niet voor gekozen bij de ontwikkeling van de applicatie.

RESTful4

RESTful (REpresentational State Transfer) is een nieuwere versie van SOAP. Het is het benaderen van webservices op een andere manier met een "uniform interface". De webservices zijn benaderbaar met vijf verschillende functies (GET, POST, HEAD, DELETE en

PUT). De benadering vanuit XML opzicht is ook net iets anders opgebouwd. Resources zijn namelijk ook via pointer bereikbaar, in plaats van de gehele XML tree terugsturen.

RESTful is een optie in in Android maar is ook lastiger te implementeren. De voordelen van RESTful zijn niet van toepassing voor de applicatie. Voor het implementeren van de webservice calls is een simpelere optie handiger.

Andere opties

Naast bovenstaande opties zijn er ook nog meer mogelijkheden gevonden op internet. Na onderzoek waren deze ook niet toepasbaar genoeg voor de Android applicatie. Hierdoor worden ze ook verder niet uitgebreid besproken. De andere opties zijn:

- SAX (Simple API form XML)
- oData
- NuSOAP

Uiteindelijke keuze: HttpClient

HttpClient is een client-side HTTP communicatie component binnen Android. Aan de hand van de HttpClient klasse kan er verbinding worden gemaakt met de webservices via HTTP.

Deze optie is gekozen voor de Find My Owner applicatie omdat deze een simpele implementatie heeft. De basis die nodig is voor de applicatie is gemakkelijk te implementeren met HttpClient.

De oplossing die gekozen is niet de meest nette oplossing. Bij de eerste opzet gaat de data namelijk over HTTP en kan er gemakkelijk een hacker een injectie uitvoeren waardoor de data uitgelezen wordt. Voor het gebruik van de webservices is namelijk enkel een *Authentication key* nodig, die op die manier makkelijk te onderscheppen is. Als deze key bekend is kan er gemakkelijk ingebroken worden op het systeem.

Om de HttpClient uiteindelijk verantwoordelijker te gebruiken, kan er gebruik gemaakt worden van Https (Secure HTTP).

In de volgende paragraaf is uitgebreid beschreven hoe het HttpClient is toegepast in de code.

5.2.2.2 Uitwerking gebruik HttpClient

Om de webservices aan te spreken wordt er gebruik gemaakt van:

WSDL

De WSDL (WebService Description Language) wordt via de bovenstaand beschreven URL aangesproken vanuit de PHP functies.

PHP functies

Los van de applicatie zijn de verschillende benodigde functies uitgewerkt in PHP. De webservices worden aangesproken in de PHP functies. De variabelen die hiervoor nodig zijn worden meegestuurd met de aanroep vanuit Android.

Android applicatie / HttpClient component

In Android wordt de HttpClient aangemaakt en wordt daarmee de PHP pagina aangeroepen. Voor elke functie is een losse PHP pagina gemaakt. Om de functies te gebruiken zijn

gegevens (variabelen) nodig. Deze worden vanuit Android meegegeven als een *NameValuePair.*

Als de functie is aangeroepen komt er vanuit ASK een response. Deze wordt opgevangen door de PHP functie en wordt weer teruggegeven aan de applicatie. De applicatie kan aan de hand van de response kijken of het geslaagd is, of de waarde opslaan voor later gebruik (bijvoorbeeld *SessionID*.

De opzet met de aanroep van verschillende webservice functies is in de tweede oplevering van de applicatie uitgewerkt voor de volgende functies:

- *startSession*
  Het starten van een sessie aan de hand van de *authentication key.* De sessie is een uur geldig. In de applicatie is een *AlarmManager* opgezet die na 45 minuten opnieuw een sessie opzet, zodat de verbinding met ASK altijd blijft bestaan.

  De mogelijkheid is ook om elke keer als er een aanroep wordt gedaan een sessie op te zetten maar dan bestaat de kans dat er meerdere functies achter elkaar worden aangeroepen. Hierdoor worden er in korte meerdere sessies opgezet. Dit kost veel CPU tijd van de server en dat is zonde. Een controle op tijd (een timer) is het handigste op te zetten aan de hand van de *AlarmManager*.
- *createNode*
  Het creëren van een nieuwe Node met een uniek UUID en naam voor de node.
- *createAddress*
  De node heeft voor het bellen / sms-en adressen nodig. Deze kunnen toegevoegd worden aan een node aan de hand van de createAddress functie. Hierin wordt het type adres meegegeven om aan te geven om wat voor soort adres het gaat (bijvoorbeeld “PHONE” of “EMAIL”).
- *attachNode*
  Om een node (bijvoorbeeld de eigenaar) te koppelen aan een andere node (bijvoorbeeld de sms groep) wordt de functie attachNode gebruikt.

In onderstaande codefragmenten is de verbinding tussen *Find My Owner* en ASK te zien aan de hand van de *createNode* functie.

In het eerste fragment is het deel van de *createNode* functie te zien in de WSDL. In XML staat beschreven welke *request* en *reponse* functies er zijn en welke waardes hieraan meegegeven moeten worden.

```
<message name="createNodeRequest">
        <part name="sessionID" type="xsd:string"/>
        <part name="nodeUUID" type="xsd:string"/>
        <part name="data" type="tns:tupleArray"/>
</message>
<message name="createNodeResponse">
        <part name="return" type="tns:stringResponse"/>
</message>
```

Codefragment 14: Onderdeel WSDL met CreateNodeRequest / Response

In onderstaand fragment is de php functie createNode te zien. Als eerste worden de gegevens die meegestuurd zijn in variabelen gezet. Hiermee wordt de functie createNode aangeroepen en uitgevoerd. In de functie wordt de WSDL url aangeroepen aan de hand van een SoapClient.

Aan de createNode functie worden drie verschillende waardes meegegeven. Namelijk SessionID, het sessieID. Een SessionID is op te vragen aan de hand van de authenticatie key en de functie StartSession. Deze is bij deze functie al uitgevoerd en het SessionID wordt meegeegven vanuit de Android applicatie.

In de webservices kan de gebruiker zelf bepalen wat het *NodeUUID* wordt van de nieuwe node. Als de node namelijk wordt aangemaakt via de webinterface wordt er standaard een UUID aan de node gegeven.
Als de node goed wordt toegevoegd vanuit de webservices, wordt de *NodeUUID* als return teruggegeven. Het UUID is een uniek nummer en voor geen van de nodes gelijk. Op het moment dat de node al bestaat wordt er *null* teruggegeven.

Daarnaast kan er aan de node ook al extra informatie worden meegegeven. Bijvoorbeeld de naam van de node. Aan de hand van een *name-value tuple array* wordt bijvoorbeeld de naam meegegeven.

```
$sessionID     = htmlspecialchars($_POST["sessionID"]);
$nodeUUID      = htmlspecialchars($_POST["nodeUUID"]);
$name          = htmlspecialchars($_POST["name"]);

$receivedNodeUUID = createNode($sessionID, $nodeUUID, $name);

if ($receivedNodeUUID != null){
        echo $receivedNodeUUID;
} else {
        echo "Node already exists";
}

function createNode($sessionID, $nodeUUID, $name){

        $webServiceURI = 'http://ask-dev.ask-
                cs.nl/~marleine/ask/webservices/index.php?wsdl';

        $webServiceClient = new SoapClient($webServiceURI);

        $_createNode = $webServiceClient->createNode($sessionID, $nodeUUID,

        array   (
                        array   (
                                'name' => 'name',
                                'value' => $name
                                )
                        )
        );

        $NUUID = $_createNode->result;

        return $NUUID;

}
```

Codefragment 15: PHP Functie CreateNode (CreateNode.php)

Om de PHP functie aan te roepen moet er in de applicatie een verbinding worden opgezet. Dit gebeurt in de *connectToASK* functie. Hierin wordt de *HttpClient* aangemaakt met de bijbehorende URL naar de PHP pagina. Deze wordt vanuit een van de functies (bijvoorbeeld *CommunicationWithASK::createNode()*) meegegeven. Aan de *HttpPost* wordt de array met *NameValuePairs* meegegeven die de variabelen voor de functie bevat.

Deze *NameValuePairs* worden toegevoegd aan een *List<NameValuePair>* in de *CommunicationWithASK*::*createNode*() functie. Alle waardes worden in de *SharedPreferences* opgeslagen, zodat ze bereikbaar zijn vanuit verschillende locaties in de applicatie.

De response vanuit de PHP aanroep wordt opgevangen in een *HttpResponse*. Deze response wordt teruggegeven aan de functie en wordt daar vertaald (aan de hand van de *HttpHelper* klasse) naar een string. Deze string kan dan weer opgeslagen worden als *SharedPreference* of als *Log* waarde worden weergegeven.

```
public HttpResponse connectToASK(String url, List<NameValuePair> nvp) {
        // Create a new HttpClient and Post Header
        HttpClient HttpClient = new DefaultHttpClient();
        HttpPost httppost = new HttpPost(url);

        try {
                // Add your data
                httppost.setEntity(new UrlEncodedFormEntity(nvp));

                // Execute HTTP Post Request
                HttpResponse response = HttpClient.execute(httppost);
                return response;

        } catch (ClientProtocolException e) {
                return null;
        } catch (IOException e) {
                return null;
        }
}
```

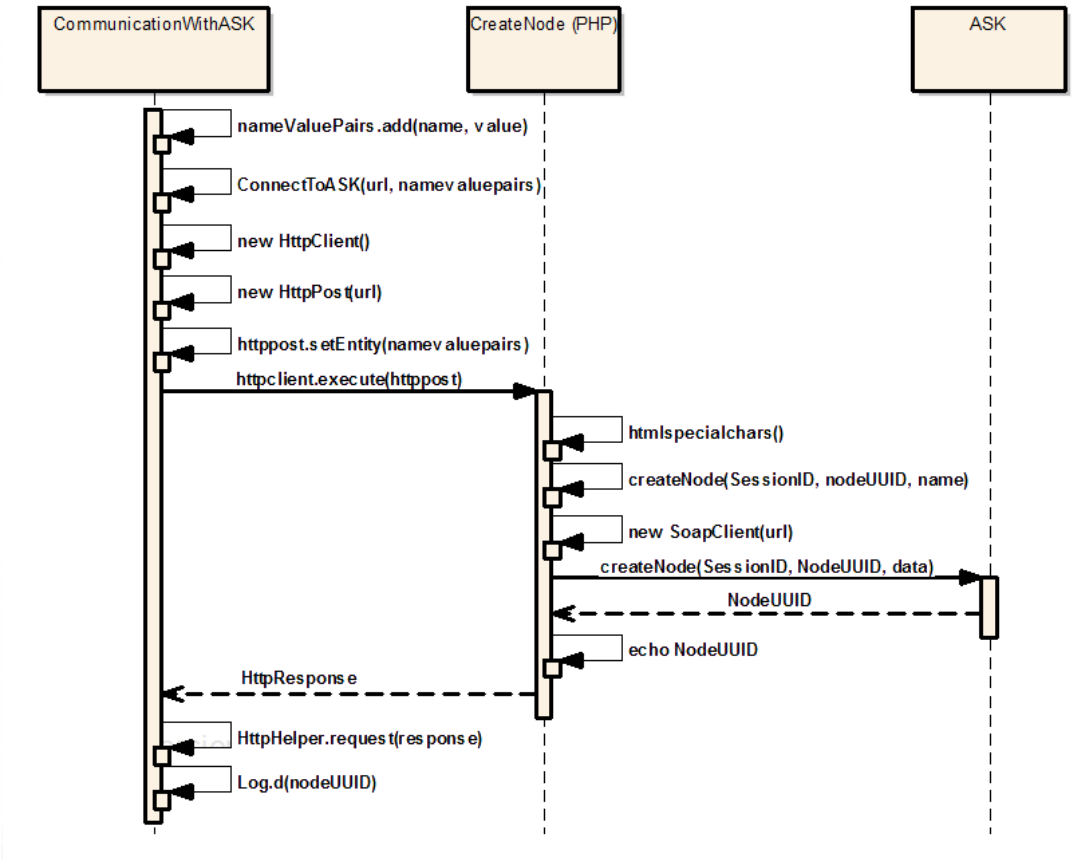
Codefragment 16: connectToASK (CommunicationWithASK.java)

```
public void createNode(){
        List<NameValuePair> nameValuePairs = new ArrayList<NameValuePair>(3);
        nameValuePairs.add(new BasicNameValuePair("sessionID",
                shPrefs.getString(Preferences.ASK_SESSIONID, "")));

        nameValuePairs.add(new BasicNameValuePair("nodeUUID",
                shPrefs.getString(Preferences.PREF_ASK_NODE_UUID, "")));
        nameValuePairs.add(new BasicNameValuePair("name",
                shPrefs.getString(Preferences.PREF_ASK_NODE_UUID, "")));

        HttpResponse responseCreateNode = connectToASK(URL_CREATENODE,
                nameValuePairs);

        String respNodeUUID = HttpHelper.request(responseCreateNode);
        if (respNodeUUID != null){
                Log.e(TAG, "NodeUUID: " + respNodeUUID);
        } else {
                Log.e(TAG, "NodeUUID: empty, Node already exists" );
        }
}
```

Codefragment 17: createNode (CommunicationWithASK.java)

In het onderstaande diagram is de verbinding weergegeven. Het diagram is opgebouwd uit het voorbeeld van de *createNode* functie.

Allereerst wordt de verbinding met de PHP pagina opgezet vanuit de applicatie. Hierbij worden ook de variabelen meegegeven. Vervolgens wordt in PHP de array met variabelen uitgepakt en wordt de functie bij ASK aangeroepen. Bij ASK wordt de Node aangemaakt en wordt het resultaat teruggegeven aan de PHP pagina. De PHP pagina geeft het resultaat terug en de applicatie pikt dit op. De applicatie verwerkt vervolgens de response in het log.



Figuur 22: Verbinding met ASK / createNode functie

6. Derde oplevering

In de derde oplevering zijn alle vind-acties uitgewerkt. Hierbij is vooral de verbinding met ASK en de opzet binnen ASK van belang.

Daarnaast zijn er kleine veranderingen gemaakt in het systeem om bugs weg te werken of om het systeem netter / beter te laten werken. Het gaat hierbij om verbeteringen als:

- Locatie van de eigenaar ophalen en verwerken (aan de hand van Google Maps API)
- Herkenningsacties aanroepen
- Alarmen voor toestand verandering
- Opslag van instellingen / constanten

In dit hoofdstuk wordt uitgebreid de werking binnen en met ASK besproken. Zo worden er voorbeelden gegeven hoe de instellingen binnen ASK zijn opgelost. De kleine aanpassingen in het systeem worden kort toegelicht.

Als eerst wordt het klassendiagram gegeven van de derde oplevering. Door de aanpassingen van de applicatie en de uitbreiding van ASK zijn er *package* veranderingen en functie veranderingen te zien in het diagram.

6.1 Samenhang en verantwoordelijkheden van de klassen

Het klassendiagram van de derde en vierde oplevering is te vinden in hoofdstuk zeven. In de vierde oplevering is er namelijk maar één onderdeel dusdanig aangepast wat verschilt met de derde oplevering. Het gaat hierbij om de toevoeging van de werking van de *Jealous* toestand.

In hoofdstuk zeven wordt dan ook het uiteindelijke klassendiagram besproken, met daarin de logica en afhankelijkheden van *package*s.

6.2 Implementatie

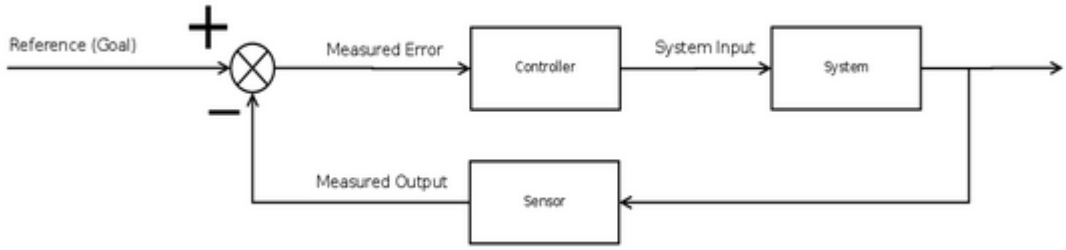
6.2.1 Vind-acties door middel van ASK en Android

Om een helder verhaal te geven bij de implementatie van de communicatie met ASK is een inleiding op het ASK platform van belang.

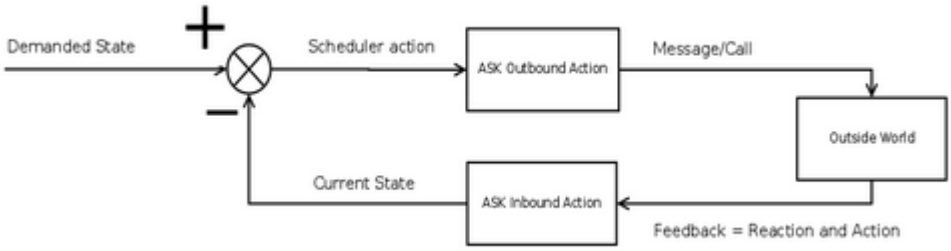
In 2004 is het ASK platform opgezet als communicatieplatform. Toen de tijd enkel nog met de mogelijkheid voor bellen maar later in de tijd zijn er verschillende media (e-mail, sms) bijgekomen. Momenteel wordt uitbreid naar andere platforms als Twitter, Facebook en LinkedIn.

ASK bevat een dynamisch netwerk van agent die reageren op toestandsveranderingen of processen waar ze verantwoordelijk voor zijn. De agenten zijn zo gebouwd dat ze een bepaalde toestand moeten bewaken, als deze toestand veranderd moet de agent een andere agent waarschuwen of contact opnemen met de mens.

De basis van ASK is gebaseerd op een simpel elektrisch systeem. Een systeem waarbij een wens (demand) aan het systeem wordt gegeven en het systeem zo werkt dat het resultaat gelijk is aan de wens. Om te controleren of het resultaat gelijk is aan de wens wordt een sensor toegevoegd. In onderstaande figuren is te zien hoe dit er in een elektrisch schema uitziet en hoe dit in ASK is geïnterpreteerd.



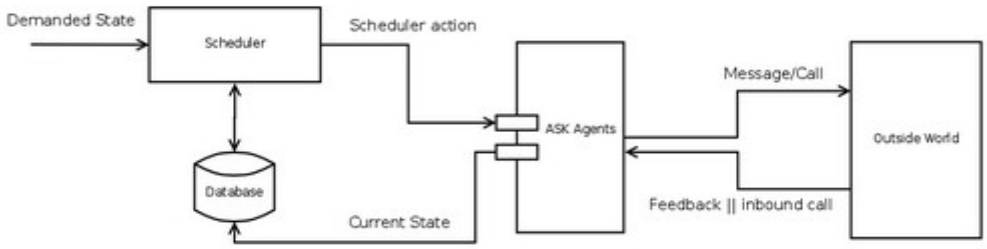
Figuur 23: Elektrisch schema (denk aan thermostaat)



Figuur 24: Elektrisch schema toegepast in ASK

In ASK wordt de wens gezien als de status die een onderdeel binnen het systeem moet hebben, bijvoorbeeld een groep moet minimaal tien personen bevatten. Om dit voor elkaar te krijgen voert het systeem een actie uit, bijvoorbeeld een groep mensen bellen. Hierna wordt opnieuw gecontroleerd of de groep tien personen bevat. Zo ja, het systeem is tevreden en wacht op de volgende toestandsverandering. Zo niet, het systeem voert nog een actie uit om het resultaat gelijk te krijgen aan de wens.

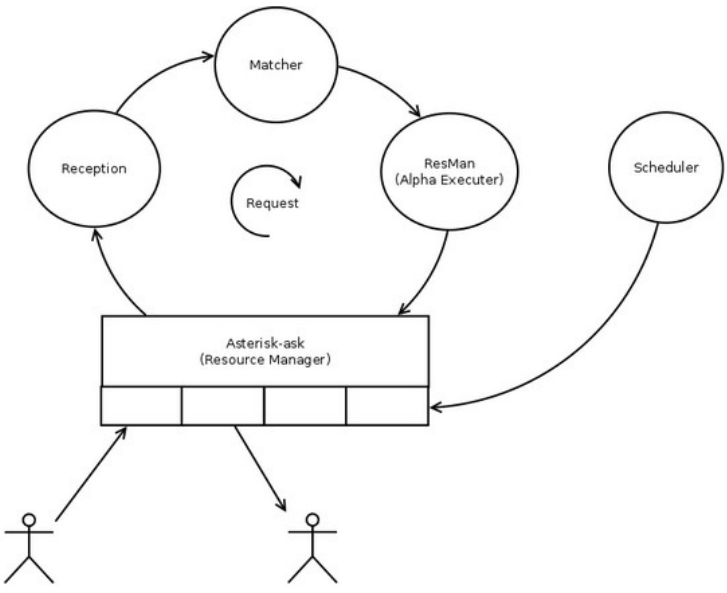
In onderstaande afbeelding is te zien hoe het schema uitgebreider in ASK zit. In ASK speelt namelijk ook een database met alle waardes en gegevens mee en de agenten zelf. De Scheduler geeft aan wanneer een actie uitgevoerd moet worden en geeft deze actie aan een van de agenten. De agent voert de opdracht uit en krijgt de toestand van de “outside world” en zet deze in de database. De Scheduler controleert vervolgens weer of de wens gelijk is aan de waarde in de database. Als dit niet zo is wordt er opnieuw een agent opgezet.



Figuur 25: Elektrisch systeem uitgebreid toegepast in ASK

Binnen het systeem zijn vier verschillende agenten te beschrijven (te zien in onderstaand figuur):

- Resource Manager
  Verantwoordelijk voor het leggen van een connectie tussen twee partijen (twee personen, één persoon en IVR-menu) via o.a. telefoon. In de huidige versie is het nog niet direct mogelijk om sms berichten via de Asterisk te sturen. Hiervoor is een apart SOAP Client opgezet om tekstberichten op te sturen (TxtXchange).
- Reception
  De receptie verzorgt de afhandeling van scripts. De scripts vertellen welke stappen er uitgevoerd moeten worden.
- Matcher
  De *matcher* zoekt bij de aanvraag voor de connectie de juiste tegenpartij uit. Het selecteren kan gaan om direct een ID maar ook via een bepaald algoritme voor de persoon met de hoogste rating.
- Scheduler
  De *scheduler* zorgt ervoor dat het systeem de acties gaat uitvoeren.



Figuur 26: Request loop binnen ASK

Binnen ASK werd tot 2010 verschil gemaakt tussen gebruikers en groepen. Zo kon een gebruiker resources, als telefoonnummer en e-mail, bevatten en een groep kon een set van gebruikers bevatten. In de vernieuwde opzet is er geen onderscheid meer tussen deze partijen. Er wordt enkel nog gesproken over *Nodes*. Een node kan meerdere nodes bevatten maar kan zelf ook resources hebben. Zo zijn beide groepen simpel gezegd gecombineerd. Het probleem hierbij is nog dat er geen duidelijk onderscheid meer gemaakt kan worden tussen een gebruiker en een groep op het moment dat bijvoorbeeld alle gebruikers binnen een groep een e-mail moeten ontvangen.

Om deze laatste reden zijn er een paar keuzes gemaakt binnen de ontwikkeling van de *Find My Owner* applicatie, om andere oplossingen te kiezen. Als het nieuwe systeem met nodes beter in ASK zit kan er een soms makkelijkere weg gekozen worden. Waar deze keuzes een rol spelen wordt later toegelicht.

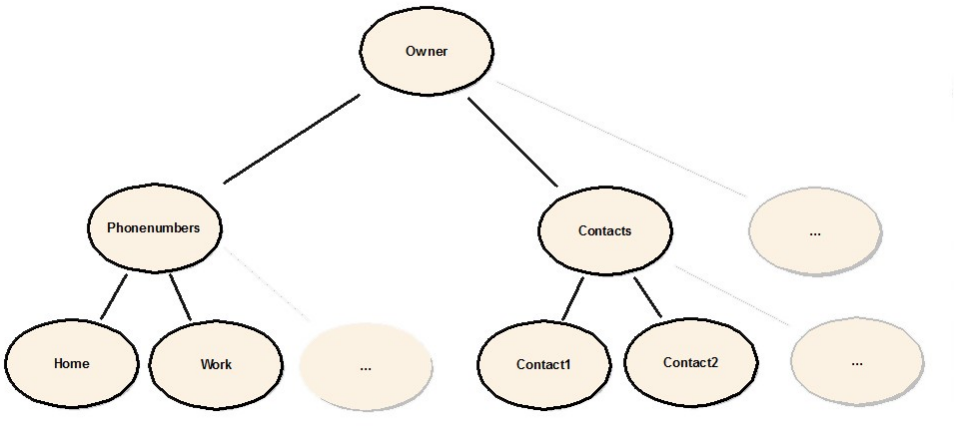
<u>Implementatie van het vinden van de eigenaar door middel van een e-mail</u>
Opvolgend worden de volgende vind-acties uitgevoerd:

- E-mail locatie naar eigenaar
- Bel de eigenaar op een ander telefoonnummer (bijv. werk- of thuisnummer)
- Neem contact op met een contactpersoon van de eigenaar
  (via e-mail, sms of e-mail)

Deze vind-acties zijn in werking naar de gebruiker toe anders maar qua implementatie lijken deze grotendeels op elkaar. Hierdoor wordt er in het implementatiedocument maar één van de vind-acties uitgewerkt. Namelijk het versturen van een e-mail naar contactpersonen van de eigenaar. Het enige verschil wat er tussen de implementatie van alle acties zit is de opzet van het bericht (telefoonbericht, sms of e-mail) zelf.

