955
1997
003

Een ITIL-conform helpdesk-ondersteunend systeem

Een scriptie ter afronding van een studie
Technische Cognitiewetenschappen
aan de
Rijksuniversiteit van Groningen

Richard Doornbos
1997

Scriptiebegeleiders:

Reinier van Oosten
Hans van Ditmarsch

968

INHOUDSOPGAVE

1. INLEIDING **1-1**

1.1 PROBLEEMSTELLING 1-2
1.2 AANPAK 1-2

2. THEORIE **2-4**

2.1 ITIL 2-4
2.1.1 Helpdesk *2-6*
2.1.1.1 Communicatie en promotie 2-7
2.1.1.2 Incidentbeheer 2-7
2.1.1.3 Rapportage aan management 2-8
2.1.2 Probleembeheer *2-8*
2.1.2.1 Probleemidentificatie en registratie 2-9
2.1.2.2 Classificatie 2-10
2.1.2.3 Allocatie mensen en middelen 2-10
2.1.2.4 Onderzoek en diagnose 2-10
2.1.2.5 Foutbeheer 2-11
2.1.2.6 Informatievoorziening 2-11
2.2 DE OBJECT ORIENTED SOFTWARE ENGINEERING (OOSE) METHODIEK 2-11
2.2.1 Systeemontwikkeling *2-12*
2.2.2 Het analysis process *2-13*
2.2.2.1 Requirements model 2-13
2.2.2.1.1 Use case model 2-14
2.2.2.1.2 Interface descriptions 2-15
2.2.2.1.3 Problem domain object model 2-15
2.2.2.2 Analysis Model 2-15
2.2.2.2.1 Entity objects 2-15
2.2.2.2.2 Interface objects 2-16
2.2.2.2.3 Control objects 2-16
2.2.2.2.4 Een voorbeeld van het analyseren van een use case 2-16
2.2.3 Construction process *2-16*
2.2.3.1 Design Model 2-16
2.2.3.2 Implementation Model 2-17
2.2.4 Test process *2-17*
2.2.4.1 Test Model 2-17
2.2.5 Gebruikte onderdelen van OOSE *2-17*

3. ITIL IS SLECHTS HET BEGIN **3-18**

3.1 PROBLEEMIDENTIFICATIE EN REGISTRATIE 3-18
3.1.1 Kenmerken van incidenten en problemen *3-19*
3.1.2 Het vergelijken van kenmerken van incidenten en problemen *3-20*

4. DE ONDERSTEUNING VAN HET CLASSIFICEREN VAN EEN INCIDENT **4-22**

4.1.1 Definities *4-23*
4.1.2 Ordenen van problemen *4-23*
4.1.2.1 Het ordenen van problemen met het theorema van Bayes 4-24
4.1.2.1.1 Theorie 4-24
4.1.2.1.2 Algoritme 4-26
4.1.2.2 Het ordenen van problemen op basis van verhoudingen 4-27
4.1.2.2.1 Theorie 4-27
4.1.2.2.2 Algoritme 4-28
4.1.3 Het ordenen van de vervolgvragen *4-28*
4.1.3.1 Het ordenen van vragen op basis van verhoudingen 4-29
4.1.3.1.1 Theorie 4-29
4.1.3.1.2 Algoritme 4-29
4.1.3.2 Het ordenen van vragen met het theorema van Bayes 4-30
4.1.3.2.1 Theorie 4-30
4.1.3.2.2 Algoritme 4-31

4.1.4 C4.5 ..... *4-32*
4.1.4.1 Het ordenen van problemen met C4.5 4-34
4.1.4.2 Het ordenen van vragen met C4.5 4-35

5. SYSTEEMONTWIKKELING 5-36

5.1 REQUIREMENTS SPECIFICATION 5-36
5.2 ANALYSIS PROCESS 5-36
5.2.1 Het requirements model *5-37*
5.2.1.1 Actoren 5-37
5.2.1.1.1 De helper 5-38
5.2.1.1.2 De probleembeheerder 5-38
5.2.1.1.3 De receptontwikkelaar 5-39
5.2.1.2 Use cases 5-39
5.2.1.2.1 Helper 5-39
5.2.1.2.1.1 Aannemen van een call 5-40
5.2.1.2.1.2 Analyse van een incident 5-40
5.2.1.2.1.3 Analyse van een incident met uitgebreide ondersteuning 5-41
5.2.1.2.1.4 Analyse van een incident zonder ondersteuning 5-41
5.2.1.2.1.5 Adviseren over het uitvoeren van een oplossing 5-42
5.2.1.2.1.6 Aanmaken van een workaround 5-42
5.2.1.2.2 Probleembeheerder 5-43
5.2.1.2.2.1 Fout invoeren 5-43
5.2.1.2.2.2 Het aanmaken van een wijziging 5-43
5.2.1.2.3 De receptontwikkelaar 5-44
5.2.1.2.3.1 Het ontwikkelen van een recept 5-44
5.2.1.2.4 Extend use cases 5-45
5.2.1.2.4.1 Zoeken 5-45
5.2.1.2.4.2 Zoeken met behulp van keywords 5-46
5.2.1.2.4.3 Zoeken met behulp van woorden uit de omschrijving 5-46
5.2.1.2.4.4 Twee manieren tegelijkertijd 5-46
5.2.1.2.4.5 Persoon zoeken 5-47
5.2.1.2.4.6 Persoon zoeken door gedeeltelijke beschrijving 5-47
5.2.1.2.4.7 Persoon zoeken uit een lijst 5-47
5.2.1.2.4.8 Toekennen van keywords 5-47
5.2.1.2.4.9 Vraag maken 5-48
5.2.1.2.4.10 Actie maken 5-48
5.2.1.3 Interface beschrijvingen 5-48
5.2.1.3.1 De interface voor het aannemen van een call 5-49
5.2.1.3.2 De interface voor de analyse van een incident 5-50
5.2.1.3.3 De interface voor het zoeken van een persoon 5-50
5.2.1.4 Problem domain 5-51
5.2.2 Analysis model *5-51*
5.2.2.1 Het aannemen van een call 5-51
5.2.2.1.1 Entity objecten 5-51
5.2.2.1.2 Interface objecten 5-52
5.2.2.1.3 Control objecten 5-52
5.2.2.2 Analyse van een incident 5-52
5.2.2.2.1 Entity objecten 5-52
5.2.2.2.2 Interface objecten 5-52
5.2.2.2.3 Control objecten 5-53
5.2.2.3 Het zoeken van een persoon 5-53
5.2.2.3.1 Entity objecten 5-53
5.2.2.3.2 Interface objecten 5-53
5.2.2.3.3 Control objecten 5-53

6. DESIGN 6-54

6.1 PROBLEM DOMAIN 6-54
6.1.1 Persoon *6-55*
6.1.1.1 Persoon associaties 6-55
6.1.1.2 Persoon attributen 6-55
6.1.2 Rol *6-55*
6.1.2.1 Rol associaties 6-56

6.1.3 Helper *6-56*
6.1.3.1 Helper associaties 6-56
6.1.4 Probleembeheerder *6-56*
6.1.4.1 Probleembeheerder associaties 6-56
6.1.5 Bedenker *6-56*
6.1.5.1 Bedenker associaties 6-56
6.1.6 Eigenaar *6-57*
6.1.6.1 Eigenaar associaties 6-57
6.1.7 Melder *6-57*
6.1.7.1 Melder associaties 6-57
6.1.8 Gebruiker *6-57*
6.1.8.1 Gebruiker associaties 6-57
6.1.8.2 Gebruiker attributen 6-57
6.1.9 Call *6-58*
6.1.9.1 Call associaties 6-58
6.1.9.2 Call attributen 6-58
6.1.10 Incident *6-59*
6.1.10.1 Incident associaties 6-59
6.1.10.2 Incident attributen 6-60
6.1.11 Probleem *6-63*
6.1.11.1 Probleem associaties 6-63
6.1.11.2 Probleem attributen 6-64
6.1.12 Fout *6-64*
6.1.12.1 Fout associaties 6-64
6.1.12.2 Fout attributen 6-65
6.1.13 Oplossing *6-65*
6.1.13.1 Oplossing associaties 6-65
6.1.13.2 Oplossing attributen 6-66
6.1.14 Wijziging *6-66*
6.1.14.1 Wijziging associaties 6-66
6.1.15 Work-around *6-67*
6.1.15.1 Work-around associaties 6-67
6.1.16 Keyword *6-67*
6.1.16.1 Een lijst 6-67
6.1.16.2 Een boom 6-67
6.1.16.3 Een netwerk 6-68
6.1.16.4 Indexeren 6-69
6.1.16.5 Keyword associaties 6-69
6.1.16.6 Keyword attributen 6-69
6.1.17 Recept *6-70*
6.1.17.1 Recept associaties 6-70
6.1.17.2 Recept attributen 6-70
6.1.18 Vragen en antwoorden *6-71*
6.1.18.1 Vragen stellen 6-71
6.1.18.2 Andere manieren om informatie te verzamelen 6-71
6.1.18.3 Vraag associaties 6-71
6.1.18.4 Vraag attributen 6-72
6.1.19 Antwoord *6-72*
6.1.19.1 Antwoord associaties 6-72
6.1.19.2 Antwoord attributen 6-73
6.1.20 Actie *6-73*
6.1.20.1 Actie associaties 6-73
6.1.20.2 Actie attributen 6-73
6.1.21 Integriteit van de associaties *6-74*
6.2 CONTROL OBJECTEN VOOR HET CLASSIFICATIE PROCES 6-74
6.2.1 Implementatie van de ordeningsmethoden *6-75*
6.2.1.1 Sorter 6-75
6.2.1.2 ProblemSorter 6-76
6.2.1.3 BayesProblemSorter 6-76
6.2.1.4 QuinlanProblemSorter 6-76
6.2.1.5 QuestionSorter 6-76

6.2.1.6 BayesQuestionSorter 6-77
6.2.1.7 QuinlanQuestionSorter 6-77
6.2.2 Andere control-objecten voor het testen van de classificatiemethoden *6-77*
6.2.2.1 SortingApplicationModel 6-77
6.2.2.2 HelpdeskDataGenerator 6-78
6.2.2.3 IncidentGenerator 6-79
6.2.2.4 ClassificationTester 6-80

7. TESTEN 7-81

7.1 INLEIDING 7-81
7.2 TESTVRAGEN 7-81
7.3 TESTOMGEVING 7-82
7.3.1 Het verloop van een simulatie *7-83*
7.3.2 De variabelen *7-84*
7.3.3 Het meten van de classificatie-prestaties *7-85*
7.3.4 Uitleg over de grafieken *7-86*
7.4 HET TESTEN 7-86
7.4.1 Welk algoritme presteert het best? *7-86*
7.4.1.1 Testopzet 7-86
7.4.1.2 Resultaten 7-87
7.4.1.2.1 Methode van Bayes 7-87
7.4.1.2.2 Methode van Quinlan 7-87
7.4.1.2.3 De verhoudingen-methode 7-88
7.4.1.3 Conclusie 7-88
7.4.2 Ondersteuning van classificatie met behulp van vragen *7-88*
7.4.2.1 Testopzet 7-89
7.4.2.2 Resultaten 7-89
7.4.2.3 Conclusie 7-89
7.4.3 Manieren om belief-waarden te combineren *7-90*
7.4.3.1 Testopzet 7-90
7.4.3.2 Resultaten 7-91
7.4.3.3 Conclusies 7-92
7.4.4 De prestaties in de aanloopperiode en daarna *7-92*
7.4.4.1 Testopzet 7-92
7.4.4.2 Resultaten 7-93
7.4.4.2.1 De standaard testomgeving 7-93
7.4.4.2.2 De testomgeving met extra IncidentGeneratoren 7-94
7.4.4.2.3 De testomgeving met extra keywords 7-95
7.4.4.2.4 De testomgeving met extra IncidentGeneratoren en extra keywords 7-95
7.4.4.3 Conclusies 7-96
7.5 CONCLUSIES OVER DE ALGORITMEN 7-97

8. CONCLUSIES EN AANBEVELINGEN 8-98

8.1 ITIL 8-98
8.2 OOSE 8-98
8.3 ONDERSTEUNING VAN ANALYSE 8-98
8.3.1 Ordenen van problemen *8-98*
8.3.2 Ordenen van vragen *8-99*
8.4 AANBEVELINGEN 8-99

9. GEBRUIKTE LITERATUUR 9-100

9.1 BOEKEN 9-100
9.2 HANDLEIDINGEN 9-100

Voorwoord

Dit is het verslag van het afstudeeronderzoek dat ik heb uitgevoerd voor mijn studie Technische Cognitiewetenschappen aan de Rijksuniversiteit van Groningen.

Het onderzoek sluit aan bij de trends binnen het bedrijfsleven om IT-afdelingen volgens de ITIL-methodiek op te zetten en om steeds meer object georiënteerde technieken te gebruiken.

ITIL biedt een goede structuur om een helpdeskondersteunend systeem op te baseren. OOSE, de gebruikte object georiënteerde methodiek voor software engineering, blijkt een krachtig gereedschap te zijn om software te ontwikkelen

De onderzochte methoden om helpdeskpersoneel te ondersteunen werken niet allemaal even goed. De methode die tijdens dit onderzoek is ontwikkeld om problemen te ordenen op waarschijnlijkheid aan de hand van de keywords van een incident presteert echter zeer goed en zou zonder meer gebruikt kunnen worden in een commerciële helpdeskapplicatie.

Tijdens dit onderzoek werd ik begeleid door Reinier van Oosten en Hans van Ditmarsch.

De dagelijkse begeleiding kwam voor rekening van Reinier van Oosten en ik zou hem graag willen bedanken voor zijn geduld en zijn enthousiasme. Ondanks zijn drukke bezigheden vond hij altijd de tijd om mij te begeleiden.

Hans van Ditmarsch begeleidde meer van een afstand en zag er op toe dat mijn werk aan de eisen van TCW zou voldoen. Ook hij toonde veel geduld en stelde vragen die mij verder hielpen bij het onderzoek en me dwongen mijn ideeën duidelijk te formuleren..

Ook wil ik Objectivity Nederland bedanken omdat ik gebruik heb mogen maken van hun software.

Tot slot gaat mijn dank uit naar de medewerkers van de helpdesk van de N.V. Nederlandse Gasunie die mij hebben laten zien hoe er op een helpdesk gewerkt wordt.

Richard Doornbos

1. Inleiding

Steeds meer mensen gebruiken computersystemen in hun werksituatie. Het omgaan met die apparatuur en software is niet altijd even eenvoudig. Het komt dan ook regelmatig voor dat mensen niet verder kunnen met hun werk omdat ze problemen hebben met de apparatuur of met de software. Die mensen hebben ondersteuning nodig. De helpdesk biedt die ondersteuning. Er kunnen twee soorten helpdesks onderscheiden worden.

De eerste soort helpdesk bevindt zich binnen een bedrijf waarin gebruik wordt gemaakt van automatiseringshulpmiddelen. Werknemers die een computerprobleem hebben bellen de helpdesk en worden dan te woord gestaan door een helpdeskmedewerker. Deze persoon weet welke apparatuur en software binnen het bedrijf gebruikt worden en kan daar goed mee omgaan.

De tweede soort helpdesk biedt service aan de klanten van een bedrijf. Als iemand een tekstverwerker koopt, dan heeft hij vaak ook recht op ondersteuning. De medewerkers van dit type helpdesk weten bijna alles van de produkten waarvoor ze ondersteuning bieden.

Er is in het bedrijfsleven veel aandacht voor helpdesks, en dat is niet verrassend. Mensen die hun werk niet kunnen doen omdat ze een computerprobleem hebben kosten een bedrijf veel geld. Het is daarom belangrijk om die problemen zo snel mogelijk op te lossen. Het bemannen van een helpdesk kost echter ook geld. Helpdeskmedewerkers moeten immers een uitgebreide kennis hebben van apparatuur en software. De meeste bedrijven hebben meerdere helpdeskmedewerkers nodig om alle problemen vlot te kunnen verhelpen.

Omdat helpdesks een belangrijke rol spelen binnen bedrijven die gebruik maken van automatiseringshulpmiddelen zijn er methoden ontwikkeld die aangeven wat de plaats van een helpdesk binnen een organisatie moet zijn en hoe een helpdesk georganiseerd moet worden. De helpdeskmodule van ITIL (Information Technology Infrastructure Library) wordt in Nederland door steeds meer bedrijven gebruikt om hun helpdesks te organiseren.

Met deze ITIL-module als leidraad hebben verschillende bedrijven software ontwikkeld die de helpdeskmedewerkers moet ondersteunen bij het uitvoeren van hun taken. Deze software voldoet over het algemeen goed. Vooral de administratieve kant van de helpdesk wordt goed ondersteund. De software biedt echter weinig hulp bij het analyseren van de problemen van gebruikers. De kennis die de helpdeskmedewerkers opdoen door het oplossen van problemen kan niet of slechts in beperkte mate worden vastgelegd. Eerder opgeloste problemen kunnen moeilijk

teruggevonden worden. Dit heeft veel dubbel werk tot gevolg, omdat problemen vaak meerdere keren door verschillende helpdeskmedewerkers opgelost worden.

1.1 Probleemstelling

De meeste vragen die de helpdesk bereiken zijn al eens eerder gesteld en bij die gelegenheid ook beantwoord. Alle problemen die mensen tegen komen tijdens het werken met computersystemen moeten volgens ITIL opgeslagen worden. Binnen ITIL worden de problemen **incidenten** genoemd. Overeenkomstige incidenten worden samen verbonden met wat ITIL een **probleem** noemt. Dat probleem wordt vervolgens verbonden met één of meer oplossingen.

De helpdeskmedewerker die een nieuw incident wil oplossen zou in de opgeslagen gegevens moeten kunnen zoeken naar een probleem dat verbonden is met incidenten die lijken op het incident dat hij behandelt. Als hij zo'n probleem kan vinden dan is het waarschijnlijk verbonden met een oplossing. Na het uitvoeren van die oplossing is het incident afgehandeld.

We kunnen dit zoeken naar een probleem zien als de classificatie van het incident. Dit classificatieproces wordt door de huidige software niet ondersteund. Een systeem dat dit wel kan heeft een aantal voordelen:

- incidenten kunnen sneller afgehandeld worden;
- de meeste incidenten kunnen afgehandeld worden door minder hoog opgeleide medewerkers;
- het is een aanzet tot een systeem waarmee gebruikers zichzelf kunnen helpen door het juiste probleem bij hun incident te zoeken.

Het spreekt vanzelf dat dit een besparing in de kosten van de helpdesk oplevert.

Het centrale probleem dat ik in dit onderzoek behandel is het classificeren van incidenten naar problemen. De probleemstelling van dit onderzoek is dan ook:

Is het mogelijk een systeem te ontwikkelen dat binnen een ITIL-kader ondersteuning biedt voor het classificeren van incidenten?

1.2 Aanpak

Om deze probleemstelling te beantwoorden heb ik eerst het gedeelte van ITIL bestudeerd dat over de helpdesk gaat. Daarna heb ik mij verdiept in één van de object georiënteerde methodieken om applicaties te ontwikkelen. Tijdens dit theoretische onderzoek werd duidelijk dat ITIL de beschrijving is van een manier waarop een helpdesk georganiseerd kan worden, maar dat

de structuur die ITIL biedt door de IT-afdeling en systeemontwikkelaars verder moet worden ingevuld.

De rest van het onderzoek valt uiteen in twee delen.

Ten eerste de ontwikkeling van een aanzet tot een commerciële helpdeskapplicatie. Ik heb geprobeerd zo dicht mogelijk bij de ITIL-methodiek te blijven. Tijdens het onderzoek is een aantal methoden ontwikkeld om ondersteuning te bieden bij het classificeren van incidenten. Deze methoden kunnen eenvoudig toegevoegd worden aan de applicatie. Tijdens het ontwikkelen en het programmeren van de applicatie heb ik object georiënteerde technieken gebruikt. Het resultaat is een object georiënteerde applicatie, die nauw samenwerkt met een object georiënteerde database.

Naast de applicatie heb ik dit verslag geschreven. Het verslag is verdeeld in acht hoofdstukken.

Het eerste hoofdstuk wordt gevormd door de inleiding en de probleemstelling.

In het tweede hoofdstuk komt de theorie aan de orde. Daarin worden ITIL en OOSE uitgebreid behandeld.

In het derde en vierde hoofdstuk wordt beschreven hoe de ITIL-activiteit Probleemidentificatie en registratie tijdens dit onderzoek nader is uitgewerkt en geïmplementeerd.

In het vijfde hoofdstuk staat beschreven hoe het helpdesk ondersteunende systeem werd ontwikkeld.

Het zesde hoofdstuk bevat een complete opsomming van de entiteiten die binnen het helpdesksysteem een rol spelen.

Het zevende hoofdstuk bevat een beschrijving van de wijze waarop het systeem getest is en een analyse van de resultaten.

Het laatste hoofdstuk bevat de conclusies en aanbevelingen op grond van de ervaringen die tijdens dit onderzoek zijn opgedaan.

2. Theorie

Dit hoofdstuk bestaat uit twee delen. Het eerste deel is een beschrijving van ITIL. In het tweede deel wordt de OOSE (Object Oriented Software Engineering) methode van Ivar Jacobson behandeld. Dit is één van de methodieken om object georiënteerde software te ontwikkelen.

2.1 ITIL

ITIL (Information Technology Infrastructure Library) is een reeks standaardbeschrijvingen om IT-dienstverlening binnen een organisatie te organiseren. Met IT-dienstverlening wordt bedoeld het deskundig beheren en exploiteren van de IT-infrastructuur en er zo voor te zorgen dat de werknemers zo veel mogelijk zonder problemen van de automatiseringshulpmiddelen gebruik kunnen maken. De IT-infrastructuur bestaat uit de hardware, software, handleidingen en procedures die de werknemers van een organisatie gebruiken voor het uitvoeren van hun taken. Het is de taak van de helpdesk de gebruikers van deze infrastructuur te ondersteunen.

Als een gebruiker moeilijkheden ondervindt tijdens het werken met de IT-infrastructuur dan spreekt men binnen ITIL van een **incident**.

Één van de uitgangspunten van ITIL is dat een incident veroorzaakt wordt door een **fout** in de IT-infrastructuur. De infrastructuur kan bijvoorbeeld een fout bevatten die er voor zorgt dat applicatie X onder bepaalde omstandigheden vastloopt. Deze fout zal één of meerdere incidenten tot gevolg hebben.

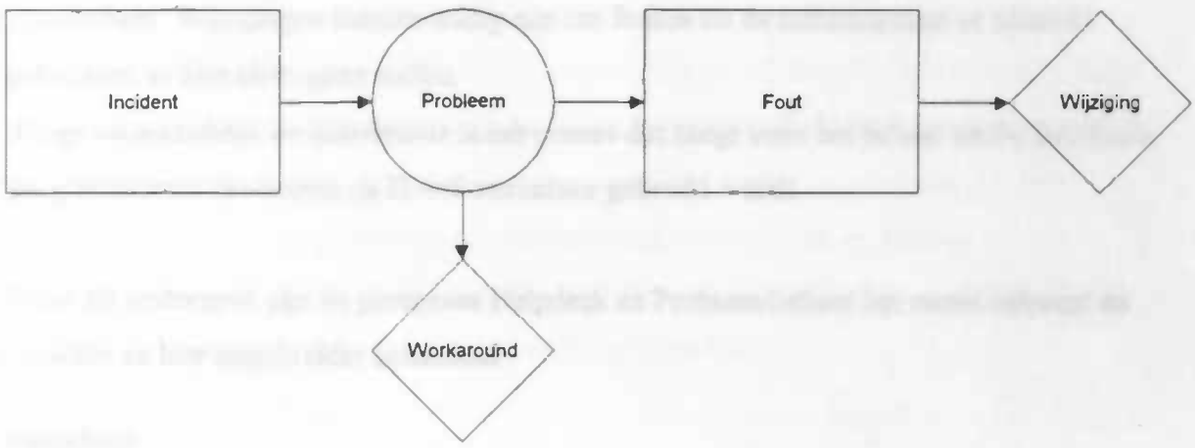
Zolang niet duidelijk is wat de oorzaak is van een incident spreken we van een **probleem**. Elk incident wordt verbonden met één probleem. Een probleem kan verbonden zijn met meerdere incidenten. Alle incidenten die gaan over het vastlopen van applicatie X worden verbonden met hetzelfde probleem.

Het oplossen van een probleem leidt in de meeste gevallen tot het vinden van de fout. Een fout kan meerdere problemen tot gevolg hebben, maar een probleem wordt altijd veroorzaakt door één fout.

Als de fout eenmaal bekend is dan kan de infrastructuur met een **wijziging** zodanig veranderd worden dat de incidenten die verbonden zijn met het probleem zich in de toekomst niet meer voordoen. De IT-infrastructuur kan bijvoorbeeld zo aangepast worden dat applicatie X niet meer vastloopt.

In sommige gevallen kan een probleem niet opgelost worden, maar wordt er wel een methode gevonden die incidenten voorkomt of herstelt. Zo'n methode word een **workaround** genoemd. Als een helpdeskmedewerker bijvoorbeeld ontdekt dat applicatie X niet vastloopt als de computer tijdens het opstarten geen verbinding maakt met het netwerk dan kan die kennis gebruikt worden om toch goed met applicatie X te kunnen werken.

ITIL eist dat alle incidenten, problemen, fouten, wijzigingen en workarounds worden opgeslagen, onder andere om rapporten te kunnen genereren.



Het doel van de automatiseringsafdeling is IT-dienstverlening te bieden van een zo hoog mogelijke kwaliteit. Dit is het geval als de dienstverlening zowel stabiel als flexibel is. De infrastructuur werkt dan zoals de gebruikers het verwachten (stabiel) en kan aangepast worden als zij dat willen (flexibel).

Binnen ITIL wordt een aantal processen beschreven die noodzakelijk zijn voor een goede IT-dienstverlening. Dit zijn:

- Configuratiebeheer
- Helpdesk
- Probleembeheer
- Wijzigingsbeheer
- Programmatuurbeheer en distributie

Configuratiebeheer brengt de componenten van de IT-infrastructuur onder controle. Met 'onder controle brengen' wordt bedoeld dat Configuratiebeheer weet welke onderdelen van de

infrastructuur zich op welke plaats in de organisatie bevinden en in welke toestand zij verkeren. Configuratiebeheer geeft de andere processen informatie over de IT-infrastructuur.

Als de IT-dienstverlening niet naar verwachting plaatsvindt dan wordt dat doorgegeven aan **Helpdesk**. Het proces Helpdesk ondersteunt de gebruikers van de IT-diensten en probeert afwijkingen in de dienstverlening zo snel mogelijk te verhelpen.

Probleembeheer is het proces dat de oorzaak van een falende dienstverlening probeert te achterhalen, door te zoeken naar de fouten die de problemen veroorzaken.

Wijzigingsbeheer zorgt ervoor dat wijzigingen aan de IT-infrastructuur gecontroleerd plaatsvinden. Dit betekent onder andere dat er overleg plaatsvindt tussen Wijzigingsbeheer en Configuratiebeheer. Wijzigingen kunnen nodig zijn om fouten uit de infrastructuur te halen of omdat gebruikers andere eisen gaan stellen.

Programmabeheer en distributie is het proces dat zorgt voor het beheer en de distributie van de programmatuur die binnen de IT-infrastructuur gebruikt wordt.

Voor dit onderzoek zijn de processen Helpdesk en Probleembeheer het meest relevant en daarom worden ze hier uitgebreider behandeld.

2.1.1 Helpdesk

De doelen van het proces Helpdesk zijn:

1. acceptatie van de IT-afdeling door de gebruikers als partner bij het uitvoeren van de bedrijfsprocessen;
2. de IT-afdeling toegankelijk maken voor klachten en opmerkingen over de automatiseringshulpmiddelen;
3. zorgen voor een continue dienstverlening door het snel afhandelen van incidenten.

Om deze doelen te realiseren worden binnen ITIL drie activiteiten van Helpdesk onderscheiden. Dit zijn:

- communicatie en promotie
- incidentbeheer
- rapportage aan het management

2.1.1.1 Communicatie en promotie

De helpdesk vormt de verbinding tussen de gebruikers en de automatiseringsafdeling. Voor de gebruikers van de IT-infrastructuur is de helpdesk het aanspreekpunt van de IT-afdeling. Als zij vragen, opmerkingen of klachten hebben over het gebruik van de automatiseringsmiddelen dan nemen zij contact op met de helpdesk.