Het opzetten van een vind-actie[2] bestaat uit twee onderdelen. Namelijk het inrichten van de agents in ASK en de functie voor het sturen van een e-mail in de applicatie.

Voordat er überhaupt een actie uitgevoerd gaat worden moet de eigenaar worden toegevoegd als node. Voor het toevoegen is een graaf opgezet (figuur 26) die weergeeft welke nodes er aan de “eigenaarnode” worden gehangen.



Figuur 27: Graaf van ownernode

2 *In deze context ook benoemd als het sturen van een e-mail naar een andere contactpersoon*

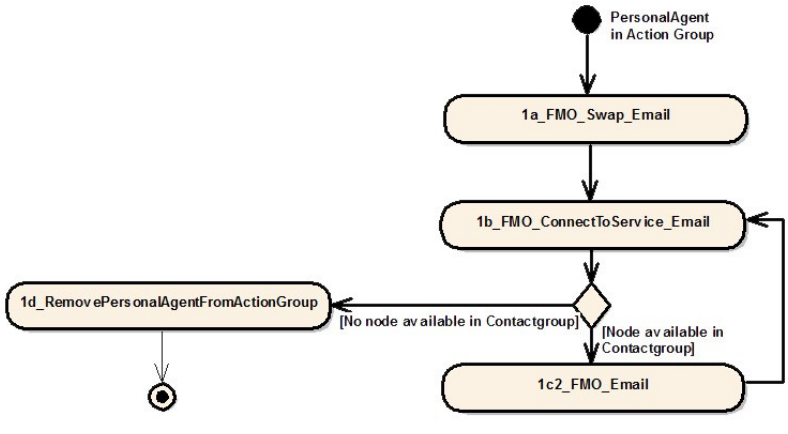
Voor het contact opnemen met de eigenaar op andere nummers en contact opnemen met andere personen zijn er namelijk ook gegevens nodig. Deze nodes worden los aangemaakt en daarna aan de eigenaar gehangen onder de groepen waar ze bij horen *(contactpersoon* of *contactgegevens*).

Het is van belang dat duidelijk is hoe de nodes gekoppeld zijn omdat de juiste nodes gezocht moeten worden bij het contact opnemen. Zo moeten de nodes uit de *contactpersoongroep* worden opgezocht als er contact opgenomen wordt met een van de contactpersonen en vica versa voor de *contactgegevensgroep*.

Binnen ASK moeten de volgende onderdelen worden ingesteld om te zorgen dat een e-mail wordt gestuurd naar een contactpersoon van de eigenaar van de telefoon.

- Een node aanmaken die functioneert als groep, bijvoorbeeld *EmailActionContacts-OfOwner.* Op het moment dat de actie uitgevoerd moet worden, wordt de *Owner-node*[3] in deze groep gehangen.
- Een agent (*scheduler node*) aanmaken. Deze agent heeft een planboard waarop aangegeven wordt op welke momenten de scheduler actief is. De agent controleert elke minuut of er een node in de groep *EmailActionContactsOfOwner* zit.
- De scheduler wordt gekoppeld aan een groep en een taak in een *job.* Een van de jobs is een *Service Availability.* Hierin wordt aangegeven dat de scheduler elke minuut op de groep moet controleren. Als er meer dan nul personen in de groep zitten moet er een actie (IVR menu) worden uitgevoerd.
- Het IVR menu zorgt dat de juiste contactpersonen worden opgezocht (in de contactpersoon groep) en zorgt dat er daadwerkelijk contact mee wordt opgenomen. De IVR stappen worden afgesloten met het verwijderen van de eigenaar uit de *EmailActionContactsOfOwner groep.*

In ASK is het niet mogelijk om alle stappen van de actie in één IVR te zetten. Hierdoor zijn er vier IVR menu's opgezet. Deze zijn weergegeven in onderstaand figuur.



Figuur 28: IVR Menu's contactopenemen met eigenaar

De basis van de actie is het in contact brengen van twee nodes. De interne gebruiker is de gebruiker waar contact mee wordt opgenomen en de externe gebruiker is de gebruiker waar vanuit contact wordt opgenomen.

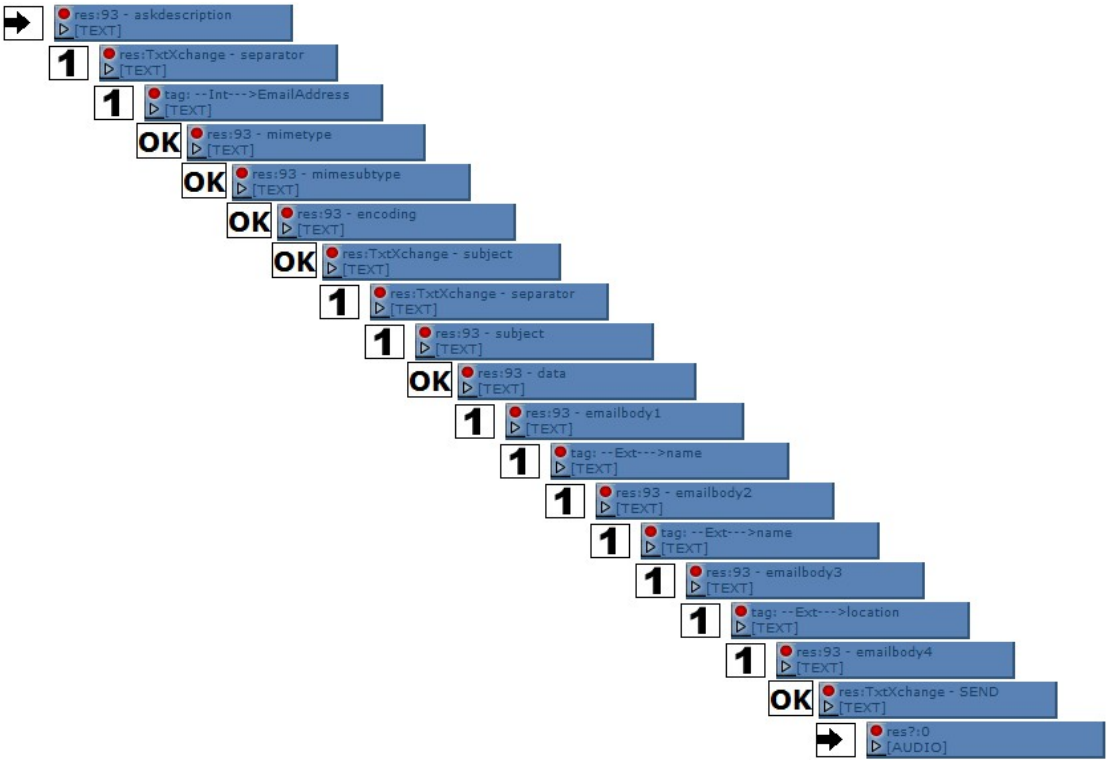
3 *De node met alle gegevens van de eigenaar*

Op het moment dat de eerste IVR wordt gestart (als er meer dan nul gebruikers in de groep zitten) staat de eigenaar aangegeven als de persoon waar contact mee opgenomen moet worden, oftewel de interne gebruiker. In het eerste IVR (*1a_FMO_Swap_Email*) wordt de eigenaar als externe gebruiker neergezet. Er moet namelijk vanuit de eigenaar contact worden opgenomen met contacten van de eigenaar.

Het zoeken van de contacten van de eigenaar gebeurt in het tweede IVR (*1b_FMO_ConnectToService_Email).* In de graaf van nodes wordt gezocht naar de contacten van de eigenaar. Het zoeken van de contacten gebeurt aan de hand van een service. In deze service staat beschreven dat de nodes gezocht moeten worden die onder de eigenaar groep vallen en die een resource *contactofowner* hebben.

Elke keer als er een contact is gevonden gaat de actie verder naar de volgende IVR. Namelijk het daadwerkelijk verzenden van de e-mail. In het derde IVR menu (*1c2_FMO_Email)* wordt de e-mail opgebouwd uit verschillende onderdelen. De verschillende stappen zijn te zien in figuur Figuur 29. De e-mail wordt net als de sms verzonden via de aparte txtXchange SOAP Client. Voor het verzenden is een protocol opgezet wat beschrijft welke onderdelen er aan de txtXchange meegegeven moeten worden om er daadwerkelijk een e-mail van te maken. Voorbeelden hiervan zijn de encoding, het subject en de body.

De body is opgebouwd uit verschillende onderdelen. Het is namelijk ook mogelijk om in de e-mail variabelen te verwerken. In het geval van e-mail en de sms wordt er gebruik gemaakt van de naam en locatie van de eigenaar. Na figuur 28 is te zien hoe de e-mail er uiteindelijk uit komt te zien. In groen is de variabele *name* gemarkeerd en in het blauw is de variabele *location*[4] gemarkeerd.



Figuur 29: IVR Menu e-mail verzenden naar contact van eigenaar

4 *Meer over het toevoegen van een locatie aan de e-mail wordt beschreven verderop in het hoofdstuk*

Hi,

Have you seen my owner Jan Jansen?

If you have, please send Jan Jansen to me or to http://common.sense-os.nl.

For some extra information, my location is: 5, Lloydstraat, Rotterdam, Zuid-Holland Latitude: 51.9035902, Longitude: 4.4598810.

Thanks in advance,
A Disappointed phone

Op het moment dat de e-mail is verzonden naar de contactpersoon wordt de volgende contactpersoon gezocht. Dit is de loop die te zien is in het diagram. Op het moment dat de service geen personen meer vindt die aan de voorwaarden voldoen en nog niet gebeld zijn, gaat de actie over naar het laatste IVR menu (*1d_FMO_RemovePersonalAgentFromAction-Group*). Met dit IVR menu wordt de eigenaar uit de groep *EmailActionContactsOfOwner* gehaald.

Hierdoor is de *EmailActionContactsOfOwner* leeg en hoeft er geen actie meer uitgevoerd te worden om de groep leeg te krijgen. Op het moment dat de eigenaar opnieuw zoek is, wordt deze weer in een van de actiegroepen gehangen.

In de applicatie er minder toegevoegd om vind-acties werkend te krijgen. Doordat de basis van communicatie met ASK eerder al is geïmplementeerd, hoeven deze enkel gebruikt te worden. ASK gaat actie uitvoeren op het moment dat de eigenaar in een van de actiegroepen is geplaatst. Oftewel het is vanuit de applicatie alleen van belang om ervoor te zorgen dat de eigenaar in een van de groepen wordt gehangen. Hiervoor zijn de functies *attachToAction* en *attachNode* voor gebruikt.

```
private static final String SMSACTION_CONTACTS_OF_OWNER = "03b17874-adaf-102e-bf75-
                                                          005056bc3799";
private static final String EMAILACTION_CONTACTS_OF_OWNER = "8e45ea88-b8b8-102e-
                                                           bf75-005056bc3799";
...

public void attachToAction(int action) {
               String ownerUUID = shPrefs.getString(Config.ASK_NODE_UUID_OWNER,"");
               switch (action) {
               case Config.FIND_SMS_OTHER_PERSONS:
                      attachNode(ownerUUID, SMSACTION_CONTACTS_OF_OWNER);
                      break;
               case Config.FIND_EMAIL_OTHER_PERSONS:
                      attachNode(ownerUUID, EMAILACTION_CONTACTS_OF_OWNER);
                      break;
               case Config.FIND_CALL_OTHER_NUMBER_OWNER:
                      attachNode(ownerUUID, CALLACTION_OWNER);
                      break;
               case Config.FIND_SEND_EMAIL:
                      attachNode(ownerUUID, EMAILACTION_OWNER);
                      break;
               case Config.FIND_CALL_OTHER_PERSONS:
                      attachNode(ownerUUID, CALLACTION_CONTACTS_OF_OWNER);
                      break;
               }
       }
```

Codefragment 18: attachToAction (CommunicationWithASK.java)

```
public void attachNode(String memNode, String parNode) {
        List<NameValuePair> nameValuePairs = new ArrayList<NameValuePair>(5);
        nameValuePairs.add(new BasicNameValuePair("sessionID", shPrefs
                        .getString(Config.ASK_SESSIONID, "")));

        nameValuePairs.add(new BasicNameValuePair("memUUID", memNode));
        nameValuePairs.add(new BasicNameValuePair("parUUID", parNode));

        HttpResponse responseAttachNode = connectToWeb(URL_ATTACHNODE,
                        nameValuePairs);

        String respAttach = ResponseToString(responseAttachNode);

        Log.e(TAG, "AttachNode: " + respAttach);
}
```

Codefragment 19: attachNode (CommunicationWithASK.java)

De *attachToAction* functie wordt gebruikt om te bepalen aan welke groep de eigenaar gehangen moet worden. Deze groep wordt vervolgens samen met het UUID van de eigenaar meegegeven aan de *attachNode* functie.

De attachNode functie roept de bijbehorende php pagina weer op om de attachNode functie van de webservices te gebruiken. Deze functie koppelt twee nodes aan elkaar als member en parent node. In dit geval wordt de eigenaar als membernode neergezet en de *Email-ActionContactsOfOwner* groep als parentnode.

6.2.2 Locatie uit Google Maps

In de applicatie wordt de locatie van de telefoon opgeslagen. Deze locatie wordt onder andere gebruikt om in een e-mail of sms weer te geven waar de telefoon zich bevindt.

De locatie wordt opgeslagen als *latitude* en *longitude.* Voor de persoon die de e-mail of sms leest is het niet reëel om dan direct te weten waar dat op zou moeten zijn. Daarom is het handig om deze waardes om te zetten in een adres.

Via de Google Maps API is het mogelijk om de *latitude* en de *longitude* mee te geven en hiervoor in xml alle bijbehorende gegevens wat betreft de locatie terug te krijgen.

In onderstaand fragment is een deel van de *xml response* te zien wanneer de coördinaten van het kantoor van Almende worden meegegeven.

```
<result>
        <type>street_address</type>
        <formatted_address>Westerstraat 50, 3016 Rotterdam,
Nederland</formatted_address>
        <address_component>
                <long_name>50</long_name>
                <short_name>50</short_name>
                <type>street_number</type>
        </address_component>
        <address_component>
                <long_name>Westerstraat</long_name>
                <short_name>Westerstraat</short_name>
                <type>route</type>
        </address_component>
        <address_component>
                <long_name>Nieuwe Werk</long_name>
                <short_name>Nieuwe Werk</short_name>
                <type>sublocality</type>
        <type>political</type>
        </address_component>
        <address_component>
                <long_name>Deelgemeente Centrum</long_name>
                <short_name>Deelgemeente Centrum</short_name>
                <type>sublocality</type>
                <type>political</type>
        </address_component>
        <address_component>
                <long_name>Rotterdam</long_name>
                <short_name>Rotterdam</short_name>
                <type>locality</type>
                <type>political</type>
        </address_component>

        <geometry>
                <location>
                        <lat>51.9088191</lat>
                        <lng>4.4796331</lng>
                </location>
        <location_type>ROOFTOP</location_type>
```

Codefragment 20: XML response van Google Maps API
via url: http://maps.googleapis.com/maps/api/geocode/json?latlng= 51.9088191,4.4796331&sensor=true

Te zien is dat elk onderdeel van het adres los is beschreven in een zogenaamde *address_component.* Om het adres in de e-mail te zetten worden een aantal van de componenten gekoppeld. In de e-mail wordt het als volgt weergegeven:

5, Lloydstraat, Rotterdam, Zuid-Holland Latitude: 51.9035902, Longitude: 4.4598810.

Om het adres op te vragen wordt er vanuit de applicatie een php pagina aangeroepen en wordt vanuit de php pagina de webservice aangeroepen en verwerkt[5].

In onderstaand fragment is de code hiervan te zien. Vanuit de applicatie wordt de laatste locatie van de telefoon doorgestuurd en de hoeveelheid data die opgehaald moet worden. Voor een sms geldt namelijk een andere lengte dan voor een e-mail.

```
<?php

$latitude             = htmlspecialchars($_POST["latitude"]);
$longitude            = htmlspecialchars($_POST["longitude"]);
$length               = htmlspecialchars($_POST["length"]);

getAddressFromGoogle($latitude, $longitude, $length);

function getAddressFromGoogle( $lat, $lng, $length ){
        // XML URL -
        $xml_url = "http://maps.googleapis.com/maps/api/geocode/xml?latlng=" . $lat .
                "," . $lng . "&sensor=true";

        # INITIATE CURL.
        $curl = curl_init();

        # CURL SETTINGS.
        curl_setopt($curl, CURLOPT_URL,"$xml_url");
        curl_setopt($curl, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt($curl, CURLOPT_CONNECTTIMEOUT, 0);

        # GRAB THE XML FILE.
        $xmlFile = curl_exec($curl);

        curl_close($curl);

        $xml = simplexml_load_string($xmlFile);

        // Get address Details for long message (in e-mail)
        if ($length == "1") {
                foreach ($xml->result->address_component as $item) {

                   if($item->type == "route"){ //5
                   echo"{$item->long_name}, ";
                   }

                   if($item->type == "street_number"){ //Lloydstraat
                   echo"{$item->long_name}, ";
                   }

                   if($item->type == "locality"){ //Rotterdam
                   echo"{$item->long_name}, ";
                   }

                   if($item->type == "administrative_area_level_1"){ //Zuid-Holland
                   echo"{$item->long_name}, ";
                   }

                   if($item->type == "postal_code"){ //3024 EZ
                   echo"{$item->long_name}, ";
                   }

                }

                // Get GPS Details
                foreach ($xml->result->geometry->location as $item) {
                   echo"Latitude: {$item->lat}, ";
                   echo"Longitude: {$item->lng}";
                }
        }
```

5 *Zie hoofdstuk 5.2.2 voor meer uitleg over de webservices en keuze voor php*

```
                // Get address details for short message (in sms)
        if ($length == "0") {
                foreach ($xml->result->address_component as $item) {

                   if($item->type == "route"){ //5
                   echo"{$item->long_name}, ";
                   }

                   if($item->type == "street_number"){ //Lloydstraat
                   echo"{$item->long_name}, ";
                   }

                   if($item->type == "locality"){ //Rotterdam
                   echo"{$item->long_name}";
                   }
                }
        }
}
?>
```

Codefragment 1: Google adres ophalen (getAddressFromGoogle.php)

In de code wordt de verbinding opgezet met de API en wordt het resultaat opgevangen als XML en uitgelezen. Elk resultaat dat als type *address_component* heeft wordt bekeken en mogelijk uitgeprint. Bij de e-mail zijn dit meer mogelijkheden dan bij de sms.

6.2.2 Werking van de alarmen

Om de status van de telefoon te veranderen wordt er gebruik gemaakt van alarmen. In hoofdstuk 4.3.6 wordt hier meer over uitgelegd. In de loop van het project is de opbouw van de alarmen veranderd en uitgebreid.

Om de toestand van de applicatie te veranderen naar een volgende toestand wordt er per toestand een alarm gezet. Om bijvoorbeeld over te gaan van herkennen naar het vinden van de eigenaar wordt het *WonderingAlarm* gezet. Na vijftien minuten gaat er een alarm af en wordt de *onReceive()* functie uitgevoerd van de *BroadcastReceiver – WonderingChecker.* Deze functie zet de toestand van de applicatie op *Disappointed* en wordt de *findAction* klasse uitgevoerd. In de *findAction* klasse worden vervolgens de acties uitgevoerd die van toepassing zijn op de toestand. In onderstaand fragment is de code van de *BroadcastReceiver* te zien.

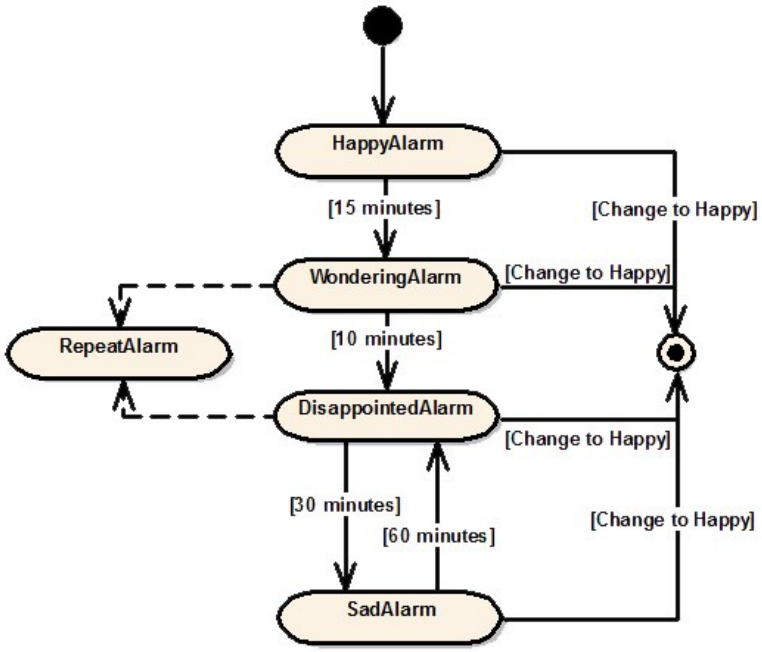
```
package nl.afst.owner.findaction;

import nl.afst.owner.config.Config;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.preference.PreferenceManager;
import android.util.Log;

public class WonderingChecker extends BroadcastReceiver {
        public static final String TAG = "WonderingChecker";
        public static final int WON_CHECK = 1;
        SharedPreferences shPrefs;

        @Override
        public void onReceive(Context c, Intent i) {
                shPrefs = PreferenceManager.getDefaultSharedPreferences(c);

        shPrefs.edit().putInt(Config.OWNER_STATE,Config.DISAPPOINTED).commit();
                c.startService(new Intent(c, FindAction.class));

        }

}
```

Codefragment 21: onReceive() (WonderingChecker.java)

In figuur 30 is de totale overgang van de alarmen te zien. De alarmen hangen samen met de overgang van toestanden. Enkel heeft de toestand *Jealous* geen alarm. Er is namelijk geen moment waarop deze uit de toestand moet komen.

Daarnaast worden de vind- en herkenningsacties na vijf minuten herhaald als er nog geen actie is ondernomen door de eigenaar. Om te zorgen dat er na vijf minuten weer een actie wordt uitgevoerd, is het *repeatAlarm* gemaakt. Deze wordt gestart bij de toestand *Wondering* en bij de toestand *Disappointed.*

Het eerste alarm dat wordt gestart is het *HappyAlarm*. Vanuit hier worden de volgende alarmen gezet. Het eindpunt kan vanuit elk toestandsalarm komen. Op het moment dat de toestand *Happy* wordt, op het moment dat de eigenaar de telefoon gebruikt, worden alle alarmen weer gestopt.



Figuur 30: Overgang van alarmen (OwnerService.java)

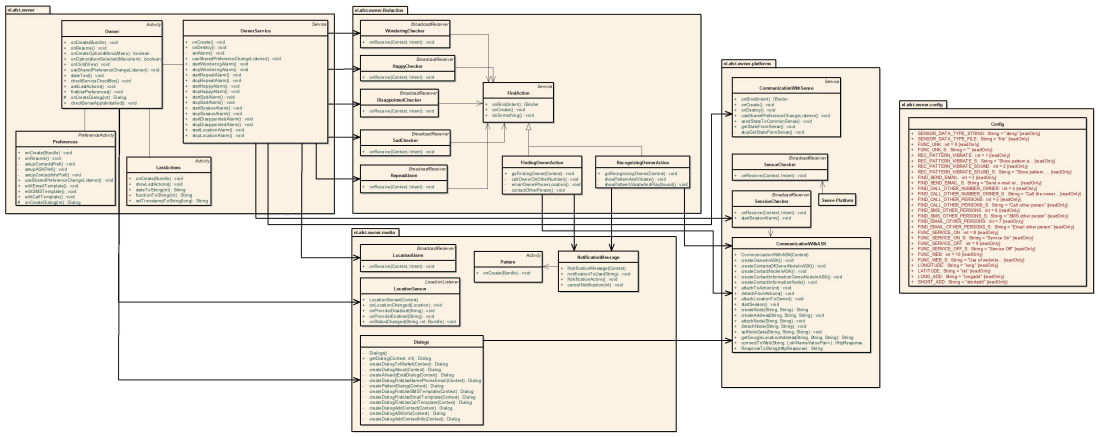
7. Vierde oplevering – Eindversie

In de vierde oplevering zijn de laatste functies aan de applicatie toegevoegd. Het belangrijkste onderdeel hierbij is de webinterface waardoor ook de toestand *Jealous* gebruikt kan worden.

Naast de webinterface is de applicatie geheel getest en zijn de fouten die hieruit gekomen zijn verbeterd in de applicatie. De resultaten van de tests zijn terug te vinden in het transition fase rapport. De verbeteringen worden in dit verslag ook verder niet besproken, enkel als het om een grote wijziging in de code gaat.

In de vierde oplevering is ook de *javadoc* voltooid. Al het commentaar bij de functies is te vinden via de website voor de ontwikkelaars.