De automatiseringsafdeling profileert zich via de helpdesk aan de gebruikers. Om een goede IT-dienstverlening te kunnen bieden is het noodzakelijk te weten welke taken de gebruikers uitvoeren en hoe ze daarbij gebruik maken van de IT-infrastructuur. Door de contacten met de gebruikers zorgt de helpdesk ervoor dat deze kennis binnen de IT-afdeling aanwezig is. De automatiseringsafdeling kan de gebruikers dan bijvoorbeeld attent maken op nieuwe mogelijkheden om bepaalde doelen te realiseren. Acceptatie van de IT-afdeling als partner bij het uitvoeren van de bedrijfsprocessen zal daardoor sneller plaatsvinden.

2.1.1.2 Incidentbeheer

Zolang alle werknemers naar tevredenheid gebruik maken van de automatiseringshulpmiddelen is de IT-dienstverlening in orde. Er kan zich echter een situatie voordoen waardoor een gebruiker niet optimaal gebruik kan maken van de faciliteiten. Dit kan het gevolg zijn van een fout in de IT-infrastructuur of een gebrek aan kennis bij de gebruiker. Zo'n afwijking van de optimale situatie wordt een incident genoemd.

Incidenten storen de werknemers bij het uitvoeren van hun taken en moeten zo snel mogelijk verholpen worden. Dit wordt incidentbeheer genoemd. De nadruk ligt hierbij op het snel opheffen van storingen zodat mensen weer verder kunnen met hun werk. Binnen incidentbeheer wordt een aantal deelactiviteiten onderscheiden.

- detectie en registratie
- classificatie en toewijzing
- diagnose en oplossing
- afsluiting

Detectie draait om het opmerken van incidenten. Incidenten kunnen in verschillende domeinen voorkomen. De helpdesk hoeft niet altijd te wachten totdat een incident door een

gebruiker wordt gemeld. In sommige domeinen (denk aan een netwerk) kunnen incidenten automatisch gedetecteerd worden. In het ideale geval kan zo'n incident opgelost worden voordat de gebruikers er iets van merken.

Als een incident opgemerkt of gemeld wordt vindt de registratie plaats. De symptomen worden genoteerd, samen met andere gegevens die latere diagnose kunnen vergemakkelijken.

Na de registratie kijkt de helpdeskmedewerker of er een oplossing bekend is voor het incident. Zo'n oplossing is opgeslagen in de registratie van bekende fouten en problemen.

Als de helpdesk op een bepaald moment meerdere incidenten moet verhelpen is het nodig de incidenten te ordenen naar prioriteit. De prioriteit van een incident wordt bepaald door drie factoren. De impact van een incident is de mate waarin het leidt tot een afwijking in de normale dienstverlening. Als de belangrijkste computer van een organisatie niet meer werkt dan heeft dat incident een hoge impact. De urgentie geeft aan hoe snel een incident opgelost moet worden. De laatste factor is de verwachte inspanning die het gaat kosten om een incident te verhelpen. Incidenten die snel op te lossen zijn, moeten snel opgelost worden, ook al zijn de impact en urgentie laag. Gebruikers weten vaak wel of een bepaald incident snel op te lossen valt en raken geïrriteerd als het te lang duurt.

Na registratie en classificatie begint de helpdesk met het diagnostiseren en zo snel mogelijk oplossen van een incident. Het gaat er om een oplossing te vinden die voor de gebruiker aanvaardbaar is. Mocht het de helpdesk niet lukken om binnen een bepaalde tijd een oplossing te vinden dan moeten er meer personen of specialisten ingezet worden. Dit heet escalatie.

Een incident kan pas afgesloten worden als er een oplossing is gevonden en uitgevoerd. Elke incident wordt gecodeerd en opgeslagen zodat er later nog eens naar gekeken kan worden en er overzichten kunnen worden gemaakt.

2.1.1.3 Rapportage aan management

De rapporten die door de helpdesk gemaakt worden bestaan uit overzichten van opgeloste en niet-opgeloste incidenten en de tijd die aan de incidenten is besteed. Door de incidenten in categorieën in te delen kunnen zwakke plekken in de IT-infrastructuur duidelijk worden.

2.1.2 Probleembeheer

De helpdesk zorgt ervoor dat medewerkers in geval van een incident zo snel mogelijk weer aan het werk kunnen. Om dienstverlening van een zo hoog mogelijk niveau te kunnen bieden zullen de oorzaken van incidenten achterhaald moeten worden.

Het doel van Probleembeheer is het zorgen voor een zo hoog mogelijke stabiliteit van IT-dienstverlening door het opsporen en verwijderen van fouten in de infrastructuur.

Hoe minder incidenten hoe stabieler de dienstverlening. Het is de taak van Probleembeheer structurele maatregelen te nemen om herhaling van incidenten te voorkomen.

Probleembeheer houdt zich bezig met problemen, fouten en incidenten. ITIL definieert een probleem als een ongewenste situatie, geïdentificeerd uit incidenten, die indicatief is voor een fout in de IT-infrastructuur.

Stel dat gebruikers onder bepaalde omstandigheden 's ochtends niet kunnen inloggen op het netwerk. Die mensen gaan bellen en melden hun incident. Al die incidenten worden geregistreerd door Helpdesk. Omdat ze op elkaar lijken worden ze allemaal verbonden met een (nieuw) probleem. Het is nu de taak van Probleembeheer om de oorzaak van het probleem te achterhalen. Dit wordt de fout genoemd. Als de fout in de IT-infrastructuur is opgespoord moeten er maatregelen genomen worden die ervoor zorgen dat de incidenten zich niet meer voordoen.

Het werk van probleembeheer kan verdeeld worden in zes activiteiten.

1. probleemidentificatie en registratie
2. classificatie
3. allocatie mensen en middelen
4. onderzoek en diagnose
5. foutbeheer
6. rapportage aan management

2.1.2.1 Probleemidentificatie en registratie

Probleembeheer houdt een lijst bij van problemen. Elk incident wordt gekoppeld aan een probleem. Bij de analyse van een nieuw incident wordt gekeken of er een probleem is met incidenten die veel lijken op het nieuwe incident. Als zo'n probleem niet gevonden kan worden dan wordt er een nieuw probleem gedefinieerd.

Indien voor een probleem een oplossing bekend is dan wordt die oplossing verbonden met het probleem.

Alle incidenten en problemen worden geregistreerd. Op die manier wordt voorkomen dat incidenten niet afgehandeld worden of dat problemen niet opgelost worden.

2.1.2.2 Classificatie

Na het definiëren van een nieuw probleem besluit Probleembeheer hoeveel aandacht er besteed gaat worden aan de oplossing ervan. Dit hangt af van de invloed van het probleem op de organisatie.

Via classificatie wordt bepaald hoe belangrijk het is dat een probleem opgelost wordt. Classificatie bestaat uit vier stappen.

1. categoriseren
2. impact
3. urgentie
4. prioriteit

In de categorisatie-stap wordt een probleem ingedeeld in een groep met overeenkomende problemen (bijvoorbeeld alle netwerkproblemen, alle printproblemen). Met impact worden de gevolgen van een probleem op overige delen van de IT-infrastructuur bedoeld. De urgentie van een probleem bepaalt hoe snel het opgelost moet worden. De prioriteit geeft aan hoe belangrijk een probleem is ten opzichte van andere problemen.

Het is van belang onderscheid te maken tussen de classificatiestap die in deze paragraaf wordt beschreven en het classificeren van incidenten naar problemen. Bij de classificatie van een probleem gaat het er om te bepalen hoe belangrijk het is dat de oorzaak van een probleem achterhaald wordt. Het doel van het classificeren van een incident is het vinden van een probleem met soortgelijke incidenten.

2.1.2.3 Allocatie mensen en middelen

Op basis van de classificatie wordt in deze stap bepaald welke mensen en middelen gebruikt zullen gaan worden om een probleem op te lossen.

2.1.2.4 Onderzoek en diagnose

Onderzoek en diagnose moet leiden tot het vinden van de werkelijke oorzaak van een probleem. Hierbij wordt niet alleen gekeken naar apparatuur, maar ook naar procedures, handleidingen en eventueel menselijk falen.

2.1.2.5 Foutbeheer

Als de oorzaak van een probleem, en dus van de incidenten, gevonden is spreken we van een bekende fout. Het doel van foutbeheer is het opheffen van fouten in de IT-infrastructuur en het minimaliseren van de gevolgen ervan.

Om een fout te verwijderen moet er meestal een wijziging in de IT-infrastructuur worden aangebracht. Het uitvoeren van zo'n wijziging is de taak van Wijzigingsbeheer. Het is echter de taak van Probleembeheer om een voorstel te doen voor een wijziging.

2.1.2.6 Informatievoorziening

Probleembeheer geeft informatie aan het management over de geconstateerde problemen, de invloed daarvan op de organisatie en de voortgang die geboekt wordt met het oplossen. De rapportage kan bijvoorbeeld bestaan uit overzichten van incidenten en problemen per afdeling of per onderwerp en de tijd en mankracht die besteed is aan het oplossen van elk probleem.

2.2 De Object Oriented Software Engineering (OOSE) methodiek

Het ontwikkelen van software duurt meestal veel langer dan gepland. Volgens Ivar Jacobson is de oorzaak daarvan dat de software engineering industrie nog in de kinderschoenen staat. In zijn boek Object Oriented Software Engineering (OOSE) beschrijft hij een methode die ertoe moet leiden dat het ontwikkelen van software meer op een industrieel proces gaat lijken.

Als analogie gebruikt hij het bouwen van een huis. Mensen bouwen al heel lang huizen en in de loop der tijd zijn er algemeen geaccepteerde ideeën ontstaan over hoe dat het beste gedaan kan worden. Bepaalde onderdelen van het bouwen van een huis kunnen daarom gemakkelijk uitbesteed worden. Voor het maken van een stevige fundering bestaan bijvoorbeeld vaste procedures. Een aannemer kan dat deel van het werk laten doen door een bedrijf dat daarin gespecialiseerd is. In de software industrie wordt daarentegen door fabrikanten steeds opnieuw het wiel uitgevonden. Verschillende bedrijven bouwen tekstverwerkers die allemaal hetzelfde kunnen.

De OOSE methode van Ivar Jacobson is bedoeld om dit te veranderen en op een industriële manier computersystemen te ontwikkelen. Uiteindelijk wil hij toe naar een situatie als in de huizenbouw. Software kan dan in elkaar gezet worden als een huis, door het bestaan van componenten, die samengevoegd kunnen worden. We zien tegenwoordig al, vooral na de komst van Java, steeds meer dat software uit componenten gaat bestaan.

WordPerfect heeft al een tekstverwerker geschreven die uit componenten is opgebouwd. Als een gebruiker een rapport wil schrijven dan begint hij met een eenvoudige

tekstverwerkingscomponent. Later kan hij via Internet een component voor spellingscontrole en een component voor het maken van grafieken oproepen. Deze samenwerkende componenten zijn voorlopig nog afkomstig van dezelfde fabrikant, maar in de toekomst zullen er steeds meer bedrijven komen die gespecialiseerde componenten maken die met allerlei andere componenten kunnen samenwerken.

2.2.1 Systeemontwikkeling

De ontwikkeling van een systeem begint nadat is vastgesteld dat er behoefte aan is. Er wordt dan een document gemaakt waarin beschreven staat wat het systeem moet kunnen. Dit wordt de requirements specification genoemd.

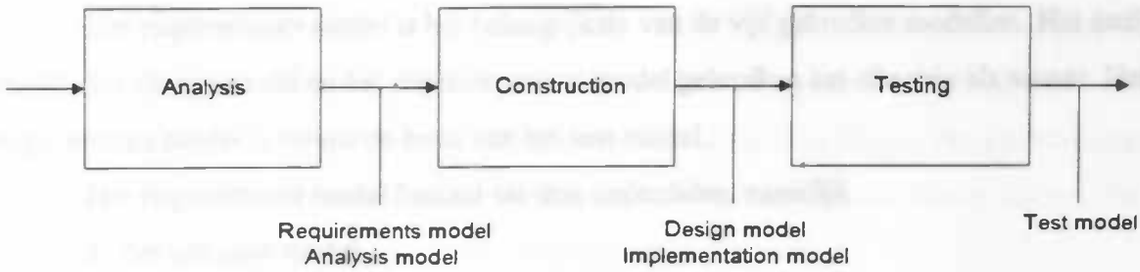
Het ontwikkelen van een computersysteem is een ingewikkelde taak, die verdeeld kan worden in drie processen.

1. Analysis process
2. Construction process
3. Testing process

Om de cómplexiteit van het ontwikkelen in de hand te houden maakt OOSE gebruik van een aantal modellen, die elk gericht zijn op een bepaald aspect van het systeem.

De vijf gebruikte modellen zijn:

1. Het **requirements model** waarin staat wat de eigenschappen van het systeem zijn en wat het moet kunnen.
2. Het **analysis model** waarin de structuur van het systeem beschreven wordt.
3. Het **design model** waarin de resultaten van het analysis process aangepast worden aan de implementatie-omgeving.
4. In het **implementation model** wordt de uiteindelijke programmacode beschreven.
5. Het **test model** beschrijft hoe het systeem wordt getest en wat de resultaten daarvan zijn.



2.2.2 Het analysis process

Het doel van het analysis process is nauwkeurig te beschrijven wat het systeem zal gaan doen. Als basis hiervoor gebruiken we de requirements specification: een tekst waarin de eisen en wensen van de opdrachtgever staan.

Het is heel belangrijk om met de toekomstige gebruikers te praten en ze mee te laten denken over de ontwikkeling van het systeem. De kans om een systeem te maken dat ook gebruikt zal gaan worden is dan het grootst.

In het analysis process houden we geen rekening met de omgeving waarin het systeem geïmplementeerd zal gaan worden, op die manier kunnen we ons concentreren op het eigenlijke probleem. Tijdens het construction process wordt het analysis model dan aangepast aan de implementatie omgeving.

Tijdens het analysis process maken we twee modellen: het requirements model en het analysis model.

In het requirements model wordt gedefinieerd wat het systeem moet kunnen en wat het niet hoeft te kunnen. We maken hierbij gebruik van de objecten uit het problem domain en interface beschrijvingen. In het requirements model wordt de complete functionaliteit van het systeem beschreven door middel van actoren en use cases.

Het analysis model neemt als invoer het requirements model. De use cases uit het requirements model worden opgesplitst in control, entity en interface objecten. De bedoeling hiervan is een robuuste en makkelijk te wijzigen structuur voor het systeem te maken

2.2.2.1 Requirements model

Het requirements model beschrijft welke taken wel en welke niet door het systeem verricht moeten worden. De toekomstige gebruikers spelen een belangrijke rol bij het ontwikkelen van dit model, dat kan dienen als een contract tussen opdrachtgever en ontwikkelaar.

Het requirements model is het belangrijkste van de vijf gebruikte modellen. Het analysis model, het design model en het implementation model gebruiken het alle drie als invoer. Het requirements model is tevens de basis van het test model.

Het requirements model bestaat uit drie onderdelen, namelijk

1. het use case model
2. de interface beschrijvingen
3. het problem domain model

2.2.2.1.1 Use case model

In het use case model wordt beschreven wat de functionaliteit van het systeem is vanuit het oogpunt van de gebruikers. Dit model bestaat uit actoren en use cases.

Actoren zijn entiteiten die geen deel uitmaken van het systeem. Een actor kan een persoon zijn, maar ook een ander systeem. Een actor is niet één bepaalde persoon, maar een rol die door meerdere personen gespeeld kan worden. Actoren gebruiken het systeem om verschillende taken uit te voeren. Zulke taken worden use cases genoemd.

In een use case wordt beschreven welke acties een actor en het systeem uitvoeren om een bepaalde taak uit te voeren.

Het kan voorkomen dat verschillende actoren in verschillende use cases dezelfde reeks handelingen uitvoeren. Deze gemeenschappelijke handelingen worden beschreven in een deel-use case die vervolgens wordt ingevoegd bij de verschillende use cases. Zo'n deel-use case noemen we een extend use case.

Het ontwerpen van een systeem met behulp van use cases en actoren wordt **use case driven design** genoemd en is kenmerkend voor de methode van Jacobson. Deze manier van ontwerpen heeft een aantal voordelen. Het is makkelijk om met de toekomstige gebruikers van het systeem te praten in termen van use cases en actoren. Mocht de manier waarop een bepaalde actor met het systeem omgaat in de toekomst wijzigen dan hoeft er meestal maar één use case veranderd te worden. De use cases worden tijdens de verdere ontwikkeling van het systeem intensief gebruikt. Tijdens het construction process wordt bijvoorbeeld steeds één use case geïmplementeerd. Tijdens het test process wordt gekeken of de use cases zonder problemen uitgevoerd kunnen worden.

2.2.2.1.2 Interface descriptions

Interface descriptions zijn voorbeelden van de toekomstige gebruikersinterfaces van het systeem. Ze kunnen de ontwikkelaars helpen om een goed beeld te krijgen van een taak doordat de gebruikers kunnen controleren of ze de taak kunnen uitvoeren met de interface. Als dat niet lukt dan begrijpen de ontwikkelaars het probleem nog niet goed.

2.2.2.1.3 Problem domain object model

Het problem domain model definieert de entiteiten die binnen het systeem gebruikt worden. Dit vergemakkelijkt de communicatie tussen ontwikkelaars en gebruikers.

Een systeem dat personeel van een luchtvaartverkeerstoren moet ondersteunen heeft waarschijnlijk onder meer vluchten, vliegtuigen en landingsbanen als problem domain objects.

2.2.2.2 Analysis Model

Als het requirements model afgerond is en goedgekeurd wordt door de gebruikers kan begonnen worden met het volgende model. In het analysis model wordt de logische structuur van het toekomstige systeem beschreven, zonder rekening te houden met de uiteindelijke implementatie omgeving. De structuur moet vooral ook makkelijk uitbreidbaar zijn.

In het analysis model worden drie soorten objecten gebruikt, namelijk

- entity objects
- interface objects
- control objects

Elk type object bevat informatie, vertoont bepaald gedrag en draagt zorg voor de presentatie van informatie, maar in verschillende mate. De reden om met deze drie soorten objecten te werken is dat in geval van wijzigingen in het systeem meestal maar één object gewijzigd hoeft te worden. De verdeling van verschillende soorten functionaliteit onderscheidt de methode van Jacobson van andere oo-methodieken.

2.2.2.2.1 Entity objects

Informatie die we langere tijd willen bewaren (denk aan personen, problemen, incidenten en fouten) modelleren we door middel van entity objects. Dit zijn vaak objecten die in een database opgeslagen gaan worden. Entity objecten laten zich meestal makkelijk vinden. Ze hebben vrijwel altijd een tegenhanger in de werkelijkheid en bevinden zich in het problem domain.

2.2.2.2.2 Interface objects

Interface objects zorgen voor de communicatie tussen het systeem en de buitenwereld. De buitenwereld bestaat uit gebruikers en andere systemen. Via de interface objecten kunnen de actoren communiceren met het systeem.

2.2.2.2.3 Control objects

Control object vormen vaak de lijm tussen interface objects en entity objects. Ze modelleren gedrag en informatie die niet onder te brengen zijn bij de andere twee typen.

2.2.2.2.4 Een voorbeeld van het analyseren van een use case

De use cases uit het requirements model worden nu gemodelleerd door middel van deze drie typen objecten.

Stel we hebben een use case voor het toevoegen van een persoon aan een database. De use case start als een gebruiker aangeeft dat hij een persoon wil toevoegen. De gebruiker speelt nu de rol van PersoonToevoeger. Het systeem presenteert dan een formulier dat de actor kan invullen. Als extraatje zorgen we ervoor dat de woonplaats van de persoon automatisch wordt ingevuld zodra zijn postcode is ingevoerd.

Deze use case kan als volgt worden opgedeeld in analysis objecten. De nieuwe persoon is een entity object. Hij bevat voornamelijk informatie, die we voor langere tijd op willen slaan. Er is ook een interface object, namelijk het invulformulier. Tot slot is er een control object dat de woonplaats bij een postcode zoekt.

2.2.3 Construction process

In het construction process worden twee modellen ontwikkeld: het design model en het implementation model.

2.2.3.1 Design Model

In het analysis process werd geen rekening gehouden met de omgeving waarin het systeem geïmplementeerd zal worden. Hiervoor zijn twee redenen aan te voeren. De eerste reden is dat het niet wenselijk is de structuur van het systeem af te laten hangen van de omgeving. Die omgeving zal namelijk tijdens het gebruiken van het systeem onvermijdelijk veranderen. De tweede reden is dat we ons beter kunnen concentreren op het werkelijke probleem als we niet worden afgeleid door allerlei implementatie-gedachten.

Het design model wordt afgeleid van het analysis model. We doen dat door het analysis model formeler te maken. Dat kan onder meer gebeuren door weer te geven hoe de verschillende objecten met elkaar communiceren. We gaan net zo lang door met het precies opschrijven van het verloop van use cases tot we bijna programmacode aan het schrijven zijn.

2.2.3.2 Implementation Model

Het implementation model bestaat alleen uit de broncode met commentaar.

2.2.4 Test process

Het enige model dat ontwikkeld wordt in dit process is het test model.

2.2.4.1 Test Model

Het test model beschrijft de tests die uitgevoerd moeten worden om te zien of het systeem goed functioneert. Het bevat ook de resultaten van die tests.

Het uittesten van het systeem begint op het laagste niveau, daarna gaat het langzaam omhoog tot de use cases en tot slot het hele systeem. Tijdens het testen worden het requirements model en het analysis model gebruikt om te kijken of het systeem functioneert zoals daarin beschreven staat.

2.2.5 Gebruikte onderdelen van OOSE

Voor het maken van het helpdeskondersteunende systeem zijn alleen de modellen gebruikt die in het analysis process en in het test process worden ontwikkeld.

Het uiteindelijke systeem is voornamelijk getest op het correct functioneren van de ontwikkelde algoritmen. Zij vormen namelijk het cruciale deel van het systeem.

3. ITIL is slechts het begin

In de helpdeskmodule van ITIL worden in theorie de opzet en het functioneren van een helpdesk beschreven. ITIL is niet meer dan een kapstok, een framework waaraan de implementatie van een helpdesk kan worden opgehangen. Uiteraard blijven sommige aspecten van een helpdesk in de beschrijving vaag. Er wordt dan alleen beschreven wát er op een bepaald moment gebeuren moet, niet hoe dat precies moet. Het is dan aan de IT-afdeling om te beslissen hoe men diverse onderdelen van de helpdesk gaat implementeren.

3.1 Probleemidentificatie en registratie

Één van de punten die in de helpdeskmodule niet uitgewerkt zijn is de activiteit Probleemidentificatie en registratie van het proces Probleembeheer. Binnen deze activiteit worden incidenten verbonden met problemen en worden problemen geïdentificeerd uit incidenten. Dit is een belangrijk onderdeel omdat dit een aanzienlijk deel van het werk van de helpdeskmedewerkers vormt.

Op het moment dat een incident wordt gemeld moet de helpdesk onderzoeken bij welk probleem het hoort. ITIL noemt dit Probleemidentificatie, maar het kan ook gezien worden als het classificeren van het incident.

Een drukbezette helpdeskmedewerker heeft geen tijd om elk bekend probleem af te lopen om te controleren of het misschien het probleem is waar zijn incident bij thuishoort. De verleiding is dan groot om een ad hoc oplossing te bedenken. Als alle helpdeskmedewerkers op die manier werken dan kunnen ze niet optimaal profiteren van de kennis die op de helpdesk aanwezig is en zal er veel dubbel werk verricht worden.

Als de helpdeskmedewerkers echter consequent bij elk incident het juiste probleem zoeken en gevonden oplossingen bewaren bij de problemen dan wordt de gezamenlijke kennis opgeslagen in het helpdesksysteem en profiteren de helpdeskmedewerkers van elkaar. Zo kan er na verloop van tijd veel efficiënter gewerkt worden. Dit heeft een verbeterde IT-dienstverlening tot gevolg.

Als we willen dat de helpdeskmedewerkers het juiste probleem zoeken voor elk incident dan moeten we er voor zorgen dat het classificeren van een incident zo soepel en zo snel mogelijk kan gebeuren. We moeten de helpdeskmedewerker dus ondersteunen bij het classificeren van incidenten.

Omdat de activiteit niet is uitgewerkt binnen ITIL heeft de IT-afdeling, en dus ook een systeemontwikkelaar die een helpdeskondersteunend systeem ontwikkelt, de vrijheid om te bepalen hoe de activiteit geïmplementeerd wordt. Tijdens dit onderzoek hebben we een methode ontwikkeld om de classificatie van incidenten snel en eenvoudig uit te voeren. Deze methode wordt in het volgende hoofdstuk beschreven.

3.1.1 Kenmerken van incidenten en problemen

Om incidenten te kunnen classificeren moeten ze over duidelijk gedefinieerde kenmerken beschikken. Deze kenmerken moeten vervolgens vergeleken worden met de kenmerken van problemen. De kenmerken van problemen worden afgeleid van de kenmerken van incidenten.

We onderscheiden een aantal soorten kenmerken:

1. Keywords
2. Een korte beschrijving
3. Proposities. Dit zijn vraag-antwoord combinaties waarbij de vragen die de helpdeskmedewerker stelt gekoppeld worden aan de antwoorden die de klant daarop geeft.

De keywords bieden de helpdeskmedewerker de mogelijkheid heel beknopt aan te geven om wat voor incident het gaat. De beschrijving is al een stuk uitgebreider en de proposities geven het analyseproces stap voor stap weer.

Een probleem wordt geïdentificeerd uit één of meer incidenten. De kenmerken van een probleem worden afgeleid van de kenmerken van de incidenten die er mee verbonden zijn. Voorlopig worden alleen de keywords en de proposities van de incidenten gebruikt om de kenmerken van een probleem te bepalen, omdat we verwachten dat die informatie voldoende is om het juiste probleem te identificeren.

Voor elk probleem wordt een keywordverdeling en een propositieverdeling bepaald.

De keywordverdeling bevat alle keywords die voorkomen bij de incidenten van het probleem. Daarnaast staat in de keywordverdeling ook hoe vaak elk keyword binnen de incidenten voorkomt.

De onderstaande keywordverdeling hoort bij een probleem dat verbonden is met vier incidenten. Elk incident heeft als keywords Printen en Netwerk. Daarnaast hebben drie van de vier incidenten ook nog het keyword Word97.

Keyword	#voorkomens
Word97	3
Printen	4
Netwerk	4

Tabel 3-1 Een voorbeeld van een keywordverdeling

Een propositieverdeling bevat informatie over de vragen die de helpdeskmedewerkers gesteld hebben aan de melders van de incidenten en de antwoorden die ze daarop gekregen hebben. Een voorbeeld zou er als volgt uit kunnen zien.

Vraag	Antwoord	#voorkomens
Staat de printer aan?	Ja	3
	Nee	1
Is de computer verbonden met de printer?	Ja	4

Tabel 3-2 Een voorbeeld van een propositieverdeling

In de propositieverdeling van het probleem hierboven is bij vier incidenten de vraag 'Staat de printer aan?' gesteld. Drie keer was het antwoord Ja, en één keer was het antwoord Nee.

3.1.2 Het vergelijken van kenmerken van incidenten en problemen

Doordat de kenmerken van een probleem zijn afgeleid van de kenmerken van de incidenten die ermee verbonden zijn is het niet moeilijk de kenmerken van een nieuw incident te vergelijken met de kenmerken van een probleem.

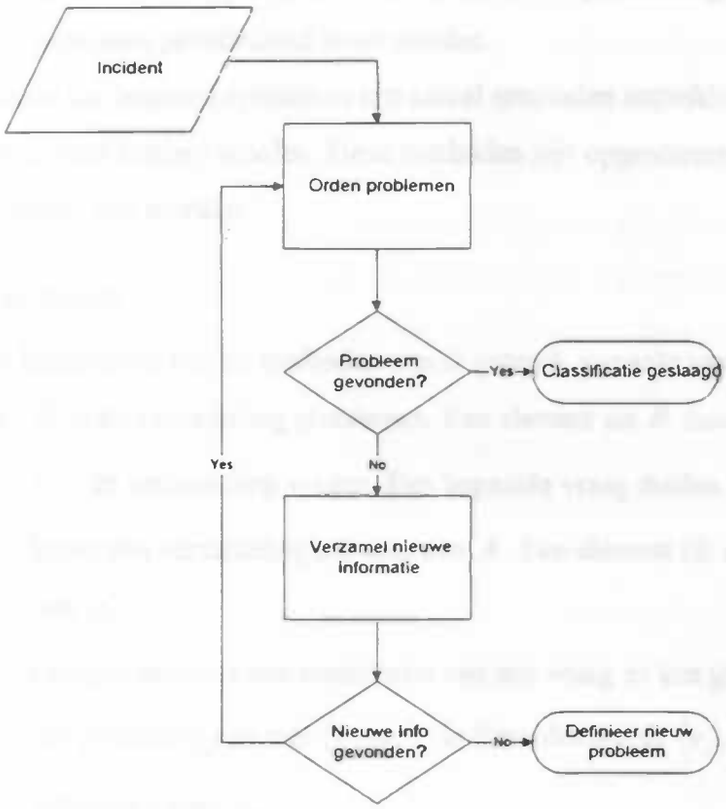
Zo kunnen we bijvoorbeeld voor elk keyword van een incident kijken of dat keyword ook voorkomt bij de incidenten van het probleem. We gebruiken hiervoor de keywordverdeling van het probleem. Hoe vaker het keyword bij de incidenten van het probleem voorkomt, hoe waarschijnlijker het is dat het incident bij het probleem thuishoort. Hoe meer keywords de incidenten van een probleem gemeenschappelijk hebben met een nieuw incident, hoe waarschijnlijker het is dat het incident bij dat probleem hoort.