7.1 Samenwerking tussen de klassen in eindversie



Figuur 31: Totaal klassediagram eindversie (Grote versie te vinden in de bijlage)

Het diagram is anders opgebouwd dan de eerste twee klassendiagrammen. In het diagram worden de lagen van abstractie weergegeven.

owner[6] is de *package* met de het hoogste instabiliteit. Vanuit deze klasse worden enkel andere klassen/*package*s aangeroepen. Als in één van de klassen in *owner* veranderd hoeft er geen verandering plaats te vinden in de andere klassen. Echter op het moment dat in de andere packages veranderingen plaatsvinden, is een grote kans dat in owner ook iets veranderd moet worden.

Het *findaction*[7] *package* is afhankelijk van platforms en media. Mochten hier veranderingen in plaatsvinden dan moet mogelijk een functie uit een van de klassen van *findaction* ook worden aangepast.

Het *media*[8] *package* is enkel afhankelijk van het *platforms*[9] *package*. Echter kan deze afhankelijkheid weggelaten worden. De *package* roept namelijk alleen functies aan van deze klassen en ontvangt hiervan geen reactie. Op het moment dat een van de functies in een klasse van *findaction* veranderd en een response krijgt of variabele nodig heeft dan moet de *Dialogs* klasse wel worden veranderd.

6 *Lees nl.afst.owner*
7 *Lees nl.afst.owner.findaction*
8 *Lees nl.afst.owner.media*
9 *Lees nl.afst.owner.platforms*

Het *platform package* is zelf nergens van afhankelijk. Wijzigingen zorgen zoals eerder gezegd wel voor wijzigingen in andere *packages.*

Het *config package* staat in bovenstaand diagram(figuur 31) los van de andere *package*s en is in bovenstaand verhaal ook niet beschreven als afhankelijkheid. Echter is deze afhankelijkheid er wel degelijk. De C*onfig* klasse definieert namelijk veel standaarden. Vanuit elke klasse wordt de *Config* klasse aangeroepen om waardes op te halen. Deze waardes vallen onder andere onder de standaard namen voor SharedPreferences. Een voorbeeld van de aanroep van de *Config* klasse is in onderstaand fragment te zien.

```
Editor edit = shPrefs.edit();
edit.putInt(Config.OWNER_STATE, Config.WONDERING).commit();

In de Config klasse staan deze waardes gedefinieerd als:
public static final String OWNER_STATE = "OwnerState";
public static final int WONDERING = 2;
```

Codefragment 22: Gebruiker klasse Config

Op het moment dat een waarde een van de variabelen veranderd in de code van de Config klasse heeft dit niet direct gevolg op de rest van de *packages.* In de *packages* hoeft niets veranderd te worden ze gebruiken namelijk alleen de naam van de variabele. Het wijzigen of verwijderen van namen heeft wel gevolgen voor de rest van de code. Echter gebeurt dit alleen als dit voor een van de andere klassen van belang is, net zoals het toevoegen van waardes.

Om het diagram overzichtelijk te houden zijn alle pijlen richting de Config klasse weggelaten. Echter hoort er in het diagram vanuit elke klasse en pijl naar de Config klasse.

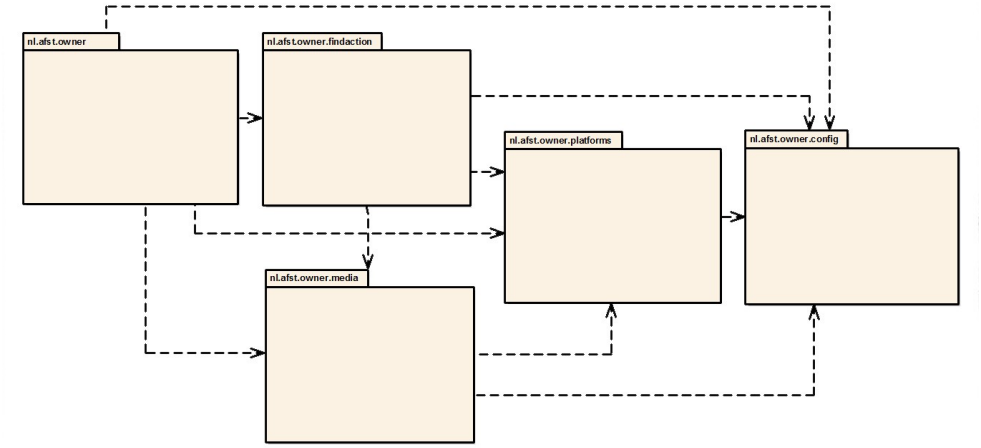
In eerdere versies waren deze standaarden ook al aanwezig maar bevonden deze zich in de *Perferences klasse.* Echter zou hierdoor de instabiliteit van de *owner package* hiervan totaal veranderen.

Een mogelijkheid is ook om de *Preferences* klasse uit de *owner package* te halen. Hierdoor verandert alleen weer de opzet van de *owner package.* De owner *package* bevat namelijk alle activities[10] van de applicatie.

Zoals beschreven is voor de standaarden niet maar één oplossing mogelijk. Voor de *Find My Owner* applicatie is gekozen voor de losse *Config klasse / package*.

10 *Weergaves zoals hoofdscherm, instellingenscherm en lastActions lijst*

In onderstaand figuur is de het *package* diagram te zien. Hierin zijn de afhankelijkheden van elke *package* te zien. Hierin is de de *config package* wel opgenomen.



Figuur 32: Package diagram

Om weer te geven hoe instabiliteit / abstractheid tussen de *package*s is, is het *SAP*[11] */ SDP*[12] toegepast op het klassendiagram / *package*diagram. Bij deze *principles* wordt aan de hand van verschillende eigenschappen de abstractheid en instabiliteit van een *package* berekend. De berekeningen en uitkomsten zijn te zien in de tabellen 2 en 3.

Nc	*Aantal klassen*	*H*	*Cohesie*
Na	*Aantal abstracte klassen*	*I*	*Instabiliteit (Ce / (Ca + Ce))*
Ca	*Afferente koppelingen ->[-]<-*	*A*	*Abstractheid (Na / Nc)*
Ce	*Efferente koppelingen <-[-]->*	*D*	*Afstand van main sequence (|A+I-1| / √2)*
R	*Aantal interne relaties*	*D'*	*Genormaliseerde afstand (|A+I-1|)*

Tabel 2: Legenda en berekening

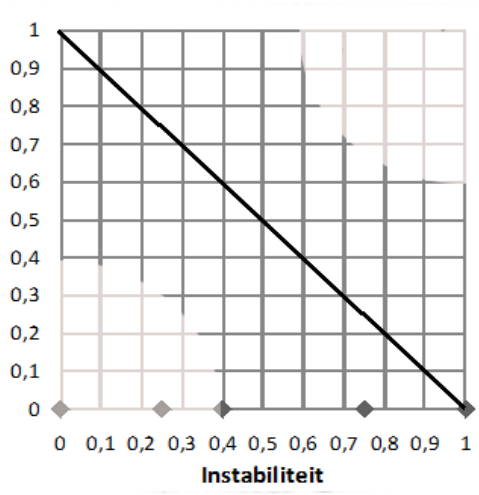
Package	Nc	Na	Ca	Ce	R	H	I	A	D	D'
Owner	4	0	21	11	3	1	0,344	0	0,464	0,656
Findaction	8	0	1	3	7	1	0,75	0	0,177	0,25
Media	5	0	3	2	1	0,4	0,4	0	0,424	0,6
Platforms	4	0	3	1	1	0,5	0,25	0	0,53	0,75
Config	1	0	21	0	0	1	0	0	0,707	1

Tabel 3: Berekening Cohesie, Instabiliteit, Abstractheid, Main Sequence

Daarnaast is de Abstractie / Instabiliteit weergegeven in figuur 33. De zwarte lijn is de main sequence. Op het moment dat een waarde op punt (0,1) of (1,0) ligt is dit perfect. Ligt een waarde op de lijn dan is het bijna perfect.

Op het moment dat de het punt in het lichtgekleurde vak links onderin zit betekent het dat de *package* moeilijk is opgezet. De *package* heeft veel andere klassen die afhankelijk zijn van de klasse(s) in dit *package*. In de Find My Owner applicatie is deze afhankelijkheid groot bij *config en platforms.*

11 *Stable Abstractions Principle*
12 *Stable Dependencies Principle*



Figuur 33: Abstractheid / Instabiliteit packages

Op het moment dat een waarde in het lichtgekleurde vak rechtsboven komt te staan is het een "nutteloos" *package*. Het *package* heeft dan in verhouding veel abstracte klassen maar deze klasse worden niet gebruikt. Want het aantal afferente kloppelingen ligt laag. Het is dan dus een weinig tot niet gebruikt *package*. In de find my owner applicatie is niet een dusdanige *package* aanwezig.

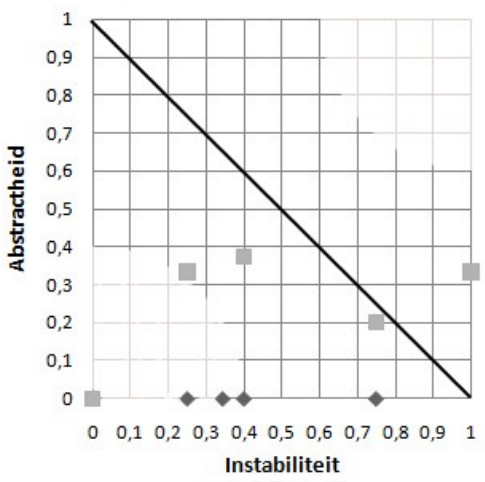
In bovenstaande berekeningen is enkel rekening gehouden met de instabiliteit van de *package*s. Bij *SAP* en *SDP* wordt er ook gekeken naar de abstractie van een *package*. Oftewel het aantal abstracte klassen in de *package*. In de applicatie zitten geen abstracte klassen enkele worden wel een aantal klassen uit de *package*s afgeleid van abstracte klassen als *Service, Activity* en *BroadcastReceiver.*

Als de waardes worden berekend voor de aanwezigheid van deze abstracte klassen komt er het volgende uit:

Package	Nc	Na	Ca	Ce	R	H	I	A	D	D'
Owner	6	2	0	11	7	1,333	1	0,333	0,236	0,333
Findaction	10	2	1	3	10	1,1	0,75	0,2	0,035	0,05
Media	8	3	3	2	4	0,625	0,4	0,375	0,159	0,225
Platforms	6	2	3	1	4	0,833	0,25	0,333	0,295	0,417
Config	1	0	21	0	0	1	0	0	0,707	1

Tabel 4: Berekening Abstractheid / Instabiliteit

De grafiek die hier bij hoort is te zien in figuur 34. De grijze vierkantjes zijn de nieuwe punten. Te zien is dat een hogere abstractie aanwezig is, waardoor een aantal punten niet meer in de moelijke zone vallen.



Figuur 34: Abstractheid / Instabiliteit packages

De tweede opzet is opgezet op een indruk te geven wat gebeurt als gewerkt wordt met meer abstracte klassen. De tweede opzet is ook niet geheel juist, want Activity, Service en BroadcastReceiver vallen niet onder de code van *Find My Owner* en behoren daardoor eigenlijk ook niet tot de packages.

7.2 Webinterface – Toestand van de applicatie veranderen naar *Jealous*

Om te zorgen dat de applicatie niet altijd opzoek gaat naar de eigenaar is het mogelijk om de applicatie stop te zetten (via knop in de applicatie). Hierdoor worden er door de applicatie geen vind-acties uitgevoerd.

Nu kan het zo zijn dat de eigenaar niet in de buurt van de telefoon is en daardoor de applicatie niet kan uitschakelen om te stoppen met vind-acties. Hiervoor is het voor de eigenaar mogelijk om via een webinterface aan te geven dat hij niet bij de telefoon in de buurt is. Hierdoor komt de applicatie in de toestand *Jealous* en wordt er geen actie uitgevoerd totdat de eigenaar terug is bij de telefoon.

De webinterface is een opgebouwd uit enkele PHP pagina's met daarin de volgende functionaliteiten:

- Inloggen bij CommonSense
- Status veranderen naar *Jealous* in CommonSense
- Locatie van de telefoon bekijken

7.2.1 Inloggen bij CommonSense

Het inloggen bij CommonSense gebeurt via de API van Sense[13]. Het inloggen gebeurt aan de hand van de *login* functie die te bereiken is via: http://api.sense-os.nl/login.json. Het verzenden van de de data gebeurt in een JSON[14] object. Het object ziet er als volgt uit:

```
$data = '{"username":"'.$username.'","password":"'.$password.'"}'
```

13 *Documentatie van de API is te vinden via http://api.sense-os.nl*
14 *JSON is Java Script Object Notation. Eerder gebruikt bij het verzenden van berichten naar Sense vanuit de applicatie*

Het wachtwoord is voor verzenden omgezet in een md5 wachtwoord. De wachtwoorden van Sense staan namelijk ook als md5 opgeslagen in de database.

Via de functie `http_post_json($url, data$)`[15] wordt de data naar CommonSense gestuurd. Vanuit CommonSense wordt een bevestiging met sessionID teruggestuurd. Met het sessionID kunnen vervolgens de andere functies van de API worden aangeroepen. Het sessionID is één dag geldig. Hierdoor hoeft niet bij elke aanroep van een functie opnieuw een sessie te worden opgestart.

7.2.2. Toestand veranderen naar *Jealous* in CommonSense

De sensor die in CommonSense veranderd moet worden is de *Find My Owner* sensor. Binnen CommonSense is het mogelijk om meerdere devices aan te sluiten waardoor het mogelijk is dat er meerdere *Find My Owner* sensoren aanwezig zijn.

Bij het versturen van nieuwe data naar CommonSense vanaf de Sense smartphone applicatie is bekend om welke device het gaat. Namelijk het device waar de data vandaan wordt gestuurd. Hierdoor kan CommonSense direct de nieuwe waarde toevoegen aan de juiste sensor.

Bij het versturen via de API is er niet bekend om welk device het gaat en moet eerst worden uitgezocht welke sensor aangepast moet worden. Bij het opzoeken van de juiste sensor wordt er vanuit gegaan dat maar op één van de twee devices de *Find My Owner* applicatie actief is. Er wordt namelijk gekeken naar de sensor die als laatste aangepast is. Hiervoor worden de volgende twee stappen uitgevoerd:

- Het ophalen van alle sensordata van de eigenaar uit CommonSense
  In deze data wordt gezocht naar alle sensoren die de naam “find my owner” hebben. Heeft een sensor deze naam, dan wordt het id van deze sensor opgeslagen in een array. De API functie die voor deze actie wordt gebruikt is:
  http://api.sense-os.nl/sensors.json?owned=1
- Het ophalen van de timestamp van de “find my owner” sensoren
  Aan de hand van een id's wordt bij CommonSense de laatste data van de sensor opgehaald. Uit de data wordt de timestamp gehaald en vergeleken met de timestamp van de andere opgehaalde sensoren. De sensor met de hoogste timestamp wordt opgeslagen als beste sensor.

Aan de hand van het id dat uit de vergelijking is gekomen kan de functie *http://api.sense-os.nl/sensors/1234/data.json* worden aanroepen. Hierin staat 1234 voor het id van de sensor. De data die wordt meegegeven ziet er als volgt uit:

```
$data = '{ "data":[{"value":"{\"State\":\"Jealous\",\"Function\":\"Use of
        webinterface\",\"Function id\":10}","date":'.time().'}]}';
```

In het JSON pakket wordt meegegeven:

- De waarde van de sensor (toestand en functie)
- De tijd waarop de sensordata wordt verstuurd

7.3 De applicatie – Toestandsverandering naar *Jealous* herkennen.

Doordat de toestand van de applicatie in CommonSense wordt veranderd naar *Jealous* is niet direct bij de applicatie de nieuwe toestand bekend. Om te voorkomen dat de applicatie

15 *Functie uit ApiCalls.php, functies voor het aanspreken van de API, opgezet door medewerker van Sense*

opzoek gaat naar die eigenaar, terwijl de eigenaar al heeft aangegeven niet aanwezig te zijn, wordt de toestand voor elke actie gecontroleerd.

Bij het opstarten van de applicatie wordt een Broadcast Intent gestuurd naar de Sense applicatie die op de smartphone draait. De Sense applicatie kan namelijk gegevens uit CommonSense halen met de *FeedbackChecker* en deze mogelijk verwerken.

Vanuit de Find My Owner applicatie wordt er een *Intent* "gegooid" naar "nl.sense_os.service.CheckFeedback". Het intent bevat extra informatie over:

- De sensor die moet worden opgehaald (In dit geval "find my owner")
- De periode waarmee de Sense applicatie de sensor moet controleren bij CommonSense
- De actie na ontvangen van de sensor gegevens uitgevoerd moet worden, hierbij een actie van de Find My Owner applicatie.

De *BroadcastReceiver FeedbackRx* vangt het Intent op (fragment x), pakt de gegevens uit en start een functie in de klasse *FeedbackChecker.* Deze functie vraagt bij CommonSense de gevraagde waarde op bij de aangegeven sensor en stuurt de waarde ingepakt in een JSON String weg als Intent naar "nl.afst.owner.SensorChecker".

De *SensorChecker* pakt de Intent uit (fragment x) en controleert of de JSON String het woord *Jealous* bevat. Als dat zo is, is de laatste toestand bij *CommonSense Jealous* en moet de toestand van de applicatie hier ook in worden veranderd. Op het moment dat de toestand *Jealous* wordt kan er geen vind-actie meer worden uitgevoerd. De applicatie gaat wachten totdat de eigenaar terug is bij de telefoon.

```
public void getStateFromSense(){
        Log.e(TAG, "getStateFromSense");
        /*Communication with Sense*/
        Intent feedbackIntent = new Intent("nl.sense_os.service.CheckFeedback");
        feedbackIntent.putExtra("sensor_name", "find my owner");
        feedbackIntent.putExtra("period", 3600 * 60 * 1);
        feedbackIntent.putExtra("broadcast_after","nl.afst.owner.platforms.-
        SensorChecker");
        sendBroadcast(feedbackIntent);
        /**/
}
```

In onderstaand codefragment is het "gooien" van de *Intent* te zien en de waardes die aan de *Intent* worden toegevoegd aan de hand van *putExtra(name, value).*

```
public class FeedbackRx extends BroadcastReceiver {

private static final String TAG = "Sense Feedback BroadcastReceiver";
public static final String ACTION_CHECK_FEEDBACK =
"nl.sense_os.service.CheckFeedback";
public static final int REQ_CHECK_FEEDBACK = 2;

public int periodCheckSensor = 0;
public String sensorName = null;
public String actionAfterCheck = null;

@Override
public void onReceive(Context context, Intent intent) {
        Log.e(TAG, "onReceive");

        periodCheckSensor     = intent.getIntExtra("period", (1000 * 60 * 2));
        sensorName            = intent.getStringExtra("sensor_name");
        actionAfterCheck      = intent.getStringExtra("broadcast_after");
```

```
        /* set the next check broadcast */
        final Intent alarmIntent = new Intent(ACTION_CHECK_FEEDBACK);
         alarmIntent.putExtra("period", periodCheckSensor);
         alarmIntent.putExtra("sensor_name", sensorName);
         alarmIntent.putExtra("broadcast_after", "actionAfterCheck");
         final PendingIntent alarmOp = PendingIntent.getBroadcast(context,
        REQ_CHECK_FEEDBACK,alarmIntent, 0);
         final long alarmTime = System.currentTimeMillis() + periodCheckSensor;
         final AlarmManager mgr = (AlarmManager)
        context.getSystemService(Context.ALARM_SERVICE);
         mgr.cancel(alarmOp);
         mgr.set(AlarmManager.RTC_WAKEUP, alarmTime, alarmOp);

         /* start the feedback check task */
         Intent checkFeedback = new Intent(FeedbackChecker.ACTION_CHECK_FEEDBACK);
         checkFeedback.putExtra("sensor_name", sensorName);
         checkFeedback.putExtra("broadcast_after", actionAfterCheck);
         Log.e(TAG, "SensorName Rx: " + sensorName);

         ComponentName component = context.startService(checkFeedback);

         if (null == component) {
             Log.w(TAG, "Could not start feedback checker");
         }
    }
}
```

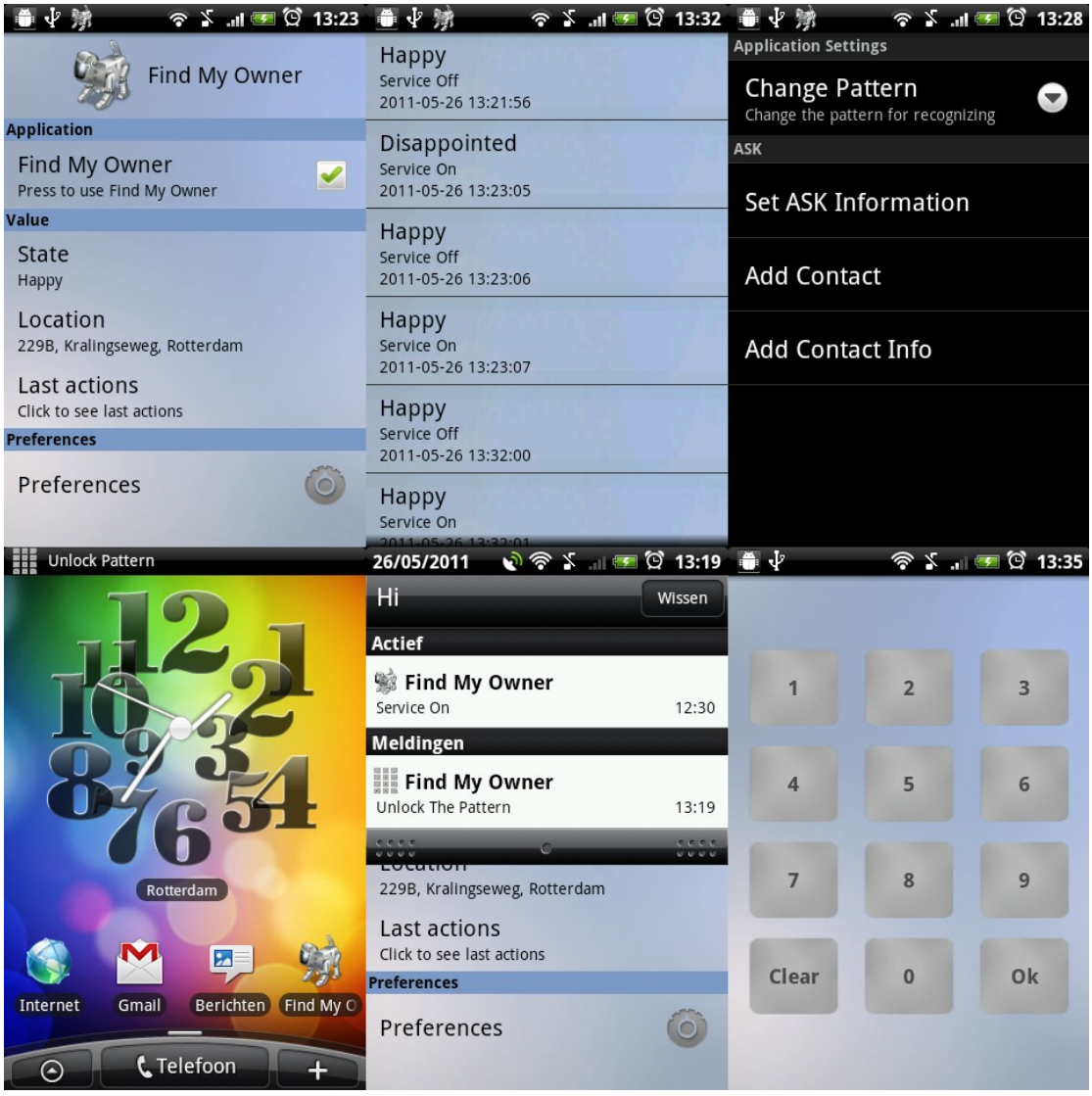
```
@Override
public void onReceive(Context context, Intent intent) {

shPrefs = PreferenceManager.getDefaultSharedPreferences(context);

String jsonString = intent.getStringExtra("json");
        if (jsonString != null){
                Log.e(TAG, "Received the Sensor data from Sense!");
                Log.e(TAG, "Jsonstring: " + jsonString);
                //jsonString

                int index = jsonString.lastIndexOf("Jealous");

                if (index == -1){
                        Log.e(TAG, "not found index is: " + index);
                        //do nothing
                } else {
                        Log.e(TAG, "Jealous found! index is: " + index);

                        Editor edit = shPrefs.edit();

                        edit.putInt(Config.OWNER_STATE, Config.JEALOUS);
                        edit.putInt(Config.OWNER_STATE_ACTION,
                         Config.FUNC_WEB);
                        edit.commit();

                }
        }
        else {
                Log.e(TAG, "Received Intent without jsondata");
        }

}
```

Voor de juiste werking van de *FeedbackChecker* in de Sense applicatie is er enkele code aangepast. De mogelijkheid van het opvragen van een sensor zat namelijk wel al in de applicatie maar deze was nog niet beschikbaar voor ander applicaties dan de Sense applicatie zelf. De *FeedbackChecker* is zo omgebouwd dat een andere applicatie gegevens over de sensor, periode en actie kan meegeven aan de Intent en de juiste sensor waarde krijgt teruggestuurd.

Deze aanpassing was nog niet uitgevoerd door Sense maar was wel van belang omdat de Sense applicatie uiteindelijk door meer verschillende applicaties aangesproken moet kunnen worden.

7.3 Eindversie van de GUI

De uiteindelijk versie van de GUI is te zien in onderstaande afbeeldingen. Onder de afbeeldingen is de toelichting te vinden over de wijzingen van de GUI ten opzichte van de GUI op in hoofdstuk 4.2.



Linksboven is het hoofdscherm van de applicatie te zien. Ten opzichte van de andere ontwerpen is deze weinig veranderd. Enkel verandert de regel onder *Location* op het moment dat de locatie bekend is. Het verkorte adres wordt dan weergegeven op deze regel.

De *LastActions* lijst bevat in de eindversie ook de tijd waarop een functie heeft plaatsgevonden.

Het *Preferences* scherm bevat nieuwe instellingen voor de communicatie met ASK. Zo moet de gebruiker zijn eigen gegevens, de gegevens van een contactpersoon en andere

contactgegevens van de eigenaar zelf invoeren. Om fouten te voorkomen is het instellingenscherm de *Service On/Off* instelling gehaald. De gebruiker kan de service enkel nog in- en uitschakelen in het hoofdscherm.
Aan de *notification bar* is het icon van de applicatie toegevoegd als actieve applicatie. Het icon in de vorm van een *pattern* is gebruikt voor de melding “Unlock pattern”. Op deze manier kan de gebruiker makkelijk de applicatie starten en uitschakelen indien nodig.

Het patroon is hetzelfde gebleven. Het patroon kan in de laatste versie ook opgeroepen worden met de “State” knop in het hoofdscherm. Hierdoor kan de gebruiker wel het patroon openen op het moment dat hij deze heeft gewist als notificatie.

In het transition fase rapport is meer te lezen over de resultaten van de gebruikerstest en de reacties over de GUI.

Verklarende woordenlijst

Woord, afkorting	Verklaring
ASK, ASK Community Systems	Dochterbedrijf van Almende dat het ASK Platform heeft opgezet
ASK, ASK webinterface	De basis van ASK, hier wordt alles gecombineerd. Dit deel wordt door ASK CS ingesteld voor een klant
ASK Desktop	Een applicatie waar ASK wordt gebruikt op de achtergrond
ASK Platform	Het ASK systeem met alle onderdelen (Communication Coordinator, API, Rules and scenario's, Users and groups, applicatiies en communicatiemiddelen)
IVR Menu	Interactive voice response, een telefonisch keuze menu. Ook wel bekend van klantenservice telefoongesprekken
Agent	Een virtuele persoonlijkheid binnen het ASK platform die één of meerdere verantwoordelijkheden binnen het systeem heeft.
Thread	Het hoofdproces verdelen in meerdere processen om verschillende taken tegelijkertijd uit te kunnen voeren
Herkennen, Recognizing, *Wondering*	Het herkennen van de eigenaar. Hierbij wordt er gekeken of de eigenaar in de buurt van de telefoon is. De toestand die hieraan gegeven wordt is *Wondering*.
Vinden, Find, *Disappointed*	Het vinden van de eigenaar. Hierbij wordt gekeken waar de eigenaar is en wordt er geprobeerd deze herenigen met de telefoon. De toestand die hieraan gegeven wordt is *Disappointed*
Toestand, State, Status (van de eigenaar / applicatie)	De toestanden van de applicatie zijn de toestanden waarin de applicatie verschillende acties gaat uitvoeren. De toestanden zijn uitgedrukt in verschillende emoties
Package	De groepering van klassen in verschillende namespaces.
BroadcastReceiver	Interrupt van bijvoorbeeld een alarm afvangen en vervolgens bijbehorende acties uitvoeren (bijvoorbeeld de *FindAction* klasse aanroepen)
Service	Een *Service* is een actie die op de achtergrond uitgevoerd wordt zonder dat de gebruiker input hoeft te leveren
Activity	Een A*ctivity* een een activiteit die op de voorgrond wordt uitgevoerd. De A*ctivity* is doorgaans gekoppeld aan een UI.

Tabel 5: Verklarende woordenlijst

Bronvermelding

1. Rien Elling, Bas Andeweg, Jaap de Jong, (2005). *Digitaal Handboek Rapportagetechniek 4.2.* Noordhoff Uitgevers (cd-rom)
2. The Eclipse Foundation, (-2011). *OpenUP.* Via http://epf.eclipse.org/wikis/openup/
3. The Eclipse Foundation, (-2011). *Iteration Plan.*
4. The Eclipse Foundation, (-2011). *Example Iteration Plan*.
5. ASK-CS, (-2011). *About ASK.* Via http://www.ASK-CS.com
6. T. van As en L. Stellingwerff, (2011). *ASK Reference Guide.* ASK Community Systems
7. T. van As, (2011). *ASK Architecture.* ASK Community Systems
8. T. van As en S. Mulder, (2011). *MyASK Architecture*. ASK Community Systems
9. T. van As, (2011). *MyASK Features, Applications & Roadmap.* ASK Community Systems
10. Sense-OS, (-2011). *About Sense.* Via http://www.Sense-OS.nl
11. S. Mulder, (2011). *CommonSense Flyer.* Sense Observation Systems
12. Mark L. Murphy, (2010). *Android Programming Tutorials.* CommonsWare
13. Mark L. Murphy, (2010). *The Busy Coder's Guide To Android Development.* CommonsWare
14. Mark L. Murphy, (2009-2010). *The Busy Coder's Guide To Advanced Android Development.* CommonsWare
15. Android, (-2011). *Android Developers.* Via http://developer.Android.com/guide/index.html
16. Nederlandse taalunie, (-2011). *Literatuurlijsten (algemeen).* Via http://taaladvies.net/taal/advies/tekst/87/#7
17. Android, (-2011) *Installing Android SDK*. Via http://developer.android.com/sdk/installing/html
18. Software innovators, (-2011). *Ontwikkelen op het Android platform*. Via http://www.software-innovators.nl/2009/06/05/ontwikkelen-op-het-android-platform
19. Oracle, (-2011). RESTful. Via http://www.oracle.com/technetwork/articles/javase/index-137171.html
20. André Boonzaaijer, (2008). *RESTful Webservices – Intro.* Via http://www.software-innovators.nl/2008/05/24/restful-webservices-intro/
21. Google Maps, (-2011). *Geocoding with Google Maps.* Via http://code.google.com/apis/maps/documentation/geocoding/
22. mlseim, (-2011). *Use of Google Maps API.* Via http://www.webdesignforums.net/threads/google-maps-api.32565/
23. Carlo Post, (2011). *txtXchange protocol*. ASK Community Systems

Vermelding van alle bronnen
Zoveel mogelijk bronnen zijn in bovenstaande lijst vermeld. Echter is hij bij ontwikkeling lastig om elke bron van codefragmenten of componenten te noemen. In ieder geval zijn zoveel mogelijk bronnen vermeld in de bovenstaande lijst.

Algemene bronnen o.a Wikipedia en Google
Wikipedia en Google worden doorgaans niet in een bronvermelding genoemd omdat dit doorverwijzingen zijn naar andere bronnen. Tijdens het uitvoeren van de opdracht is Google als zoekmachine gebruikt voor het vinden van websites die informatie bevatten over bepaalde componenten of constructies.

Wikipedia is daarbij ook gebruikt om van verschillende onderdelen een algemeen verhaal te vinden. Op het moment dat er uitgebreidere informatie is gebruikt, is de bron in de bovenstaande lijst vermeld.

Daarnaast is er voor het ontwikkelen veel gebruik gemaakt van Bigresource.com en Stackoverflow.com. Dit zijn twee fora waar veel informatie te vinden is over fouten in code van andere programmeurs.

Transition Fase Rapport

"Onderzoek en ontwikkeling van Android applicatie – Waar is mijn eigenaar?"

Auteur: Marleine van Kampen
Datum: 6 juni 2011
Versie: 1.0

Versiebeheer

Versie	Datum	Auteur	Aanpassingen
0.1	23 februari 2011	M. van Kampen	Basis opzet
0.2	21 april 2011	M. van Kampen	Uitbreiding met meer testscenario's
0.3	12 mei 2011	M. van Kampen	Uitgevoerde tests invullen en verbeteringen toevoegen
1.0	6 juni 2011	M. van Kampen	Uitwerken laatste tests en oplevering

Reviewers

Versie	Datum	Auteur
0.1	18 maart 2011	S. Mulder
0.1	18 maart 2011	L. Stellingwerff

Gerelateerde documenten

Naam	Versie	Datum	Auteur
Afstudeerscriptie	1.0	6 juni 2011	M. van Kampen
Inception Fase Rapport	1.0	21 februari 2011	M. van Kampen
Elaboration Fase Rapport	1.0	6 juni 2011	. van Kampen
Construction Fase Rapport	1.0	6 juni 2011	M. van Kampen

Distributielijst

Naam	Organisatie	Functie	Reden
S. Mulder	Sense	Technisch begeleider	Proces begeleider
T. van As	ASK	Technisch begeleider	Proces begeleider
L. Stellingwerff	ASK	Technisch begeleider	Proces begeleider
J.J. Visser	Haagse Hogeschool	Docent	1e Examinator
D.R. Stikkolorum	Haagse Hogeschool	Docent	2e Examinator

Samenvatting

Marleine van Kampen, student van De Haagse Hogeschool Delft voert gedurende zeventien weken een afstudeeropdracht uit bij ASK Community Systems en Sense Oberservation Systems. De opdracht is genaamd “Waar is mijn eigenaar”?, een Android applicatie die opzoek gaat naar de eigenaar van de telefoon.

De ontwikkeling van de applicatie is uitgevoerd in vier fases; inception, elaboration, construction en de transition fase. De transition fase is de laatste fase van de ontwikkeling. In deze fase wordt de gehele applicatie getest.

Voor het testen is een onderzoek gedaan naar de verschillende testtechnieken. Uit deze technieken is een selectie gemaakt waarmee getest wordt.

Tijdens de ontwikkeling zijn de verschillende opleveringen getest op functionaliteit. De eindversie is op eenzelfde manier getest.

Op de eindversie van de applicatie is een performance en reliability test uitgevoerd. Deze tests zijn geslaagd.

Om te weten hoe de gebruikers met de applicatie omgaan en waar deze tegen aanlopen met het gebruik is een gebruikerstest opgesteld. Medewerkers van Sense en ASK en vrienden/familie hebben de applicatie getest. Uit de test zijn verschillende fouten gekomen die verder opgelost zijn.

De applicatie voldoet aan de eisen die aan het begin van de ontwikkeling opgesteld zijn. Waar de test niet aan voldoet zijn eisen die door een ander manier van implementeren zijn weggevallen. De hoofdeisen zijn in ieder geval goed doorgevoerd.

Inhoudsopgave

I. INLEIDING....6

I.I AANLEIDING....6
I.II DOEL....6
I.III SCOPE....6

1. GLOBAAL TESTPLAN....7

1.1 ONDERZOEK TESTTECHNIEKEN....7
1.2 V-MODEL....9
1.3 TE GEBRUIKEN TESTTECHNIEKEN....9

2. GUI....10

2.1 GUI....10
2.2 EISEN....10
2.3 CHECKLIST EISEN....10

3. TEST APPLICATIE NA EERSTE OPLEVERING....11

3.1 INLEIDING....11
3.2 ACTIVITEITENDIAGRAM....11
3.3 TESTCASES....12
3.4 TESTRAPPORT....13
3.5 ISSUES....13

4. TEST APPLICATIE NA TWEEDE OPLEVERING....14

4.1 INLEIDING....14
4.2 ACTIVITEITENDIAGRAM....14
4.3 TESTCASES....15
4.4 TESTRAPPORT....17
4.5 ISSUES....18

5. TEST APPLICATIE NA DERDE OPLEVERING....19

5.1 ACTIVITEITENDIAGRAM....19
5.2 TESTCASES....20
5.3 TESTRAPPORT....23
5.5 ISSUES....25

6. EINDVERSIE....26

6.2 ACTIVITEITENDIAGRAM....26
6.3 TESTCASES....27
6.4 TESTRAPPORT....33
6.5 ISSUES....36

7. STRESS- EN LOADTEST....37

7.1 STRESS- EN LOADTEST....37
7.2.1 TESTCASE....37
7.2.2 TESTRAPPORT....37

8. RELIABILITY TEST....38
8.1 TESTSCENARIO'S....38
8.2 TESTRAPPORT....39
8.3 ISSUES....39
9. GEBRUIKERSTEST....40
9.1 DE ENQUÊTE....40
9.2 RESULTATEN OVER DE WERKING VAN DE APPLICATIE....43
9.3 RESULTATEN OVER DE GEBRUIKSVRIENDELIJKHEID VAN DE APPLICATIE....44
9.4 GEBRUIK VAN DE APPLICATIE IN HET ALGEMEEN....44
9.4 CONCLUSIES....45
10. UITBREIDINGEN EN TOEPASSINGEN....46
10.1 UITBREIDINGEN....46
10.2 TOEPASSINGEN....47
VERKLARENDE WOORDENLIJST....48
BRONVERMELDING....49

I. Inleiding

I.I Aanleiding

Door Marleine van Kampen, afstudeerder van De Haagse Hogeschool Delft wordt gedurende zeventien weken gewerkt aan het project “Waar is mijn eigenaar?”. Het project is een onderzoek naar en ontwikkeling van een Android applicatie bovenop de bestaande platformen van de dochterbedrijven, Sense en ASK, van het Rotterdamse onderzoeksinstituut Almende.

I.II Doel

Het doel van de transistion fase is het afronden van de ontwikkeling van de Find My Owner applicatie. Deze afronding omvat voornamelijk het testen van de applicatie. Aan de hand van verschillende tests wordt aangetoond dat de applicatie voldoet aan de eisen.

Daarnaast valt onder de afronding ook het geven van een aanbeveling over het verder te ontwikkelen product en de gebruikershandleiding bij de applicatie.

I.III Scope

Het document bevat voornamelijk de tests die van toepassing zijn op de Find My Owner applicatie.

Het document wordt afgesloten met de gebruikershandleiding en de aanbeveling. Omdat de applicatie kan worden gebruikt int meerde toepassingsgebieden is er ook een lijst gemaakt met mogelijke toepassingen.

1. Globaal testplan

Voor het testen van de “Find My Owner” applicatie moeten verschillende tests worden uitgevoerd. Hierbij gaat het om tests van de code maar ook om gebruikerstests. Als eerste is er een globaal testplan waarmee bepaald wordt welke tests er opgezet en uitgevoerd worden in de verschillende iteraties.

1.1 Onderzoek testtechnieken

De verschillende tests kunnen niet door middel van één testtechniek worden uitgevoerd. Voor de tests moet dus eerst onderzocht worden welke verschillende technieken er bestaan. Aan de hand van deze informatie kan bepaald worden welke technieken gebruikt worden.

Los van de testtechnieken moet er als eerste een onderscheid gemaakt worden tussen blackbox testen, whitebox testen en glassbox testen.

- Blackbox testen
  Totaal geen kennis van de code van de applicatie. Bij blackbox testen wordt de applicatie getest op werking en uiterlijk. Door een stappenplan kan door de GUI worden heen gelopen om te kijken of de eisen hiervoor kloppen.
- Whitebox testen
  De code is volledig bekend bij de tester, in de meeste gevallen wordt een whitebox test ook uitgevoerd door de programmeur zelf. Voorbeelden van whitebox testen zijn debugen en unit tests.
- Glassbox testen[1]
  Glassbox testen ligt tussen het white- en blackbox testen in. Bij het testen is er wél kennis van de code maar mag deze code niet aangepast worden. Bij het testen mag de kennis van de code dus gebruikt worden.

Daarnaast zijn er twee verschillende soorten testen, namelijke statische en dynamische testen.

- Statisch testen
  Statisch testen wordt uitgevoerd zonder het systeem daadwerkelijk uit te voeren. Een voorbeeld van statisch testen is de syntaxiscontrole door de compiler.
- Dynamisch testen
  Bij dynamische testen vindt controle van het draaiende systeem plaats. Een voorbeeld van dynamisch testen is het draaien van een computerprogramma en controleren of het programma een functie correct uitvoert.

1 *Ook wel Pinkbox testen genoemd*

Testtechniek	Beschrijving
Inspectie	Bij inspectie wordt er met een team gekeken naar het systeem, van code tot documenten. Hierbij kunnen vroegtijdig fouten uit het systeem gehaald worden, en kan de rest van het team iets leren over de opbouw van het systeem.
Algoritme test	De algoritme test richt zich op het verwerken van gegevens. Het systeem wordt getest op de verwerkingsalgoritmes. Er wordt niet gekeken naar de gegevens maar de verwerking van de gegevens.
Beslissingstabellen test	Aan de hand van tabellen bepalen of het systeem het juiste resultaat geeft. Aan de hand van verschillende condities wordt bepaald wat het resultaat moet zijn. Door het uitvoeren van de bepaalde condities wordt gekeken of het verwachte resultaat is behaald.
Syntactische test	Testen of de beperkingen aan de invoer (soms ook uitvoer) goed zijn geïmplementeerd. Denk aan invoervelden, in- en uitgaande berichten. Maar ook: werken de functietoetsen, is de lay-out van de schermen goed is (volgens de specificaties).
Equivalentieklassen test	Bij de equivalentieklassen test worden de ingevoerde waardes getest. Door het invullen van geldige en ongeldige waardes wordt het systeem getest op de controle van ingevoerde gegevens. Stel 0-100 = A en > 100 = B, getest wordt de invoer 99 en 101.
Grenswaarden test	Bij de grenswaarden test worden de ingevoerde waardes getest. Anders dan bij de equivalentieklassen test, wordt er bij de grenswaarden test gekeken naar de grenswaardes. Stel 0-100 = A en >100 = B, getest wordt invoer van 99, 100 en 101.
Procescyclus test	De procescyclustest richt zich op de procedures die binnen het systeem geïmplementeerd zijn. Hierbij wordt gekeken naar het geautomatiseerde deel van het systeem. Er wordt gekeken of de procedures die de gebruiker moet uitvoeren voor het invullen van gegevens goed verwerkt zijn in het systeem.
Coverage	Aan de hand van activiteitendiagrammen van het systeem worden de verschillende stappen van het systeem doorlopen. Dit kan gaan over het testen van de paden, condities of beslispunten. In sommige gevallen kan hier ook een combinatie van gemaakt worden.
State transition test	Test de overgangen van een state transition diagram (toestandsdiagrammen). Net als bij Coverage is het mogelijk om verschillende combinaties van elementen van de toestandsdiagrammen te testen. Testgevallen kunnen bepaald worden door mogelijke ‘paden’ te definiëren langs de transities. Ook ongeldige gevallen kunnen getest worden (die leveren dus geen transitie op).
Stress- en load test	Een load test is een test waarin het programma wordt belast met een bepaalde hoeveelheid representatieve data (en bijvoorbeeld aantal gebruikers). Je wilt dan weten of het programma voldoet aan de prestatie eisen. Bij een stress test voer je de load steeds verder op om te kijken bij welke load je nog net aan de eisen kunt voldoen.
Reliability test	Dit is een duurtest, die de stabiliteit van een systeem test. Het programma blijft dus vooral langdurig lopen. Het is een soort van load test waarbij de nadruk niet ligt op welke load het programma aan kan, maar op of de performance in de loop van de tijd niet terugloopt (of dat het programma niet crasht).
Usability test	Doel van usability testen is te testen hoe gebruikers het programma gebruiken. Het gaat hier om zaken als performance, (hoe snel kunnen mensen bepaalde taken uitvoeren), hoeveel fouten mensen maken bij het gebruiken van je programma, vinden mensen het een prettig programma etc.

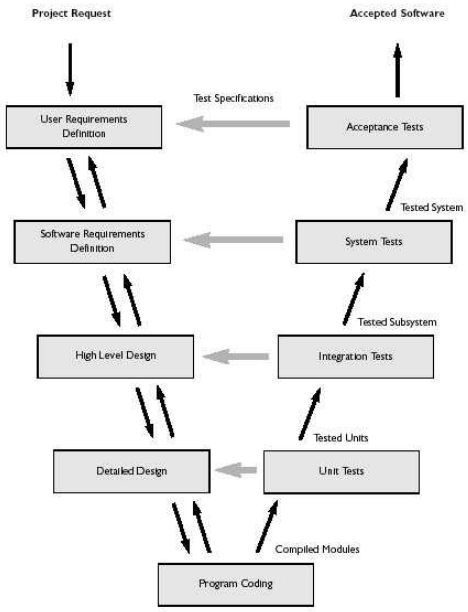
Tabel 1: Overzicht testtechnieken

1.2 V-Model

Getest wordt volgens het V-Model:
Aan de hand van elk onderdeel dat gemaakt wordt voor het systeem, wordt later in de ontwikkeling een test uitgevoerd.

Als eerst worden de stappen Analyse, Ontwerp en Implementatie uitgevoerd. Uit deze stappen komen eisen, diagrammen en code. Op het moment dat een deel code af is wordt er vanaf daar begonnen met testen. Als eerste wordt het stukje code getest. Als er meer stukken code beschikbaar zijn wordt een uitgebreidere test uitgevoerd. Bijvoorbeeld aan de hand van de stappen van het activiteitendiagram.

Uiteindelijk als alle code klaar is wordt er gekeken of er aan alle eisen is voldaan door verschillende acceptatietesten.



Figuur 1: V-Model

Om te voldoen aan de eisen van de gebruiker wordt er een gebruikerstest uitgevoerd. Verschillende personen gebruiken de applicatie en geven hier een oordeel over.

1.3 Te gebruiken testtechnieken

Aan de hand van de beschikbare testtechnieken en het gebruik van het V-Model is besloten de volgende testtechnieken toe te passen.

Inspectie	Wekelijks wordt met de begeleiders de ontwerpen en de code doorgesproken. Uit deze bespreking komen verbeteringen naar voren en deze worden weer verwerkt in volgende iteraties.
Syntactische test	De GUI is simpel maar bevat enkele invoervelden. Deze worden getest of deze geen rare taferelen gaan uitvoeren. Van te voren wordt bepaald welke sittuaties er mogelijk kunnen optreden.
Procescyclustest	De algemene cyclus voor het systeem wordt doorlopen, inclusief wat er onder water gebeurt (om de eigenaar te vinden).
Coverage	Voor de verschillende onderdelen in het systeem worden activiteitendiagrammen gemaakt. Deze worden doorgelopen om bijvoorbeeld te kijken of alle paden worden doorlopen. State tranisition test Voor het systeem gelden verschillende toestanden. Deze toestanden zijn verwerkt in het toestandsdiagrammen. Met deze test wordt bijvoorbeeld onderzocht of de applicatie daadwerkelijk wel een keer de eigenaar gaat zoeken of dat deze niet in de status “Wondering” komt.
Stress- load- en reliabilitytest	Aan de applicatie zijn verschillende eisen wat betreft batterijduur gesteld. In de stress- en load test worden deze aspecten getest.
Usability test	Aan de hand van gebruikerstesten wordt gekeken of de applicatie bruikbaar en bedienbaar is voor de gebruikers.

2. GUI

2.1 GUI

In de tweede iteratie is gewerkt aan het ontwerp en de basis implementatie van de GUI. In de ontwerpfase is het activiteitendiagram gemaakt. Het activiteitendiagram geeft vooral de verloop van schermen in de GUI weer. Door het doorlopen van het activiteitendiagram kan gekeken worden of alles verwerkt is in de GUI.

Aan het eind van de ontwikkeling van de applicatie wordt de GUI opnieuw getest. Het kan dan zijn dat er onderdelen bij gekomen zijn of dat er werking is weggevallen. Daarnaast wordt aan het einde van de ontwikkeling een gebruikerstest uitgevoerd met gebruikers die wel en niet bekend zijn met het MyASK systeem.

2.2 Eisen

Voor de GUI zijn de volgende eisen opgesteld:

- Voor de gebruiker zou de interface (GUI) simpel en duidelijk moeten zijn (*should*)
  In de GUI zou de gebruiker kunnen aangeven of hij zijn telefoon laat liggen (door uitschakelen services) (*could*)
- Bij de eerste keer gebruik van de applicatie wordt de gebruiker gevraagd een stemband op te nemen voor belacties naar collega's / vrienden (De gebruiker moet een template voor gesprekken kunnen opzetten) (*should*)
- Bij de eerste keer gebruik van de applicatie wordt de gebruiker gevraagd of hij een standaard e-mail of zelf geschreven e-mail wil verzenden (De gebruiker moet een template voor e-mails kunnen opzetten) (*should*)

2.3 Checklist eisen

Aan de hand van het doorlopen van de applicatie kan gekeken worden of er voldaan is aan deze eisen.

De eisen worden los doorlopen maar ook verwerkt in de tests van de verschillende versies en in de gebruikerstest. Uit de gebruikerstest wordt direct gehaald of de gebruiker de eis ook goed verwerkt vindt.

Eis	√
Voor de gebruiker zou de interface (GUI) simpel en duidelijk moeten zijn (*should*)	√
In de GUI zou de gebruiker kunnen aangeven of hij zijn telefoon laat liggen (door uitschakelen services) (*could*)	√
Bij de eerste keer gebruik van de applicatie wordt de gebruiker gevraagd een stemband op te nemen voor belacties naar collega's / vrienden (De gebruiker moet een template voor gesprekken kunnen opzetten) (*should*)	X
Bij de eerste keer gebruik van de applicatie wordt de gebruiker gevraagd of hij een standaard e-mail of zelf geschreven e-mail wil verzenden (De gebruiker moet een template voor e-mails kunnen opzetten) (*should*)	X

Tabel 2: Checklist GUI Eisen

De opzet van sms, telefoon en e-mail zijn in de ASK installatie op een andere manier opgelost waardoor het maken van templates niet meer van toepassing is in de applicatie. Het is in een uitbreiding wel mogelijk om de gebruiker te laten bepalen wat er in de e-mail komt. Enkel moet er dan een andere opzet gemaakt worden in ASK om per gebruiker een andere template te bouwen.