De vragen en antwoorden van een incident kunnen we op dezelfde manier vergelijken met de vragen en antwoorden van de incidenten van een probleem. Hiervoor gebruiken we de propositieverdelingen van de problemen.

We weten nu wat de kenmerken zijn van incidenten en problemen en hoe we ze kunnen gebruiken om te kijken of een incident bij een probleem thuishoort. In het volgende hoofdstuk beschrijven we een aantal algoritmen die de helpdeskmedewerkers ondersteunen bij het vinden van het juiste probleem voor een incident. Deze algoritmen maken gebruik van de kenmerken van incidenten en problemen.

4. De ondersteuning van het classificeren van een incident

Het doel van het analyseren en classificeren van een incident is het te verbinden met een probleem. Dit proces verloopt als volgt.



Voordat de classificatie begint kent de helpdeskmedewerker het incident een aantal keywords en een omschrijving toe. Op basis van deze keywords kan het systeem een ordening aanbrengen in de bekende problemen. Problemen met keywords die ook bij het incident voorkomen zijn waarschijnlijker dan problemen waarbij dat niet het geval is. Als na het ordenen van de problemen het juiste probleem gevonden is wordt het incident met dit probleem verbonden en is de classificatie afgelopen.

Als het juiste probleem nog niet is gevonden moet de helpdeskmedewerker extra informatie verzamelen over het incident. De eenvoudigste manier om extra informatie te verzamelen is het toekennen van een extra keyword aan het incident. Een andere manier is het stellen van een vraag aan de gebruiker.

Het systeem sorteert in dat geval de vragen die bij de problemen voorkomen, rekening houdend met de rangorde van de problemen en het discriminerend vermogen van de vragen. De

helpdeskmedewerker kiest één van de vragen en stelt die aan de klant. De klant geeft vervolgens antwoord op de vraag. De vraag en het gegeven antwoord vormen samen een propositie, die verbonden wordt met het incident.

Met de extra informatie kan het systeem dan opnieuw de problemen ordenen.

Dit proces van problemen ordenen, extra informatie verzamelen en opnieuw ordenen herhaalt zich totdat de helpdeskmedewerker het juiste probleem gevonden heeft, of besluit dat er een nieuw probleem gedefinieerd moet worden.

Voor het helpdesksysteem is een aantal methoden ontwikkeld waarmee problemen en vragen geordend kunnen worden. Deze methoden zijn opgenomen in een applicatie waarmee hun werking getest kan worden.

4.1.1 Definities

Voor het beschrijven van de methoden wordt gebruik gemaakt van de volgende definities:

- B is de verzameling problemen. Een element uit B duiden we aan met b_l;
- V is de verzameling vragen. Een bepaalde vraag duiden we aan met v_j. Bij elke vraag hoort een verzameling antwoorden A. Een element uit die verzameling duiden we aan met a_k;
- Een propositie is een combinatie van een vraag en een gegeven antwoord. We duiden een propositie aan met $\langle v_j, a_k \rangle$. In formules wordt $\langle v_j, a_k \rangle$ voor de leesbaarheid vervangen door r_{jk}.

4.1.2 Ordenen van problemen

Er zijn twee redenen om de problemen te sorteren. De eerste reden is dat de helpdeskmedewerker dan kan zien welke problemen, gegeven de beschikbare informatie, het meest waarschijnlijk zijn. Als hij het juiste probleem herkent dan kan hij het meteen selecteren. De andere reden is dat bij het sorteren van de vragen gebruik gemaakt kan worden van de ordening van de problemen.

We gaan er steeds van uit dat de keywords en de proposities die verbonden zijn met een incident onafhankelijk van elkaar zijn. Als we dat niet zouden doen moeten we rekening houden met de onderlinge afhankelijkheden van de keywords en proposities. Dat maakt de algoritmen een stuk ingewikkelder en daardoor ook een stuk trager.

Om te onderzoeken hoe goed de verschillende algoritmen presteren hebben we er voor gekozen om de problemen eerst alleen op de keywords of alleen op de proposities van het incident te ordenen. Het is niet moeilijk om later op basis van zowel de keywords als de proposities te ordenen.

Het ordenen van problemen verloopt in principe als volgt. Voor elk keyword of elke propositie van incident h berekenen we voor elk probleem een belief-waarde, als dat met het algoritme kan. Deze belief-waarde geeft aan hoe sterk het keyword of de propositie een aanwijzing is dat incident h verbonden moet worden met probleem b_i.

Na afloop zijn de meeste problemen verbonden met één of meer belief-waarden. Voor de problemen die niet verbonden zijn met een belief-waarde zijn geen aanwijzingen gevonden in de beschikbare informatie. Voor elk van de problemen die wel verbonden zijn met een belief-waarde geldt dat er tenminste één aanwijzing is dat het incident bij dat probleem thuishoort.

Het aantal belief-waarden van een probleem hoeft niet gelijk te zijn aan het totale aantal keywords of proposities. Binnen sommige algoritmen is het namelijk niet mogelijk een belief-waarde te berekenen voor een probleem en een keyword als dat keyword bij geen enkel incident van het probleem voorkomt.

De belief-waarden van een probleem worden vervolgens gecombineerd tot één belief-waarde. Dit kan op een groot aantal manieren gedaan worden. In paragraaf 7.4.3 staat beschreven welke manieren tijdens het testen onderzocht zijn.

De problemen kunnen nu gesorteerd worden op basis van de totale belief-waarde.

4.1.2.1 Het ordenen van problemen met het theorema van Bayes

4.1.2.1.1 Theorie

Met het theorema van Bayes kan de kans op een bepaalde hypothese bepaald worden, uitgaande van de informatie die beschikbaar is. Zo is het mogelijk de kans te berekenen dat iemand de rode hond heeft als bekend is dat hij koorts heeft.

Het theorema wordt als volgt gedefinieerd: $P(H_i|E) = \frac{P(E|H_i) \cdot P(H_i)}{\sum_{n=1}^{k} (P(E|H_n) \cdot P(H_n))}$. In woorden zegt het theorema dat de kans op een specifieke hypothese H_i, gegeven informatie E, gelijk is aan de kans op E, gegeven H_i, vermenigvuldigd met de kans op H_i gedeeld door een bepaalde

som. Deze som wordt bepaald door voor elke mogelijk hypothese H de kans op E, gegeven H te vermenigvuldigen met de kans op H en al deze producten op te tellen.

Om de kans op hypothese H_i (de patiënt heeft de rode hond) te bepalen, gegeven de informatie E (patiënt heeft koorts) doen we dus het volgende.

1. Bereken voor elke hypothese:
   1.1 de kans op E (koorts), gegeven dat de hypothese correct is
   1.2 de kans op de hypothese
   1.3 het product van beide kansen.
2. Tel al deze producten bij elkaar op.
3. Deel om $P(H_i|E)$ te bepalen het product $P(E|H_i)\cdot P(H_i)$ door het bij stap 2 berekende totaal.

Omdat steeds een deel van de som gedeeld wordt door de totale som geldt altijd $0 \le P(H_i|E) \le 1$.

Het werken met dit theorema levert in de praktijk een aantal problemen op. Rode hond is niet de enige mogelijke ziekte. De patiënt kan ook de bof hebben of mazelen of een ernstige verkoudheid. Welke mogelijke hypothesen gaan we bekijken?

Een tweede probleem is dat voor elke hypothese de kansen op een groot aantal mogelijke symptomen bekend moeten zijn. Voor elke ziekte moeten we weten wat de kans is op koorts, rode vlekken, een loopneus en nog veel meer. In de meeste praktijkgevallen is het onmogelijk om al die kansen te bepalen. Bij sommige ziekten komt een bepaald symptoom zo weinig voor dat het onmogelijk $P(E|H_i)$ te bepalen. We kunnen echter niet zomaar stellen dat $P(E|H_i)$ gelijk is aan 0, omdat we dan de hypothese helemaal uitsluiten.

Het eerste probleem is voor het helpdesksysteem geen probleem. De verzameling hypothesen is gelijk aan de verzameling problemen die bekend zijn binnen het systeem.

Het tweede probleem is wat lastiger. Als een keyword niet voorkomt bij de incidenten van een probleem dan wordt er geen belief-waarde voor dat probleem en dat keyword bepaald. We besteden voor dat probleem dan geen aandacht aan het keyword.

4.1.2.1.2 Algoritme

De informatie E die bekend is van een incident bestaat uit de keywords die er aan toegekend zijn en de gestelde vragen en de antwoorden daarop.

Het algoritme dat we gebruiken om de problemen te sorteren met behulp van het theorema van Bayes bestaat uit twee stappen.

1. Ga elk stukje beschikbare informatie E_j af.
    1.1. Ga alle problemen af
        1.1.1. Bepaal voor elk probleem $P(E|b_i) \cdot P(b_i)$
        1.1.2. Tel al deze producten bij elkaar op
        1.1.3. Bepaal voor elk probleem de kans $P(b_i|E_j)$ door $P(E|b_i) \cdot P(b_i)$ te delen door de bij 1.1.2 berekende som.
        1.1.4. Bewaar die kans bij het probleem.
2. Voor elk probleem:
    2.1 Bepaal uit de bewaarde kansen de belief-waarde voor het probleem

Een klein voorbeeld is te zien in de tabel hieronder. Voor elk probleem zijn de kansen op $P(E|b_i)$ en $P(H_i)$ bepaald en vermenigvuldigd. De producten worden opgeteld (samen 0.60) en het product van elk probleem wordt vervolgens gedeeld door deze som. Het derde probleem komt met 0.53 als meest waarschijnlijke uit de bus.

	$P(E\|b_i)$	$P(H_i)$	$P(E\|b_i) \cdot P(H_i)$	$P(b_i\|E)$
Probleem b_1	0.30	0.10	0.03	0.05
Probleem b_2	0.50	0.50	0.25	0.42
Probleem b_3	0.80	0.40	0.32	0.53
totaal	1.60	1.00	0.60	1.00

4.1.2.2 Het ordenen van problemen op basis van verhoudingen

4.1.2.2.1 Theorie

De achtergrondgedachte van deze methode is dat we de verhouding berekenen tussen de frequentie waarmee antwoord a_k op vraag v_j wordt gegeven bij de incidenten van een probleem en de frequentie waarmee antwoord a_k op vraag v_j wordt gegeven bij alle incidenten.

Bij het bepalen van de belief-waarde van probleem b_i gaan we er vanuit dat het incident dat we aan het analyseren zijn thuishoort bij b_i. Zo sluiten we een probleem dat geen enkele propositie overeenkomstig heeft met het incident niet direct uit.

De frequentie waarmee antwoord a_k op vraag v_j wordt gegeven binnen de incidenten van b_i is $P(r_{jk}|b_i) = \dfrac{\#\text{incidenten van } b_i \text{ waar } \langle v_j, a_k \rangle \text{ voorkomt}}{\#\text{incidenten van } b_i \text{ waar } v_j \text{ wordt gesteld}}$.

De frequentie waarmee antwoord a_k op vraag v_j wordt gegeven bij alle incidenten is gelijk aan $P(a_k|v_j) = \dfrac{\#\text{aantal incidenten met } \langle v_j, a_k \rangle}{\#\text{aantal incidenten met } v_j}$

Als we deze twee frequenties op elkaar delen krijgen we $\dfrac{P(r_{jk}|b_i)}{P(a_k|v_j)}$. Dit is de belief-waarde voor probleem b_i en propositie $\langle v_j, a_k \rangle$.

$P(r_{jk}|b_i)$ is een positief getal dat kleiner of gelijk is aan 1. Hoe vaker antwoord a_k op vraag v_j werd gegeven binnen de incidenten van het probleem, hoe dichter het getal bij 1 zal liggen. Hetzelfde geldt voor $P(a_k|v_j)$, alleen kijken we nu naar alle incidenten en niet naar de incidenten van één bepaald probleem.

Omdat $P(a_k|v_j)$ voor alle problemen hetzelfde is kunnen we de deling door $P(a_k|v_j)$ ook zien als een normalisatie-stap.

Als $\dfrac{P(r_{jk}|b_i)}{P(a_k|v_j)}$ groter is dan 1 dan betekent dat dat binnen de incidenten van b_i vaker dan gemiddeld antwoord a_k op vraag v_j werd gegeven. We beschouwen dit als informatief.

Stel bijvoorbeeld eens dat binnen een probleem b_i geldt dat $P(r_{jk}|b_i) = 8/10$. Dat betekent dat elke tien keer dat vraag v_j gesteld werd acht keer antwoord a_k werd gegeven. Laat de verhouding gezien over alle incidenten, $P(a_k|v_j)$, gelijk zijn aan $2/10$. Als deze twee waarden op elkaar gedeeld worden krijgen we $\frac{P(r_{jk}|b_i)}{P(a_k|v_j)} = \frac{8/10}{2/10} = 4$. Antwoord a_k wordt binnen het probleem vier keer zo vaak gegeven als buiten het probleem. Als we nu aan een gebruiker vraag v_j stellen en hij geeft als antwoord a_k, dan is dat een duidelijke aanwijzing voor probleem b_i.

4.1.2.2.2 Algoritme

Het algoritme dat we gebruiken om de problemen te sorteren door te kijken naar de verhoudingen bestaat uit twee stappen.

1. Ga elk stukje beschikbare informatie E_j af.
    - 1.1 Bepaal de frequentie van E_j binnen alle incidenten
    - 1.2 Ga alle problemen af
        - 1.2.1 Bepaal de frequentie van E_j binnen de incidenten van b_i
        - 1.2.2 Deel die frequentie door de frequentie van 1.1.
        - 1.2.3 Bewaar dit quotiënt bij het probleem.
2. Voor elk probleem:
    - 2.1 Bepaal uit de bewaarde quotiënten de belief-waarde voor het probleem

4.1.3 Het ordenen van de vervolgvragen

Het doel van het ordenen van vragen is de helpdeskmedewerker een lijst te presenteren met vragen die een discriminerende werking hebben op de gesorteerde problemen. Zulke vragen maken sommige problemen waarschijnlijker en andere minder waarschijnlijk.

We hebben twee methoden uitgewerkt om vast te stellen wat het discriminerend vermogen van een vraag is. De eerste methode bepaalt de discriminatie waarde van een vraag aan de hand van de frequenties waarmee antwoorden op de vraag gegeven worden. De tweede methode maakt weer gebruik van het theorema van Bayes. Beide methoden zijn eigenlijk slechts uitbreidingen van de methoden om problemen te ordenen.

4.1.3.1 Het ordenen van vragen op basis van verhoudingen

4.1.3.1.1 Theorie

Laten we kijken naar één probleem, b_i, en één vraag, v_j. Stel dat op vraag v_j twee antwoorden mogelijk zijn. Na het stellen van v_j kunnen we dan twee proposities verwachten: r_{j1} of r_{j2}. Als het eerste antwoord heel weinig voorkomt, terwijl het tweede antwoord veel vaker wordt gegeven, dan is de vraag discriminerend. Aan de hand van het antwoord op v_j kunnen we dan immers bepalen of b_i wel of niet waarschijnlijk is. Als de klant het eerste antwoord geeft, dan is het vrij onwaarschijnlijk dat het incident bij dit probleem thuishoort, als de klant het tweede antwoord geeft dan is dat een stuk waarschijnlijker.

Als beide antwoorden even vaak worden gegeven dan is v_j niet discriminerend voor b_i, omdat we dan niets kunnen zeggen over de waarschijnlijkheid van b_i. Het maakt niet uit welk antwoord de klant geeft, omdat beide antwoorden even waarschijnlijk zijn voor dit probleem.

De spreiding in de frequenties van de antwoorden die de klanten op een vraag gegeven hebben is dus interessant. Een grote spreiding in de frequenties wil zeggen dat een bepaald antwoord vaak gegeven is, terwijl andere antwoorden minder vaak gegeven werden. We gebruiken dit gegeven om de vragen te sorteren.

4.1.3.1.2 Algoritme

De totale discriminatie waarde van een vraag bepalen we als volgt:

1. Ga elk probleem af waarbij de vraag voorkomt
   - 1.1 Bereken de frequenties van de antwoorden.
   - 1.2 Bereken de standaarddeviatie van de frequenties
   - 1.3 Vermenigvuldig de standaarddeviatie met het belief in het probleem
2. De discriminatiewaarde is gelijk aan de som van deze producten

De frequentie waarmee antwoord a_k op vraag v_j wordt gegeven binnen de incidenten van b_i is $P(r_{jk}|b_i) = \frac{\#\text{incidenten van } b_i \text{ waar } \langle v_j, a_k \rangle \text{ voorkomt}}{\#\text{incidenten van } b_i \text{ waar } v_j \text{ wordt gesteld}}$. Eerst gaan we voor vraag v_j alle gegeven antwoorden n af en berekenen de frequentie waarmee elk antwoord voorkomt. Daarna berekenen we de standaarddeviatie van die frequenties.

De standaarddeviatie geeft namelijk aan hoe groot de spreiding in de frequenties is. Hoe groter de spreiding, hoe groter de standaarddeviatie en hoe groter het discriminerend vermogen van de vraag voor het probleem.

De formule om de standaarddeviatie van een reeks getallen te bepalen is: $\sqrt{\frac{\sum_{k=1}^{n}(\bar{x}-x_k)^2}{n}}$.

In dit geval bepalen we eerst het gemiddelde van de berekende frequenties. Dit is gelijk aan $1/n$, waarbij n het aantal antwoorden is. De som van de frequenties is immers 1. Elke keer dat de vraag gesteld wordt geeft de klant één van de mogelijke antwoorden. Als we het gemiddelde bepaald hebben trekken we dat van elke frequentie af, en nemen we het kwadraat. Deze kwadraten tellen we op en we delen de som door n. Tot slot nemen we de wortel van het quotiënt. Dit noemen we de discriminatiewaarde $D(v_j|b_i)$ van vraag j voor probleem b_i. De discriminatiewaarde geeft aan hoe discriminerend een bepaalde vraag is voor een bepaald probleem.

Om de totale discriminatiewaarde van een vraag te bepalen berekenen we de discriminatiewaarde van de vraag voor elk probleem en vermenigvuldigen we deze waarde met het belief in het probleem. De som van deze producten is de totale discriminatiewaarde voor de vraag: $\mathrm{T}(v_j)=\sum Bel(b_i)\cdot D(v_j|b_i)$.

$\mathrm{T}(v_j)$ geeft dan aan hoe interessant het is om v_j te stellen. We kunnen de vragen nu sorteren op $\mathrm{T}(v_j)$ en presenteren aan de helpdeskmedewerker.

4.1.3.2 Het ordenen van vragen met het theorema van Bayes

4.1.3.2.1 Theorie

De ondersteuning bij het analyseren en classificeren van incidenten heeft tot doel te laten zien welke problemen het meest waarschijnlijk zijn.

Na de eerste ordening die in de problemen is aangebracht op basis van de keywords gaat de helpdeskmedewerker vragen stellen om beter onderscheid tussen de verschillende mogelijke problemen te kunnen maken.

Stel nu dat de helpdeskmedewerker een willekeurige vraag stelt. De klant geeft antwoord op die vraag en de problemen kunnen opnieuw geordend worden. Hoogstwaarschijnlijk is de ordening nu veranderd. Sommige problemen zijn waarschijnlijker geworden, andere minder

waarschijnlijk. Bij het ordenen van de vragen willen we die vragen bovenaan de lijst zetten die een zo groot mogelijk effect hebben op de ordening.

Om te bepalen welke vragen dat zijn gebruiken we een brute aanpak. We stellen stuk voor stuk alle vragen en geven daarop alle mogelijke antwoorden. Bij elke vraag en antwoord combinatie berekenen we de nieuwe standaarddeviatie in de geordende problemen. Hoe groter de standaarddeviatie hoe beter de scheiding is tussen de mogelijke problemen en de veel minder waarschijnlijke problemen. Voor elke vraag wordt het gemiddelde bepaald van de standaarddeviaties die we krijgen op de antwoorden. De vraag met de hoogste gemiddelde standaarddeviatie komt bovenaan in de lijst te staan.

Probleem 1	0,1	0,2	0,5	0,75	0,9	0,4
Probleem 2	0,1	0,1	0,15	0,2	0,035	0,4
Probleem 3	0,1	0,1	0,1	0,01	0,025	0,05
Probleem 4	0,1	0,1	0,1	0,01	0,025	0,05
Probleem 5	0,1	0,1	0,1	0,01	0,015	0,05
Probleem 6	0,1	0,1	0,01	0,01	0	0,05
Probleem 7	0,1	0,1	0,01	0,01	0	0
Probleem 8	0,1	0,1	0,01	0	0	0
Probleem 9	0,1	0,05	0,01	0	0	0
Probleem 10	0,1	0,05	0,01	0	0	0
Standaarddeviatie	0,0000	0,0408	0,1502	0,2364	0,2814	0,1599

4.1.3.2.2 Algoritme

Het algoritme dat we gebruiken om de vragen te sorteren bestaat uit de volgende stappen

1. Ga alle vragen bijlangs

    1.1 Bepaal de discriminatie waarde van elke vraag

        1.1.1 Ga alle antwoorden op de vraag bijlangs.

            1.1.1.1 Voeg vraag + antwoord toe aan het incident

            1.1.1.2 Sorteer de problemen

        1.1.1.3 Bereken de standaarddeviatie

        1.2.1 Tel alle berekende standaarddeviaties bij elkaar op

        1.2.2 Deel door het aantal antwoorden

4.1.4 C4.5

C4.5 is een classificatie-algoritme dat ontwikkeld is door J.R. Quinlan.

Als invoer neemt het een aantal voorbeelden. Elk voorbeeld behoort tot een bepaalde klasse en wordt beschreven door een vast aantal attributen.

Op basis van deze voorbeelden wordt een beslissingsboom gebouwd, aan de hand waarvan een nieuw voorbeeld geclassificeerd kan worden. Het algoritme staat uitgebreid beschreven in: C4.5: Programs for machine learning (Quinlan, 1993).

De beslissingsboom wordt op de volgende manier ontwikkeld. We beginnen met een verzameling voorbeelden T , die elk tot één bepaalde klasse uit $\{C_1, C_2 \ldots, C_k\}$ behoren.

Met behulp van informatie theorie kunnen we uitrekenen hoeveel informatie (die we meten in bits) er nodig is om te bepalen tot welke klasse een voorbeeld behoort. Dit doen we door te kijken hoeveel voorbeelden van elke klasse er in de verzameling zitten. De berekende hoeveelheid informatie wordt $\inf(T)$ genoemd.

Door een vraag te stellen kunnen we de verzameling voorbeelden in twee of meer deelverzamelingen verdelen. Opnieuw geldt dat we uit kunnen rekenen hoeveel informatie er nog nodig is om een voorbeeld toe te kennen aan de juiste klasse. Dit kan met de volgende formule:

$$\inf_X(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \cdot \inf(T_i)$$

In de formule wordt voor elke deelverzameling bepaald hoeveel informatie er nodig is om de voorbeelden te classificeren, en die informatie wordt geschaald door te vermenigvuldigen met de verhouding tussen het aantal voorbeeld in de deelverzameling en het aantal voorbeelden in de oorspronkelijke verzameling. In de formule staat X voor de vraag en is n het aantal mogelijke antwoorden op de vraag.

Als we nu $\inf(T) - \inf_X(T)$ uitrekenen dan hebben we de *gain* van de vraag bepaald. Stel dat we vóór het stellen van de vraag nog 0.9 bits informatie nodig hadden, maar dat we na het stellen van de vraag nog maar 0.3 bits nodig hebben. De gain is dan 0.6. Hoe groter de gain van een vraag hoe beter het is. Des te groter de gain des te minder informatie we nog moeten verzamelen om de classificatie goed uit te voeren.

Verschillende vragen hebben een verschillende gain voor verschillende verzamelingen. Door nu voor elke verzameling, te beginnen met de verzameling van alle voorbeelden, te bepalen welke vraag de grootste gain heeft kunnen we een boom maken.

Het algoritme daarvoor ziet er zo uit:

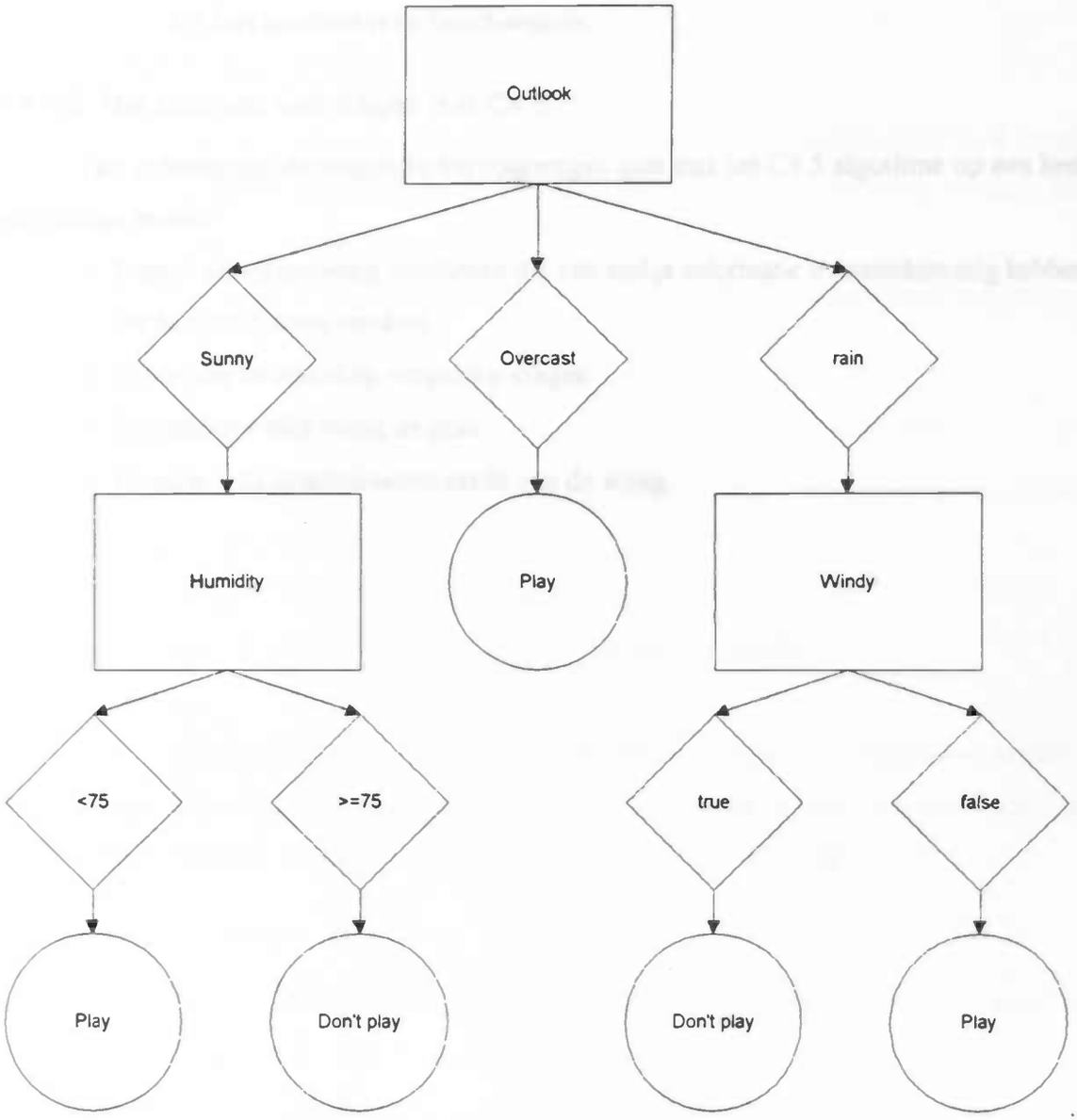
1. Voor elke (deel) verzameling:
    1.1. Bepaal de vraag met de grootste gain
    1.2. Verdeel de voorbeelden in deelverzamelingen op basis van hun antwoord op de vraag.
    1.3. Begin weer bij stap 1.