3. Test applicatie na eerste oplevering

3.1 Inleiding

De applicatie is in vier verschillende versies uitgewerkt. Elke versie bevat een extra deel van het eindresultaat. Per versie is een test opgezet om te testen of de eisen van die versie goed verwerkt zijn.

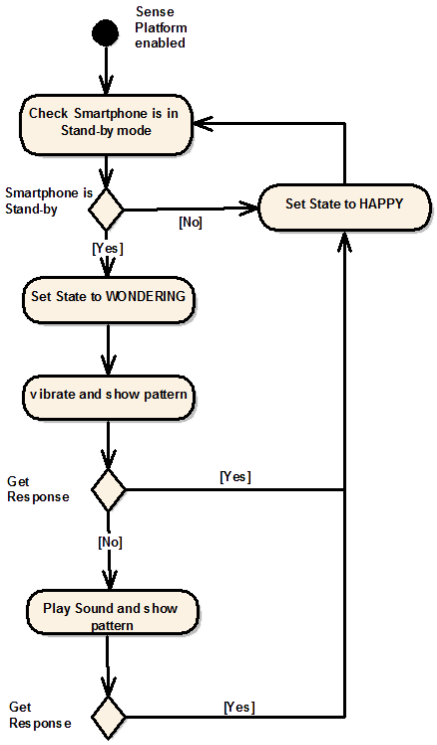
Tijdens het ontwikkelen wordt regelmatig getest of stukken code juist werken. De kleine tests zijn niet verwerkt in het testplan. In het testplan gaat het voornamelijk om de werking van elke versie en de eisen die ermee samenhangen.

In de eerste versie van de applicatie zijn de volgende onderdelen verwerkt:

- Basiscode
- Werking achter de GUI
- Alle herkenningsfuncties zonder samenwerking met Sense en ASK

Aan de hand van verschillende testcases worden de onderdelen getest. In het testrapport is te lezen wat deze tests hebben opgeleverd.

3.2 Activiteitendiagram



Figuur 2: Activiteiten Diagram - Eerste oplevering

In bovenstaand diagram is te zien welk onderdeel van het totale activiteitendiagram is uitgewerkt in de eerste oplevering. Aan de hand van het diagram zijn de de testcases (in de volgende paragraaf) opgebouwd.

3.3 Testcases

Conditie	V1.1	V1.2	V1.3
Happy	1	1	0
Wondering	0	0	1
Eigenaar gebruikt telefoon	0	1	1
Wondering (Recognizing)	1	0	0
Happy (Idle)	0	1	1

Tabel 3: Tabelgestuurde test - Scenario's uit het activiteitendiagram

Testcase	Omschrijving
Testcase ID	V1.1
Omschrijving	De eigenaar gebruikt zijn telefoon niet
Doelstelling	Na 15 minuten gaat de telefoon opzoek naar zijn eigenaar
Pre-conditie	De eigenaar gebruikt zijn telefoon niet
Testprocedure	1. Laat de telefoon liggen 2. Reageer na 15 minuten op de herkenningsacties
Post-conditie	Na 15 minuten gaat de telefoon opzoek naar zijn eigenaar
Verwacht resultaat	De status van de applicatie is Happy en na 15 minuten Wondering

Testscenario V1.1

Testcase	Omschrijving
Testcase ID	V1.2
Omschrijving	De eigenaar gebruikt de telefoon, hierdoor blijft de status van de applicatie is Happy en wacht de applicatie tot dat de eigenaar de telefoon 15 minuten niet gebruikt.
Doelstelling	Geen status veranderingen
Pre-conditie	De status van de applicatie is Happy
Testprocedure	1. Haal de telefoon uit stand-by stand (knop bovenop en teken het patroon) 2. Gebruik de telefoon gedurende een paar secondes 3. Zet de telefoon in stand-by stand
Post-conditie	De telefoon gaat pas na 15 minuten ná in stand-by stand zetten van de telefoon opzoek naar de eigenaar
Verwacht resultaat	De status van de applicatie is Happy

Testscenario V1.2

Testcase	Omschrijving
Testcase ID	V1.3
Omschrijving	De eigenaar heeft langer dan 15 minuten zijn telefoon niet gebruikt. De telefoon gaat kijken of de eigenaar in de buurt is. Doordat de eigenaar reageert, gaat de applicatie niet verder zoeken is hij weer tevreden.
Doelstelling	De eigenaar herkennen, de eigenaar reageert
Pre-conditie	Status is Wondering
Testprocedure	1. Gebruik de telefoon 2. Zet de telefoon in stand-by stand 3. Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken 4. De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet 5. De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer
Post-conditie	Status is Happy
Verwacht resultaat	De applicatie wacht weer tot dat de eigenaar 15 minuten afwezig is

Testscenario V1.3

3.4 Testrapport

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V1.1		
PRE	Status is Happy		
1	Laat de telefoon liggen	√	
2	Reageer na 15 minuten op de herkenningsacties	√	
POST	Status is Happy / Wondering	√	

Test V1.1

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V1.2		
PRE	De status van de applicatie is Happy		
1	Haal de telefoon uit stand-by stand (knop bovenop en teken het patroon)	√	
2	Gebruik de telefoon gedurende een paar secondes	√	
3	Zet de telefoon in stand-by stand	√	
POST	Status is Happy	√	

Test V1.2

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V1.3		
PRE	Status is Wondering		
1	Gebruik de telefoon	√	
2	Zet de telefoon in stand-by stand	√	
3	Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken	√	
4	De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet	√	
5	De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer	√	Telefoon ging na het goedkeuren van het patroon weer opnieuw opzoek. Wordt opgelost in tweede versie
POST	Status is Happy	√	

Test V1.3

3.5 Issues

- *Opnieuw herkenningsfuncties uitvoeren na het reageren op een herkenningsactie*
  Opgelost door de alarmen die afhangen van de herkenningsacties beter te configureren.

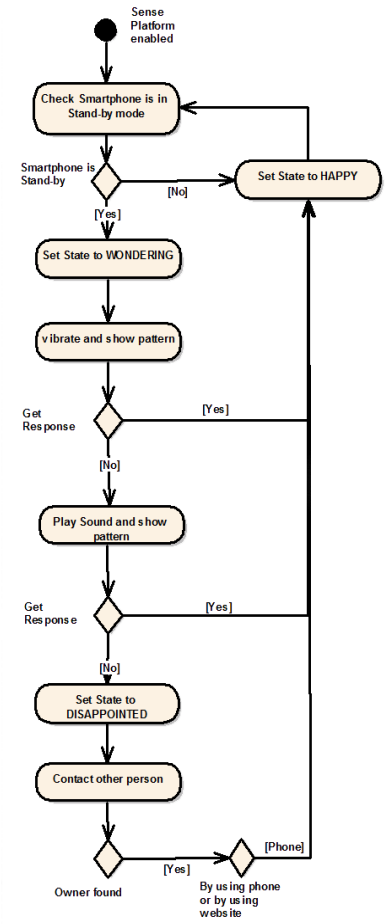
4. Test applicatie na tweede oplevering

4.1 Inleiding

In de tweede oplevering is de verbinding met Sense en ASK opgezet. Hierdoor kan er ook al een vind-actie worden uitgevoerd. In de volgende paragrafen is meer te vinden over de test van de tweede oplevering.

4.2 Activiteitendiagram

In bovenstaand figuur is het tweede deel van het activiteitendiagram te zien. Hierbij is de basis hetzelfde als in de eerst versie van de applicatie. Enkel is de mogelijkheid voor de toestand *Disappointed* erbij gekomen en daarbij ook een vind-actie "*Contact other person".*



Figuur 3: Activiteiten Diagram - Tweede oplevering

4.3 Testcases

Conditie	V2.1	V2.2	V2.3	V2.4	V2.5
Happy	1	1	0	0	0
Wondering	0	0	1	1	0
Disappointed	0	0	0	0	1
Eigenaar gebruikt telefoon	0	1	0	1	1
Wondering (Recognizing)	1	0	0	0	0
Disappointed (Finding)	0	0	1	0	0
Happy (Idle)	0	1	0	1	1

Tabel 4: Tabelgestuurde test - Scenario's uit het activiteitendiagram

Testcase	Omschrijving
Testcase ID	V2.1
Omschrijving	De eigenaar gebruikt zijn telefoon niet
Doelstelling	Na 15 minuten gaat de telefoon opzoek naar zijn eigenaar
Pre-conditie	De eigenaar gebruikt zijn telefoon niet
Testprocedure	1. Laat de telefoon liggen 2. Reageer na 15 minuten op de herkenningsacties
Post-conditie	Status is Happy / Wondering
Verwacht resultaat	De status van de applicatie is Happy en na 15 minuten Wondering

Testscenario V2.1

Testcase	Omschrijving
Testcase ID	V2.2
Omschrijving	De eigenaar gebruikt de telefoon, hierdoor blijft de status van de applicatie Happy en wacht de applicatie tot dat de eigenaar de telefoon 15 minuten niet gebruikt.
Doelstelling	Geen status veranderingen
Pre-conditie	Status is Happy
Testprocedure	1. Haal de telefoon uit stand-by stand (knop bovenop en teken het patroon) 2. Gebruik de telefoon gedurende een paar secondes 3. Zet de telefoon in stand-by stand
Post-conditie	De telefoon gaat pas na 15 minuten na in stand-by stand zetten van de telefoon opzoek naar de eigenaar
Verwacht resultaat	Status is Happy

Testscenario V2.2

Testcase	Omschrijving
Testcase ID	V2.3
Omschrijving	De eigenaar heeft langer dan 15 minuten zijn telefoon niet gebruikt. De telefoon gaat kijken of de eigenaar in de buurt is. Doordat de eigenaar niet reageert, gaat de applicatie over op een zoekactie.
Doelstelling	De eigenaar herkennen, maar de eigenaar reageert niet
Pre-conditie	Status is Happy
Testprocedure	1. Gebruik de telefoon 2. Zet de telefoon in stand-by stand 3. Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken 4. De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet 5. De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer niet
Post-conditie	Status is Disappointed
Verwacht resultaat	De applicatie gaat opzoek naar de eigenaar

Testscenario V2.3

Testcase	Omschrijving
Testcase ID	V2.4
Omschrijving	De eigenaar heeft langer dan 15 minuten zijn telefoon niet gebruikt. De telefoon gaat kijken of de eigenaar in de buurt is. Doordat de eigenaar reageert, gaat de applicatie niet verder zoeken is hij weer tevreden.
Doelstelling	De eigenaar herkennen, de eigenaar reageert
Pre-conditie	Status is Wondering
Testprocedure	1. Gebruik de telefoon 2. Zet de telefoon in stand-by stand 3. Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken 4. De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet 5. De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer
Post-conditie	Status is Happy
Verwacht resultaat	De applicatie wacht weer tot dat de eigenaar 15 minuten afwezig is

Testscenario V2.4

Testcase	Omschrijving
Testcase ID	V2.5
Omschrijving	De eigenaar heeft langer dan 15 minuten zijn telefoon niet gebruikt. De telefoon gaat kijken of de eigenaar in de buurt is. Doordat de eigenaar niet reageert op de herkenningsacties gaat de applicatie over op zoekacties, hier reageert de eigenaar op.
Doelstelling	De eigenaar vinden, de eigenaar reageert door zijn telefoon weer te gebruiken
Pre-conditie	Status is Disappointed
Testprocedure	1. Gebruik de telefoon 2. Zet de telefoon in stand-by stand 3. Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken 4. De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet 5. De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer niet 6. De applicatie belt op een ander nummer, reageer niet 7. De applicatie stuurt per e-mail de locatie van de telefoon, reageer door de telefoon te gebruiken
Post-conditie	Status is Happy
Verwacht resultaat	De applicatie wacht weer tot dat de eigenaar 15 minuten afwezig is

Testscenario V2.5

4.4 Testrapport

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V2.1		
PRE	Status is Happy	√	
1	Laat de telefoon liggen	√	
2	Reageer na 15 minuten op de herkenningsacties	√	
POST	Status is Happy / Wondering	√	

Test V2.1

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V2.2		
PRE	Status is Happy		
1	Haal de telefoon uit stand-by stand (knop bovenop en teken het patroon)	√	
2	Gebruik de telefoon gedurende een paar secondes	√	
3	Zet de telefoon in stand-by stand	√	
POST	Status is Happy	√	

Test V2.2

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V2.3		
PRE	Status is Happy		
1	Gebruik de telefoon	√	
2	Zet de telefoon in stand-by stand	√	
3	Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken (status is Wondering)	√	
4	De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet	√	
5	De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer niet	√	
POST	Status is Disappointed	√	

Test V2.3

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V2.4		
PRE	Status is Wondering		
1	Gebruik de telefoon	√	
2	Zet de telefoon in stand-by stand	√	
3	Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken	√	
4	De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet	√	Problemen met de volgorde van uitvoer van functies. De applicatie speelt ook direct geluid af
5	De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer	√	
POST	Status is Happy	√	

Test V2.4

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V2.5		
PRE	Status is Disappointed		
1	Gebruik de telefoon	√	
2	Zet de telefoon in stand-by stand	√	
3	Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken	√	
4	De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet	√	
5	De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer niet	√	
6	De applicatie belt op een ander nummer, reageer niet	√	Tegelijkertijd e-mail sturen en bellen.
7	De applicatie stuurt per e-mail de locatie van de telefoon, reageer door de telefoon te gebruiken	√	
POST	Status is Happy	√	

Test V2.5

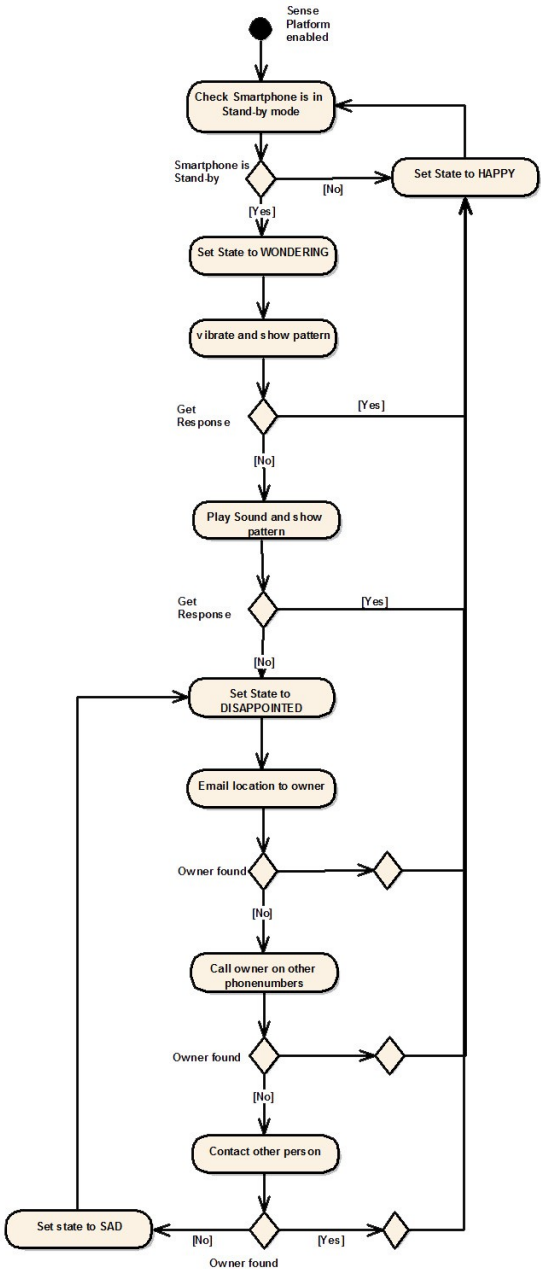
4.5 Issues

- *De functies worden tegelijkertijd uitgevoerd. Er zit geen wachttijd tussen het uitvoeren van de eerste functie en de tweede. Hierdoor wordt er meteen een geluidsignaal afgespeeld of tegelijkertijd een e-mail en een telefoontje.*
  Opgelost door een *RepeatAl* *arm* aan te maken. Het alarm gaat elke vijf minuten "af", waardoor de vindacties opnieuw worden aangeroepen. Afhankelijk van de toestand van de applicatie (Wondering of Jealous) worden de juiste vindacties uitgevoerd.

5. Test applicatie na derde oplevering

In de derde oplevering is onder andere de toestand Sad toegevoegd. Deze toestand zorgt ervoor dat de applicatie wacht met opnieuw zoeken naar de eigenaar. Er kunnen zich situaties voordoen waarbij de eigenaar niet vindbaar is voor de applicatie. Om niet constant te zoeken wordt er een tussenpoze van één uur gezet. Als de eigenaar de telefoon in de tussentijd gebruikt wordt de toestand gewoon Happy.

5.1 activiteitendiagram



Figuur 4: Activiteitendiagram na derde oplevering

5.2 Testcases

Conditie	V3.1	V3.2	V3.3	V3.4	V3.5	V3.6	V3.7	V3.8
Happy	1	1	0	0	0	0	0	0
Wondering	0	0	1	1	0	0	0	0
Disappointed	0	0	0	0	1	1	0	0
Sad	0	0	0	0	0	0	1	1
Eigenaar gebruikt telefoon	0	1	0	1	0	1	0	1
Wondering (Recognizing)	1	0	0	0	0	0	0	0
Disappointed (Finding)	0	0	1	0	0	0	0	0
Happy (Idle)	0	1	0	1	0	1	0	1
Sad (Idle)	0	0	0	0	1	0	1	0

Tabel 5: Tabelgestuurde test - Scenario's uit het activiteitendiagram

Testcase	Omschrijving
Testcase ID	V3.1
Omschrijving	De eigenaar gebruikt zijn telefoon niet
Doelstelling	Na 15 minuten gaat de telefoon opzoek naar zijn eigenaar
Pre-conditie	De eigenaar gebruikt zijn telefoon niet
Testprocedure	1. Laat de telefoon liggen 2. Reageer na 15 minuten op de herkenningsacties
Post-conditie	Status is Happy / Wondering
Verwacht resultaat	De status van de applicatie is Happy en na 15 minuten Wondering

Testscenario V3.1

Testcase	Omschrijving
Testcase ID	V3.2
Omschrijving	De eigenaar gebruikt de telefoon, hierdoor blijft de status van de applicatie is Happy en wacht de applicatie tot dat de eigenaar de telefoon 15 minuten niet gebruikt.
Doelstelling	Geen status veranderingen
Pre-conditie	Status is Happy
Testprocedure	1. Haal de telefoon uit stand-by stand (knop bovenop en teken het patroon) 2. Gebruik de telefoon gedurende een paar secondes 3. Zet de telefoon in stand-by stand
Post-conditie	De telefoon gaat pas na 15 minuten ná in stand-by stand zetten van de telefoon opzoek naar de eigenaar
Verwacht resultaat	Status is Happy

Testscenario V3.2

Testcase	Omschrijving
Testcase ID	V3.3
Omschrijving	De eigenaar heeft langer dan 15 minuten zijn telefoon niet gebruikt. De telefoon gaat kijken of de eigenaar in de buurt is. Doordat de eigenaar niet reageert, gaat de applicatie over op een zoek actie.
Doelstelling	De eigenaar herkennen, maar de eigenaar reageert niet
Pre-conditie	Status is Wondering
Testprocedure	1. Gebruik de telefoon 2. Zet de telefoon in stand-by stand 3. Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken 4. De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet 5. De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer niet
Post-conditie	Status is Disappointed
Verwacht resultaat	De applicatie gaat opzoek naar de eigenaar

Tabel 6: Testscenario V3.3

Testcase	Omschrijving
Testcase ID	V3.4
Omschrijving	De eigenaar heeft langer dan 15 minuten zijn telefoon niet gebruikt. De telefoon gaat kijken of de eigenaar in de buurt is. Doordat de eigenaar reageert, gaat de applicatie niet verder zoeken is hij weer tevreden.
Doelstelling	De eigenaar herkennen, de eigenaar reageert
Pre-conditie	Status is Wondering
Testprocedure	1. Gebruik de telefoon 2. Zet de telefoon in stand-by stand 3. Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken 4. De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet 5. De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer
Post-conditie	Status is Happy
Verwacht resultaat	De applicatie wacht weer tot dat de eigenaar 15 minuten afwezig is

Testscenario V3.4

Testcase	Omschrijving
Testcase ID	V3.5
Omschrijving	De eigenaar heeft langer dan 15 minuten zijn telefoon niet gebruikt. De telefoon gaat kijken of de eigenaar in de buurt is. Doordat de eigenaar niet reageert op de herkenningsacties gaat de applicatie over op zoekacties
Doelstelling	De eigenaar vinden, de eigenaar reageert niet
Pre-conditie	Status is Happy
Testprocedure	1. Gebruik de telefoon 2. Zet de telefoon in stand-by stand 3. Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken 4. De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet 5. De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer niet 6. De applicatie belt op een ander nummer, reageer niet 7. De applicatie stuurt per e-mail de locatie van de telefoon, reageer niet 8. De applicatie gaat andere personen bellen, laat ze reageren als niet wetend waar de eigenaar is
Post-conditie	Status is Sad
Verwacht resultaat	De applicatie wacht 60 minuten en gaat weer een zoekactie uitvoeren.

Testscenario V3.5

Testcase	Omschrijving
Testcase ID	V3.6
Omschrijving	De eigenaar heeft langer dan 15 minuten zijn telefoon niet gebruikt. De telefoon gaat kijken of de eigenaar in de buurt is. Doordat de eigenaar niet reageert op de herkenningsacties gaat de applicatie over op zoekacties, hier reageert de eigenaar op.
Doelstelling	De eigenaar vinden, de eigenaar reageert door zijn telefoon weer te gebruiken
Pre-conditie	Status is Disappointed
Testprocedure	1. Gebruik de telefoon 2. Zet de telefoon in stand-by stand 3. Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken 4. De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet 5. De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer niet 6. De applicatie belt op een ander nummer, reageer niet 7. De applicatie stuurt per e-mail de locatie van de telefoon, reageer door de telefoon te gebruiken
Post-conditie	Status is Happy
Verwacht resultaat	De applicatie wacht weer tot dat de eigenaar 15 minuten afwezig is

Testscenario V3.6

Testcase	Omschrijving
Testcase ID	V3.7
Omschrijving	De eigenaar heeft langer dan 15 minuten zijn telefoon niet gebruikt. De telefoon gaat kijken of de eigenaar in de buurt is. Doordat de eigenaar niet reageert op de herkenningsacties gaat de applicatie over op zoekacties, ook hier reageert de eigenaar niet op. Na een uur gaat de applicatie opnieuw opzoek naar de eigenaar.
Doelstelling	De eigenaar vinden, de eigenaar reageert nog steeds niet
Pre-conditie	Status is Sad
Testprocedure	1. Gebruik de telefoon 2. Laat de applicatie alle zoek-acties uitvoeren en reageer niet 3. Na 60 minuten voert de applicatie opnieuw vind-acties uit, reageer niet
Post-conditie	Status is Sad
Verwacht resultaat	De applicatie wacht 60 minuten en gaat weer een zoekactie uitvoeren.