Een voorbeeld van een beslissingsboom zien we hieronder. Met behulp van deze boom kunnen we bepalen of het wel (Play) of geen (Don't play) goed weer is om te golfen.

De vraag die voor de verzameling van alle voorbeelden de grootste gain heeft is de vraag naar de Outlook. Op die vraag zijn drie antwoorden mogelijk: Sunny, Overcast en Rain. De voorbeelden die voor het attribuut Outlook de waarde Sunny hebben komen in één verzameling, die met de waarde Overcast in een tweede en de voorbeelden met de waarde Rain in de laatste.

De verzameling problemen die voor het attribuut Outlook de waarde Overcast hebben bevat problemen van één klasse (namelijk: Play) en hoeft dus verder niet opgesplitst te worden. Als de Outlook van een voorbeeld Overcast is dan kan er golf gespeeld worden.

De deelverzamelingen Sunny en Rain bevatten allebei nog voorbeelden Play en Don't Play. Door nu voor elke verzameling te berekenen welke vraag de voorbeelden het beste verdeeld in Play en Don't play oftewel: welke vraag de grootste gain heeft, kunnen we de rest van de boom afmaken.



4.1.4.1 Het ordenen van problemen met C4.5

Het ordenen van de problemen op basis van keywords met behulp van de C4.5 methode gaat als volgt.

1. Maak een lege verzameling
2. Ga elk keyword van het te classificeren incident bijlangs
   2.1 Voeg alle incidenten die verbonden zijn met het keyword toe aan de verzameling
3. Bepaal de beliefwaarde voor elk probleem
   3.1 Tel hoeveel incidenten uit de verzameling thuishoren bij het probleem
   3.2 Deel dat aantal door het totale aantal incidenten in de verzameling

3.3 Het quotiënt is de belief-waarde

4.1.4.2 Het ordenen van vragen met C4.5

Het ordenen van de mogelijke vervolgvragen gaat met het C4.5 algoritme op een heel eenvoudige manier.

1. Bepaal de verzameling incidenten die een stukje informatie overeenkomstig hebben met het te classificeren incident.
2. Bepaal de verzameling mogelijke vragen.
3. Bepaal voor elke vraag de gain.
4. De gain is de discriminatiewaarde van de vraag.

5. Systeemontwikkeling

In dit hoofdstuk wordt beschreven hoe het helpdeskondersteunende systeem is ontwikkeld. We beginnen met de requirements specification en voeren daarna de activiteiten uit die in het analysis process van OOSE moeten gebeuren.

Bij de ontwikkeling is gebruik gemaakt van VisualWorks als programmeeromgeving en van Objectivity als object georiënteerde database.

5.1 Requirements specification

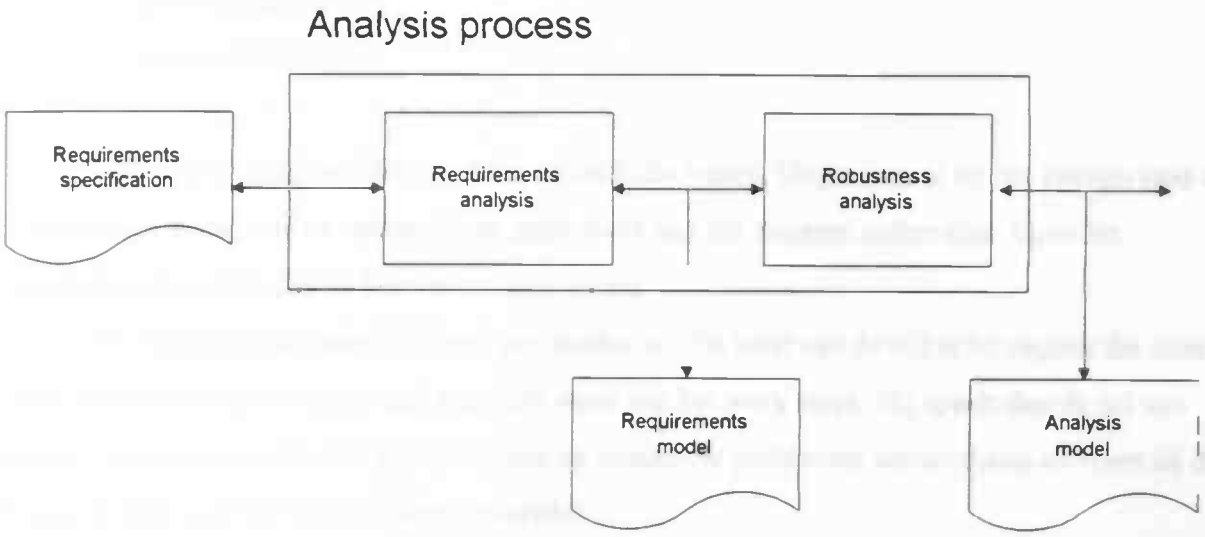
Het te ontwikkelen helpdeskondersteunende systeem moet voldoen aan de volgende eisen:

1. ITIL-conform werken
2. uitstekende zoekmogelijkheden hebben
3. goede ondersteuning bieden bij de classificatie van incidenten

Vooral de eerste eis legt beperkingen op aan het systeem dat ontwikkeld kan worden. ITIL-objecten zoals incident, probleem en fout en het traject van incident naar probleem naar fout naar wijziging bepalen voor een groot deel de structuur van het systeem.

5.2 Analysis process

Tijdens het analysis proces worden vanuit de requirements specification twee modellen ontwikkeld: het requirements model en het analysis model (zie figuur)



Het requirements model beschrijft welke functionaliteit het systeem moet bieden.

Het analysis model bevat een beschrijving van de structuur van het systeem.

5.2.1 Het requirements model

Het requirements model bestaat uit drie onderdelen:

1. het use case model
2. interface beschrijvingen
3. het problem domain model

Het use case model bevat actoren en use cases. Actoren zijn de mensen of andere systemen die zullen communiceren met het systeem. Use cases kunnen gezien worden als dialogen tussen een actor en het systeem.

Vaak maken de ontwikkelaars een programma dat niet de vereiste functionaliteit heeft, maar dat de gebruikers toont hoe de schermen van het systeem er gaan uitzien. Aan de hand van deze interface beschrijvingen kunnen de toekomstige gebruikers zien of de ontwikkelaars een goed beeld hebben van hun wensen.

Het problem domain model bevat definities van entiteiten die binnen het systeem van belang zijn.

5.2.1.1 Actoren

Voor een helpdeskondersteunend systeem kan een aantal actoren onderscheiden worden:

1. helper
2. probleembeheerder
3. receptontwikkelaar

Deze actoren komen niet zo maar uit de lucht vallen. Uitgangspunt bij het zoeken naar de actoren is de vraag wie de belangrijkste gebruikers van het systeem zullen zijn. Voor het helpdesksysteem zijn dat de helpdeskmedewerkers.

De helpdeskmedewerker speelt een dubbelrol. De helft van de tijd is hij degene die contact heeft met de klanten en ze zo snel mogelijk weer aan het werk helpt. Hij speelt dan de rol van helper. Op andere momenten zoekt hij naar de fouten die problemen veroorzaken en voert hij die fouten in. Dit is de rol van probleembeheerder.

De rol van receptontwikkelaar komt naar voren bij het uitschrijven van de use cases voor de probleembeheerder. Een recept is een handelingsvoorschrift dat onder andere gebruikt wordt bij het uitvoeren van oplossingen.

5.2.1.1.1 De helper

De helper is de persoon die de klant te woord staat wanneer hij een incident meldt. Hij is ook degene die het inbrengt in het systeem, naar een bekende oplossing zoekt en adviseert bij het uitvoeren van die oplossing.

De helper heeft de volgende taken:

1. het aannemen van calls
2. het invoeren van incidenten
3. incident verbinden met problemen
4. adviseren over een workaround of wijziging
5. workaround definiëren.

5.2.1.1.2 De probleembeheerder

De probleembeheerder zoekt naar de oorzaak van problemen die door de helpdesk geconstateerd worden. Als het lukt die oorzaak te vinden dan maakt hij één of meer wijzigingen die de gevonden fout uit de IT-infrastructuur verwijderen.

De taken van de probleembeheerder zijn:

1. het bestuderen van de problemen
2. het achterhalen van de onderliggende fouten
3. het creëren van wijzigingen.

Het zoeken naar de oorzaak van een probleem (de fout) wordt niet direct door het systeem ondersteund, hoewel de probleembeheerder natuurlijk naar opgeslagen problemen en wijzigingen kan kijken.

5.2.1.1.3 De receptontwikkelaar

Deze actor maakt een recept dat bestaat uit acties en vragen. Een recept is opgebouwd uit een aantal stappen. Door die stappen uit te voeren kan een bepaald resultaat bereikt worden, bijvoorbeeld het beantwoorden van een vraag of het doorvoeren van een wijziging.

De helper en de probleembeheerder worden vaak voor korte tijd receptontwikkelaar als ze een oplossing gevonden hebben die ze willen bewaren.

5.2.1.2 Use cases

Bij elke actor hoort een aantal use cases die door de actor uitgevoerd kunnen worden. Tijdens het zoeken naar use cases zijn twee punten van belang. Ten eerste moet iedere use case een duidelijk omschreven taak zijn. Ten tweede moet er rekening mee gehouden worden dat het systeem in de toekomst gewijzigd moet kunnen worden. Wijzigingen moeten zo eenvoudig mogelijk doorgevoerd kunnen worden. Tijdens de ontwikkeling van het analysis model komt dit punt weer ter sprake.

De makkelijkste manier om use cases te vinden is door te kijken welke taken een actor, de helper bijvoorbeeld, uitvoert met het systeem. Als een klant contact opneemt met de helpdesk wordt dat vastgelegd door middel van een call. Deze taak kan gezien worden als een use case, omdat het een taak is met een duidelijk begin en eind. Het is best mogelijk dat de wijze van vastlegging in de toekomst moet veranderen. Dat heeft waarschijnlijk tot gevolg dat ook de interface hiervoor moet veranderen. Door de beschrijving van de use case voor het vastleggen van een call apart te houden van de overige taken van de helpdeskmedewerker hoeft dan alleen deze eenvoudige use case gewijzigd te worden.

5.2.1.2.1 Helper

Het is de taak van de helper het incident van de klant te registreren, naar een bestaande oplossing te zoeken en eventueel te helpen bij het uitvoeren van die oplossing.

Voor de helper kunnen vier use cases onderscheiden worden.

1. Aannemen van een call
2. Analyse van een incident
3. Adviseren over het uitvoeren van een oplossing
4. Aanmaken van een workaround.

5.2.1.2.1.1 Aannemen van een call

Deze use case begint als de helpdeskmedewerker aangeeft dat hij een nieuwe call wil invoeren. De helpdesk vraagt de klant nu om zijn naam, om te bepalen of hij recht heeft op hulp. De klant geeft aan wat zijn probleem is. De helpdeskmedewerker voert vervolgens een beschrijving van het incident in.

1. De helpdeskmedewerker geeft aan dat hij een nieuwe call wil invoeren.
2. Het systeem maakt een nieuwe call aan en vult de standaardwaarden zoals de helpdeskmedewerker en datum en tijd in.
3. De helpdeskmedewerker vraagt de klant om zijn naam en controleert of hij recht heeft op hulp (extend use case: Persoon zoeken). Indien de klant geen recht heeft op hulp wordt de use case afgesloten.
4. Het systeem maakt een nieuw incident aan, initialiseert het, verbindt het met de call en vult de helpdeskmedewerker, datum en tijd en standaardwaarden voor melder en gebruiker (de klant van de call) in.
5. De klant vertelt wat zijn probleem is.
6. De helpdeskmedewerker kent het incident een aantal keywords toe (extend use case: Toekennen van keywords) en vult de omschrijving van het incident in.

De use case is nu afgelopen. In de meeste gevallen zal de helpdeskmedewerker onmiddellijk doorgaan met de use case 'Analyse van een incident'

5.2.1.2.1.2 Analyse van een incident

De analyse van een nieuw incident wordt uitgevoerd nadat de keywords en omschrijving zijn toegekend. De helpdeskmedewerker probeert nu een probleem te zoeken om het incident mee te verbinden. Het systeem biedt twee manieren om deze taak te ondersteunen.

Bij de eerste methode analyseert het systeem de bekende informatie over het incident en de bekende problemen en presenteert een lijst met problemen die gesorteerd is op de waarschijnlijkheid van het probleem en een lijst vragen die gesorteerd is op discriminerend vermogen van de vraag.

De andere methode is geschikt voor de ervaren helpdeskmedewerker. Hij zoekt zelf de vragen op die hij aan de klant wil stellen en probeert op die manier tot een probleem te komen.

Als de helpdeskmedewerker een probleem kan vinden waar het incident bij lijkt te passen dan koppelt hij het incident hier aan. Lukt het niet een probleem te vinden dan wordt een nieuw probleem gedefinieerd. Het probleem moet dan aan een persoon of groep worden toegewezen. Dit kan bijvoorbeeld een specialist zijn, of een externe organisatie in geval van een hardware storing.

5.2.1.2.1.3 Analyse van een incident met uitgebreide ondersteuning

Deze use case start nadat de helpdeskmedewerker een nieuw incident heeft aangemaakt en de keywords en omschrijving heeft ingevuld.

1. Het systeem bepaalt aan de hand van de keywords en de omschrijving van het incident en de antwoorden van de klant op eventueel al gestelde vragen, de verzameling mogelijke problemen.
2. De helpdeskmedewerker kiest een vraag uit de lijst en stelt de vraag aan de klant.
3. Eventueel ondersteunt de helpdeskmedewerker de klant bij het bepalen van het antwoord op de vraag.
4. De helpdeskmedewerker voert het antwoord op de vraag in.
5. Het systeem heeft nu extra informatie over het incident van de klant. Op basis van die informatie worden bepaalde problemen minder waarschijnlijk terwijl andere waarschijnlijker worden.

Als er nu slechts enkele (of zelfs maar één) problemen overblijven dan kan helpdeskmedewerker bekijken of hij het incident aan één van deze problemen wil toekennen. Zo niet, dan moet er een nieuw probleem gedefinieerd worden. Zijn er nog te veel mogelijke problemen dan gaat de use case verder bij stap 1.

5.2.1.2.1.4 Analyse van een incident zonder ondersteuning

De ervaren helpdeskmedewerkers weten vaak al wat het probleem is van een klant voordat deze uitgesproken is. Ze proberen hun hypothese te bevestigen door een aantal vragen te stellen aan de klant.

1. Met behulp van de keywords ontsluit de helper de verzameling vragen die in de loop der tijd zijn ingevoerd.
2. Hij kiest één van de vragen of definieert een nieuwe vraag (extend use case: Vraag/actie maken) en stelt deze aan de klant.
3. Eventueel ondersteunt de helper de klant bij het bepalen van het antwoord op de vraag.

4. De helper voert het antwoord op de vraag in.
5. Met deze nieuwe informatie (en eventueel met de toegekende keywords en omschrijving) bepaalt het systeem een lijst van mogelijke problemen.
6. Zolang zijn hypothese niet ontkracht wordt door een antwoord van de klant gaat de helper weer verder met stap 1.
7. Als de hypothese niet langer houdbaar is bekijkt de helper de problemen die het systeem heeft bepaald aan de hand van de vragen en antwoorden.
8. Eventueel kan nu de use case die hierboven wordt beschreven gestart worden.

5.2.1.2.1.5 Adviseren over het uitvoeren van een oplossing

Als voor het probleem waaraan een incident is toegekend een wijziging bekend is dan kan die uitgevoerd worden. Vaak is er geen wijziging bekend maar heeft een andere helper een workaround gecreëerd voor één van de incidenten die verbonden zijn met het probleem. De helper kan zo'n workaround dan ook voor het huidige incident uitvoeren.

De helper bepaalt in overleg met de klant welke van de beschikbare oplossingen (wijzigingen en workarounds) geschikt is. Welke oplossing er gekozen wordt hangt onder meer af van de moeilijkheid van de oplossing, het kennisniveau en de vaardigheid van de klant en de mogelijke gevolgen van de oplossing. De helpdeskmedewerker kan de klant helpen bij het uitvoeren van de oplossing.

5.2.1.2.1.6 Aanmaken van een workaround

Tijdens het analyseren van een incident kan de helper een tijdelijke oplossing ontdekken die de klant weer snel op weg helpt. Zo'n oplossing noemen we een workaround. Als de helper denkt dat een dergelijk incident nog wel eens gemeld zal gaan worden dan kan hij besluiten de workaround op te slaan bij het incident. Dit gaat als volgt in zijn werk.

1. De helper geeft aan dat hij een workaround wil toekennen aan het huidige incident.
2. Het systeem biedt de mogelijkheid om een bestaande workaround te zoeken of een nieuwe te definiëren.
3. De helper geeft aan dat hij een nieuwe workaround wil maken.
4. Het systeem presenteert de interface van een workaround.

5. De helper voert een beschrijving van de workaround in, geeft aan wat de gevolgen van het uitvoeren van de workaround zijn en kent tot slot een aantal keywords toe aan de workaround.
6. Eventueel kan de helper nu een recept opzoeken en het aan de workaround toekennen. In de meeste gevallen zal zo'n recept nog niet bestaan. De helper verandert dan in een receptontwikkelaar en definieert het recept, waarna hij weer helper wordt. Het recept wordt nu toegekend aan de workaround.
7. De helper slaat de workaround op.

5.2.1.2.2 Probleembeheerder

De helper legt calls en incidenten vast en verbindt incidenten met problemen. Problemen worden veroorzaakt door fouten in de IT-infrastructuur. Het is de taak van de probleembeheerder om die onderliggende fout op te sporen en een wijziging te ontwikkelen die de fout verwijdert. Het zoeken naar de oorzaak van een probleem wordt niet door het helpdesksysteem ondersteund.

Een groot deel van het werk waarbij de probleembeheerder van het systeem gebruik maakt van het helpdesksysteem bestaat uit zoeken. De functionaliteit die het systeem biedt om te zoeken wordt verderop beschreven door middel van een aantal extend use cases.

Voor de probleembeheerder kunnen we twee use cases onderscheiden:

1. Invoeren van een fout
2. Aanmaken van een wijziging

5.2.1.2.2.1 Fout invoeren

Deze use case wordt door de probleemoplosser uitgevoerd nadat hij een fout in de IT-architectuur gevonden heeft.

1. Eerst beschrijft hij de fout door middel van een tekstveld.
2. Dan kent hij de fout een aantal keywords toe (extend use case: Toekennen van keywords).
3. Vervolgens verbindt hij de fout met één of meer problemen (extend use case: Probleem zoeken).

5.2.1.2.2.2 Het aanmaken van een wijziging

Een fout in de infrastructuur moet worden verholpen door een wijziging.

1. De probleembeheerder geeft aan dat hij een wijziging wil toekennen aan de huidige fout.
2. Het systeem biedt de mogelijkheid om een bestaande wijziging te zoeken of een nieuwe te definiëren.
3. De probleembeheerder geeft aan dat hij een nieuwe wijziging wil maken.
4. Het systeem presenteert de interface van een wijziging.
5. De probleembeheerder voert een beschrijving van de wijziging in, geeft aan wat de gevolgen van het uitvoeren van de wijziging zijn en kent tot slot een aantal keywords toe aan de wijziging.
6. Eventueel kan de probleembeheerder nu een recept opzoeken en het aan de wijziging toekennen. In de meeste gevallen zal zo'n recept nog niet bestaan. De probleembeheerder verandert dan in een receptontwikkelaar en definieert het recept, waarna hij weer probleembeheerder wordt. Het recept wordt nu toegekend aan de wijziging.
7. De probleembeheerder slaat de wijziging op.

5.2.1.2.3 De receptontwikkelaar

Als voor een bepaald probleem een snelle oplossing bestaat dan kan de helper een workaround definiëren. Het belangrijkste onderdeel van een workaround is het recept. Het recept beschrijft wat de klant moet doen om de workaround uit te voeren. Maar de workaround is niet de enige plaats waar een recept van pas kan komen. Ook bij het doorvoeren van een wijziging kan een recept erg nuttig zijn. Daarnaast kunnen recepten gebruikt worden voor vragen en acties. Een recept kan dan de stappen beschrijven die de klant moet nemen om een vraag te beantwoorden of een actie uit te voeren.

Recepten bestaan uit vragen en acties. Het is dus mogelijk dat er een recursieve structuur ontstaat. Een recept voor een workaround kan bijvoorbeeld bestaan uit drie acties. Voor het uitvoeren van elke actie kan dan weer een recept bestaan.

De enige use case voor de receptontwikkelaar is:

- het ontwikkelen van een recept.

5.2.1.2.3.1 Het ontwikkelen van een recept

1. De receptontwikkelaar selecteert de eerste stap van het recept. Een stap is een actie of een vraag.

2. Na elke stap is een aantal antwoorden mogelijk. De mogelijke antwoorden voor een actie zijn 'gelukt' en 'mislukt'. De mogelijke antwoorden op een vraag zijn de antwoorden die voor die vraag gedefinieerd zijn. De receptontwikkelaar selecteert een antwoord.
3. Vervolgens geeft hij aan wat de volgende stap zal zijn als het geselecteerde antwoord wordt gegeven.
4. Als de stappen van het recept bekend zijn kent de receptontwikkelaar een aantal keywords toe aan het recept, waarna het wordt opgeslagen.

Het is mogelijk dat een recept de vorm van een boom krijgt, maar meestal zal het een eenvoudige reeks handelingen zijn.

De receptontwikkelaar zal regelmatig nieuwe vragen en acties willen definiëren. Dit wordt beschreven met extend use case, omdat ook de helpdeskmedewerker van deze functionaliteit gebruik maakt.

5.2.1.2.4 Extend use cases

Een extend use case beschrijft een taak die in verschillende use cases door verschillende actoren wordt uitgevoerd. Een voorbeeld van een extend use case is het zoeken van een persoon. Het zoeken van een persoon (bijvoorbeeld een klant) is een taak die in verschillende use cases uitgevoerd moet worden. Door die taak als een extend use case te beschouwen hoeft de taak maar één keer beschreven te worden. Binnen een use case kan dan verwezen worden naar de extend use case

5.2.1.2.4.1 Zoeken

Zowel de helper als de probleembeheerder moet regelmatig een incident, probleem, fout, wijziging of workaround opzoeken in het systeem.. Het is daarom noodzakelijk dat eenvoudig en snel kan gebeuren. In de volgende paragrafen gebruik ik steeds het voorbeeld van iemand die een incident wil opzoeken. Het systeem biedt twee manieren om een incident op te sporen.

1. via de keywords
2. via de omschrijving

5.2.1.2.4.2 Zoeken met behulp van keywords

Keywords worden geassocieerd met incidenten, problemen, fouten, wijzigingen en workarounds. Stel nu dat een helper een incident zoekt. Dat is het moment waarop de use case begint.

1. Het systeem presenteert nu alle bekende incidenten, de keywords die daarin voorkomen, en een lijstje met de keywords van het incident.
2. De gebruiker selecteert nu een keyword en geeft het systeem opdracht te zoeken.
3. Het systeem reageert met het inkorten van de lijst incidenten. Alleen incidenten waarvan tenminste één keyword geselecteerd is blijven over. Hoe meer keywords van een incident geselecteerd zijn hoe hoger het staat in de lijst. Ook het lijstje met beschikbare keywords wordt kleiner.

De gebruiker kan ook een keyword deselecteren. De lijst incidenten en lijst keywords worden dan weer langer. Door keywords te selecteren en te deselecteren kan de gebruiker de lijst incidenten steeds verder beperken. Uiteindelijk blijven er dan maar een paar incidenten over die stuk voor stuk bekeken kunnen worden.

5.2.1.2.4.3 Zoeken met behulp van woorden uit de omschrijving

De omschrijving van een incident is een tekstveld. Het systeem houdt automatisch een index bij van interessante woorden en weet dus bijvoorbeeld in welke incidenten het woord 'opstarten' voorkomt.

Deze index kan gebruikt worden om een incident terug te vinden. Het systeem presenteert een lijst van alle bekende incidenten, een lijst van alle woorden die in de omschrijvingen van die incidenten voorkomen en een lijst van de woorden die in de omschrijving van het incident voorkomen. Door nu één of meer woorden te selecteren kan de gebruiker de gepresenteerde lijst incidenten inkorten, net als bij het zoeken via keywords.

5.2.1.2.4.4 Twee manieren tegelijkertijd

Het moet natuurlijk ook mogelijk zijn de twee beschreven manieren tegelijkertijd te gebruiken. Een helpdeskmedewerker kan dan bijvoorbeeld zoeken naar een incident met het keyword 'Word' en de woorden 'tabellen' en 'converteren' in de omschrijving.

5.2.1.2.4.5 Persoon zoeken

Bij het invoeren van calls en incidenten moet de helper regelmatig een persoon opzoeken. Hiervoor zijn twee manieren beschikbaar.

5.2.1.2.4.6 Persoon zoeken door gedeeltelijke beschrijving

De gegevens van de beller, melder of gebruiker worden in een aantal invulvelden op het scherm weergegeven.

1. De helper vult één of meer van die velden in en geeft het systeem opdracht te zoeken.
2. Het systeem presenteert een lijst van personen die aan de gedeeltelijke beschrijving voldoen.
3. De helper kan de lijst op verschillende manieren sorteren en zo de juiste persoon opzoeken of het systeem opdracht de lijst van alle bekende personen te laten zien. Dan komen we bij de volgende use case.

5.2.1.2.4.7 Persoon zoeken uit een lijst

1. Het systeem presenteert de lijst van alle bekende personen.
2. De gebruiker laat de lijst sorteren op het gewenste veld (stel: achternaam).
3. De helper typt nu een deel van de gezochte achternaam in.
4. Het systeem scrollt naar de getypte achternaam.

5.2.1.2.4.8 Toekennen van keywords

Het toekennen van keywords is een use case die veel wordt gebruikt, aangezien een groot aantal domein objecten van keywords voorzien kunnen worden.

1. Het systeem laat drie lijsten naast elkaar zien. In de eerste twee lijsten verschijnen straks delen van het keywords netwerk, in de derde lijst de toegekende keywords.
2. Afhankelijk van het type object waaraan de keywords toegekend worden presenteert het systeem in de eerste lijst een aantal ingangen tot het netwerk van keywords. Voor een incident zouden dat bijvoorbeeld applicaties, netwerk en hardware kunnen zijn.
3. De helper klikt op één van de ingangen
4. Het systeem presenteert de daaronderliggende keywords in de eerste lijst.
5. Door nu op keyword A te klikken verschijnen de keywords die met A verbonden zijn in de tweede lijst.
6. Door met de rechter muisknop op een keyword te klikken kan de helper dit keyword selecteren.

7. De gebruiker kan de tweede lijst in de eerste lijst laten afbeelden. Door nu in de eerste lijst weer een keyword te selecteren verschijnen in de twee lijst de daarmee verbonden keywords. Op deze manier kan de helper door het netwerk van keywords navigeren.

5.2.1.2.4.9 Vraag maken

Deze use case start als een actor het systeem laat weten dat hij een nieuwe vraag wil invoeren. Nadat het systeem een nieuwe vraag heeft aangemaakt en een aantal velden alvast heeft ingevuld is het de beurt aan de actor.

Hij voert eerst de tekst van de vraag in ('Wat staat er onderin het scherm?'). Daarna geeft hij aan welke antwoorden er mogelijk zijn op de vraag. Hij kan hierbij kiezen uit een aantal typen (getal, tijd, datum, ja, nee).

Als de vraag niet door elke klant direct te beantwoorden is kan de actor een recept definiëren voor de vraag. Als hij dat gedaan heeft kan de vraag toegevoegd worden aan het systeem.

5.2.1.2.4.10 Actie maken

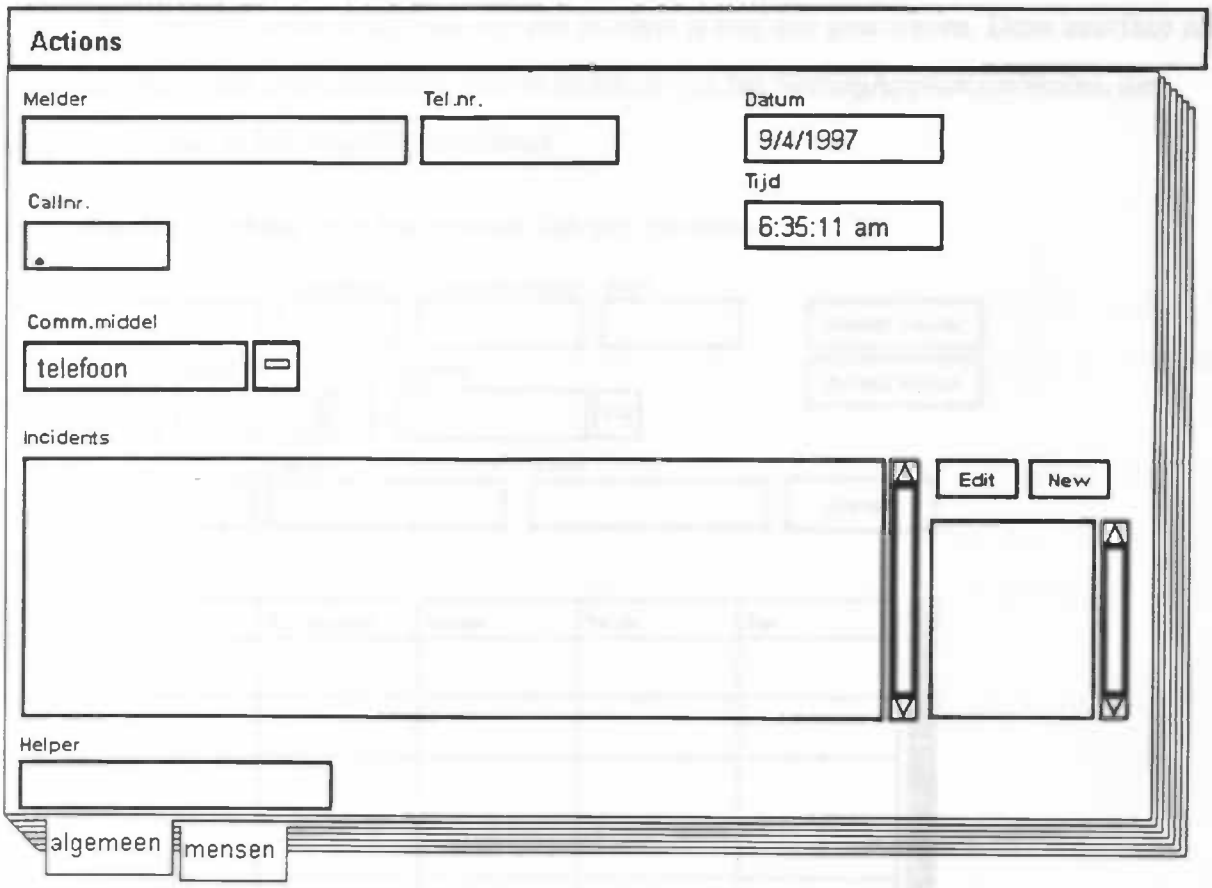
Het maken van een actie gaat als volgt. De actor geeft aan dat hij een nieuwe actie wil maken. Het systeem creëert een actie en vult een aantal velden alvast in.

De actor omschrijft dan de actie in een tekstveld, geeft aan wie de actie uit moet gaan voeren en kent de actie een aantal keywords toe. De keywords maken het mogelijk een actie snel weer terug te vinden. Eventueel kan de actor nu een recept definiëren voor het uitvoeren van de actie.

5.2.1.3 Interface beschrijvingen

Het afbeelden van alle interface beschrijvingen die voor het helpdesksysteem zijn gemaakt neemt teveel ruimte in beslag en daarom geven we hieronder alleen de interfaces voor de use cases die in het analysis model verder worden uitgewerkt.

5.2.1.3.1 De interface voor het aannemen van een call



Dit is de interface die gebruikt wordt om een call te registreren. Linksboven komt de melder van de call te staan, met daarnaast zijn of haar telefoonnummer.

Onder de melder staan het nummer van de call en het communicatiemiddel waarmee de call is gemeld.

Onder het kopje incidenten staat een lijst met incidenten die al eerder zijn gemeld door de melder van deze call, met daarnaast twee knoppen om een oud incident te wijzigen (edit) of een nieuw incident aan te maken. Onder deze knoppen verschijnen de keywords van het incident dat in de lijst wordt geselecteerd.

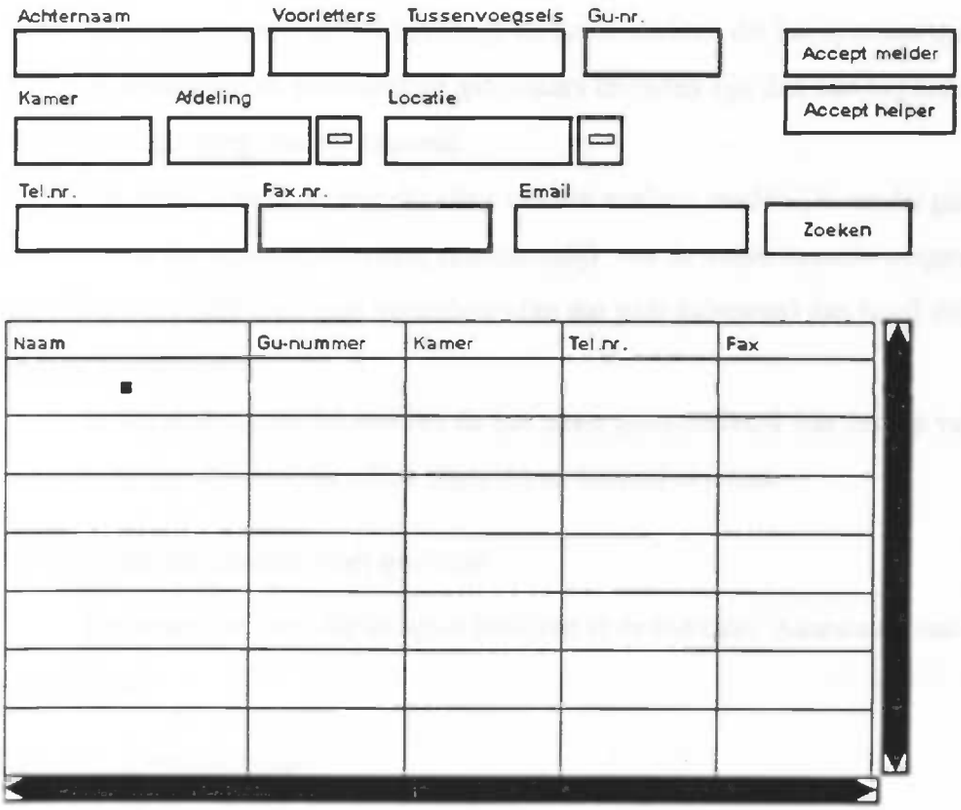
Helemaal onderaan staat de naam van de helpdeskmedewerker die de call aanneemt vermeldt.

Deze interface is een zogenaamde tabbladen-interface. Door op het tabblad mensen te klikken verschijnt er een andere interface en kan de gebruiker de melder opzoeken in een lijst. De interface om personen te zoeken wordt verderop behandeld.

5.2.1.3.2 De interface voor de analyse van een incident

De interface voor de analyse van een incident is nog niet geschreven. Deze interface zal voor een groot deel overeenkomen met de interface van het SortingApplicationModel, die beschreven staat in het volgende hoofdstuk.

5.2.1.3.3 De interface voor het zoeken van een persoon



Met deze interface kan gezocht worden naar een persoon. Als de interface wordt geopend verschijnt in de onderste helft een lijst van alle bekende personen. Door de invulvelden bovenin het scherm van waarden te voorzien en vervolgens op de knop zoeken te drukken wordt de lijst ingeperkt tot alleen die personen die aan de opgegeven criteria voldoen.

Als de gebruiker boven in één van de kolommen klikt wordt de lijst gesorteerd op dat attribuut.

Door op één van de knoppen rechts bovenin te drukken kan de geselecteerde persoon als helper of als melder geaccepteerd worden. Deze knoppen maken normaal gesproken geen deel uit van de interface om een persoon te zoeken, maar zijn toegevoegd omdat het bovenstaande plaatje de interface is voor het zoeken van een persoon in de use case voor het aannemen van een call.

5.2.1.4 Problem domain

Het problem domain bevat beschrijvingen van de entiteiten (probleem, fout, etc.) die in het systeem gebruikt worden. Het problem domain vormt het grootste gedeelte van het volgende hoofdstuk.

5.2.2 Analysis model

Het requirements model beschrijft de functionaliteit die het systeem moet gaan bieden. Als het model klaar is en de toekomstige gebruikers tevreden zijn dan kan begonnen worden met de verdere structurering van het systeem.

De eerste stap is de ontwikkeling van het analysis model. Dit model gaat de logische structuur van het systeem bevatten, onafhankelijk van de implementatie-omgeving. Als die omgeving in de toekomst gaat veranderen (en dat gaat gebeuren) dan hoeft dit model niet aangepast te worden.

In het analysis model worden de use cases gemodelleerd met behulp van drie typen objecten: entity objecten, interface objecten en control objecten.

5.2.2.1 Het aannemen van een call

De eerste use case die we gaan bekijken is de use case 'Aannemen van een call' voor de actor Helper.

5.2.2.1.1 Entity objecten

Het is vaak zo dat de objecten die beschreven worden in het domain model in het analysis model terugkomen als entity objecten. Één methode om de entity objecten binnen een use case te vinden is daarom te kijken welke objecten uit het domain model gebruikt worden in de use case. In dit geval hebben we te maken met de volgende entity objecten:

- Call
- Helper
- Melder
- Incident

5.2.2.1.2 Interface objecten

In de eerste stap van de use case maakt het systeem een nieuwe call aan en vult deze met bekende informatie. De gebruiker wil de call kunnen zien. Het systeem moet daarom een representatie van de call weergeven. Dit is de taak van een interface object.

De volgende stap is dat de gebruiker aangeeft wie de melder van de call is. Dit gebeurt door middel van een extend use case, die straks beschreven zal worden.

De gebruiker selecteert nu een oud incident of geeft opdracht een nieuw incident te definiëren. Dit incident moet afgebeeld worden en daar wordt een interface object voor gebruikt.

Voor deze use case zijn dus twee interface objecten nodig:

- call userinterface
- incident userinterface

5.2.2.1.3 Control objecten

Voor deze use case is het niet nodig control objecten te gebruiken. Het gaat hier om een eenvoudige administratieve use case. In use case 'Analyse van een incident' zal zo'n object wel noodzakelijk zijn.

5.2.2.2 Analyse van een incident

Deze use case is wat ingewikkelder dan de vorige, omdat het systeem nu zelf iets gaat berekenen, namelijk een lijst met mogelijke problemen.

5.2.2.2.1 Entity objecten

De entity objecten voor deze use case zijn vrij eenvoudig te vinden. Bij het bepalen van de waarschijnlijkheid van een probleem kijkt het systeem naar de gestelde vragen en de antwoorden daarop. Aan de hand daarvan wordt een lijst met interessante vervolgvragen berekend. De entity objecten zijn dus:

- problem
- vraag
- antwoord

5.2.2.2.2 Interface objecten

Om het zoeken te ondersteunen is een aparte interface nodig waarin de gestelde vragen en antwoorden, de meest zinvolle vervolgvragen en de mogelijke problemen worden weergegeven.

5.2.2.2.3 Control objecten

Voor deze use case is tenminste één control object nodig. Dit object moet, gegeven de gestelde vragen en de antwoorden daarop, berekenen welke vragen het best gesteld kunnen worden. Deze functionaliteit kan misschien nog het beste verdeeld worden over twee control objecten. Het eerste object bepaalt dan welke problemen het meest waarschijnlijk zijn. Het tweede control object berekent vervolgens aan de hand van die problemen wat de meest zinvolle vervolgvraag is. Hoe al die berekeningen precies verlopen is op dit moment nog niet belangrijk.

5.2.2.3 Het zoeken van een persoon

Het zoeken naar een persoon is een veel voorkomende deeltaak. Het is daarom belangrijk dat een actor snel een persoon kan vinden.

5.2.2.3.1 Entity objecten

De enige entity objecten die in deze use case een rol spelen zijn personen.

5.2.2.3.2 Interface objecten

De interface objecten zijn in deze use case het belangrijkst. De interface voor het zoeken van een persoon bestaat uit een interface voor het afbeelden en invoeren van een persoon en uit een aantal knoppen en een lijst.

5.2.2.3.3 Control objecten

Er is voor deze use case één control object nodig. Dit object kan een lijst personen sorteren op basis van een aantal door de gebruiker ingevoerde kenmerken. Daarnaast kan het object de lijst sorteren op één of meer velden.

6. Design

In dit hoofdstuk worden de entiteiten van het helpdesksysteem beschreven. Het hoofdstuk valt uiteen in twee delen. Het eerste deel bevat het problem domain van het helpdesksysteem. Het tweede deel is een uitgebreide beschrijving van de (control-) objecten die gebruikt worden bij het classificeren van incidenten.

6.1 Problem domain

In het problem domain model worden de entiteiten beschreven waarmee het systeem te maken heeft. Deze domein objecten worden onder meer gebruikt bij het definiëren van de use cases en tijdens het overleg met gebruikers.

Elk domain object wordt eerst kort beschreven. Daarna geven we aan met welke andere objecten het domain object geassocieerd kan zijn. Een incident kan bijvoorbeeld geassocieerd zijn met zijn een probleem. Tot slot beschrijven we de attributen van het object. Een attribuut van een persoon is bijvoorbeeld zijn achternaam.

Domein objecten die te maken hebben met personen

- Persoon
- Rol
    - Helper
    - Probleembeheerder
    - Eigenaar
    - Bedenker
    - Melder
    - Gebruiker

ITIL domein objecten

- Call
- Incident
- Problem
- Fout
- Workaround

- Wijziging

Analyse objecten

- Keyword
- Recept
- Vraag
- Antwoord
- Actie

6.1.1 Persoon

Een Persoon staat model voor een willekeurig persoon.

6.1.1.1 Persoon associaties

- Rollen

Een Persoon kan verschillende rollen op zich nemen. Zo kan iemand voor het systeem een helper, een klant en een melder zijn. Hij heeft dan drie rollen.

6.1.1.2 Persoon attributen

- Voorletters
- Tussenvoegsels
- Achternaam
- Telefoonnummer
- Faxnummer
- Emailadres

Deze attributen spreken voor zich.

6.1.2 Rol

Een Persoon kan verschillende rollen spelen. Iemand die normaal gesproken de rol van helpdeskmedewerker speelt kan op een zeker moment de rol van melder op zich nemen als hij zelf een incident meldt bij de helpdesk.

6.1.2.1 Rol associaties

- Persoon

Persoon

De persoon die de rol speelt.

6.1.3 Helper

Helper is één van de rollen die een Persoon kan spelen. Het is belangrijk om te weten wie een klant geholpen heeft en daarom wordt bij elke call en bij elk incident opgeslagen wie de helper was.

6.1.3.1 Helper associaties

- Call
- Incident

6.1.4 Probleembeheerder

Probleembeheerder is een andere rol die een Persoon op zich kan nemen.

6.1.4.1 Probleembeheerder associaties

- Probleem
- Fout

Probleem

De problemen waar de Probleembeheerder verantwoordelijk voor is.

Fout

De fouten die de Probleembeheerder beheert.

6.1.5 Bedenker

De rol van bedenker krijgt die persoon toegekend die een actie, workaround of Wijziging toevoegt aan het systeem.

6.1.5.1 Bedenker associaties

- actie
- workaround
- wijziging

Actie

Dit is de actie die de bedenker bedacht heeft.

Workaround

Dit is de workaround die de bedenker ingevoerd heeft.

Wijziging

Dit is de Wijziging die de bedenker ingevoerd heeft.

6.1.6 Eigenaar

Een incident heeft altijd een eigenaar die er verantwoordelijk voor is, totdat het incident afgesloten wordt.

6.1.6.1 Eigenaar associaties

- Incident

Incident

De incidenten waarvoor de eigenaar verantwoordelijk is.

6.1.7 Melder

Een melder is een persoon die een call of een incident meldt.

6.1.7.1 Melder associaties

- Call
- Incident

6.1.8 Gebruiker

De gebruiker is de eigenaar van de computer waarop het incident is geconstateerd. Meestal zijn de melder en de gebruiker van een incident gelijk, maar het kan voorkomen dat iemand een incident meldt voor iemand anders.

6.1.8.1 Gebruiker associaties

- incident

6.1.8.2 Gebruiker attributen

- Service level

6.1.9 Call

Als een klant contact opneemt met de helpdesk spreken we van een **call.** Een call gaat altijd over één incident. Elke call wordt zo spoedig mogelijk ingevoerd door de helpdeskmedewerker.

6.1.9.1 Call associaties

- Melder
- Helper
- Incident

Melder

Melder is de melder van de call. De melder wordt gekozen uit een lijst, die op verschillende manieren te doorzoeken moet zijn. Melder is een verplicht veld, dat wil zeggen dat een call met een melder verbonden moet zijn voordat het opgeslagen kan worden.

Helper

De medewerker die de call heeft aangenomen. Dit veld wordt automatisch ingevuld, maar kan wel gewijzigd worden. Dit veld is verplicht.

Incident

Het incident waar de call over gaat. Dit veld is verplicht.

6.1.9.2 Call attributen

- Callnummer
- Datum/tijd
- Communicatiemiddel
- Soort call
- Omschrijving

Callnummer

Het callnummer wordt pas toegekend op het moment dat de call wordt opgeslagen. Gaten zijn namelijk niet toegestaan.

Datum en tijd

Deze velden worden automatisch ingevuld, maar kunnen gewijzigd worden.

Communicatiemiddel

Dit attribuut kan één van de volgende waarden aannemen: telefoon (standaard), fax, post, email of persoonlijk.

Soort call

Dit attribuut geeft de helpdesk de mogelijkheid onderscheid te maken tussen verschillende soorten calls. Mogelijke waarden zijn: klacht, vraag, melding en incident (de standaardwaarde).

Omschrijving

In dit tekstveld kan de helpdeskmedewerker eventuele opmerkingen over de call kwijt. Dit tekstveld is niet verplicht. Het systeem moet kunnen zoeken naar woorden uit dit veld.

6.1.10 Incident

We spreken van een **incident** als een klant moeilijkheden ondervindt tijdens het werken met de IT-infrastructuur en daarvan melding maakt bij de helpdesk.

6.1.10.1 Incident associaties

- Call
- Keyword
- Melder
- Gebruiker
- Helper
- Configuration Item (CI)
- Problem
- Vragen en antwoorden
- Eigenaar
- Oplossing

Call

Het kan voorkomen dat de helpdesk meerdere calls krijgt over hetzelfde incident. Een melder kan bijvoorbeeld bellen om te vragen hoe ver de helpdesk is met de afhandeling van een eerder gemeld incident.

Keyword

Met behulp van keywords wordt een incident heel kort beschreven. Aan de hand de keywords kan een incident ook weer teruggevonden worden.

Melder

De melder van een incident is in de meeste gevallen ook degene die het probleem heeft (de gebruiker), maar dat hoeft niet altijd.

Gebruiker

De gebruiker is de persoon die in zijn werk gestoord wordt door een fout in de infrastructuur. De standaardwaarde voor deze associatie is de melder.

Helper

De helpdeskmedewerker die het incident aanmaakt. Dit is meestal degene die ook de call heeft ingebracht.

Configuration Item (CI)

Dit veld is niet verplicht maar kan ingevuld worden als het incident een hardwarestoring betreft. Alle apparatuur wordt beschreven in de configuratiedatabase.

Probleem

Elk incident wordt na de analysefase verbonden met één probleem. Meestal is dit een al bestaand probleem, maar de helpdeskmedewerker kan ook een nieuw probleem definiëren, als hij geen bestaand probleem kan vinden waaraan het incident gekoppeld kan worden.

Vragen en antwoorden

Tijdens het zoeken naar een probleem om het incident aan te verbinden kan de helpdeskmedewerker vragen stellen aan de klant. Die vragen en de antwoorden daarop worden verbonden met het incident en vormen zo de derde manier van incidentbeschrijving (naast keywords en omschrijving).

Eigenaar

De persoon of groep waaraan het incident wordt toegewezen. Het incident wordt standaard toegekend aan de medewerker die de call invoert.

Oplossing

Dit veld is een verwijzing naar een wijziging of een workaround

6.1.10.2 Incident attributen

- Incidentnummer
- Status
- Soort incident
- Omschrijving

- Datum/tijd van aanmaken
- Uiterste datum/tijd gereed
- Geplande datum afgehandeld
- Logboek

Incidentnummer

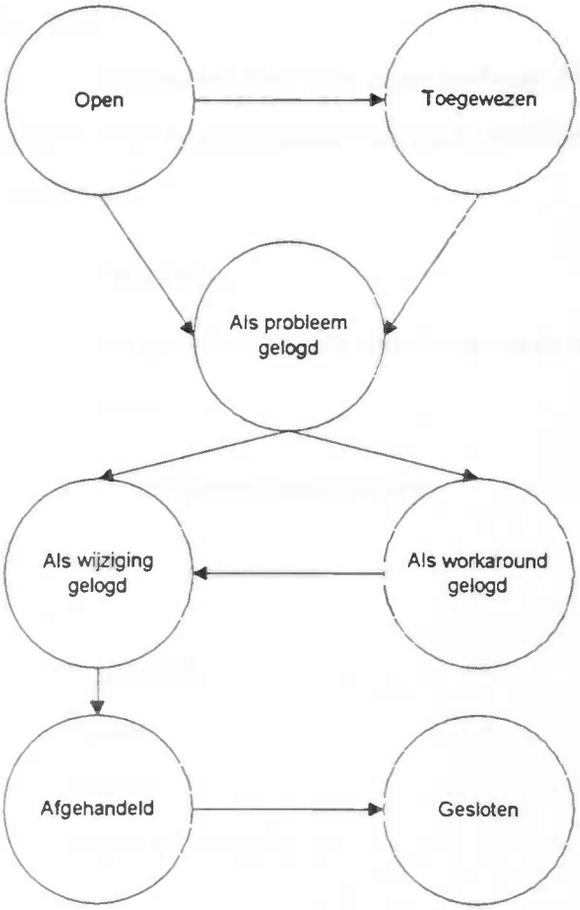
Het incidentnummer wordt toegekend als het incident wordt opgeslagen. Lege incidentnummers zijn verboden omdat elk incident te traceren moet zijn.

Status

De status van een incident geeft aan in welke fase van behandeling een incident verkeert. Mogelijke waarden voor dit veld zijn:

1. open (standaard)
2. toegewezen
3. als probleem gelogd
4. als wijziging gelogd
5. als workaround gelogd
6. afgehandeld
7. gesloten

De status van een incident is **'open'** zolang de analysefase nog niet is afgesloten. Als de status **'toegewezen'** is wil dat zeggen dat het incident aan iemand anders dan de invoerder in behandeling is gegeven. Dit kan bijvoorbeeld een pakketbeheerder zijn of een reparateur. Als een incident de status 'open' of 'toegewezen' heeft is het nog niet aan een probleem toegekend. **'Als probleem gelogd'** wil zeggen dat het incident is verbonden met een probleem waarvan de oorzaak nog niet gevonden is. Er is dan ook nog geen wijziging.
'Als wijziging gelogd' wil zeggen dat het incident verbonden is met een wijziging die (binnenkort) wordt uitgevoerd.
'Als workaround gelogd' wil zeggen dat er een workaround is uitgevoerd voor dit incident. Een incident krijgt de status **'Afgehandeld'** als er een wijziging is uitgevoerd die het probleem oplost. De status van een incident mag pas **'Gesloten'** worden als de klant op de hoogte is gesteld van de oplossing en tevreden is.



Soort incident

Incidenten kunnen in een aantal soorten verdeeld worden. De mogelijke waarden voor dit attribuut zijn nog onbekend.

Omschrijving

De waarde van dit attribuut is een omschrijving van het incident door de helpdeskmedewerker.

Datum/tijd van aanmaken

De datum en tijd waarop het incident werd aangemaakt. Deze gegevens worden automatisch ingevuld.

Uiterste datum/tijd gereed

Dit tijdstip wordt automatisch door het systeem bepaald.

Geplande datum afgehandeld

Deze datum kan eventueel worden ingevuld door de helpdeskmedewerker. Als een pakketbeheerder bijvoorbeeld heeft laten weten dat een wijziging op een bepaalde datum beschikbaar zal zijn dan kan die datum hier ingevuld worden.

Logboek

Elk incident heeft een eigen logboek. Het systeem houdt hierin bij wat er met het incident gebeurt. Ook helpdeskmedewerkers en probleembeheerders kunnen hier opmerkingen aan toevoegen.

6.1.11 Probleem

Een **probleem** is één van de manieren waarop een fout in de IT-infrastructuur zich kan manifesteren.

6.1.11.1 Probleem associaties

- Incident
- Fout
- Workaround
- Eigenaar
- Keyword
- Vraag + antwoord

Incident

Tijdens het analyseproces wordt een incident verbonden met een probleem. Als er geen probleem gevonden kan worden voor een incident dan wordt er een nieuw probleem gedefinieerd. Het incident wordt dan verbonden met het nieuwe probleem.
De eigenschappen van een probleem worden bepaald door de eigenschappen van de incidenten die ermee verbonden zijn.

Fout

Elk probleem verwijst naar één fout in de IT-infrastructuur.

Workaround

Aan een probleem kunnen meerdere workarounds gekoppeld zijn. Een workaround bestaat uit een reeks handelingen die er voor zorgen dat de klant verder kan met zijn werk. Het is geen echte oplossing voor het probleem. Zie verder bij workaround.

Eigenaar

De eigenaar van een probleem is degene die het gecreëerd heeft of toegewezen heeft gekregen.

Keyword

Een probleem heeft naast de keywordverdeling (zie bij attributen) van zijn incidenten ook eigen keywords.

6.1.11.2 Probleem attributen

- Omschrijving
- Keywordverdeling
- Vraag + antwoordverdeling
- Logboek

Omschrijving

In de omschrijving kunnen de helpdeskmedewerker en de probleembeheerder een tekst kwijt die het probleem omschrijft.

Keywordverdeling

De keywordverdeling wordt afgeleid uit de incidenten die verbonden zijn met een problem.

Vraag + antwoordverdeling

Een boomstructuur die per vraag aangeeft hoe vaak elk antwoord op die vraag gegeven is binnen de incidenten die verbonden zijn met het probleem.

Logboek

Hierin kunnen aantekeningen opgenomen worden. Het systeem houdt in het logboek bij wat er met het probleem gebeurt, bijvoorbeeld wanneer het is gemaakt, welke incidenten er wanneer aan toegekend zijn etc.

6.1.12 Fout

Een fout in de IT-infrastructuur veroorzaakt incidenten. Vaak kan een fout zich op meerdere manieren manifesteren. Die verschillende manieren noemen we **problemen**. Een fout kan dus verwijzen naar meerdere problemen.

6.1.12.1 Fout associaties

- Problem
- Wijzigingen
- Eigenaar
- Keywords

Problem

Dit zijn de verschillende manieren waarop een fout tot uiting kan komen.

Wijzigingen

Dit zijn de mogelijke wijzigingen die de fout in de infrastructuur opheffen.

Keywords

Aan de hand van keywords kan een fout opgezocht worden. De helpdeskmedewerker kan zelf keywords toekennen. Het is ook mogelijk de keywords van de fout af te leiden uit de keywords van de problemen.

6.1.12.2 Fout attributen

- Omschrijving
- Logboek

Omschrijving

Een omschrijving van de fout in woorden.

Logboek

Logboek van de fout.

6.1.13 Oplossing

Een **oplossing** verwijst naar een recept. Na het uitvoeren van dat recept kan de klant weer verder met zijn werk. Er zijn twee soorten oplossingen: wijzigingen en workarounds.

Een **wijziging** is een officiële wijziging in de IT-infrastructuur met als gevolg de verwijdering van een fout. Problemen en incidenten die veroorzaakt werden door de fout mogen daarna niet meer voorkomen.

Een **workaround** is een tijdelijke oplossing. De werkelijke oorzaak wordt niet aangepakt.

6.1.13.1 Oplossing associaties

- Keyword
- Eigenaar
- Incident
- Recept

Keyword

Een aantal keywords aan de hand waarvan een oplossing teruggezocht kan worden.

Eigenaar

Degene die de oplossing gecreëerd heeft.

Incident

De incidenten waarvoor de oplossing is uitgevoerd.

Recept

Een reeks van acties en vragen.

6.1.13.2 Oplossing attributen

- Omschrijving
- Gevolgen

Omschrijving

Hier wordt kort omschreven wat de oplossing doet.

Gevolgen

Dit tekstveld beschrijft de mogelijke gevolgen van de oplossing. Aan de hand van de gevolgen kan de helpdeskmedewerker bijvoorbeeld bepalen of een oplossing uitgevoerd moet worden. Soms is het middel erger dan de kwaal. Als er meerdere oplossingen mogelijk zijn kan de beschrijving van de gevolgen helpen bij het kiezen.

6.1.14 Wijziging

Een **wijziging** is een oplossing. Een wijziging kan verbonden zijn met meerdere fouten en een fout kan meerdere wijzigingen hebben.

6.1.14.1 Wijziging associaties

- Fout
- Specialist

Fout

Dit zijn de fouten waarmee de wijziging verbonden is.