Testscenario V3.7

Testcase	Omschrijving
Testcase ID	V3.8
Omschrijving	De eigenaar heeft langer dan 15 minuten zijn telefoon niet gebruikt. De telefoon gaat kijken of de eigenaar in de buurt is. Doordat de eigenaar niet reageert op de herkenningsacties gaat de applicatie over op zoekacties, ook hier reageert de eigenaar niet op. Na een uur gaat de applicatie opnieuw opzoek naar de eigenaar.
Doelstelling	De eigenaar vinden, de eigenaar gebruikt binnen de 60 minuten zijn telefoon weer
Pre-conditie	Status is Sad
Testprocedure	1. Gebruik de telefoon 2. Laat de applicatie alle zoek-acties uitvoeren en reageer niet 3. Wacht 10 minuten nadat de telefoon geen acties meer uitvoert 4. Gebruik de telefoon
Post-conditie	Status is Happy
Verwacht resultaat	De applicatie wacht weer tot dat de eigenaar 15 minuten afwezig is

Testscenario V3.8

5.3 Testrapport

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V3.1		
PRE	Status is Happy		
1	Laat de telefoon liggen	√	
2	Reageer na 15 minuten op de eerste herkenningsactie	√	
POST	Status is Wondering /Happy	√	

Test V3.1

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V3.2		
PRE	Status is Happy		
1	Haal de telefoon uit stand-by stand (knop bovenop en teken het patroon)	√	
2	Gebruik de telefoon gedurende een paar secondes	√	
3	Zet de telefoon in stand-by stand	√	
POST	Status is Happy	√	

Test V3.2

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V3.3		
PRE	Status is Wondering	√	
1	Gebruik de telefoon	√	
2	Zet de telefoon in stand-by stand	√	
3	Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken	√	
4	De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet	√	
5	De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer niet	√	
POST	Status is Disappointed	√	

Test V3.3

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V3.4		
PRE	Status is Wondering	√	
1	Gebruik de telefoon	√	
2	Zet de telefoon in stand-by stand	√	
3	Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken	√	
4	De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet	√	
5	De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer	√	
POST	Status is Happy	√	

Test V3.4

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V3.5		
PRE	Status is Happy		
1	Gebruik de telefoon	√	
2	Zet de telefoon in stand-by stand	√	
3	Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken	√	
4	De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet	√	
5	De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer niet	√	
6	De applicatie belt op een ander nummer, reageer niet	√	
7	De applicatie stuurt per e-mail de locatie van de telefoon, reageer niet	√	
8	De applicatie gaat andere personen bellen, laat ze reageren als niet wetend waar de eigenaar is	√	
POST	Status is Sad	X	De toestand wordt niet Sad

Test V3.5

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V3.6		
PRE	Status is Disappointed		
1	Gebruik de telefoon	√	
2	Zet de telefoon in stand-by stand	√	
3	Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken	√	
4	De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet	√	
5	De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer niet	√	
6	De applicatie belt op een ander nummer, reageer niet	√	
7	De applicatie stuurt per e-mail de locatie van de telefoon, reageer door de telefoon te gebruiken	√	
POST	Status is Happy	√	

Test V3.6

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V3.7		
PRE	Status is Sad	√	
1	Gebruik de telefoon	√	
2	Laat de applicatie alle zoek-acties uitvoeren en reageer niet	X	
3	Na 60 minuten voert de applicatie opnieuw vind-acties uit, reageer niet	X	
POST	Status is Sad	X	De toestand wordt niet Sad

Test V3.7

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V3.8		
PRE	Status is Sad		
1	Gebruik de telefoon	√	
2	Laat de applicatie alle zoek-acties uitvoeren en reageer niet	√	
3	Wacht 10 minuten nadat de telefoon geen acties meer uitvoert	√	
4	Gebruik de telefoon	√	
POST	Status is Happy	√	

Test V3.8

5.5 Issues

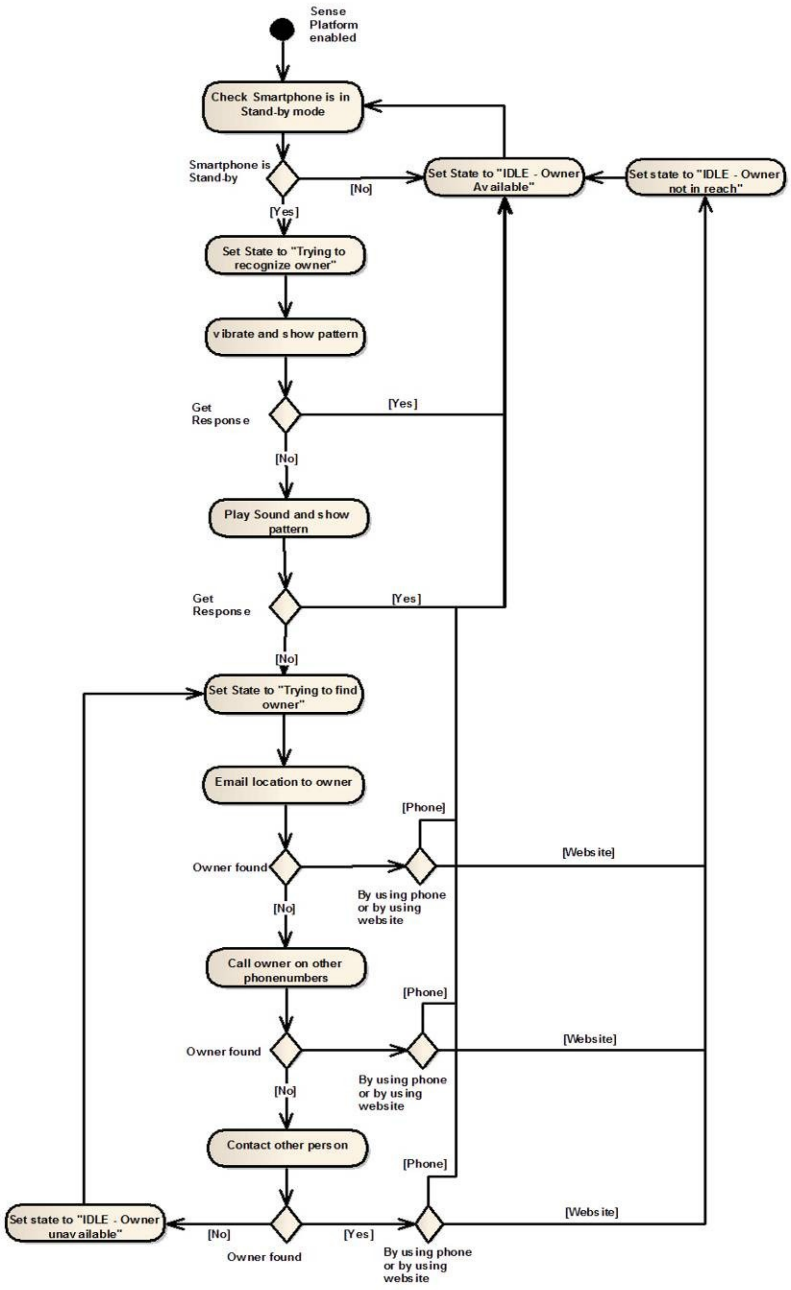
- *Overgang van Disappointed naar Sad geeft problemen.*
  Opgelost door de alarmen voor het veranderen van toestand anders te configureren. Het alarm van *Sad* bleek op direct uitvoeren te staan, waardoor elke keer als de toestand *Sad* werd, het alarm werd gestart en afging. Hierdoor werd de toestand direct weer op *Disappointed* gezet.

6. Eindversie

De eindversie van de applicatie is getest zoals de eerdere opleveringen. Naast de tabelgestuurde test zijn er ook performance, reliability en gebruikerstesten uitgevoerd met de eindversie. Meer informatie over deze testen is te vinden in volgende hoofdstukken.

De voornamelijkste verandering in de eindversie is de toevoeging van de webinterface. Hierdoor kan de eigenaar zich afwezig melden bij de telefoon.

6.2 Activiteitendiagram



Figuur 5: Activiteitendiagram - Totale applicatie

6.3 Testcases

Conditie	V4.1	V4.2	V4.3	V4.4	V4.5	V4.6	V4.7	V4.8	V4.9	V4 .10	V4 .11	V4 .12	V4 .13	V4 .14
Happy	1	1	1	0	0	0	0	0	0	0	0	0	0	0
Wondering	0	0	0	1	1	1	0	0	0	0	0	0	0	0
Disappointed	0	0	0	0	0	0	1	1	1	0	0	0	0	0
Sad	0	0	0	0	0	0	0	0	0	1	1	1	0	0
Jealous	0	0	0	0	0	0	0	0	0	0	0	0	1	1
Eigenaar gebruikt telefoon	0	1	0	0	1	0	0	1	0	0	1	0	0	1
Eigenaar meldt zich via webinterface	0	0	1	0	0	1	0	0	1	0	0	1	0	0
Wondering (Recognizing)	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Disappointed (Finding)	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Happy (Idle)	0	1	0	0	1	0	0	1	0	0	1	0	0	1
Sad (Idle)	0	0	0	0	0	0	1	0	0	1	0	0	0	0
Jealous (Idle)	0	0	1	0	0	1	0	0	1	0	0	1	1	0

Tabel 7: Tabelgestuurde test - Scenario's uit het activiteitendiagram

Benodigheden
Android smartphone (v2.1) met volle batterij
Find My Owner Android applicatie
Website CommonSense (common.sense-os.nl)
Testscenario's T1 t/m T14

Tabel 8: Benodigdheden testen

Testcase	**Omschrijving**
Testcase ID	V4.1
Omschrijving	De eigenaar gebruikt zijn telefoon niet
Doelstelling	Na 15 minuten gaat de telefoon opzoek naar zijn eigenaar
Pre-conditie	De eigenaar gebruikt zijn telefoon niet
Testprocedure	1. Laat de telefoon liggen 2. Reageer na 15 minuten op de herkenningsacties
Post-conditie	Status is Happy / Wondering
Verwacht resultaat	De status van de applicatie is Happy idle en na 15 minuten Wondering

Testscenario V4.1

Testcase	Omschrijving
Testcase ID	V4.2
Omschrijving	De eigenaar gebruikt de telefoon, hierdoor blijft de status van de applicatie is Happy en wacht de applicatie tot dat de eigenaar de telefoon 15 minuten niet gebruikt.
Doelstelling	Geen status veranderingen
Pre-conditie	Status is Happy
Testprocedure	1. Haal de telefoon uit stand-by stand (knop bovenop en teken het patroon) 2. Gebruik de telefoon gedurende een paar secondes 3. Zet de telefoon in stand-by stand
Post-conditie	De telefoon gaat pas na 15 minuten ná in stand-by stand zetten van de telefoon opzoek naar de eigenaar
Verwacht resultaat	Status is Happy

Testscenario V4.2

Testcase	Omschrijving
Testcase ID	V4.3
Omschrijving	De eigenaar weet dat zijn telefoon niet in de buurt is en gaat naar de webinterface.
Doelstelling	De applicatie niet te laten zoeken naar de eigenaar omdat bekend is dat deze afwezig is.
Pre-conditie	Status is Happy
Testprocedure	1. Gebruik de telefoon even 2. Ga naar de webinterface 3. Geef aan dat de eigenaar niet in de buurt is van de telefoon 4. Wacht 15 minuten
Post-conditie	De telefoon gaat na 15 minuten niet opzoek naar zijn eigenaar
Verwacht resultaat	Status is Jealous

Testscenario V4.3

Testcase	Omschrijving
Testcase ID	V4.4
Omschrijving	De eigenaar heeft langer dan 15 minuten zijn telefoon niet gebruikt. De telefoon gaat kijken of de eigenaar in de buurt is. Doordat de eigenaar niet reageert, gaat de applicatie over op een zoek actie.
Doelstelling	De eigenaar herkennen, maar de eigenaar reageert niet
Pre-conditie	Status is Wondering
Testprocedure	1. Gebruik de telefoon 2. Zet de telefoon in stand-by stand 3. Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken 4. De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet 5. De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer niet
Post-conditie	Status is Disappointed
Verwacht resultaat	De applicatie gaat opzoek naar de eigenaar

Testscenario V4.4

Testcase	Omschrijving
Testcase ID	V4.5
Omschrijving	De eigenaar heeft langer dan 15 minuten zijn telefoon niet gebruikt. De telefoon gaat kijken of de eigenaar in de buurt is. Doordat de eigenaar reageert, gaat de applicatie niet verder zoeken is hij weer tevreden.
Doelstelling	De eigenaar herkennen, de eigenaar reageert
Pre-conditie	Status is Wondering
Testprocedure	1. Gebruik de telefoon 2. Zet de telefoon in stand-by stand 3. Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken 4. De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet 5. De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer
Post-conditie	Status is Happy
Verwacht resultaat	De applicatie wacht weer tot dat de eigenaar 15 minuten afwezig is

Testscenario V4.5

Testcase	Omschrijving
Testcase ID	V4.6
Omschrijving	De eigenaar heeft langer dan 15 minuten zijn telefoon niet gebruikt. De telefoon gaat kijken of de eigenaar in de buurt is. Doordat de eigenaar reageert, gaat de applicatie niet verder zoeken is hij weer tevreden.
Doelstelling	De eigenaar herkennen, de eigenaar reageert
Pre-conditie	Status is Wondering
Testprocedure	1. Gebruik de telefoon 2. Zet de telefoon in stand-by stand 3. Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken 4. De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet 5. Ga naar de website en meldt dat de eigenaar niet in de buurt is van de telefoon
Post-conditie	Status is Jealous
Verwacht resultaat	De applicatie wacht weer tot dat de eigenaar 15 minuten afwezig is

Testscenario V4.6

Testcase	Omschrijving
Testcase ID	V4.7
Omschrijving	De eigenaar heeft langer dan 15 minuten zijn telefoon niet gebruikt. De telefoon gaat kijken of de eigenaar in de buurt is. Doordat de eigenaar niet reageert op de herkenningsacties gaat de applicatie over op zoekacties
Doelstelling	De eigenaar vinden, de eigenaar reageert niet
Pre-conditie	Status is Disappointed
Testprocedure	1. Gebruik de telefoon 2. Zet de telefoon in stand-by stand 3. Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken 4. De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet 5. De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer niet 6. De applicatie belt op een ander nummer, reageer niet 7. De applicatie stuurt per e-mail de locatie van de telefoon, reageer niet 8. De applicatie gaat andere personen bellen, laat ze reageren als niet wetend waar de eigenaar is
Post-conditie	Status is Sad
Verwacht resultaat	De applicatie wacht 60 minuten en gaat weer een zoekactie uitvoeren.

Testscenario V4.7

Testcase	Omschrijving
Testcase ID	V4.8
Omschrijving	De eigenaar heeft langer dan 15 minuten zijn telefoon niet gebruikt. De telefoon gaat kijken of de eigenaar in de buurt is. Doordat de eigenaar niet reageert op de herkenningsacties gaat de applicatie over op zoekacties, hier reageert de eigenaar op.
Doelstelling	De eigenaar vinden, de eigenaar reageert door zijn telefoon weer te gebruiken
Pre-conditie	Status is Disappointed
Testprocedure	1. Gebruik de telefoon 2. Zet de telefoon in stand-by stand 3. Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken 4. De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet 5. De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer niet 6. De applicatie belt op een ander nummer, reageer niet 7. De applicatie stuurt per e-mail de locatie van de telefoon, reageer door de telefoon te gebruiken
Post-conditie	Status is Happy
Verwacht resultaat	De applicatie wacht weer tot dat de eigenaar 15 minuten afwezig is

Testscenario V4.8

Testcase	Omschrijving
Testcase ID	V4.9
Omschrijving	De eigenaar heeft langer dan 15 minuten zijn telefoon niet gebruikt. De telefoon gaat kijken of de eigenaar in de buurt is. Doordat de eigenaar niet reageert op de herkenningsacties gaat de applicatie over op zoekacties, hier reageert de eigenaar op.
Doelstelling	De eigenaar vinden, de eigenaar reageert via de website
Pre-conditie	Status is Disappointed
Testprocedure	1. Gebruik de telefoon 2. Zet de telefoon in stand-by stand 3. Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken 4. De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet 5. De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer niet 6. De applicatie belt op een ander nummer, reageer niet 7. De applicatie stuurt per e-mail de locatie van de telefoon, reageer via de website
Post-conditie	Status is Jealous
Verwacht resultaat	De applicatie wacht weer tot dat de eigenaar de telefoon weer gebruikt

Testscenario V4.9

Testcase	Omschrijving
Testcase ID	V4.10
Omschrijving	De eigenaar heeft langer dan 15 minuten zijn telefoon niet gebruikt. De telefoon gaat kijken of de eigenaar in de buurt is. Doordat de eigenaar niet reageert op de herkenningsacties gaat de applicatie over op zoekacties, ook hier reageert de eigenaar niet op. Na een uur gaat de applicatie opnieuw opzoek naar de eigenaar.
Doelstelling	De eigenaar vinden, de eigenaar reageert nog steeds niet
Pre-conditie	Status is Sad
Testprocedure	1. Gebruik de telefoon 2. Laat de applicatie alle zoek-acties uitvoeren en reageer niet 3. Na 60 minuten voert de applicatie opnieuw vind-acties uit, reageer niet
Post-conditie	Status is Sad
Verwacht resultaat	De applicatie wacht 60 minuten en gaat weer een zoekactie uitvoeren.

Testscenario V4.10

Testcase	Omschrijving
Testcase ID	V4.11
Omschrijving	De eigenaar heeft langer dan 15 minuten zijn telefoon niet gebruikt. De telefoon gaat kijken of de eigenaar in de buurt is. Doordat de eigenaar niet reageert op de herkenningsacties gaat de applicatie over op zoekacties, ook hier reageert de eigenaar niet op. Na een uur gaat de applicatie opnieuw opzoek naar de eigenaar.
Doelstelling	De eigenaar vinden, de eigenaar gebruikt binnen de 60 minuten zijn telefoon weer
Pre-conditie	Status is Sad
Testprocedure	1. Gebruik de telefoon 2. Laat de applicatie alle zoek-acties uitvoeren en reageer niet 3. Wacht 10 minuten nadat de telefoon geen acties meer uitvoert 4. Gebruik de telefoon
Post-conditie	Status is Happy
Verwacht resultaat	De applicatie wacht weer tot dat de eigenaar 15 minuten afwezig is

Testscenario V4.11

Testcase	Omschrijving
Testcase ID	V4.12
Omschrijving	De eigenaar heeft langer dan 15 minuten zijn telefoon niet gebruikt. De telefoon gaat kijken of de eigenaar in de buurt is. Doordat de eigenaar niet reageert op de herkenningsacties gaat de applicatie over op zoekacties, ook hier reageert de eigenaar niet op. Na een uur gaat de applicatie opnieuw opzoek naar de eigenaar.
Doelstelling	De eigenaar vinden, de eigenaar meldt zich binnen 60 minuten via de website
Pre-conditie	Status is Sad
Testprocedure	1. Gebruik de telefoon 2. Laat de applicatie alle zoek-acties uitvoeren en reageer niet 3. Wacht 10 minuten nadat de telefoon geen acties meer uitvoert 4. Gebruik de telefoon
Post-conditie	Status is Jealous
Verwacht resultaat	De applicatie wacht weer tot dat de eigenaar de telefoon weer gebruikt

Testscenario V4.12

Testcase	Omschrijving
Testcase ID	V4.13
Omschrijving	De eigenaar heeft via de website aangegeven dat hij niet aanwezig is bij zijn telefoon. De applicatie wacht dan tot dat de eigenaar weer gebruik maakt van zijn telefoon.
Doelstelling	De status blijft Jealous, de eigenaar heeft de telefoon nog steeds niet gebruikt.
Pre-conditie	Status is Jealous
Testprocedure	1. Meldt via de website dat de eigenaar niet aanwezig is bij de telefoon 2. Gebruik de telefoon niet
Post-conditie	Status is Jealous
Verwacht resultaat	De applicatie wacht weer tot dat de eigenaar de telefoon weer gebruikt

Testscenario V4.13

Testcase	Omschrijving
Testcase ID	V4.14
Omschrijving	De eigenaar heeft via de website aangegeven dat hij niet aanwezig is bij zijn telefoon. De applicatie wacht dan tot dat de eigenaar weer gebruik maakt van zijn telefoon.
Doelstelling	De eigenaar gaat de telefoon weer gebruiken
Pre-conditie	Status is Jealous
Testprocedure	1. Meldt via de website dat de eigenaar niet aanwezig is bij de telefoon 2. Gebruik de telefoon weer
Post-conditie	Status is Happy
Verwacht resultaat	De applicatie wacht weer tot dat de eigenaar 15 minuten afwezig is

Testscenario V4.14

6.4 Testrapport

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V4.1		
PRE	Status is Happy		
1	Laat de telefoon liggen	√	
2	Reageer na 15 minuten op de herkenningsacties	√	
POST	Status is Happy / Wondering	√	

Test V4.1

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V4.2		
PRE	Status is Happy		
1	Haal de telefoon uit stand-by stand (knop bovenop en teken het patroon)	√	
2	Gebruik de telefoon gedurende een paar secondes	√	
3	Zet de telefoon in stand-by stand	√	
POST	Status is Happy	√	

Test V4.2

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V4.3		
PRE	De status is Happy		
1	Gebruik de telefoon even	√	
2	Ga naar de webinterface	√	
3	Geef aan dat de eigenaar niet in de buurt is van de telefoon	√	
4	Wacht 15 minuten	√	
POST	De status is Jealous	√	

Test V4.3

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V4.4		
PRE	Status is Wondering		
1	Gebruik de telefoon	√	
2	Zet de telefoon in stand-by stand	√	
3	Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken	√	
4	De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet	√	
5	De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer niet	√	
POST	Status is Disappointed	√	

Test V4.4

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V4.5		
PRE	Status is Wondering		
1	Gebruik de telefoon	√	
2	Zet de telefoon in stand-by stand	√	
3	Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken	√	
4	De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet	√	
5	De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer	√	
POST	Status is Disappointed	√	

Test V4.5

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V4.6		
PRE	Status is Wondering		
1	Gebruik de telefoon	√	
2	Zet de telefoon in stand-by stand	√	
3	Wacht 15 minuten totdat de telefoon	√	
4	De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet	√	
5	Ga naar de website en meldt dat de eigenaar niet in de buurt is van de telefoon	√	
POST	Status in Happy	√	

Test V4.6

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V4.7		
PRE	Status is Disappointed		
1	Gebruik de telefoon	√	
2	Zet de telefoon in stand-by stand	√	
3	Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken	√	
4	De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet	√	
5	De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer niet	√	
6	De applicatie belt op een ander nummer, reageer niet	√	
7	De applicatie stuurt per e-mail de locatie van de telefoon, reageer niet	√	
8	De applicatie gaat andere personen bellen, laat ze reageren als niet wetend waar de eigenaar is	√	
POST	Status is Sad	√	

Test V4.7

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V4.8		
PRE	Status is Disappointed		
1	Gebruik de telefoon	√	
2	Zet de telefoon in stand-by stand	√	
3	Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken	√	
4	De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet	√	
5	De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer niet	√	
6	De applicatie belt op een ander nummer, reageer niet	√	
7	De applicatie stuurt per e-mail de locatie van de telefoon, reageer door de telefoon te gebruiken	√	
POST	Status is Happy	√	

Test V4.8

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V4.9		
PRE	Status is Disappointed		
1	Gebruik de telefoon	√	
2	Zet de telefoon in stand-by stand	√	
3	Wacht 15 minuten totdat de telefoon de eigenaar gaat zoeken	√	
4	De applicatie laat de telefoon vibreren en laat een patroon zien, reageer niet	√	
5	De applicatie laat de telefoon een geluid afspelen en laat een patroon zien, reageer niet	√	
6	De applicatie belt op een ander nummer, reageer niet	√	
7	De applicatie stuurt per e-mail de locatie van de telefoon, reageer via de website	√	
POST	Status is Jealous	√	

Test V4.9

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V4.10		
PRE	Status is Sad		
1	Gebruik de telefoon	√	
2	Laat de applicatie alle zoek-acties uitvoeren en reageer niet	√	
3	Na 60 minuten voert de applicatie opnieuw vind-acties uit, reageer niet	√	
POST	Status is Sad	√	

Test V4.10

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V4.11		
PRE	Status is Sad		
1	Gebruik de telefoon	√	
2	Laat de applicatie alle zoek-acties uitvoeren en reageer niet	√	
3	Wacht 10 minuten nadat de telefoon geen acties meer uitvoert	√	
4	Gebruik de telefoon	√	
POST	Status is Happy	√	

Test V4.11

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V4.12		
PRE	Status is Sad		
1	Gebruik de telefoon	√	
2	Laat de applicatie alle zoek-acties uitvoeren en reageer niet	√	
3	Wacht 10 minuten nadat de telefoon geen acties meer uitvoert	√	
4	Gebruik de telefoon	√	
POST	Status is Jealous	√	

Test V4.12

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V4.13		
PRE	Status is Jealous		
1	Meldt via de website dat de eigenaar niet aanwezig is bij de telefoon	√	
2	Gebruik de telefoon niet	√	
POST	Status is Jealous	√	

Test V4.13

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V4.14		
PRE	Status is Jealous		
1	Meldt via de website dat de eigenaar niet aanwezig is bij de telefoon	√	
2	Gebruik de telefoon weer	√	
POST	Status is Happy	√	

Test V4.14

6.5 Issues

– *De webinterface geeft niet direct dezelfde toestand als de toestand in de applicatie.*

Probleem moet worden opgelost in uitbreiding van de applicatie. Het probleem ligt bij de synchronisatie van de toestanden. De webinterface vraagt de toestand op die bekend is in CommonSense en de applicatie stuurt per minuut informatie naar CommonSense waardoor de toestand niet up-to-date is.

7. Stress- en loadtest

De tests uit hoofdstuk zes zijn tests voor alle versies van de applicatie. De hierovolgende tests worden enkel uitgevoerd op de eindversie van de *Find My Owner* applicatie.

7.1 Stress- en Loadtest

Bij een stress- en loadtest wordt onder andere gekeken wat gebeurt als er meerdere personen tegelijkertijd de applicatie gebruiken. In het geval van de Find My Owner applicatie is deze test niet van toepassing. De applicatie wordt los per telefoon geïnstalleerd en heeft geen samenwerking met de andere gebruikers.

Hierbij kan wel gelden dat er meerdere gebruikers tegelijkertijd actie vragen van het ASK platform omdat er bijvoorbeeld een aantal vindacties tegelijkertijd worden uitgevoerd. De test of het ASK Platform deze load aankan valt buiten de scope van het project.

Wel kan er getest worden wat gebeurt als de gebruiker veel vraagt van de applicatie. Bijvoorbeeld het regelmatig in- en uitschakelen van de service en instellingen wijzigen.

7.2.1 Testcase

Testcase	**Omschrijving**
Testcase ID	V.SL1
Doelstelling	Het testen of de applicatie binnen korte tijd veel acties op de juiste manier kan verwerken.
Testprocedure	Gedurende een kwartier worden er meerdere acties afwisselend uitgevoerd op de applicatie. Denk hierbij aan: • In- en uitsckalen van service • Last Actions lijst openen • Google maps openen • Preferences openen • Verwerking binnen ASK • Wachten op een herkenningsactie en het patroon wel openen maar niet intypen
Verwacht resultaat	De applicatie reageert naar behoren

Testscenario V.SL1

7.2.2 Testrapport

Testcase	**Omschrijving**	**Resultaat**	**Opmerking**
Test ID	V.SL1	Voldoende	*Zie beschrijving van test hieronder*

Test V.SL1

De stress- en loadtest is direct met de gebruikerstest uitgevoerd. Een aantal gebruikers zijn tegelijk gestart met het testen van de applicatie. Hierdoor werden er in ASK tegelijkertijd meerdere nodes aangemaakt. De toevoeging hiervan verliep goed[2].