Specialist

De persoon die de wijziging goed moet keuren.

6.1.15 Work-around

Een **workaround** is een reeks acties die er voor zorgen dat een vastgelopen klant weer verder kan met zijn werk. Een workaround is een tijdelijke oplossing. De werkelijke oplossing wordt gevonden door het probleem te onderzoeken, de fout vast te stellen en een wijziging te bedenken. Een workaround kan verbonden zijn met meerdere problemen.

6.1.15.1 Work-around associaties

- Probleem

Probleem

Dit zijn de problemen waarmee de workaround verbonden is.

6.1.16 Keyword

Aan alle ITIL-objecten en aan acties, vragen en recepten kunnen keywords worden toegekend. Deze keywords worden gebruikt om te zoeken naar de opgeslagen objecten.

Er zijn verschillende manieren om de keywords te organiseren. Hieronder staan er drie beschreven.

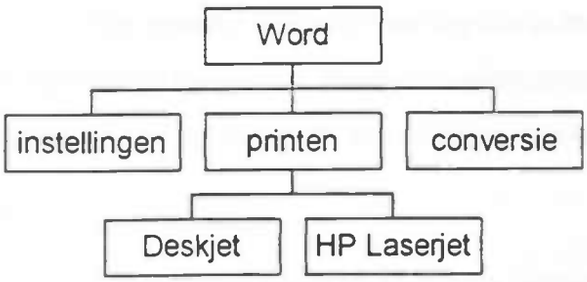
6.1.16.1 Een lijst

De keywords kunnen in een lange lijst geplaatst worden. Door de eerste letters van een keyword in te tikken kan de helpdeskmedewerker dan naar een keyword zoeken. Dit moet een paar keer gedaan worden als er meerdere keywords geselecteerd moeten worden.

Binnen een enkele lijst is het moeilijk om relaties tussen keywords aan te geven. Met een relatie wordt in dit geval bedoeld dat twee keywords een keer samen aan een object zijn toegekend.

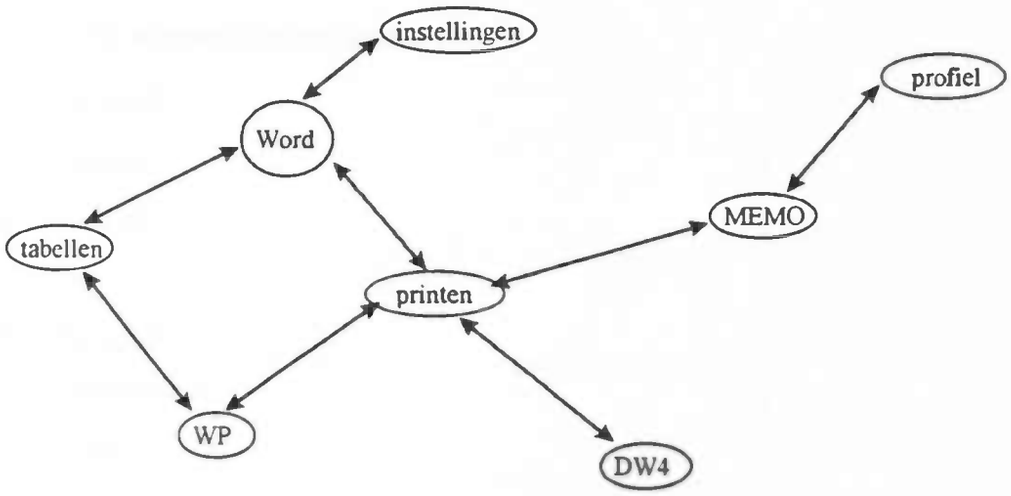
6.1.16.2 Een boom

In een boom kunnen relaties tussen keywords wel aangegeven worden. Het keyword **Word** kan een aantal keywords onder zich krijgen. Als de helpdeskmedewerker kiest voor **Word**, dan krijgt hij een lijst te zien met de keywords: **printen, instellingen** en **conversie**. Kiest hij vervolgens voor **printen** dan verschijnen **Deskjet** en **HP Laserjet.** Maakt hij een keuze voor **Deskjet** dan kunnen al dan niet ook de bovenliggende keywords worden geselecteerd. Deze methode wordt in een aantal helpdesk-pakketten al toegepast.



6.1.16.3 Een netwerk

De derde manier om structuur in de keywords weer te geven is het netwerk. Keywords worden met andere keywords verbonden, op een wijze zoals hieronder staat aangegeven. Zo is het keyword **printen** verbonden met een aantal tekstverwerkings-programma's waarmee wel eens problemen met het printen voorkomen. Tegelijkertijd is het keyword **Word** ook verbonden met het keyword **printen.**



Als een klant een probleem heeft met tabellen in Word en hij noemt eerst het woord tabellen, dan kan de helpdeskmedewerker het keyword **tabellen** alvast selecteren. Via **tabellen** kan hij dan bij **Word** en **WP** komen.

De klant kan ook beginnen met: 'Ik heb een probleem met Word...', de helpdeskmedewerker kan dan het keyword **Word** selecteren en kan daarna kiezen uit **instellingen, tabellen** en **printen.** Er zijn dus verschillende manieren om door het netwerk te navigeren. Keywords kunnen geselecteerd worden door op de rechtermuisknop te drukken en vervolgens Selecteren te kiezen in het popup-menu.

Het systeem verbindt twee keywords met elkaar zodra ze een keer samen aan een probleem zijn toegekend. Helpdeskmedewerkers kunnen keywords toevoegen. Dit moet wel gecontroleerd gebeuren om een chaos te voorkomen.

6.1.16.4 Indexeren

Elk keyword houdt bij aan welke incidenten, problemen, fouten, wijzigingen, workarounds, vragen en acties het is toegekend. Op die manier is het mogelijk om bijvoorbeeld snel alle problemen te vinden met de keywords **Word** en **printen**.

Een keyword kan ook bijhouden hoe vaak het tegelijk met andere keywords wordt gebruikt. Met deze informatie kan bijvoorbeeld de lijst met verbindingen van een keyword gesorteerd worden, zodat de meest gebruikte verbinding bovenaan staat. Het is echter maar de vraag of de helpdeskmedewerkers dat prettig vinden. Uit onderzoek is gebleken dat mensen het liefst werken met een interface die niet regelmatig wijzigt.

6.1.16.5 Keyword associaties

- Keyword
- Incident
- Problem
- Fout
- Wijziging
- Workaround
- Actie
- Vraag
- Recept

Deze associaties spreken voor zich.

6.1.16.6 Keyword attributen

- Keyword

Keyword

Een keyword kan met meerdere keywords verbonden zijn.

6.1.17 Recept

Binnen het helpdesksysteem komt het regelmatig voor dat de helpdeskmedewerker of de klant een aantal handelingen uit moet voeren en/of een aantal vragen moet beantwoorden om bijvoorbeeld een **vraag** te beantwoorden of om een **workaround** of **wijziging** uit te voeren. Zo'n reeks handelingen en vragen noemen we een **recept**.

6.1.17.1 Recept associaties

- Keywords
- gebruikteVraag
- gebruikteActie
- Receptontwikkelaar
- Actie
- Vraag
- Wijziging
- Workaround

gebruikteVraag

Dit zijn de vragen die onderdeel uitmaken van het recept.

gebruikteActie

Dit zijn de acties die onderdeel uitmaken van het recept.

Receptontwikkelaar

Dit is de persoon die het recept gemaakt heeft.

Actie, Vraag, Wijziging, Workaround

Dit zijn de vragen, wijzigingen en workarounds waarin het recept gebruikt wordt. In het algemeen zal een recept maar naar één van de vier verwijzen.

6.1.17.2 Recept attributen

- Gevolgen

Gevolgen

De gevolgen van het uitvoeren van actie worden opgeslagen bij de actie. De gevolgen van het uitvoeren van een recept kunnen afgeleid worden uit de gevolgen van de acties die opgenomen zijn in het recept.

6.1.18 Vragen en antwoorden

Tijdens de analysefase probeert de helpdeskmedewerker een incident te koppelen aan een probleem. Hij doet dit door informatie te verzamelen over het deel van de IT-infrastructuur waar het incident zich voordoet.

6.1.18.1 Vragen stellen

Één van de manieren om informatie te vergaren is het stellen van vragen aan de klant. De klant geeft antwoord op die vragen. Hieronder staat een aantal vragen dat gesteld zou kunnen worden.

1. Staat de printer aan?
2. Kunt u wel printen vanuit een ander pakket?
3. Wat staat er in de taakbalk?
4. Op welke frequentie staat de refresh-rate van uw monitor ingesteld?
5. Heeft u misschien de schermbeveiliging aanstaan?

Sommige vragen zijn eenvoudig te beantwoorden. De eerste vraag is er eentje. Om antwoord te kunnen geven op andere vragen moet de klant soms een aantal handelingen uitvoeren. Afhankelijk van het kennisniveau van de klant moet de helpdeskmedewerker hem daarbij begeleiden door een recept te schrijven.

Soms zijn er verschillende manieren om tot een antwoord op een vraag te komen.

6.1.18.2 Andere manieren om informatie te verzamelen

Het stellen van vragen aan de klant is niet de enige manier om informatie te verzamelen. De helpdeskmedewerker kan ook buiten de klant om informatie verzamelen. Hij kan bijvoorbeeld kijken welke printprocessen in de wachtrij van de server staan en welke servers draaien of down zijn.

6.1.18.3 Vraag associaties

- Problem
- Incident
- Keyword
- Recept
- Recept

- Antwoord

Problem

Problemen waarin de vraag voorkomt.

Incident

Incidenten waarin de vraag voorkomt.

Keyword

De keywords waarmee een vraag teruggevonden kan worden

Recept

De recepten waarin de vraag gebruikt wordt (gebruikteVraag bij Recept)

Recept

Het recept voor het beantwoorden van de vraag.

Antwoord

De antwoorden die mogelijk gegeven kunnen worden op de vraag. Dit kan een getal zijn, ja of nee, of een tijdstip.

6.1.18.4 Vraag attributen

- Vraagtekst

Vraagtekst

Bijvoorbeeld: wat gebeurt er als u ... doet?

6.1.19 Antwoord

Een antwoord is sterk verbonden met een vraag. Elk antwoord hoort bij een enkele vraag. Er zijn dus verschillende antwoorden 'ja', die elk het antwoord zijn op een andere vraag.

6.1.19.1 Antwoord associaties

- Vraag
- Incident

Vraag

De vraag waar het antwoord bij hoort.

Incident

De incidenten waar dit antwoord is gegeven op de vraag die bij dit antwoord hoort.

6.1.19.2 Antwoord attributen

- Antwoord

Antwoord

Het antwoord zelf, bijvoorbeeld: 'ja', 'nee', '123'.

6.1.20 Actie

Een actie is een handeling die door de klant of de helpdeskmedewerker wordt uitgevoerd. Voor het uitvoeren van een actie kan een recept bestaan.

6.1.20.1 Actie associaties

- Recept
- Vraag
- Workaround
- Wijziging
- Keyword
- Recept

Recept

De recepten waarin de actie gebruikt wordt. (gebruikteActie bij Recept)

Vraag, Workaround en Wijziging

De vragen, workarounds en wijzigingen die een recept hebben waarin de actie wordt gebruikt.

Keyword

Keywords kunnen worden gebruikt om een bepaalde actie terug te vinden.

Recept

Het recept voor het uitvoeren van de actie.

6.1.20.2 Actie attributen

- Uitvoerder
- Omschrijving van de handeling
- Gevolgen

Uitvoerder

De persoon die de actie uit moet voeren. Dit kan de klant zijn die een instelling in een pakket moet veranderen, maar ook een systeembeheerder die een password moet resetten.

Omschrijving van de handeling

Voorbeelden:

1. Stel de refresh-frequentie in op 72Hz.
2. Voeg een kolomeinde in.
3. Wijzig het decimaalteken in een punt.

Gevolgen

Hier worden de eventuele gevolgen van het uitvoeren van de actie beschreven.

6.1.21 Integriteit van de associaties

Binnen het helpdesksysteem verwijst elk geclassificeerd incident naar een probleem en elk probleem naar één of meer incidenten. De relatie tussen een probleem en een incident moet integer zijn, dat wil zeggen: wanneer een bepaald incident naar een probleem wijst dan moet het probleem ook naar het incident wijzen. Deze integriteit moet gewaarborgd blijven. Als we besluiten dat het incident bij een ander probleem thuishoort dan laten we het naar dat probleem wijzen. Het probleem waaraan het incident eerder was toegekend mag dan niet meer naar het incident wijzen. Een incident hoort immers maar bij één probleem.

De object georiënteerde database die we gebruikt hebben voor het helpdesk systeem biedt de mogelijkheid te specificeren wat de relatie is tussen twee klassen objecten. Voorbeelden zijn de één-op-één relatie (de getrouwd-zijn-met relatie) en de één-op-n relatie (incident-probleem).

Als we eenmaal duidelijk gemaakt hebben hoe de relatie is dan wordt er automatisch programmacode gegenereerd die er voor zorgt dat de relaties integer blijven. Dit bespaart een hoop tijd bij het programmeren.

6.2 Control objecten voor het classificatie proces

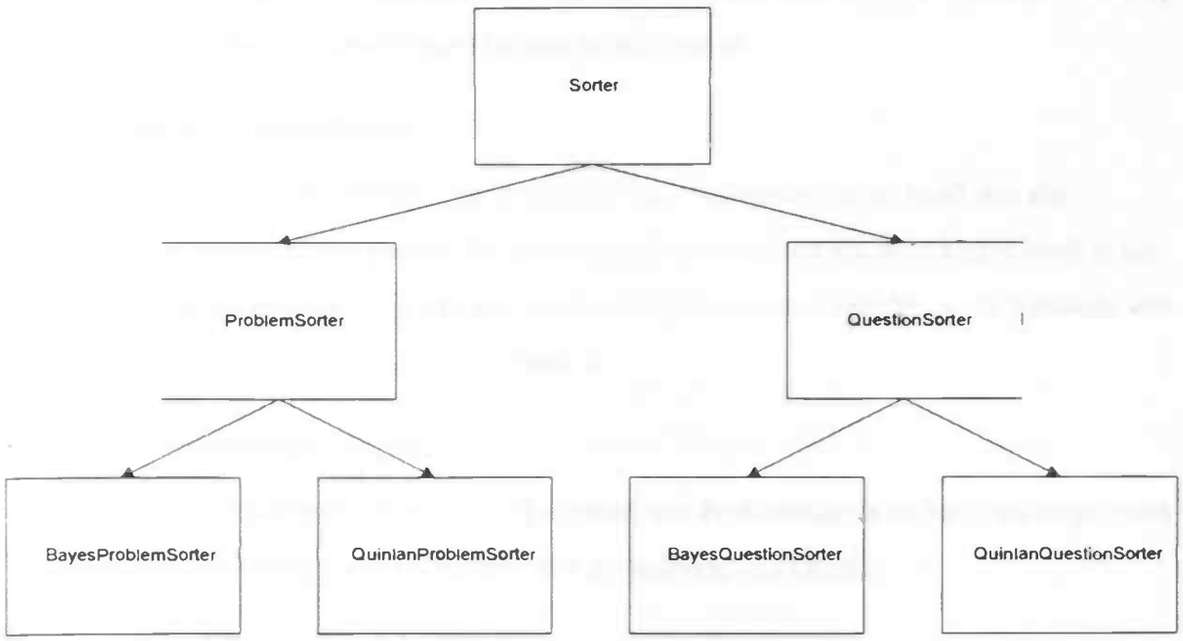
In deze paragraaf wordt beschreven welke control-objecten ontwikkeld zijn om de classificatiemethoden te implementeren en te testen. We beschrijven de functie van elk object en de manier waarop het communiceert met andere objecten.

6.2.1 Implementatie van de ordeningsmethoden

Om problemen en vragen te kunnen ordenen volgens de verschillende methoden is een aantal control-objecten ontwikkeld. Dit zijn:

- Sorter
- ProblemSorter
- BayesProblemSorter
- QuinlanProblemSorter
- QuestionSorter
- BayesQuestionSorter
- QuinlanQuestionSorter

Deze objecten staan in een hiërarchische relatie tot elkaar. Deze relatie wordt weergegeven in de onderstaande figuur.



6.2.1.1 Sorter

Sorter is een abstracte klasse en heeft twee subklassen: ProblemSorter en QuestionSorter. Sorter kan een aantal kansen bepalen:

- de kans op een keyword over alle incidenten
- de kans op een keyword binnen de incidenten van een probleem

- de kans op een probleem
- de kans op een propositie binnen een probleem.

Verder biedt Sorter via een variabele de mogelijkheid om te verwijzen naar een applicatie waaruit data kan worden gehaald.

6.2.1.2 ProblemSorter

ProblemSorter is een subklasse van Sorter en bevat de implementatie van de verhoudingen-methode om problemen te sorteren.

Andere objecten kunnen ProblemSorter de opdracht geven om de bekende problemen te ordenen op basis van de keywords of de proposities van een gegeven incident. Na afloop van het ordenen kan de belief-waarde voor elk probleem en een gesorteerde lijst problemen worden opgevraagd.

ProblemSorter bevat ook de functionaliteit om de belief-waarden van een probleem op verschillende manieren te combineren tot één belief-waarde.

6.2.1.3 BayesProblemSorter

De klasse BayesProblemSorter is afgeleid van ProblemSorter en heeft dus alle functionaliteit van die klasse geërfd. De enige extra functionaliteit die deze klasse biedt is het sorteren van problemen op basis van keywords of proposities met behulp van de methode van Bayes zoals die beschreven staat in hoofdstuk 2.

6.2.1.4 QuinlanProblemSorter

De klasse QuinlanProblemSorter is afgeleid van ProblemSorter en biedt als enige extra functionaliteit het ordenen van problemen met de methode van Quinlan.

6.2.1.5 QuestionSorter

De klasse QuestionSorter ordent vragen op basis van hun discriminerende vermogen. De klasse heeft een verwijzing naar een ProblemSorter voor informatie over (de ordening van) problemen. Via deze ProblemSorter kan de QuestionSorter bijvoorbeeld achterhalen wat de belief-waarde is voor een bepaald probleem.

Andere objecten kunnen aan QuestionSorter vragen wat het discriminerende vermogen van een bepaalde vraag is. Daarnaast kan QuestionSorter een lijst vragen doorgeven die geordend zijn op discriminerend vermogen.

6.2.1.6 BayesQuestionSorter

De klasse BayesQuestionSorter is afgeleid van QuestionSorter en biedt de mogelijkheid om vragen te ordenen met behulp van de methode van Bayes zoals beschreven in hoofdstuk 2.

6.2.1.7 QuinlanQuestionSorter

De klasse QuinlanQuestionSorter is afgeleid van QuestionSorter en biedt de mogelijkheid om vragen te ordenen met behulp van de methode van Quinlan zoals beschreven in hoofdstuk 2.

6.2.2 Andere control-objecten voor het testen van de classificatiemethoden

Naast de objecten die de verschillende ordeningsmethoden implementeren hebben we nog drie objecten ontwikkeld die ons helpen de prestaties van de algoritmen te testen.

- SortingApplicationModel
- HelpdeskDataGenerator
- IncidentGenerator
- ClassificationTester

6.2.2.1 SortingApplicationModel

Het SortingApplicationModel is een applicatie waarmee we twee dingen kunnen doen.

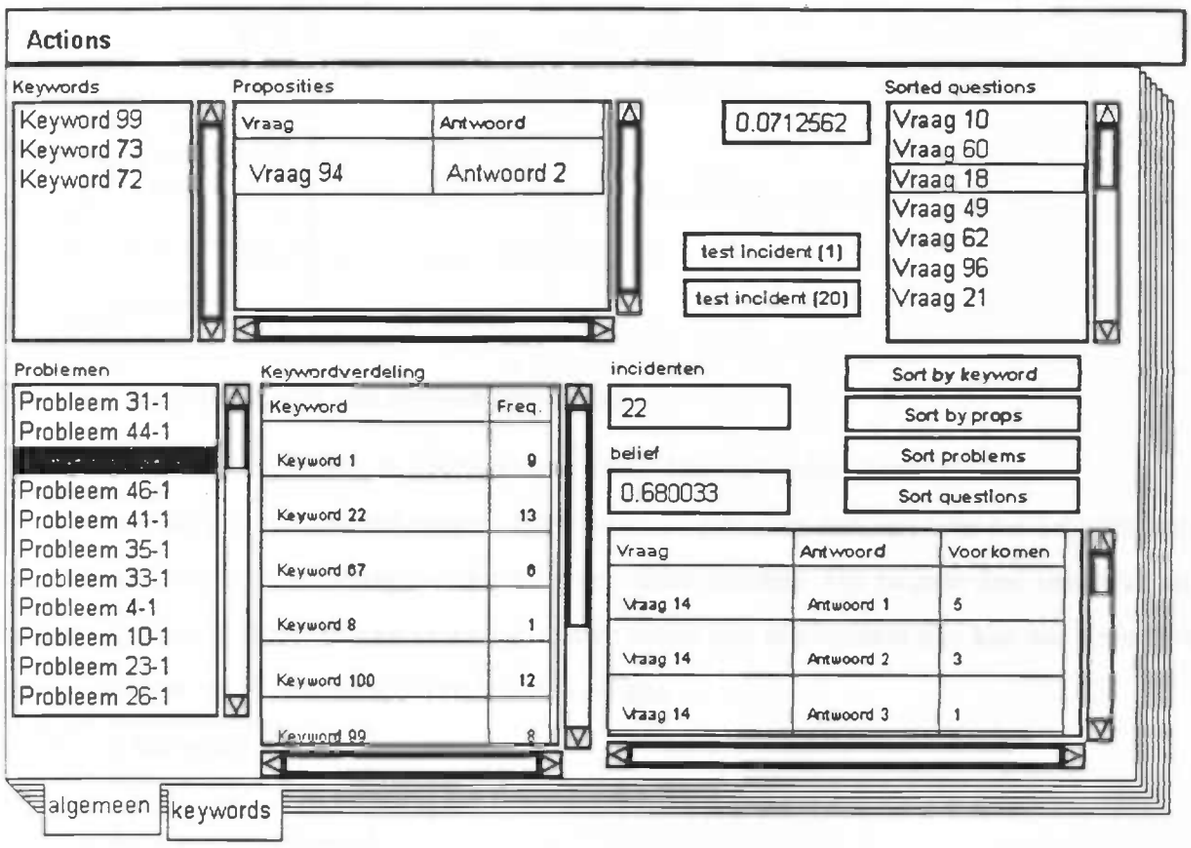
Ten eerste kan de gebruiker van de applicatie zelf een helpdeskmedewerker na doen. Hij kan een nieuw incident definiëren, er keywords aan toekennen en het systeem de problemen en vragen laten ordenen. Daarna kan hij een vraag stellen en er een antwoord op geven. Zo is het mogelijk de werking van de algoritmen nauwkeurig te onderzoeken.

Ten tweede kan de gebruiker het systeem opdracht geven zelf een groot aantal incidenten te genereren en te classificeren en daarvan een rapport te maken. Op die manier kunnen de prestaties van de algoritmen over grote aantallen incidenten bestudeerd worden. We gebruiken deze mogelijkheid om het systeem te testen, zoals in het volgende hoofdstuk beschreven staat.

De interface van SortingApplicationModel bestaat uit de volgende onderdelen:

Linksboven de keywords van het huidige incident, daarnaast de proposities die ermee verbonden zijn. Onder de keywords een lijst van alle bekende problemen. Naast de problemen

staan de keywordverdeling van het geselecteerde probleem, het aantal incidenten dat verbonden is met het probleem en het huidige belief in het probleem. Helemaal rechts staat de vragenverdeling van het geselecteerde probleem. Boven de vragenverdeling is een aantal knoppen geplaatst om problemen en vragen te sorteren. Boven de knoppen staat de lijst met gesorteerde vragen. Links naast de lijst de discriminatiewaarde van de geselecteerde vraag. Onder deze discriminatiewaarde staat twee knoppen om automatisch incidenten te genereren en testen.



Voor het genereren en classificeren van grote aantal incidenten zijn nog drie andere objecten nodig: de HelpdeskDataGenerator, de IncidentGeneratoren en de ClassificationTester, die in de volgende paragrafen aan bod komen.

6.2.2.2 HelpdeskDataGenerator

De HelpdeskDataGenerator creëert een testomgeving. Deze testomgeving bestaat uit keywords, vragen en IncidentGeneratoren. De gebruiker kan zelf bepalen hoeveel keywords, vragen en IncidentGeneratoren er aangemaakt moeten worden.

6.2.2.3 IncidentGenerator

De taak van een IncidentGenerator is incidenten te genereren. Een IncidentGenerator kan gezien worden als een gesimuleerde fout. Als een IncidentGenerator gecreëerd wordt dan krijgt hij een keywordverdeling en een propositieverdeling. De keywordverdeling bevat standaard 8 keywords. Elk keyword is verbonden met een bepaalde kans. Als de IncidentGenerator een nieuw incident creëert wordt het verbonden met 3 keywords. Hoe groter de kans die verbonden is met een keyword hoe vaker het keyword met een incident verbonden zal worden.

Word	0.8
Printen	0.7
Pagina-instellingen	0.6
Kantlijnen	0.5
Marges	0.4
Voettekst	0.2

Tabel 6-1 Voorbeeld van een keywordverdeling

De propositieverdeling bestaat uit twee delen. Het eerste deel werkt als een keywordverdeling, maar dan met vragen. Elke vraag is verbonden met een kans die aangeeft hoe waarschijnlijk het is dat de vraag voorkomt bij een nieuw incident. Het tweede deel ziet er uit als in de onderstaande tabel. Als een vraag verbonden wordt met een incident dan kan met deze tabel bepaald worden wat het antwoord op de vraag zal zijn.

Vraag 1	Antwoord 1	0.9
	Antwoord 2	0.1
Vraag 2	Antwoord 1	0.4
	Antwoord 2	0.6

Tabel 6-2 Voorbeeld van een propositieverdeling

Op basis van deze twee verdelingen genereert een IncidentGenerator in opdracht van bijvoorbeeld een ClassificationTester incidenten.

En IncidentGenerator kan ook een extra keyword of propositie toevoegen aan een gegeven incident. Dit gebeurt ook weer volgens de verdelingen.

In uitgebreide simulaties gebruiken we een aantal IncidentGeneratoren die we incidenten laten genereren. Op die manier kunnen we een praktijksituatie nabootsen.

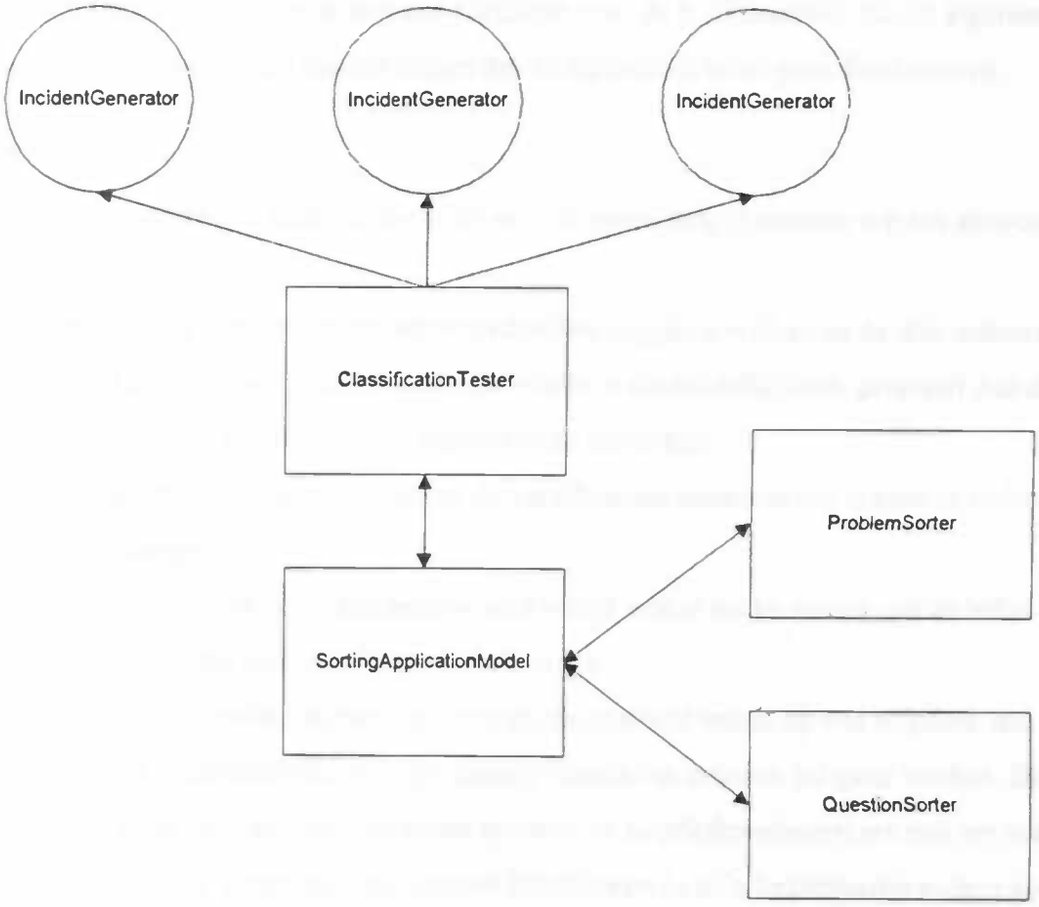
6.2.2.4 ClassificationTester

Een ClassificationTester is bedoeld om diverse praktijksituaties na te bootsen. SortingApplicationModel en ClassificationTester werken nauw samen. De ClassificationTester zorgt er voor dat één van de IncidentGeneratoren een incident aanmaakt en laat dat vervolgens door SortingApplicationModel classificeren.

De gebruiker kan zelf instellen hoe de classificatie moet plaatsvinden. Zo kan bijvoorbeeld opgegeven worden hoeveel extra informatie er over een incident verzameld mag worden en welke manier gebruikt moet worden om de problemen te ordenen.

Als er meerdere incidenten achter elkaar geclassificeerd worden genereert ClassificationTester een rapport met testresultaten.

De manier waarop de verschillende objecten samenwerken om het classificatie proces te simuleren wordt in het volgende plaatje weergegeven.



7. Testen

7.1 Inleiding

Om de bruikbaarheid van het ontwikkelde helpdesk ondersteunende systeem te bepalen moet het getest worden in een omgeving die de praktijksituatie zoveel mogelijk benaderd. Één van de manieren om dit te doen is een praktijksituatie te simuleren. In zo'n simulatie kan het systeem op verschillende manieren onderzocht worden. Naar aanleiding van de simulatie kan het dan gewijzigd of aangepast worden.

In dit hoofdstuk wordt beschreven op welke manieren en onder welke omstandigheden het systeem is getest. Eerst gaan we in op de vragen die we willen beantwoorden, dan volgt een korte uiteenzetting over de manier van testen. Vervolgens gaan we in op de verschillende testen en de verkregen resultaten. We sluiten af met een conclusie over de bruikbaarheid van de algoritmen en aanpassingen die er toe zouden kunnen leiden dat de algoritmen beter gaan functioneren.

7.2 Testvragen

Voorafgaand aan het testen formuleren we een aantal vragen waarop we een antwoord zoeken.

De eerste vraag waarop we een antwoord willen krijgen is welke van de drie manieren om problemen te sorteren het best werkt. Als er één manier is die duidelijk beter presteert dan de andere twee, dan kunnen we die voor de andere testen gebruiken.

De tweede vraag is: hoe goed werken de verschillende manieren om vragen te ordenen op discriminerend vermogen?

Een andere vraag die we graag beantwoord willen zien is welke manier om de belief-waarden te combineren tot één belief-waarde de beste is.

Als een helpdesk ondersteunend systeem geïntroduceerd wordt op een helpdesk dan is er nog geen informatie in opgenomen over het domein waarin het systeem zal gaan werken. Deze informatie wordt in de loop der tijd verzameld doordat de helpdeskmedewerkers met het systeem werken. We willen graag weten hoe het systeem functioneert in zo'n beginsituatie en hoe goed het presteert na een aanloopperiode.

Als er eenmaal voldoende informatie over het domein verzameld is willen we weten hoe het systeem reageert op nieuwe fouten in de IT-infrastructuur, die in de praktijk altijd zullen voorkomen.

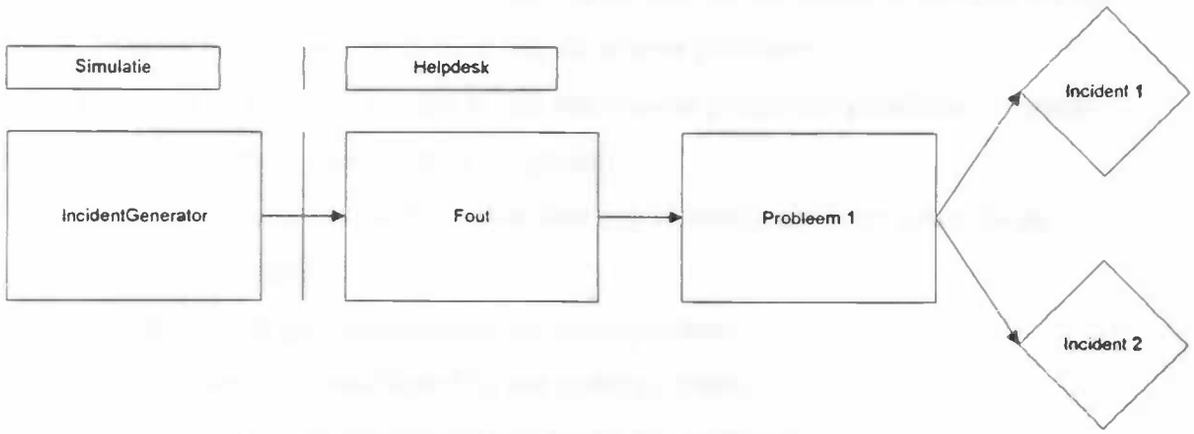
In totaal zijn er dus zes vragen die onderzocht moeten worden:

1. Is één van de methoden om problemen te ordenen duidelijk beter dan de andere?
2. Hoe goed kunnen we door het ordenen van vragen de classificatie ondersteunen?
3. Wat is de beste manier om belief-waarden te combineren?
4. Hoe functioneert het systeem in de aanloopsituatie?
5. Hoe goed zijn de prestaties na de aanloopperiode?
6. Hoe reageert het systeem op nieuwe fouten in de IT-architectuur?

7.3 Testomgeving

Om het helpdesksysteem te testen is er een testomgeving ontwikkeld. Deze testomgeving bestaat uit twee delen. Het eerste deel is het helpdesksysteem zelf, zoals dat in de praktijk gebruikt zou worden. Het tweede deel, het simulatiedeel, ligt daar als het ware overheen en bestuurt het eerste deel. Dit simulatiedeel bootst de werkelijkheid na. Het genereert incidenten, laat die door het helpdesksysteem classificeren en legt de prestaties vast. Het simulatiedeel kan op een aantal manieren worden ingesteld om de verschillende vragen te beantwoorden.

Een belangrijk onderdeel van de simulatieomgeving vormen de IncidentGeneratoren. Zij representeren samen een IT-infrastructuur met daarin een aantal fouten. De voornaamste functie van een IncidentGenerator is de incidenten te genereren, die anders zouden ontstaan doordat gebruikers hun werk doen. Elke IncidentGenerator wordt verbonden met een keywordverdeling, een propositieverdeling en de fout die hij representeert. Dit is de relatie tussen het simulatiedeel en de praktijk.



7.3.1 Het verloop van een simulatie

De eerste stap van de simulatie bestaat uit het opzetten van de testomgeving. We doen dit door een aantal IncidentGeneratoren te creëren. Deze IncidentGeneratoren vormen samen de IT-infrastructuur.

Daarna begint de simulatie. We kiezen één van de IncidentGeneratoren en geven de opdracht om een incident te genereren. Dit komt overeen met het ontstaan van een incident doordat een gebruiker werkt met de infrastructuur. Het incident moet geclassificeerd worden door het helpdesksysteem. Aangezien er nog geen problemen bestaan wordt er een nieuw probleem gedefinieerd en wordt het incident ondergebracht bij dit nieuwe probleem. Het probleem wordt vervolgens verbonden met de fout van de IncidentGenerator die het incident maakte. In de werkelijkheid wordt deze relatie niet meteen gelegd, maar pas later door de probleembeheerder, als er onderzoek is gedaan naar het probleem. Binnen de simulatie is het nodig om deze relatie nu al te leggen, omdat we later willen weten of de classificatie van andere incidenten goed is verlopen.

We selecteren nu een opnieuw een IncidentGenerator en laten een incident ontstaan. Omdat er nu maar één probleem is, zal het systeem het incident bij dat probleem classificeren. Dit is niet het juiste probleem (tenzij we twee keer dezelfde IncidentGenerator geselecteerd hebben). De oplossing die in de praktijk verbonden is met het probleem zal dan niet werken. De helpdeskmedewerker zal op zoek gaan naar een probleem dat wel verbonden is met een goede oplossing, of een nieuw probleem definiëren. Aangezien er maar één probleem is wordt er een nieuw probleem aangemaakt, dat verbonden wordt met de fout van de IncidentGenerator die het incident maakte. Het incident wordt verbonden met dit nieuwe probleem.

In het begin van de simulatie worden er dus veel nieuwe problemen gedefinieerd, omdat het systeem als het ware kennis maakt met de omgeving.

Als een IncidentGenerator voor de tweede keer een incident maakt dan zijn er na de classificatie drie situaties mogelijk:

1. Het incident wordt geclassificeerd bij het juiste probleem.
2. Het incident wordt geclassificeerd bij een ander probleem.
3. Het incident kan niet geclassificeerd worden bij één probleem.

In het eerste geval is er niets aan de hand. De classificatie is correct, de gesuggereerde oplossing zal werken en het incident wordt verbonden met het probleem.

In het tweede geval zal de gesuggereerde oplossing niet werken en zal de helpdeskmedewerker proberen een ander probleem te zoeken. We simuleren dit eerst door het systeem de gesorteerde lijst met problemen af te laten gaan, totdat het juiste probleem gevonden is. Later verzamelt het systeem nieuwe informatie over het testincident in de vorm van keywords of proposities om zo te discrimineren tussen de mogelijke problemen.

In het laatste geval moet er een nieuw probleem gedefinieerd worden. Dit probleem wordt vervolgens verbonden met de fout van de IncidentGenerator. In de praktijk betekent dit dat een fout verschillende problemen veroorzaakt. Het is echter niet eenvoudig om een manier te vinden om te bepalen wanneer de classificatie niet eenduidig is. We laten deze laatste mogelijkheid voorlopig dan ook buiten beschouwing.

7.3.2 De variabelen

Het simulatiedeel van de testomgeving bevat een aantal variabelen. Door deze variabelen verschillende waarden toe te kennen kunnen de prestaties van het helpdesksysteem onder verschillende omstandigheden onderzocht worden.

De volgende eigenschappen zijn instelbaar.

- Het aantal IncidentGeneratoren waarmee de simulatie begint
  (standaard: 25)
- Wel of geen toename van het aantal IncidentGeneratoren tijdens de simulatie
  (standaard: geen toename)
- Tempo waarmee IncidentGeneratoren worden toegevoegd
  (standaard: één IncidentGenerator erbij per honderd geclassificeerde incidenten)
- Het aantal keywords in de keywordverdeling van de IncidentGeneratoren
  (standaard: 8)
- Het aantal keywords dat wordt toegekend aan een nieuw incident
  (standaard: 3)
- Het aantal vragen in de propositieverdeling
  (standaard: 8)
- Het totale aantal keywords binnen het systeem
  (standaard: 100)
- Het totale aantal vragen binnen het systeem
  (standaard: 100)

- Wel of geen extra keywords toekennen aan een incident om de classificatie te verbeteren (standaard: niet doen)
- Wel of geen extra proposities toekennen aan een incident om de classificatie te verbeteren (standaard: niet doen)
- Het algoritme dat gebruikt wordt om te classificeren (standaard: verhoudingen-algoritme)

7.3.3 Het meten van de classificatie-prestaties

Als we willen dat het helpdeskondersteunenende systeem ook werkelijk gebruikt gaat worden dan moet het goed onderscheid kunnen maken tussen de verschillende problemen in het domein. Na afloop van de analyse van een incident moet het juiste probleem zo hoog mogelijk in de lijst staan. De helpdeskmedewerkers zijn ervaren mensen en het systeem is er om deze mensen te ondersteunen. Het is daarom niet nodig dat het juiste probleem altijd op de eerste plaats komt te staan. Met hun kennis en ervaring kunnen de helpdeskmedewerkers het juiste probleem wel vinden binnen een beperkte verzameling problemen.

Als het juiste probleem bovenaan staat na de classificatie dan noemen we de classificatie perfect. Als het juiste probleem op de eerste of de tweede positie staat dan noemen we de classificatie geslaagd. Staat het juiste probleem bij bovenste vijf problemen dan vinden we de classificatie acceptabel. We noemen de classificatie mislukt als het juiste probleem niet bij de eerste vijf problemen staat.

Positie van het juiste probleem	Oordeel
1	Perfect
1,2	Geslaagd
1,2,3,4,5	Acceptabel
6, ...	Mislukt

Tabel 7-1 Definities voor de classificatieprestaties

Tijdens het testen laten we steeds twintig incidenten classificeren. We kijken voor elk incident op welke plaats het juiste probleem staat. In de tabel hieronder staat het resultaat van de classificatie van een twintigtal incidenten. Voor elf van die incidenten stond het juiste probleem na classificatie op de eerste plaats. Voor twee van de twintig incidenten stond het op de vierde plaats. In totaal stond voor zeventien van de twintig incidenten het juiste probleem bij de eerste vijf.

We laten telkens zeker vijftig keer twintig incidenten classificeren. Als resultaat krijgen we dan een tabel met vijftig rijen zoals die hieronder. Uit deze tabel kunnen we dan afleiden hoe snel het systeem leert en hoe goed de prestaties na verloop van tijd zijn.

#1	#2	#3	#4	#5	Totaal
11	3	1	2	0	17

Tabel 7-2 Voorbeeld van het resultaat van een classificatie

7.3.4 Uitleg over de grafieken

In alle grafieken staat op de x-as het aantal incidenten dat het systeem in totaal geclassificeerd heeft tijdens een simulatie.

Op de y-as staan steeds twee verschillende lijnen. De ene lijn laat zien hoe vaak het juiste probleem op de eerste of de tweede plaats stond. De andere lijn laat zien hoe vaak het juiste probleem bij de eerste vijf problemen stond. Deze laatste lijn ligt dus altijd op hetzelfde niveau als de eerste lijn, of hoger.

7.4 Het testen

In deze paragraaf gaan we op zoek naar de antwoorden op de gestelde vragen. Per vraag beschrijven we eerst hoe de test wordt opgezet en uitgevoerd, daarna geven we de resultaten en tot slot een conclusie, met daarin het antwoord op de vraag.

7.4.1 Welk algoritme presteert het best?

In de theorie hebben we drie manieren beschreven om problemen te ordenen op basis van de keywords of de proposities die met een testincident verbonden zijn. Voordat we het helpdesk systeem uitgebreid gaan testen willen we graag weten welke methode de problemen het beste ordent.

7.4.1.1 Testopzet

Eerst definiëren we een testomgeving. Uit oriënterende simulaties is gebleken dat een omgeving van 50 IncidentGeneratoren de verschillende eigenschappen van de algoritmen goed doet uitkomen. Naast 50 IncidentGeneratoren creëren we ook duizend incidenten, die we automatisch aan de juiste problemen toekennen. In die omgeving laten we elk algoritme 200 incidenten classificeren. Zo kunnen we goed zien hoe de verschillende algoritmen presteren na een aanloopperiode.

De resultaten staan samengevat in een tabel. De tabel toont de resultaten van de classificatie als de problemen geordend worden op basis van de keywords van het testincident.

7.4.1.2 Resultaten

7.4.1.2.1 Methode van Bayes

#1	#2	#3	#4	#5	Totaal
0	5	5	2	0	12
5	4	3	2	0	14
0	5	5	2	1	13
2	2	3	3	3	13
1	4	3	3	2	13
4	5	3	1	3	16
1	2	6	3	4	16
4	6	2	5	1	18
1	5	2	6	4	18
3	5	4	1	3	16

Tabel 7-3 Resultaten van de methode van Bayes

7.4.1.2.2 Methode van Quinlan

#1	#2	#3	#4	#5	Totaal
10	4	0	2	2	18
12	4	3	0	0	19
11	5	1	0	0	17
11	5	2	1	1	20
8	8	3	0	1	20
13	1	2	1	0	17
13	4	2	1	0	20
13	3	0	2	2	20
11	5	3	0	1	20
13	6	1	0	0	20

Tabel 7-4 Resultaten van de methode van Quinlan

7.4.1.2.3 De verhoudingen-methode

#1	#2	#3	#4	#5	Totaal
20	0	0	0	0	20
20	0	0	0	0	20
18	2	0	0	0	20
20	0	0	0	0	20
20	0	0	0	0	20
20	0	0	0	0	20
19	1	0	0	0	20
20	0	0	0	0	20
20	0	0	0	0	20
19	1	0	0	0	20

Tabel 7-5 Resultaten van de verhoudingen-methode

7.4.1.3 Conclusie

De methode die de problemen ordent met de methode van Bayes presteert het minst van de drie. De totaalscore is weliswaar heel behoorlijk en wordt ook beter na meer classificaties, maar het aantal geslaagde classificaties is laag. De methode van Quinlan presteert beter dan de methode van Bayes. Het aantal geslaagde classificaties is veel hoger, namelijk meer dan de helft.

De methode die problemen sorteert op basis van verhoudingen komt als beste naar voren. Bijna alle incidenten worden met deze methode perfect geclassificeerd. De rest van de testen worden daarom uitgevoerd met de verhoudingen-methode.

7.4.2 Ondersteuning van classificatie met behulp van vragen

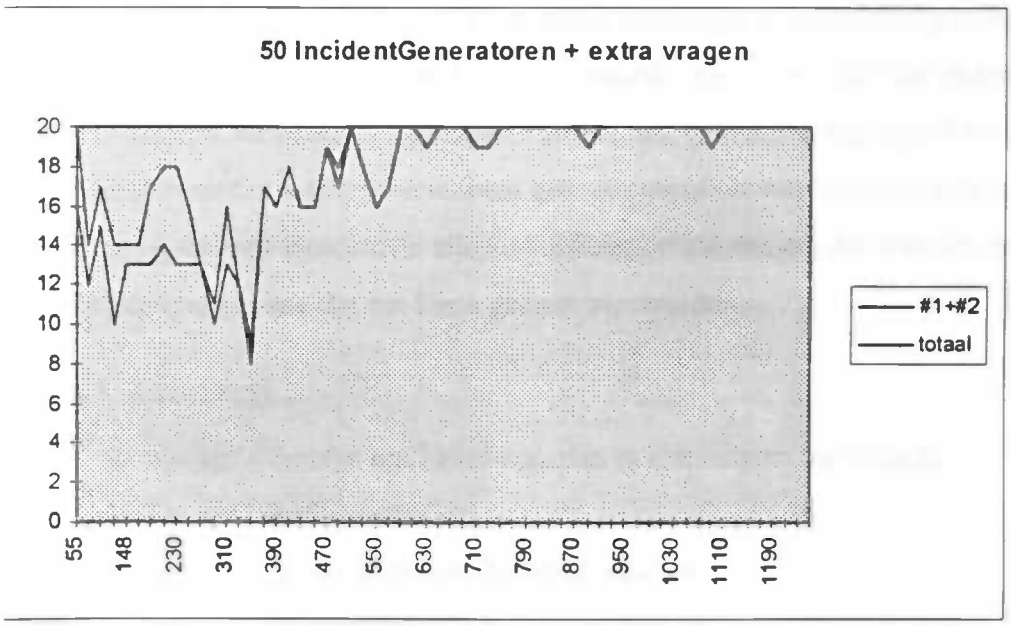
Als een helpdeskmedewerker een incident analyseert, verbindt hij het eerst met een aantal keywords. Als de classificatie met behulp van deze keywords niet goed is, dan is er blijkbaar meer informatie nodig. Deze informatie kan bestaan uit een extra keyword, of uit een vraag die aan de klant gesteld wordt. In deze test simuleren we het stellen van een vraag aan de klant door de helpdeskmedewerker als de classificatie op basis van keywords niet bevredigend is.

7.4.2.1 Testopzet

We beginnen in een testomgeving met 50 IncidentGeneratoren. De IncidentGeneratoren laten we eerst 200 incidenten classificeren die verbonden zijn met 3 keywords uit de keywordverdeling en 3 proposities uit de propositieverdeling van de IncidentGenerator. Zo geven we het systeem de kans om de omgeving al wat te leren kennen. Daarna laten we het systeem duizend incidenten classificeren die wel met drie keywords verbonden zijn, maar met slechts één propositie. Als de classificatie van een incident niet perfect is ordent het systeem de vragen. De vraag die bovenaan staat wordt gesteld. Als de vraag voorkomt in de propositieverdeling van de IncidentGenerator dan wordt aan de hand van die verdeling bepaald wat het antwoord op de vraag zal zijn. Als de vraag niet voorkomt in de propositieverdeling dan kijkt het systeem naar de incidenten waarbij de vraag wel is gesteld, bepaalt de verdeling van de gegeven antwoorden en genereert dan een antwoord dat in overeenstemming is met deze verdeling.

7.4.2.2 Resultaten

In de onderstaande grafiek staan de resultaten weergegeven van de test met 50 IncidentGeneratoren.



7.4.2.3 Conclusie

In de grafiek valt te zien dat het systeem na ongeveer 600 incident behoorlijk goed classificeert en dat ook blijft doen. Dit kan op twee manieren verklaard worden. Het kan zijn dat het systeem alleen al na ordening op basis van de keywords tot goede classificatie prestaties komt

en dat er helemaal geen extra vragen gesteld worden of dat deze extra vragen niet bijdragen tot een betere classificatie.

De andere mogelijkheid is dat het systeem niet goed genoeg classificeert op basis van de keywords alleen en dat het systeem deze prestaties opkrikt door extra vragen te stellen.

Uit de grafiek met daarin de resultaten van een simulatie waarin alleen op basis van keywords geclassificeerd werd, blijkt dat het systeem al na zo'n 400 incidenten net zo goed presteert als de simulatie hierboven na 600 incidenten. Het stellen van vragen heeft dus geen positief effect op de prestaties. Daar komt nog bij dat een simulatie met 50 IncidentGeneratoren waarin keywords toegevoegd worden als de classificatie niet perfect is al na 270 incidenten alle volgende incidenten perfect classificeert.

Het is dus beter het systeem alleen op keywords te laten ordenen en eventueel keywords toe te voegen dan een door het systeem gesuggereerde vraag te stellen.

We kunnen concluderen dat het ordenen van de vragen door het systeem niet goed genoeg lukt.

7.4.3 Manieren om belief-waarden te combineren

Tijdens het ordenen van de problemen gaat het systeem alle informatie af die bekend is over het te classificeren incident. Als een stukje informatie een aanwijzing is voor een bepaald probleem, dan krijgt dat probleem een belief-waarde toegekend voor dat stukje informatie. Nadat alle informatie is verwerkt zijn de meeste problemen verbonden met één of meer belief-waarden. Deze belief-waarden worden vervolgens gecombineerd tot één belief-waarde op basis waarvan de problemen geordend worden. Er zijn verschillende manieren om die waarden te combineren en een belangrijke vraag is hoe dat het beste gedaan kan worden.

7.4.3.1 Testopzet

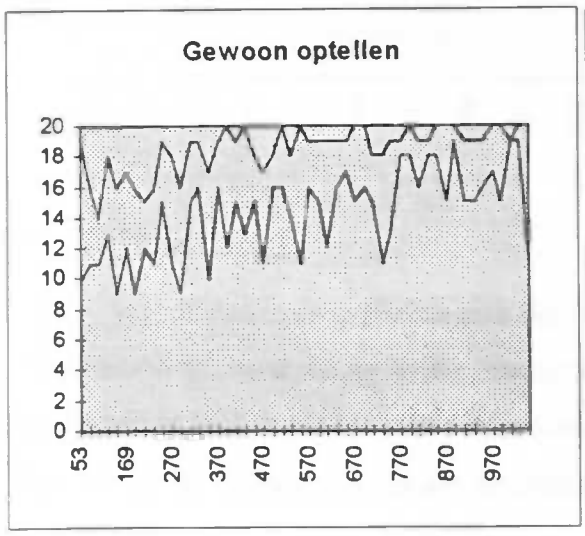
Er zijn vijf manieren om belief-waarden te combineren onderzocht.

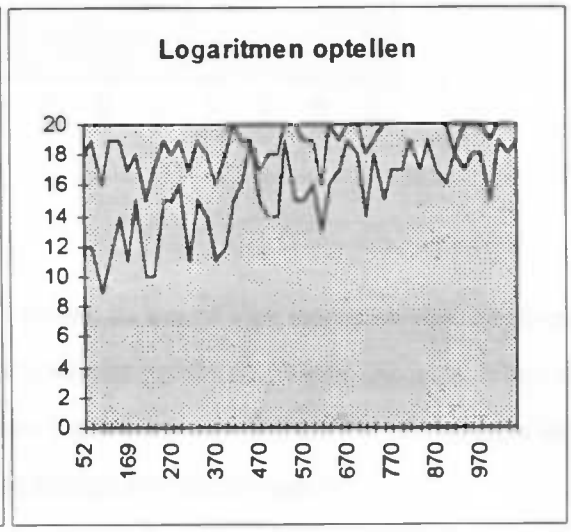
1. Tel alle belief-waarden op
2. Neem het gemiddelde van de belief-waarden
3. Tel de logaritmen van alle belief-waarden op
4. Neem het gemiddelde van de logaritmen van de belief-waarden.
5. Neem de grootste belief-waarde.

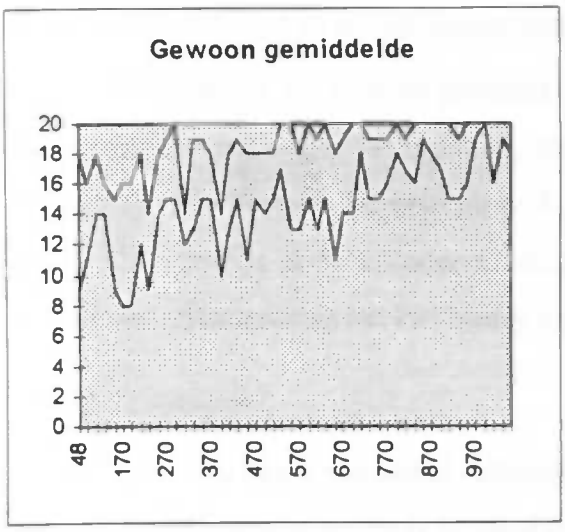
In een omgeving van 50 IncidentGeneratoren hebben we 1000 incidenten laten classificeren met de verhouding-methode. De manier waarop de belief-waarden werden gecombineerd varieerde. Door de resultaten te vergelijken kunnen we zien welke methode het beste functioneert.

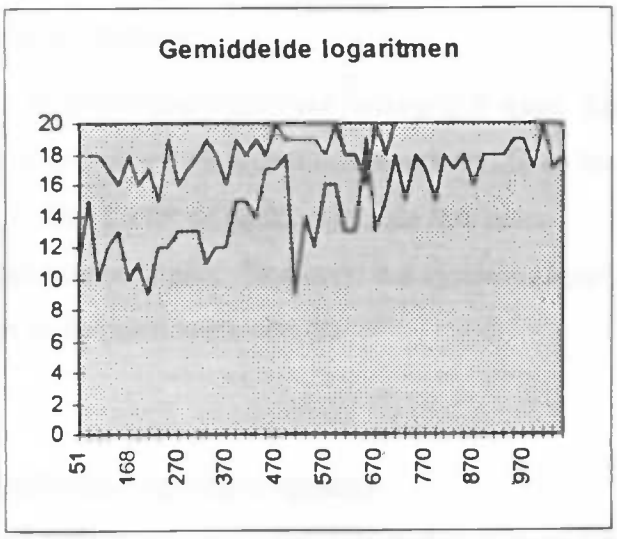
7.4.3.2 Resultaten

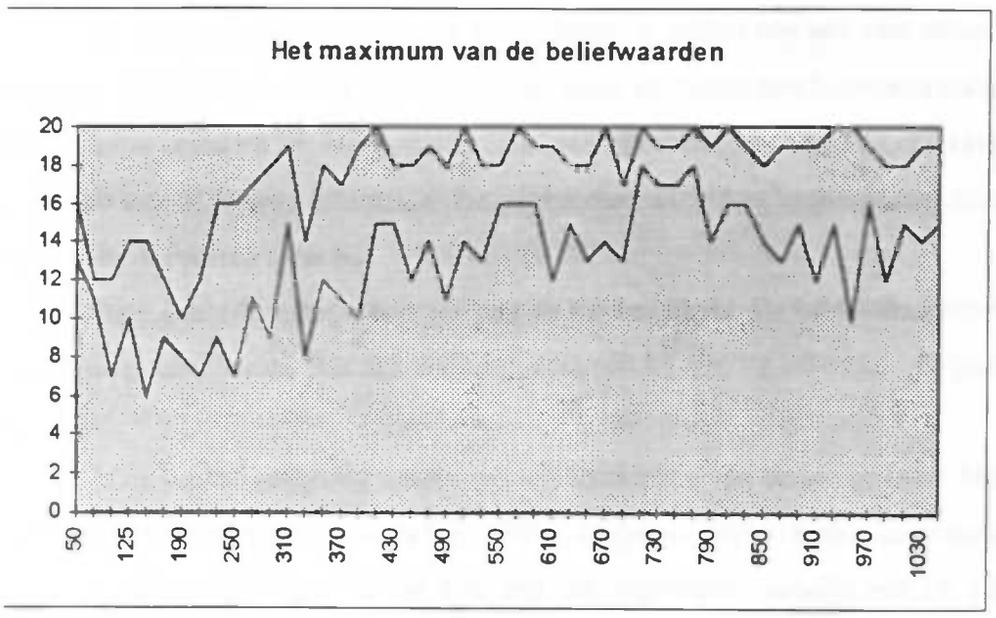
De resultaten kunnen het mooist weergegeven worden in grafiekvorm. De bovenste lijn geeft steeds aan hoe vaak de classificatie acceptabel was, de onderste lijn geeft aan hoe vaak de classificatie geslaagd was.