Verder wordt de applicatie maar door één gebruiker tegelijkertijd gebruikt. Hier is met één applicatie op getest. Deze test heeft maar geen problemen opgeleverd.

[2] *In de gebruikerstest is meer te lezen over het toevoegen van de nodes. Hier traden wel problemen mee op maar dit had niet te maken met het tegelijkertijd meerdere nodes toevoegen.*

8. Reliability test

Onderstaande lijst is de lijst met eisen die gesteld is aan de betrouwbaarheid van de applicatie.

Eis	Klassificatie
De applicatie moet op geen enkel moment andere applicaties verstoren.	Must
De batterij van de telefoon moet minstens 12 uur meegaan. Hierbij wordt er vanuit gegaan dat enkel het Sense platform en de applicatie draaien (oftewel geen intensief gebruik van andere applicaties die het batterijniveau verlagen. De batterijduur geldt voor een nieuwe telefoon	Must

Tabel 9: Selectie van de betrouwbaarheidseisen (R = reliability / P = Performance)

In bovenstaande lijst zijn de functionele betrouwbaarheidseisen weggelaten. Deze zijn in het originele pakket van eisen onderverdeeld onder reliability maar na herzien van de lijst paste deze beter bij de functionele eisen. Het gaat hier over eisen wat betreft de tijd waar binnen de applicatie de eigenaar moet gaan zoeken. Deze eisen worden namelijk ook al getest bij de testscenario's per opgeleverde versie.

Daarnaast zijn er aan de lijst één performance eis toegevoegd. Deze zijn namelijk van toepassing op de reliability test.

8.1 Testscenario's

Testcase	Omschrijving
Testcase ID	V.R1
Doelstelling	De applicatie moet op geen enkel moment andere applicaties verstoren
Testprocedure	Gedurende langere tijd de smartphone gebruiken voor verschillende toepassingen. Hierbij moet de *Find My Owner* applicatie op de achtergrond draaien.
Verwacht resultaat	Er is niets van te merken dat de Find My Owner applicatie op de achtergrond draait

Testscenario V.R1

Testcase	Omschrijving
Testcase ID	V.R2
Doelstelling	De batterij van de telefoon moet minstens 12 uur meegaan. Hierbij wordt er vanuit gegaan dat enkel het Sense platform en de applicatie draaien (oftewel geen intensief gebruik van andere applicaties die het batterijniveau verlagen. De batterijduur geldt voor een nieuwe telefoon
Testprocedure	De telefoon gedurende 12 uur weinig gebruiken en af en toe controleren op de batterijduur Enkele randvoorwaarden aan de test: • De Sense applicatie moet draaien met de sensoren ingeschakeld • De Find My Owner applicatie moet draaien • De applicatie moet 6 keer op zoek gaan naar de eigenaar • De andere applicaties worden amper gebruikt • De internet verbinding is via 3G en anders WiFi
Verwacht resultaat	De batterij gaat 12 uur mee als de Sense en Find My Owner applicatie draaien

Testscenario V.R2

8.2 Testrapport

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V.R1	√	Van de Find My Owner applicatie is niets te merken als deze draait. Deze voert namelijk geen acties uit op het moment dat de eigenaar de telefoon gebruikt.

Test V.R1

Testcase	Omschrijving	Resultaat	Opmerking
Test ID	V.R1	X	*zie issues

Test V.R2

8.3 Issues

- De *Find My Owner* applicatie is uiteindelijk niet uitgebreid getest op de laatst beschikbare Android telefoon. Momenteel is de HTC Desire S een van de nieuwste telefoons met daarbij ook een langere batterijduur. De telefoons, de HTC Desire en de HTC Legend, waar de applicatie op ontwikkeld is zullen de twaalf uur batterijduur met grote kans niet halen. De batterijduur is namelijk afhankelijk van de netwerkverbinding (3G of WiFi) en hoeveel informatie de applicatie verstuurd. De batterijen van de testtelefoons gaan beide bij normaal gebruik hooguit vijftien uur mee. Het gebruik van de applicaties[3] vermindert deze duur enkel.

  De HTC Desire S en andere nieuwe toestellen halen de twaalf uur hoogstwaarschijnlijk wel. Een batterij van deze telefoons gaat bij weinig gebruikt als ongeveer 36 uur mee.

  Om de applicatie te laten voldoen aan de minimale 12 uur moeten er ook nog een aantal aanpassingen gedaan worden in de applicatie. Waaronder de sample time voor ophalen versturen van gegevens naar Sense.

3 *Sense platform en Find My Owner*

9. Gebruikerstest

Om te valideren of de applicatie ook bruikbaar is voor verschillende soorten gebruikers is een gebruikerstest opgesteld. Deze test is afgenomen bij drie verschillende groepen:

- Medewerkers van ASK-CS
  Deze medewerkers zijn op de hoogte van de werking van het ASK platform
- Medewerkers van Sense-OS
  Deze medewerkers zijn op de hoogte van de werking van het Sense platform
- Vrienden en familie
  Deze gebruikers zijn niet bekend met het Sense platform noch het ASK platform. Wel zijn deze gebruikers in het bezit van een Android smartphone.

De test is opgezet als een enquete. Gebruikers worden met enkele uitleg en vragen door de applicatie geleid. Aan de hand van deze stappen wordt er gekeken of er aan de verwachting van de applicatie wordt voldaan. Dat wil zeggen, of voor de ontwikkelaar alle elementen goed werken en voor de gebruiker of de applicatie is wat de gebruiker verwacht.

9.1 De enquête

De test is opgezet in het Engels vanwege Engelse medewerkers van ASK-CS. In de test zijn indrukken en resultaten van acties gevraagd.

Hello, *For my graduation internship I'm developing the Find My Owner app. An application which is searching for it's owner when the owner is not using his phone.* *The last weeks I have tested the application by myself and now I want to know what you think of it and which problems you find by using the application.* *The test is set up in different steps of using the application. Do not use the application before the test. Just follow the steps.* *You can find the app in the market by the name "Find My Owner". For the app you also need the Sense platform app activated. Use for the test the same e-mail address for the Find My Owner app as for the Sense app.* *Please be honest! Post every error you encounter while you're testing.* *Thanks in advance, good luck!*
For this test you need to install two Android applications. *First you need to install the Find My Owner app from the Android market. You can find it under the keywords "Find my owner".* *After that you need to install the latest version(23-05-2011) of the Sense platform . You can also find this one in the market under the keywords "Sense platform".* *First start with using the Sense app. You need to register with your e-mail address and a password.* *More information about the Sense app you can find in the app or on www.sense-os.nl / common.sense-os.nl* *When the applications are installed. You can start this test.*
1. Start the application. but do not start the service (checkbox). Tell me, what is your first impression?

2. What parts are directly unclear?

3. When you start, you got detailed information about the app. Did you read it? If you did, was the story clear?

   If you want to read it again, you can find it at the menu (menu -> FAQ)

4. Enable the application (by clicking on the checkbox) It shows some setting dialogs. Are the dialogs clear? *

   ◉ Yes

   ○ No, because _____

5. Go to the preference screen (by clicking on "Preferences" or "Menu -> Preferences". Is the information about the preferences clear?

   ○ Yes

   ○ No, because _____

From now on we start testing the find actions of the application.

First some information about the emotional states of the smartphone.

Happy: When you use the phone regularly, the state of the phone stays happy. It's the idle state of the phone.

Wondering: When the app thinks you're not around, he'll try to recognize you near of him.

Disappointed: The app didn't found you around and starts to find you on other places. He will send you an e-mail with his location and after that he tries to call you on another phone number (like work, home). When you don't respond on these messages, he will contact others by e-mail, phone or sms.

Sad: The app tried to find you for thirty minutes, but couldn't find you. He will wait for an hour for you and after that, he tries again to find you.

Jealous: You can use an website to tell your phone you're not around. With this action, the phone don't tries to find you, but wait for you till you use the phone again.

6. Put the screen off and wait sixteen minutes. What happened?
   You can put the screen on in between because the state becomes Happy when you unlock the pattern

   *When nothing happens for twenty minutes. You can try to put the application on and off by clicking on the checkbox. After that you can put the screen off again. *

   ○ The state has become Wondering, I had to unlock a pattern

   ○ The state was still Happy, nothing happened

   ○ Other.. _____

7. Put the screen off again and wait twenty-five minutes.
   Don't unlock the pattern until twenty-eight minutes are over.

   What happened? *

   ○ The state has become Disappointed, I received an e-mail from ASK/HOST

- The state has become Disappointed but I didn't received any e-mail
- The state is still Happy

8. Put off the screen and wait thirty forty-five minutes. What happened?
   You can do other things in between *
   - The last state is Sad, I've got an e-mail, I've been called on another phonenumber and my colleagues are called.
   - The last state is Disappointed. I've got an e-mail, I've been called on another phonenumber and my colleagues are called. The app is still trying to find me
   - The state is still Happy

With the website it's possible to make the phone Jealous. Because, you can say you're somewhere else and the phone will stop trying to find you.

You can reach the website with the following link:
http://tiny.cc/FindMyOwner

You can login with your Sense username and password

9. Open the website and login. What is your first impression?
10. Did you find the information button? *
    - Yes, the information was useful
    - Yes, but the information was not useful, because
    - No
11. Click on show location. Is your location the same as the location which is shown on your phone? *
    - Yes
    - No, because
12. Click on Show Current State of emotion. Can you see your emotional state? *
    - Yes
    - No, I got the following information:
13. Click on I am somewhere else. Did the state change to Jealous? *
    - Yes
    - No, when I click on the button I got an error:
14. What do you think of the application after using it for a while?
15. This application is a proof-of-concept. The application will be used in other applications and will be improved.

    Other applications will be applications like:
    DEAL MyASK - Package delivery
    Heart Attack - Recognizing a heart attack
    Epilepsy - Recognizing a epileptic seizure

    After you know this information, do you think the application is useful? What would you like to see different?
16. After all, do you have questions or comments you can't post anywhere else?
17.
18. For the test it is useful to know which android phone and version you used and of course who you are.
    Name

Android phone	
Android version	

9.2 Resultaten over de werking van de applicatie

Uit de gebruikerstest zijn totaal verschillende resultaten gekomen. Op de ene smartphone werkte de applicatie direct goed. Op de andere smartphone ontstonden problemen. Vooraf is rekening gehouden met meerdere versies van het Android O.S. De applicatie heeft als minimale eis Android versie 2.1. Bij de test is gebruik gemaakt van versie 2.1, 2.2 en 2.3.

Gebruiker wordt niet altijd goed aangemaakt in ASK

Op het moment dat de gebruiker gegevens invoert gaan deze gegevens naar ASK. Op sommige momenten was de verbinding niet goed genoeg en ontstond er een probleem met het aanmaken van de gebruiker in ASK. Het gevolg van deze fout was dat de contacten van de eigenaar en zijn overige contactinformatie niet aan de gebruiker in ASK werd gekoppeld.

Contacten en contactinformatie wordt niet altijd goed aangemaakt in ASK

Door een fout in de code werden de contactgegevens niet goed gekoppeld aan de node in ASK en aan de groepen voor contacten.

Bij Android versie 2.3 werd de data niet goed opgeslagen en verzonden naar ASK. Hierdoor werden er wel netjes nodes aangemaakt voor de gegevens maar werd er lege data aan de nodes bij de resources ingevuld.

Aan de kant van de gebruikers lag ook een fout waardoor er lege data ontstond. De gebruiker ging er vanuit dat de contactgegevens nog niet direct ingevuld hoefde te worden en daardoor kreeg de gebruiker het voor elkaar de dialogen weg te klikken. Deze dialogen komen terug als er nog geen data is ingevuld maar vaker werd deze opnieuw weggeklikt.

Dialogen verdwijnen bij kantelen van scherm

Bij het kantelen van het scherm verdwenen de dialogen. Hierdoor kon er verder geen informatie worden ingevuld maar werd de actie na sluiten wel uitgevoerd, namelijk het verzenden van (niet ingevulde) data naar ASK.

Bij eerste keer actief niet altijd direct opzoek naar eigenaar

Na het instellen van de gegevens voor ASK en de vind-acties, wordt er niet altijd gereageerd op de “Screen off action”. Hierdoor wordt er geen alarm gestart voor het herkennen en vinden van de eigenaar.

Uitval van Nexus One twee maal tijdens het testen van de applicatie

Tijdens het testen van de applicatie is een Nexus One toestel met versie 2.2 van Android tweemaal uitgevallen. De reden hiervoor is onduidelijk

Bij het aanklikken van location “No location available”

Door te klikken op de knop “Show location” wordt eerst de locatie veranderd naar “No location available”, na nogmaals klikken wordt Google maps wel gestart en de juiste locatie weergegeven.

Toestand in de webinterface is niet direct gelijk aan toestand in de applicatie
Tussen de applicatie en de de webinterface (en CommonSense) zit een verschil in toestand. Na verloop van tijd is deze toestand wel gelijk maar niet direct. Probleem ontstaat door de synchronisatie van toestand van de applicatie naar CommonSense en de verandering naar Jealous naar de applicatie terug.

Hierdoor kan ook tussendoor de applicatie naar een volgende toestand veranderen en deze naar CommonSense sturen. In de webinterface staat dan bijvoorbeeld “Disappointed” als status.

9.3 Resultaten over de gebruiksvriendelijkheid van de applicatie

In de enquête zijn verschillende vragen gesteld over de eerste indruk van onderdelen van de applicatie. De resultaten hierop zijn over het algemeen positief. Enkel worden sommige punten aangemerkt:

Het “FAQ” aan het begin van de applicatie is duidelijk maar misschien te lang
Als basis uitleg van de applicatie is een FAQ geschreven. Deze is op een klein scherm vrij lang om te lezen waardoor gebruikers deze sneller zullen wegklikken. Echter bevat de FAQ de belangrijkste kenmerken en uitleg van de applicatie.

Knopweergaven in de applicatie
State, Location, Last actions en Preferences op de zelfde manier weergegeven. Echter zijn dit niet allemaal altijd knoppen. De *State* regel is namelijk alleen een knop als het pattern geopend moet worden.

Toetsenbord overlapt dialoog
Tijdens het invoeren van informatie in dialogen wordt het schermtoetsenbord over het dialoog weergegeven. Helaas is dit een fout van Android en is dit niet direct in de applicatie op te lossen.

Een van de voorbeelden als antwoord op vraag één:
FAQ: a lot of text, but quite clear.
I like the 'human emotional states', it makes it all quite understandable.
Though jealous is not a very obvious one.
The little mechanical dog adds to the whole "find my owner"-feel, which is nice.
I don't really know what you mean by 'pattern', but I guess that will become clear.
Main menu: Looks nice. Not too many options, which keeps it clear.

9.4 Gebruik van de applicatie in het algemeen

Aan het eind van de applicatie is de vraag gesteld of mensen de applicatie zouden gaan gebruiken.

Hierop is over het algemeen geantwoord dat er langer gebruik gemaakt moet worden van de applicatie om te weten hoe deze in dagelijks gebruik werkt. Daarnaast is er in de testversie enkel nog gewerkt met de screen-off action. In latere versies wordt er gebruik gemaakt van meerdere sensoren.

Een van de voorbeelden van de reacties op deze vraag:
Ik begreep aanvankelijk niet helemaal dat het 'pattern' een soort unlock screen password was.Ik kan niet inzien welke contacten ik al heb toegevoegd. Dit is wel prettig als je er meer wilt toevoegen. Syncen met Google/Facebook zou natuurlijk helemaal tof zijn! Ik ben niet gemaild, maar wel gebeld door ASK. Wel vervelend dat op één drukken de status niet verandert, dat moet je weer met de hand met het pattern doen. Toen mijn status op 'jealous' stond, zag ik geen duidelijke knop om mijn status weer terug te veranderen.

9.4 Conclusies

Na de test is gebleken dat de applicatie nog niet volledig werkt volgens de eisen en wensen. Een aantal fouten waren voor het testen bekend en een aantal fouten waren bij het functioneel testen nog niet naar voren gekomen.

De gebruikerstest is zeer nuttig geweest voor de ontwikkeling van de applicatie. Gebruikers kwamen met mogelijkheden voor uitbreidingen en kwamen in situaties waar bij het testen van de applicatie geen rekening mee is gehouden.

10. Uitbreidingen en toepassingen

De applicatie “Find My Owner” is ontwikkeld als een proof-of-concept van een applicatie die opzoek gaat naar zijn eigenaar. Aan de hand van het proof-of-concept kunnen allerlei uitbreidingen en toepassingen bedacht worden.

10.1 Uitbreidingen

De uitbreidingen zijn de mogelijkheden die in de applicatie toegevoegd kunnen worden na oplevering van het proof-of-concept.

Locatiebepaling aan de hand van de agenda van de eigenaar

In de huidige applicatie wordt aan de hand van een standaardgroep personen de eigenaar gezocht. Een uitbreiding kan zijn dat bij het uitkiezen van de personen die gebeld moeten worden, de agenda van de eigenaar geraadpleegd wordt. Zo kan er op het tijdstip van contact opnemen gekeken worden waar de eigenaar volgens zijn agenda moet zijn. Zo kan de eigenaar om 10 uur 's ochtends op zijn werk zitten, dan wordt de groep collega's gebruikt. 's Avonds om 8 uur kan de eigenaar bij de hockeyvereniging zitten, dan wordt er contact opgenomen met de groep hockey.

ASK en Sense nemen het waken over als de telefoon uit staat

Op het moment dat de telefoon uitstaat, maar wel de service aanstond zou het mooi zijn als Sense en ASK samen de eigenaar proberen te herenigen met de telefoon. Voor deze optie wordt de functie al doorgestuurd naar CommonSense. Aan de hand van het functie id kan gekeken worden in welke toestand de applicatie zich bevond en met welke functie de applicatie bezig was. Op deze manier kunnen Sense en ASK gemakkelijk doorgaan op de applicatie.

De agent van de andere persoon laten bepalen of er gebeld, ge-sms't of gemaild wordt

In het proof-of-concept wordt door de applicatie bepaald of een kennis eigenaar gebeld, ge-sms't of gemaild wordt om de eigenaar te vinden. Netter zou zijn dat de personal agent in ASK van de kennis gaat bepalen welke actie er uitgevoerd wordt. De personal agent weet namelijk waar de kennis zich op het moment van contact opnemen bevindt. Als deze in vergadering is, kan een sms beter passen dan een telefoontje.

Meerdere sensoren om te herkennen of de telefoon in de buurt van de eigenaar is

In het proof-of-concept wordt er enkel gebruik gemaakt van de stand-by functie en de bewegingssensor van de telefoon, om te herkennen of de eigenaar in de buurt van de telefoon is. Voor latere toepassingen zijn deze twee opties te weinig om te herkennen. Zo kan de telefoon stil op een bureau naast de eigenaar liggen.

De telefoon moet eigenlijk zo min mogelijk op zoek gaan naar zijn eigenaar en dus dit soort situaties ook herkennen. Mogelijke sensoren die gebruikt kunnen worden zijn:

- Microfoon: Het herkennen van de stem van de eigenaar in de buurt van de telefoon
- Agenda: Aan de hand van de agenda van de eigenaar bepalen waar de eigenaar op dat moment zou moeten zijn en aan de hand van de locatie bepalen of de telefoon er zelf ook is.

Gegevens beheren via de webinterface

In plaats van de contactgegevens invoeren in de applicatie is het handig om via een website de gegevens in te kunnen voeren. De webinterface werkt veel samen met ASK, waardoor het geen probleem moet zijn om ook contactgegevens ook te beheren via de webinterface. Hier moet wel rekening worden gehouden met de synchronisatie met de applicatie (zoals NodeUUID's en vernieuwde gegevens.

10.2 Toepassingen

Pakketbezorger

In eerdere voorbeelden is het idee van de pakketbezorger al gegeven. Een pakket moet bezorgd worden bij een ontvanger en er worden meerdere pakketten op een dag bezorgd. Voor de pakketbezorger en het bedrijf is het prettig als dit beide zo efficiënt mogelijk gebeurd. Hiervoor zijn DEAL, Sense en ASK een applicatie aan het maken. Deze informeert de ontvanger over de komst van het pakketje binnen een bepaalde tijd en stippelt voor de bezorger een zo logisch mogelijke route uit.

Find My Owner kan in dit verhaal een rol spelen voor de betrouwbaarheid. Zo kan een telefoon de locatie van de ontvanger aangeven als 100 km verwijderd van de pakketbezorger maar kan de ontvanger wel degelijk thuis zijn. De applicatie moet dus eerst controleren of de eigenaar in de buurt is van de telefoon om zeker te weten dat hij geen verkeerde locatie doorgeeft.

Enkelband voor huisarrest (Elektronisch huisarrest)

Op het moment dat een gevangene thuis zijn straf mag uitzitten, krijgt deze een enkelband. Deze enkelband zorgt ervoor dat de eigenaar het huis niet kan verlaten. De enkelband kan vervangen worden door de zoekapplicatie. Op het moment dat de gevangene niet reageert op zijn "smartphone-enkelband", gaat de applicatie verdere acties ondernemen. Nu is de kans dat de gevangene zijn telefoon uitzet. Op dat moment nemen ASK en Sense het over en wordt er contact opgenomen of komt er versterking naar het huis toe.

Hartstilstand

Sense is momenteel bezig met het ontwikkelen van een onderdeel van de Sense applicatie die kan controleren of de eigenaar op de grond ligt met een hartstilstand. Dit kan voor elke persoon zijn maar ook voor personen die extra gevoelig zijn voor een hartstilstand. Bij deze applicatie is toch wel gewenst dat de eigenaar zijn telefoon altijd bij zich heeft. De controle die Find My Owner uitvoert kan ook in deze toepassing van belang zijn.

Diabetes, CPLD of chronische ziektes

De applicatie kan de patiënt herinneren aan momenten dat deze medicijnen in moet nemen of metingen moet uitvoeren. Daarnaast kan er gekeken worden naar de activiteiten die de patiënt uitvoert en hier ook op inspringen als er mogelijk teveel of te weinig activiteit is.

Verklarende woordenlijst

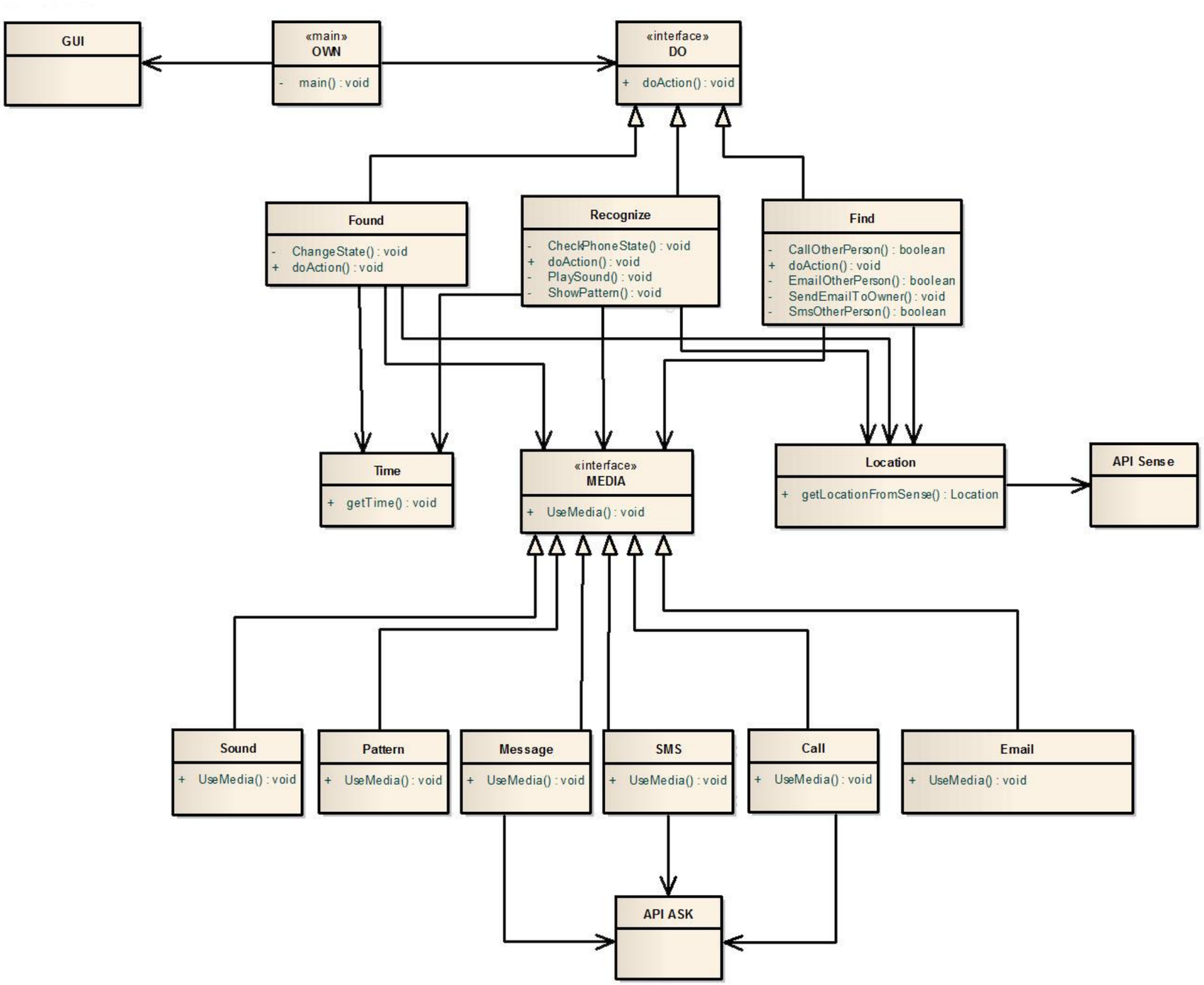
Woord, afkorting	Verklaring
ASK, ASK Community Systems	Dochterbedrijf van Almende dat het ASK Platform heeft opgezet
ASK, ASK webinterface	De basis van ASK, hier wordt alles gecombineerd. Dit deel wordt door ASK CS ingesteld voor een klant
ASK Desktop	Een applicatie waar ASK wordt gebruikt op de achtergrond
ASK Platform	Het ASK systeem met alle onderdelen (Communication Coordinator, API, Rules and scenario's, Users and groups, applicatiies en communicatiemiddelen)
IVR Menu	Interactive voice response, een telefonisch keuze menu. Ook wel bekend van klantenservice telefoongesprekken
Agent	Een virtuele persoonlijkheid binnen het ASK platform die één of meerdere verantwoordelijkheden binnen het systeem heeft.

Tabel 10: Verklarende woordenlijst

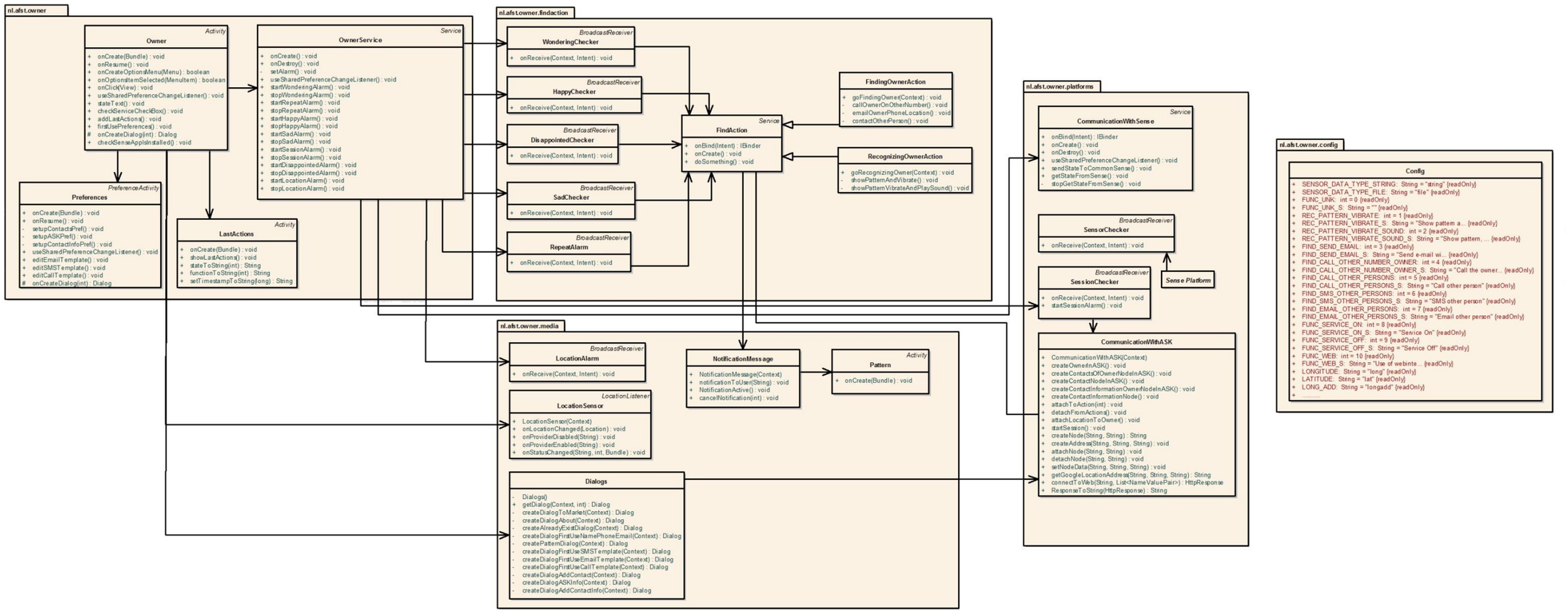
Bronvermelding

1. Rien Elling, Bas Andeweg, Jaap de Jong, (2005). *Digitaal Handboek Rapportagetechniek 4.2.* Noordhoff Uitgevers (cd-rom)
2. The Eclipse Foundation, (-2011). *OpenUP.* Via http://epf.eclipse.org/wikis/openup/
3. Clintrialaudit.com, (-2011). *V Model. Via http://www.clintrialaudit.com/validation.html*
4. Nederlandse taalunie, (-2011). *Literatuurlijsten (algemeen).* Via http://taaladvies.net/taal/advies/tekst/87/#7
5. Doug Rosenberg, (-2011). *Use case modeling for acceptance testing.* Via http://tiny.cc/z8slf
6. Glenford J. Myer, (2004). The art of software testing, Second edition. Via http://tiny.cc/mjpu2

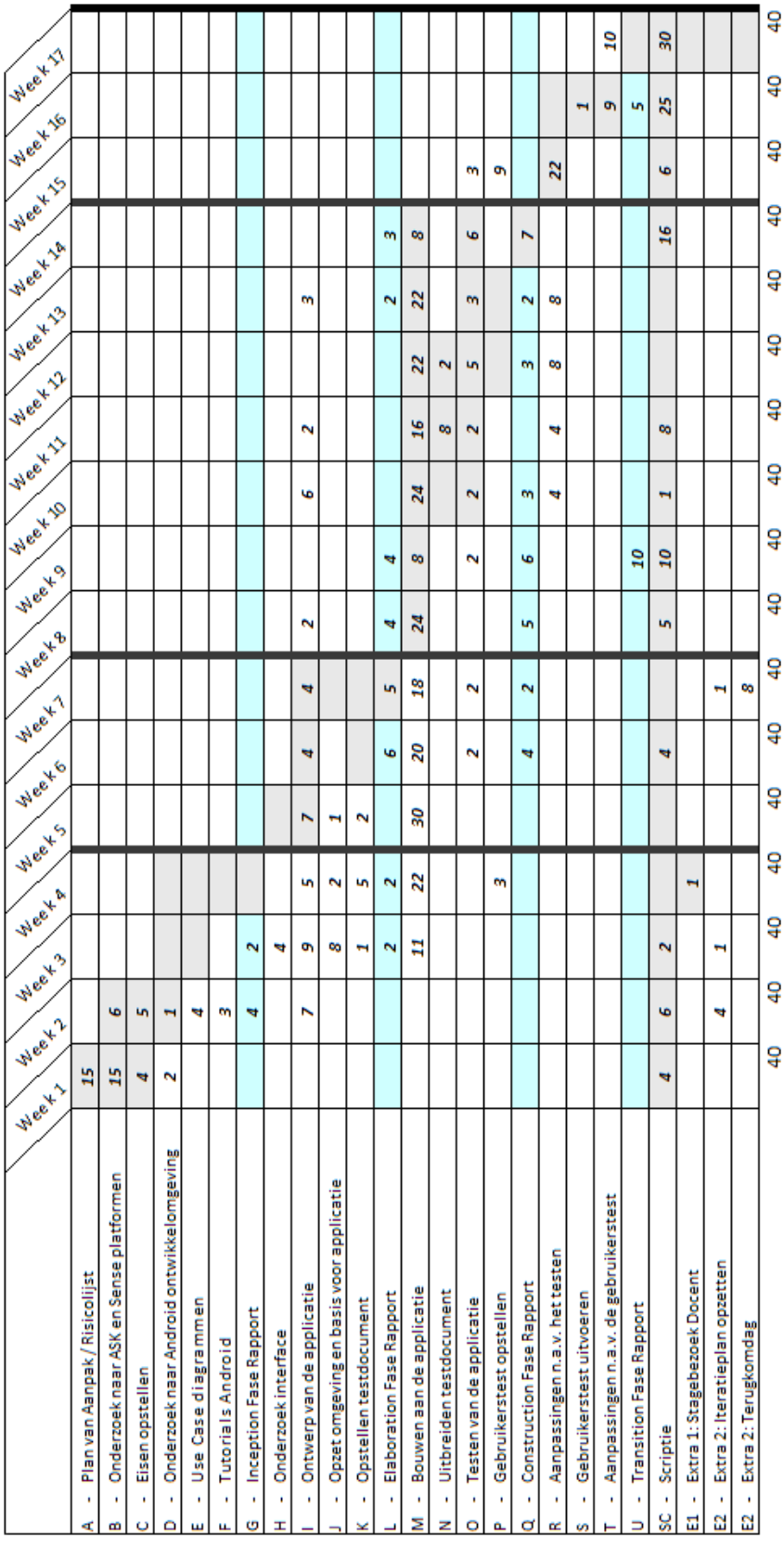
Bijlage F1: Klassendiagram basisontwerp



Bijlage F2: Klassendiagram eindversie



Urenverantwoording



	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15	Week 16	Week 17	
A - Plan van Aanpak / Risicolijst	15																	40
B - Onderzoek naar ASK en Sense platformen	15	6																40
C - Eisen opstellen	4	5																40
D - Onderzoek naar Android ontwikkelomgeving	2	1																40
E - Use Case diagrammen		4																40
F - Tutorials Android		3																40
G - Inception Fase Rapport		4	2															40
H - Onderzoek interface			4															40
I - Ontwerp van de applicatie		7	9	5	7	4	4	2		6	2		3					40
J - Opzet omgeving en basis voor applicatie			8	2	1													40
K - Opstellen testdocument			1	5	2													40
L - Elaboration Fase Rapport			2	2		6	5	4	4				2	3				40
M - Bouwen aan de applicatie			11	22	30	20	18	24	8	24	16	22	22	8				40
N - Uitbreiden testdocument											8	2						40
O - Testen van de applicatie						2	2		2	2	2	5	3	6	3			40
P - Gebruikerstest opstellen				3											9			40
Q - Construction Fase Rapport						4	2	5	6	3		3	2	7				40
R - Aanpassingen n.a.v. het testen										4	4	8	8		22			40
S - Gebruikerstest uitvoeren																1		40
T - Aanpassingen n.a.v. de gebruikerstest																9	10	40
U - Transition Fase Rapport									10							5		40
SC - Scriptie	4	6	2			4		5	10	1	8			16	6	25	30	40
E1 - Extra 1: Stagebezoek Docent				1														40
E2 - Extra 2: Iteratieplan opzetten		4	1				1											40
E2 - Extra 2: Terugkomdag							8											40

Hans Abbink:

“Het einde van top-down technologie!”

Hans Abbink onderzoekt hoe de dingen zich organiseren. In de natuur komt geen enkele soort of populatie voor die een centraal punt inricht van waaruit het alles regelt en coördineert. Dat hebben wij mensen bedacht. Dom. “Centraliseren is het creëren van een ‘single point of failure’.”

Hans Abbink zit er ontspannen bij. Ondanks dat de interviewer een half uur te laat is. Tja, vind maar eens een parkeerplekkie in hartje Rotterdam. De garage onder de Erasmusbrug, op slechts vijf minuten lopen, biedt uiteindelijk soelaas. Het stadspand waarin zijn bedrijf Almende is gevestigd oogt voornaam en ademt grandeur uit, maar mist de high tech sfeer die je zou verwachten van een bedrijf waar technologisch pionierswerk wordt verricht. Het is er, laten we zeggen, prettig rommelig. Er worden hier dingen bedacht. En dat is een waarheid als een koe. Hans Abbink is namelijk onderzoeker pur sang. Almende, het bedrijf dat hij tien jaar geleden samen met partner Peet van Tooren oprichtte, houdt zich primair bezig met de research naar zelflerende en zelforganiserende systemen. De technologische innovaties die daaruit voortkomen – met name op het gebied van agent-software – worden via Almende Investments door dochterbedrijven doorvertaald naar allerlei toepassingen. En niet zonder succes. De markt blijkt meer en meer overtuigd te raken van de toegevoegde waarde van IT-technologie die uitgaat van individuele belangen en zelforganisatie. Opmerkelijk, de meeste bedrijven werken immers nog steeds binnen een gestandaardiseerd systeem waarin alle informatie en kennis centraal wordt opgeslagen en verstrekt. Top-down dus. En dat lijkt nogal haaks te staan op de uitgangspunten van Almende.

Klopt dat?

“Als je het zo bekijkt misschien wel, maar vergis je niet, men staat wel degelijk open voor nieuwe inzichten die leiden tot verbetering. Denk maar aan het ‘nieuwe werken’. Bedrijven zien in toenemende mate in dat IT-toepassingen die de gebruiker direct ondersteunen in zijn specifieke taak – zonder tussenkomst van een manager, een administratie of een ander centraal punt in de organisatie – vele malen efficiënter zijn dan traditionele top-down systemen. Wij ontwikkelen ideeën voor software waarmee IT op een nieuwe manier kan worden ingezet. Een manier die gebaseerd is op zelforganisatie en zelfleerzaamheid. Een manier die simpelweg beter aansluit bij hoe mensen zijn.”

Hoe zijn mensen dan?

“Om die vraag te kunnen beantwoorden moet je eerst weten hoe het in de natuur geregeld is. Daarin is niets top-down georganiseerd en zie je dus nooit dat het collectief belang centraal wordt opgelegd aan de leden van de groep. Nergens. Wat je wel ziet is dat groepen zelforganiserend zijn en de individuen binnen die groep zich zelflerend hun taken eigen maken. Daarmee dienen ze zichzelf én het collectief. Kijk je met die ogen naar bedrijven dan zie je een verzameling mensen die allemaal wat anders willen dan de organisatie. Jouw belang bijvoorbeeld is niet het organisatiebelang, namelijk dat je omzet genereert, maar dat je dit interview afneemt, veilig terugrijdt naar je kantoor en het daar uitwerkt naar een aardig stuk. Daarin wil je ondersteund worden. Door nu mensen te faciliteren met IT-oplossingen die niet gestandaardiseerd, maar zelflerend, zelforganiserend en toegespitst zijn op de belangen van de gebruiker, zorg je dat ze voor elkaar krijgen wat ze willen en hun taken beter kunnen uitoefenen. En dat kan natuurlijk op verschillende niveaus.”

lees verder op pagina 6

door **Paul van Nieuwenburg**

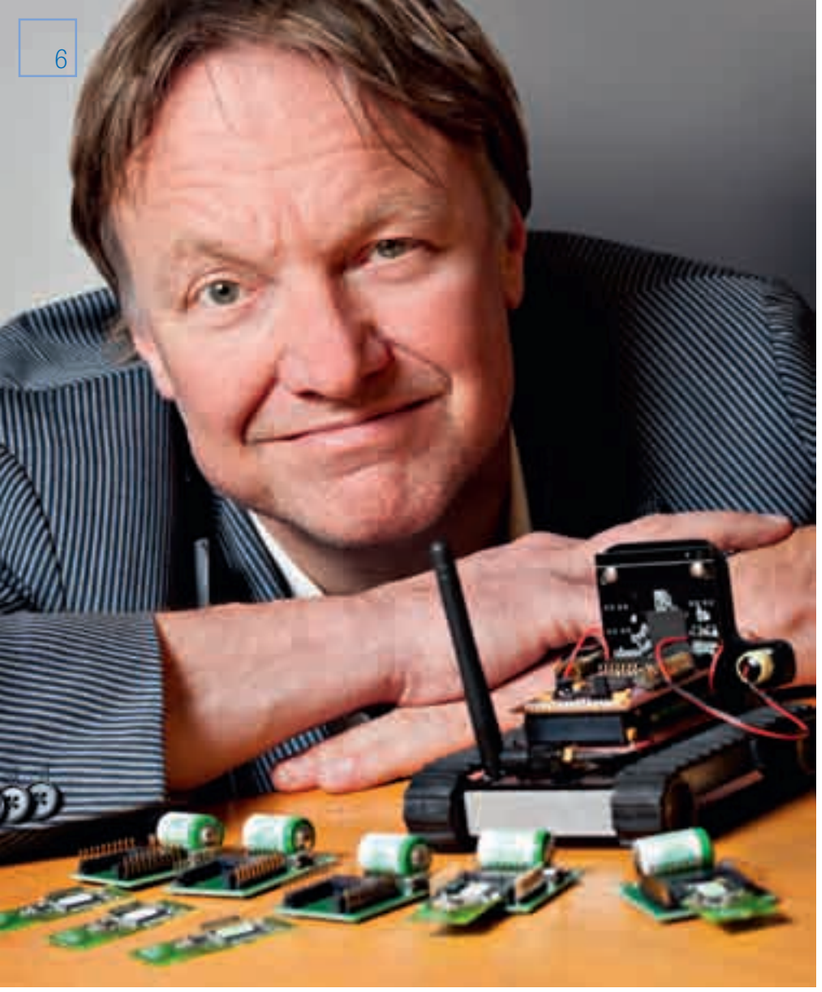
C.V.

HET BEDRIJF

Almende richt zich op het onderzoeken van ICT-toepassingen voor hybride-netwerken waarin mensen en computers samenwerken. Doel is een vorm van zelforganisatie die zich baseert op het idee dat mensen, organisaties en software niet moeten worden voorgeprogrammeerd of top-down beheerd, maar elkaar ‘spontaan’ ondersteunen. Deze aanpak resulteert in uiterst doelgericht en efficiënt handelen in verschillende sectoren.

AAN HET ROER

Binnen Almende treedt Hans Abbink op als visionair. Hij zet de lijnen uit voor onderzoek en commerciële strategie. Hij richtte Almende in 2000 op met Peet van Tooren, die nog steeds CTO is bij Almende. Hans Abbink was in 1984 mede-oprichter van AND Software (later AND International Publishers). Hij studeerde psychologie (Leiden) en econometrie (Erasmus).



‘Smartphones en internet hebben de weg vrijgemaakt voor zelf-organiserende software’

Je bedoelt op consumenten- en op bedrijfsniveau?
“Ja. Maar je kunt ook denken in termen van klein en groot. En natuurlijk in termen van autonome en complexere systemen waarin meerdere gebruikers zitten.”

Geef eens een voorbeeld.
“De adaptieve cruisecontrol, heb je misschien weleens van gehoord. Normaal stel je zo’n ding in op een bepaalde snelheid en dat is het dan. Aan die functie heeft men toegevoegd dat de auto zijn snelheid matigt zodra een sensor aangeeft dat een voorligger binnen een bepaalde afstand komt. De cruisecontrol slaat die data op en past voortaan in vergelijkbare situaties altijd zijn snelheid aan. De mate waarin hangt af van de ingegeven snelheid en de snelheid van de voorligger. Dit is een goed voorbeeld van een simpel autonoom en zelflerend systeem. Je kunt je voorstellen dat je op basis van dit principe tal van praktische toepassingen kunt bedenken. Wat dacht je van de koffieautomaat op het werk. Onhandig toch dat je elke keer moet intoetsen wat je wilt. Veel beter zou het zijn al ie je herkent en je meteen een bakkie serveert zoals jij ’m het liefst hebt. Heet, sterk, dubbel suiker, geen melk.”

En die complexere systemen met meerdere gebruikers?
“Binnen DevLab hebben we een platform ontwikkeld voor een wireless sensornetwerk, MyriaNed (WSN).”

Eh... DevLab?
“DevLab, ja, dat is een initiatief binnen FHI waarin een aantal kleinere technologische bedrijven – twaalf dacht ik – met universiteiten en technische hogescholen wereldwijd samenwerken op het gebied van research en innovatie. Doel is kennisuitwisseling en het creëren van midden en lange termijnbusinesskansen. Was voor mij reden om lid te worden van FHI.”

Oh?
“Ja, ik liep ooit eens Siebren de Vries van Chess tegen het lijf. In China. Hij is één van de drijvende krachten achter DevLab. Ik was meteen verkocht.”

Ik heb ’m ooit eens gesproken, inspirerende man. Kan me er dus wat bij voorstellen. Maar goed, je had het over MyriaNed.
“Klopt, MyriaNed maakt gebruik van een disseminatieprotocol dat Gossip heet. Nou weet jij als man van de wereld natuurlijk wat gossip betekent. Juist: roddelen. En dat is feitelijk wat dit protocol doet. Het verspreidt informatie door een netwerk van nodes. Net zoals in de natuur, zonder centrale, zonder hiërarchie, zonder namen. Alle nodes voorzien elkaar van informatie en kunnen al naar gelang noodzaak of wenselijkheid deel blijven uitmaken van het netwerk of het verlaten. Iedere node in het netwerk weet alles. Je hoeft dus maar één willekeurige op je computer aan te sluiten en voila, je hebt alle data. Op basis van dit netwerk, dat zich dus totaal anders gedraagt dan de traditionele netwerken, kunnen allerlei toepassingen bedacht worden die aansluiten op een andere – natuurlijkere en efficiëntere – manier van werken. Gossip, dat overigens niet door ons is bedacht, past perfect bij onze visie op zelforganiserende systemen. We hebben het wel aangepast en verbeterd voor toepassing in het MyriaNed WSN.”

Kun je wat van die toepassingen noemen?
“Op dit moment hebben we een project met WSN lopen in de ouderenzorg. Dat betreft indoor T&T en het monitoren van activiteiten. Een andere toepassing is een systeem dat de lokale condities in kassen meet en aangeeft wanneer en waar bepaalde omstandigheden aangepast moeten worden. Daarnaast hebben we voor de Koninklijke Nederlandse Reddingsmaatschappij agent-software ontwikkeld die in de gaten houdt of er te allen tijde acht mensen – het bemanningsaantal van een reddingsboot – in de buurt en stand-by zijn. Zodra iemand zich afmeldt verzoekt het systeem een andere persoon om paraat te staan. In geval van alarm roept het systeem de juiste personen op om uit te rukken. Software-agents kunnen een bepaalde actie uitvoeren of voor een bepaald belang opkomen, bijvoorbeeld dat van hun ‘baasje’, door onderling te communiceren en te onderhandelen. We hebben het hier dus over zelflerende software, veelal als App geïnstalleerd in smartphones, die zich aanpast aan veranderende situaties en zelf actie onderneemt. Je kunt je voorstellen dat zonder tussenkomst van een centrale de communicatie en daarmee ook de handelingen sneller en efficiënter verlopen. Om nog maar te zwijgen over de faalfactor die hierdoor aanzienlijk verkleind wordt.”

En dat gaat gewoon via een Appje, hup de smartphone in?
“Zo simpel kan het inderdaad zijn. Smartphones en het internet hebben de weg vrijgemaakt voor zelforganiserende software. Wij zien IT in die hoedanigheid als het brein van de maatschappij.”

Het brein van de maatschappij... gaat dat niet wat ver?
“Jij denkt aan science fictionboekjes en aan robots die willoze slaven van ons willen maken, he? Wat ik bedoel is dat IT het ons mogelijk maakt snel en frequent te communiceren. Dat IT ons wereldbeeld verscherpt en aanvult. En dat IT ons helpt bij samenwerking en coördinatie. Maar nu we het toch over het brein hebben, wist je dat er onderzoek is gedaan naar de kleine hersenen. En dat we op dit moment al zover zijn dat zelfs een klein stukje tissue, dat door middel van elektroden aangesloten wordt op de besturing van een helikopter, al kan leren hoe het werkt? Geef de wetenschap nog tien jaar...”

Tjeempie, over science fiction gesproken... Even terug naar zelforganisatie, je hebt daar een duidelijke visie op. Betrek je die uitsluitend op technologie?
“Binnen Almende en Almende Investments wel. In de Stichting Zelforganisatie praten we ook met partijen buiten de technologie over dat thema. Om in een bredere maatschappelijke context gedachten uit te wisselen en van elkaar te leren.”

Over maatschappelijke context gesproken, hoe kijk jij tegen Nederland als technologische natie aan?
“Ik ben eigenlijk erg optimistisch. Ik weet het, er wordt zo hier en daar met angst en beven naar de nieuwe economieën gekeken. En sommigen maken zich grote zorgen over de uittocht van talent. Ik vind dat we niet zo klein moeten denken. Er komt ook talent naar Nederland. Veel talent. En neem China, India en Brazilië, ik vind het prima dat die landen meedoen. Dat biedt ons allerlei kansen op het gebied van samenwerking. Gooi de grenzen maar open, dat is alleen maar goed.”

Zie je in deze ontwikkelingen ook een rol weggelegd voor FHI?
“Jazeker. Ik heb het natuurlijk al over DevLab gehad. De ontwikkelingen die daar plaatsvinden bevallen me uitstekend. Samenwerken, het delen van kennis en inzichten; naarmate de jaren vorderen zie ik het vertrouwen enorm toenemen. Er zit heel veel waarde in dit initiatief. FHI als overkoepelende organisatie bestaat natuurlijk uit een prachtige verzameling bedrijven die veel aan elkaar kunnen hebben. Op verschillende terreinen. Gebeurt natuurlijk al. Maar zou volgens mij nog soepeler verlopen als FHI zo ingericht was dat er zelforganisatie kon optreden. Ach ja, wishful thinking...” ¶