7.4.3.3 Conclusies

We kunnen in de grafieken zien dat er tussen de eerste vier manieren van combineren niet veel verschil zit, maar dat de laatste manier wat minder goede resultaten oplevert. Voor de rest van de testen hebben we het gemiddelde van de logaritmen van de belief-waarden gebruikt, omdat deze manier na verloop van tijd alle incidenten acceptabel classificeerde.

7.4.4 De prestaties in de aanloopperiode en daarna

De vraag hoe het systeem presteert in de aanloopperiode is een belangrijke vraag. Als de aanloopperiode erg lang is dan is dat erg vervelend voor de helpdeskmedewerkers. Zij zullen pas na lange tijd voordeel van het systeem hebben. Hoe korter de aanloopperiode hoe beter.

De prestaties na de aanloopperiode zijn ook belangrijk. Hoe beter het systeem classificeert, des te meer ondersteuning het kan bieden aan de helpdeskmedewerkers.

7.4.4.1 Testopzet

Voor deze test is een aantal verschillende testomgevingen opgezet.

De eerste testomgeving is de standaard-testomgeving (zie de paragraaf over de variabelen). In deze omgeving bevindt zich een aantal IncidentGeneratoren. Dit aantal groeit niet tijdens de simulatie. De gegenereerde incidenten hebben elk drie keywords.

In de tweede omgeving groeit het aantal IncidentGeneratoren wel tijdens de simulatie. Na ongeveer 100 geclassificeerde incidenten komt er een nieuwe IncidentGenerator bij.

In de derde omgeving hebben we wederom te maken met een vast aantal IncidentGeneratoren. Als na classificatie het juiste probleem niet bovenaan staat krijgt het incident er een extra keyword bij. Het systeem classificeert het incident dan nog een keer, rekening houdend met de nieuwe informatie. De gegenereerde incident beginnen met drie keywords en krijgen er hoogstens twee bij.

Deze procedure komt overeen met de werkelijkheid. De helpdeskmedewerker probeert een incident te classificeren. Als dat niet lukt verzamelt hij nieuwe informatie en probeert hij het opnieuw.

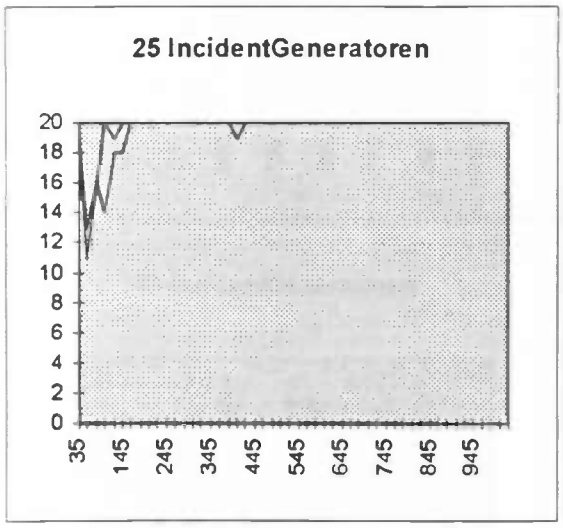
In de laatste omgeving vinden we een combinatie van de vorige twee. Het aantal IncidentGeneratoren groeit en de testincident krijgen er, indien nodig, keywords bij.

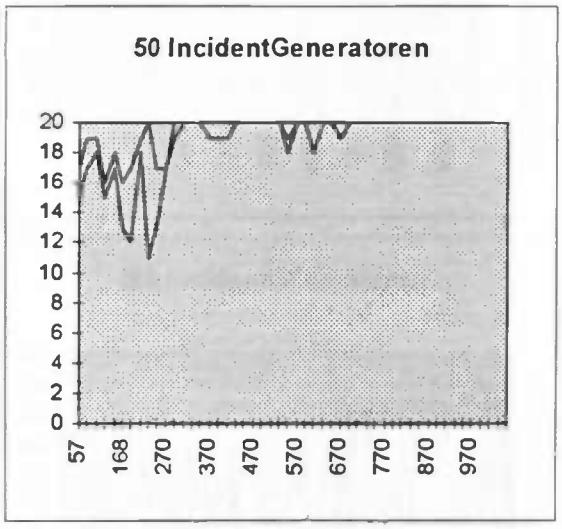
In elke omgeving werd vier keer een test uitgevoerd, namelijk met 25, 50, 100 en 200 IncidentGeneratoren. Er werden steeds duizend incidenten geclassificeerd.

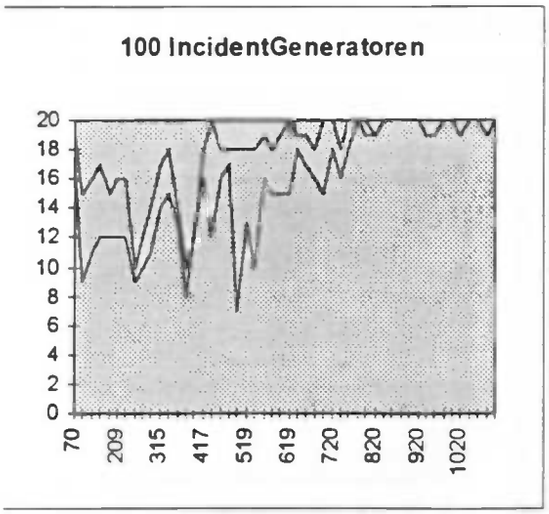
7.4.4.2 Resultaten

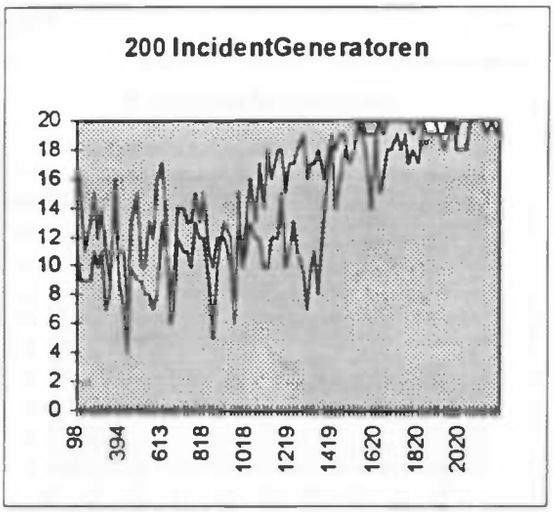
De resultaten worden uitgesplitst per testomgeving.

7.4.4.2.1 De standaard testomgeving

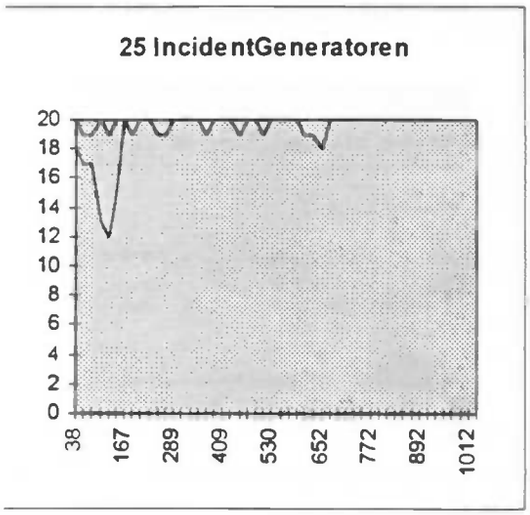


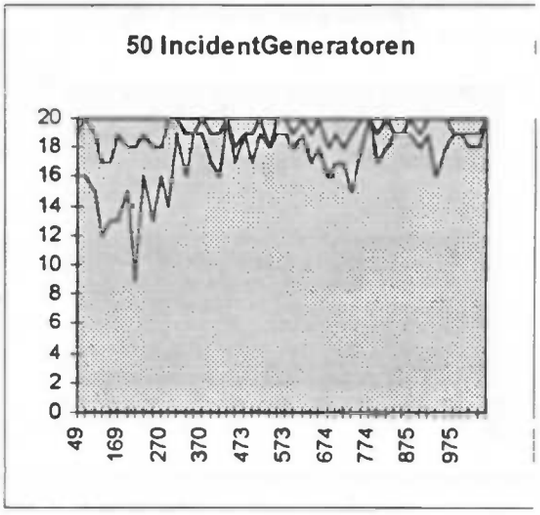


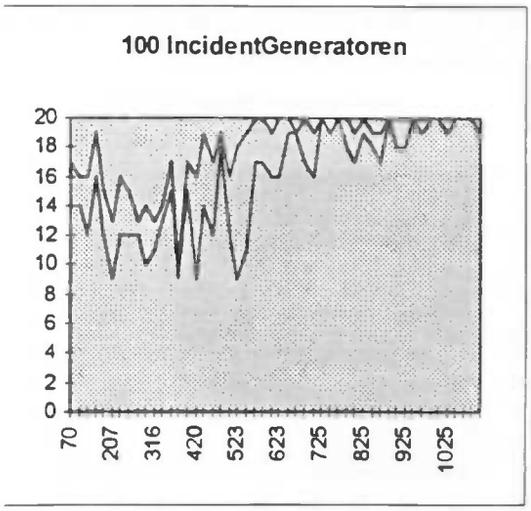


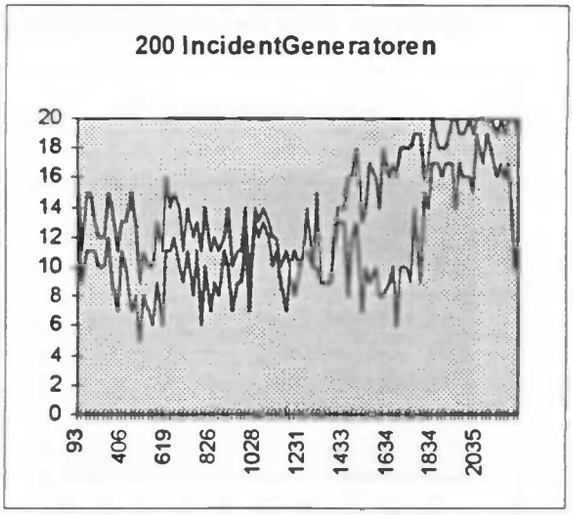


7.4.4.2.2 De testomgeving met extra IncidentGeneratoren

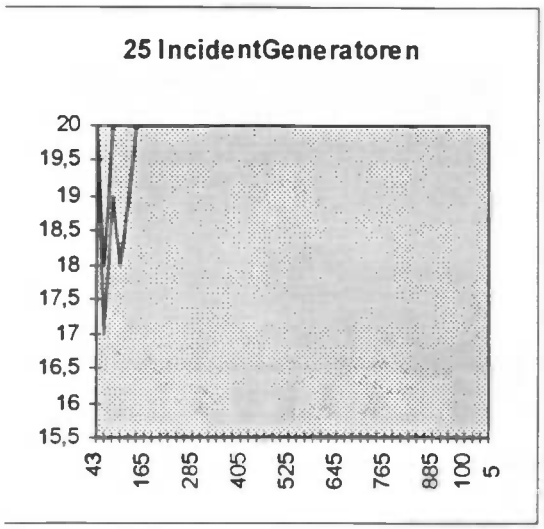


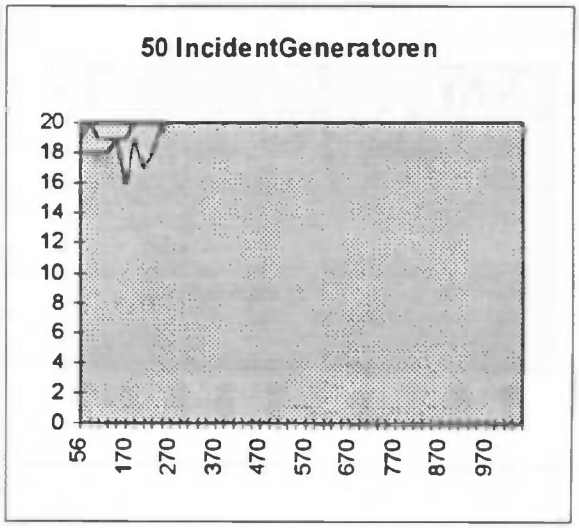


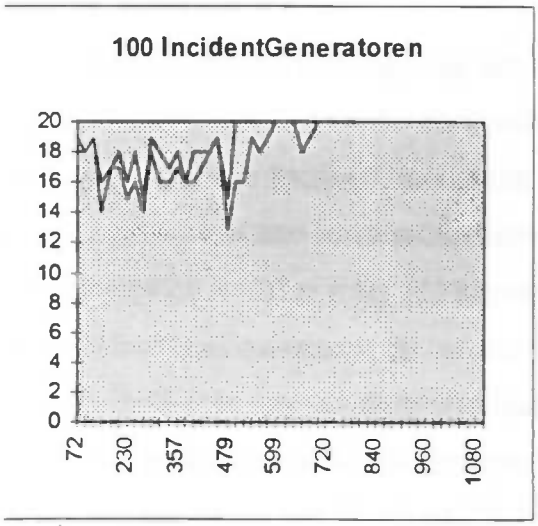


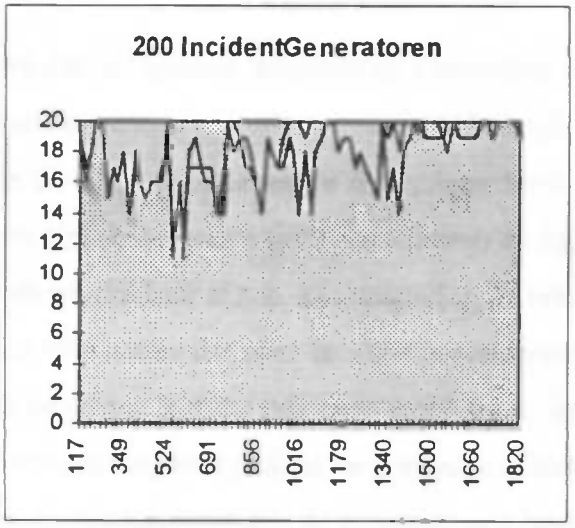


7.4.4.2.3 De testomgeving met extra keywords

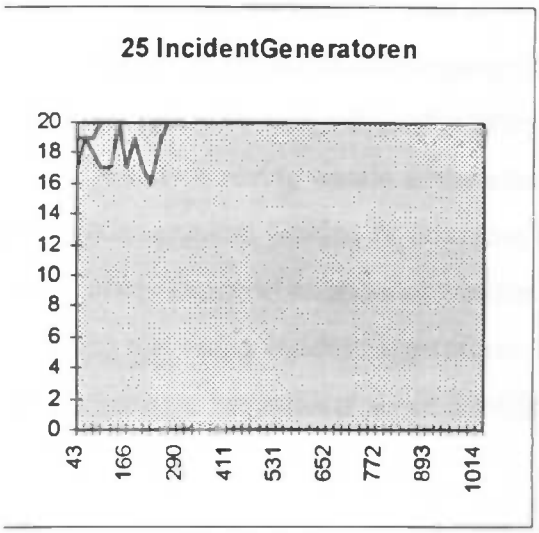


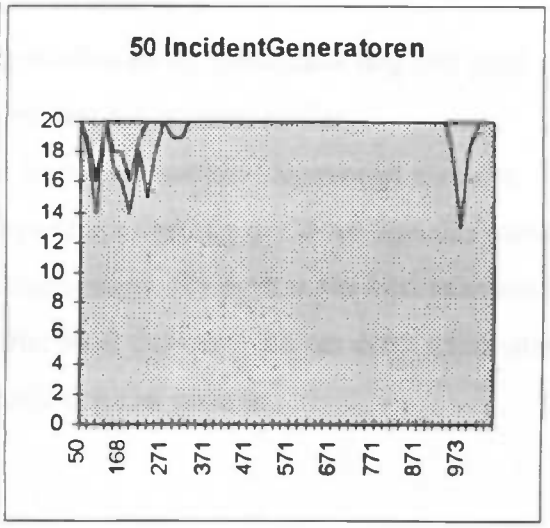


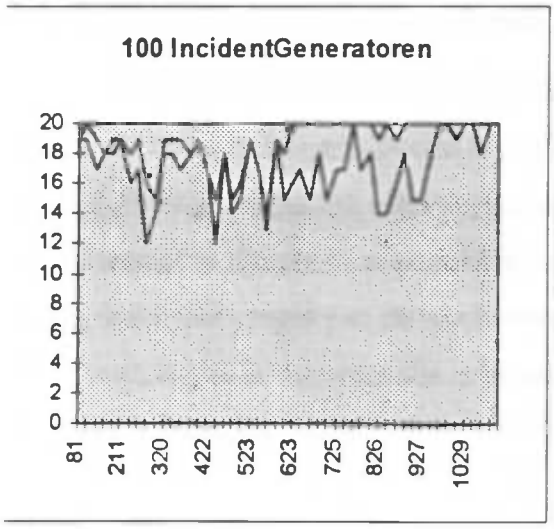


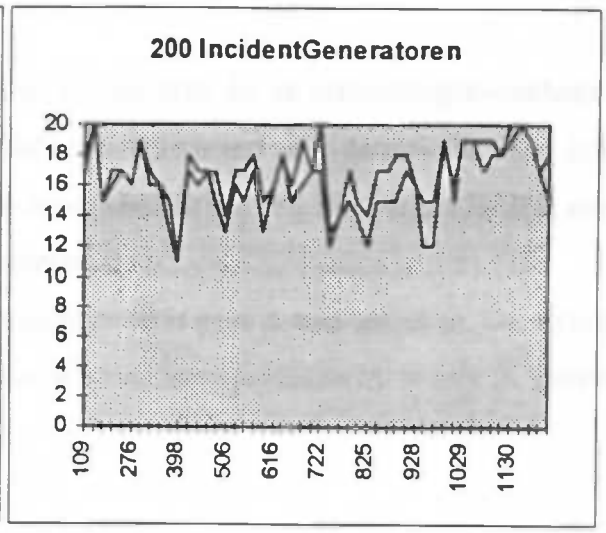


7.4.4.2.4 De testomgeving met extra IncidentGeneratoren en extra keywords









7.4.4.3 Conclusies

In de standaard testomgeving zien we dat het systeem langzaam de kenmerken van de IncidentGeneratoren leert en dat de classificatieprestaties na verloop van tijd behoorlijk goed worden. Hoe minder IncidentGeneratoren in de omgeving hoe sneller het systeem leert. Maar ook in de omgeving met 200 IncidentGeneratoren leert het systeem goed classificeren en dat is toch vrij opmerkelijk omdat er maar 100 keywords beschikbaar zijn in elke omgeving. In een omgeving met veel IncidentGeneratoren zal het daarom voorkomen dat twee IncidentGeneratoren ongeveer dezelfde incidenten creëren. In de standaard omgeving lijkt dat nog geen probleem te zijn.

Als er nieuwe IncidentGeneratoren worden toegevoegd gaan de prestaties achteruit. In de testomgevingen met 25, 50 en 100 IncidentGeneratoren zijn de prestaties nog steeds goed en vergelijkbaar met de standaard omgeving. In de omgeving met 200 IncidentGeneratoren gaat het totaal aan het eind al behoorlijk naar boven.

Met extra keywords die toegevoegd worden als de classificatie nog niet goed is worden de prestaties onder alle omstandigheden beter en leert het systeem sneller.

In een omgeving waarin zowel extra keywords worden toegevoegd als extra IncidentGeneratoren worden de prestaties behoorlijk omhoog getild ten opzichte van de omgeving waarin alleen IncidentGeneratoren worden toegevoegd. Dit geldt in het bijzonder voor de omgeving met vijftig IncidentGeneratoren. Het heeft dus zeker zin om extra informatie te verzamelen over het incident als de classificatie nog niet goed is.

7.5 Conclusies over de algoritmen

Na deze verschillende tests kunnen we concluderen dat de verhoudingen-methode die tijdens dit onderzoek is ontwikkeld heel goede prestaties levert en in de praktijk zeker bruikbaar zal zijn. Het toevoegen van extra keywords als de classificatie nog niet perfect is blijkt een hele goede manier te zijn om de prestaties te verbeteren.

Het ordenen van vragen met de beschreven algoritmen is geen succes gebleken. De verschillende algoritmen slagen er onvoldoende in de meest discriminerende vragen op te sporen. Hiervoor is nog meer onderzoek nodig.

8. Conclusies en aanbevelingen

In dit laatste hoofdstuk geven we een aantal conclusies die getrokken kunnen worden uit de ervaringen die opgedaan zijn tijdens het onderzoek.

8.1 ITIL

ITIL is heel goed te gebruiken om ondersteuning te bieden aan de gebruikers van een IT-infrastructuur in een organisatie. ITIL beschrijft heel duidelijk wat er op elk moment gebeuren moet. Hoe dat gedaan moet worden blijft echter onbesproken.

Door de duidelijke structuur van ITIL was het geen groot probleem om het te implementeren. Het administratieve gedeelte was na een week werken al heel redelijk geïmplementeerd. Dit was mede te danken aan de gebruikte ontwikkelmethode.

8.2 OOSE

Het zijn vooral de use cases die het ontwikkelen van een systeem met OOSE tot een vrij eenvoudige taak maken. Als de entiteiten in het problem domain eenmaal goed gedefinieerd zijn kan het systeem per use case geïmplementeerd worden.

De gebruikte ontwikkelomgeving, die bestond uit VisualWorks en Objectivity, bood veel ondersteuning tijdens het ontwikkelen.

Één van de grote voordelen van Objectivity is dat relaties tussen objecten automatisch integer gehouden worden en dat de database volledig geïntegreerd is in VisualWorks.

VisualWorks biedt een complete programmeeromgeving die erg prettig werkt en eventueel naar eigen wens aangepast en uitgebreid kan worden. Tijdens het ontwikkelen hebben ik gebruik gemaakt van een extensie van VisualWorks, geschreven door Howard (Howard, 1995), die het ontwikkelen van applicaties nog eenvoudiger maakt dan het al was.

8.3 Ondersteuning van analyse

8.3.1 Ordenen van problemen

Het grootste deel van de tijd die ik besteed heb aan dit onderzoek heb ik gestoken in het bedenken, implementeren en testen van verschillende methoden om problemen en vragen te ordenen op basis van de beschikbare informatie.

Niet alle methoden werken even goed. De methoden van Bayes en Quinlan om problemen te ordenen doen het aardig, maar niet zo goed dat helpdeskmedewerkers er heel erg enthousiast over zouden worden. Met de verhoudingen-methode kunnen problemen echter uitstekend geordend worden. Zo goed zelfs, dat het volgens mij interessant is om de techniek in te bouwen in een commerciële helpdeskapplicatie.

8.3.2 Ordenen van vragen

De methoden om vragen te ordenen op discriminerend vermogen werken niet goed. Het probleem ligt volgens mij bij de verdeling van de vragen over de problemen. Het totale aantal vragen zal in de praktijk waarschijnlijk groot worden, terwijl het aantal vragen dat bij meerdere problemen voorkomt relatief klein zal blijven. Om onderscheid te kunnen maken tussen twee problemen is het nodig dat bij beide problemen dezelfde vraag voorkomt, met een verschillende verdeling van de antwoorden. In de praktijk zal dat volgens mij niet zo vaak voorkomen. Tijdens het testen waren de prestaties ronduit slecht.

8.4 Aanbevelingen

Het ordenen van problemen op basis van keywords gaat heel goed, zeker als er na een eerste ordening nog één of meer extra keywords worden toegevoegd aan het incident. Ik denk dat het nuttig zou zijn om de gebruiker na de eerste ordening een lijst te presenteren met keywords die voorkomen bij de eerste vijf problemen. Als de gebruiker uit die lijst een keyword kiest dan zal dat de classificatie-prestaties waarschijnlijk erg ten goede komen.

Het bepalen van het discriminerend vermogen van vragen moet nog nader onderzocht worden. Als het lukt om daarvoor een goede methode te vinden dan is een computersysteem dat gebruikers helpt zelf hun problemen op te lossen binnen handbereik.

Ook het idee om vragen te beantwoorden en acties of oplossingen uit te voeren met behulp van recepten is volgens mij erg krachtig en zou beslist nader onderzocht en toegepast moeten worden.

9. Gebruikte literatuur

Hieronder staat een overzicht van de literatuur die tijdens het onderzoek is gebruikt. Het overzicht bestaat uit twee delen. Het eerste deel bevat de gebruikte boeken. Het tweede deel de handleidingen die gebruikt werden tijdens het ontwikkelen van het helpdesksysteem.

9.1 Boeken

Hopkins, T en Horan, B. 1995. **Smalltalk, an introduction to application development using VisualWorks**. Prentice Hall.

Howard, T. 1995. **The Smalltalk Developer's Guide to VisualWorks**. Sigs Books New York

Koppens, S en Meyberg, B. 1993. **Operationeel beheer van informatiesystemen**. Deventer: Kluwer Bedrijfswetenschappen Deventer.

Mood, A.M. en Graybill, F.A. 1963. **Introduction to the Theory of Statistics.** New York: McGraw-Hill

Quinlan, J.R. 1993. **C4.5: Programs for machine learning**. Morgan-Kaufman.

Rich, E en Knight, K. 1991. **Artificial Intelligence.** New York: McGraw-Hill

9.2 Handleidingen

VisualWorks Cookbook. 1995. ParkPlace Digital, Inc.

Objectivity/DB Guide. 1995. Objectivity Inc.

INDEX

A

aanloopperiode 7-81
actor 2-13; 2-14
analysis model 2-12; 2-15; 5-37; 5-51
analysis process 2-13

B

Bayes 4-24
belief-waarde 4-24; 7-81
beslissingsboom 4-32

C

C4.5 4-32
call 6-58
classificatie 1-2
classificeren 2-10; 3-18; 4-22; 7-82
control object 2-16

D

design model 2-12; 2-17
discriminerend vermogen 4-22; 4-28; 7-81; 8-99

E

entity object 2-15
extend use case 2-14; 5-45

F

fout 2-4; 2-9; 2-11

G

gain 4-32

H

helpdesk 1-1; 2-6; 2-7
helper 5-37; 5-38

I

implementation model 2-12; 2-17
incident 1-2; 2-4; 2-9; 2-15; 4-27; 6-59
incidentbeheer 2-7
Information Technology Infrastructure Library *Zie* ITIL
interface beschrijving 5-37
interface object 2-16
IT-dienstverlening 2-4; 2-5
ITIL 2-4; 8-98
IT-infrastructuur 2-4; 6-65
Ivar Jacobson 2-11

K

keyword 3-19; 5-47; 6-67
keywordverdeling 3-19; 6-79

O

Object Oriented Software Engineering 2-11
Objectivity 8-98
OOSE 2-11; 5-36
oplossing 6-65

P

probleem 1-2; 2-4; 2-9; 4-26; 4-30; 6-63
Probleembeheer 2-9
probleembeheerder 5-37; 5-38; 5-43
probleemstelling 1-2
problem domain 2-15; 6-54
problem domain model ... 5-37 *Zie* problem domain
propositie 3-19; 4-23; 4-27
propositieverdeling 3-20; 6-79

R

recept 5-38; 6-70; 8-99
receptontwikkelaar 5-38; 5-39
requirements model 2-12; 2-13; 5-36; 5-51
requirements specification 2-12; 2-13; 5-36

S

simulatie 7-81; 7-82
standaarddeviatie 4-30

T

test model 2-12; 2-17
testomgeving 7-82; 7-83
theorema van Bayes 4-24

U

use case 2-13; 2-14
use case driven design 2-14
use case model 2-14; 5-37

V

VisualWorks 8-98

W

wijziging 2-4; 5-43; 6-65; 6-70
workaround 2-5; 6-65; 6-70