Formele talen voor een formele RE

Bachelorscriptie Informatiekunde

Radboud Universiteit Nijmegen

Hensley Hanse

23-01-2007

Inhoudsopgave

Inleiding

Probleemstelling

In dit onderzoek is voor de probleemstelling een vraag gekozen die tevens als belangrijkste vraag dient, waarop het onderzoek antwoord moet geven.

Onderzoeksvraag:

> *"Welke formele talen en/of technieken en methodes die daarbij horen zijn bedoeld of geschikt voor formele Requirements Engineering?"*

Verantwoording

Tot heden zijn er een groot aantal talen ontwikkeld, waarvan er beweerd wordt dat ze geschikt en/of bedoeld zijn voor het specificeren in een systeem ontwikkelproces. Van deze talen ontbreekt er enig overzicht van welke talen het hier allemaal over gaan en wat precies hun afzonderlijke eigenschappen zijn. Nu zijn er onderzoekers die geïnteresseerd zijn in formele RE(Requirements Engineering), waarbij er getracht wordt om RE formeel te maken, maar die geen overzicht hebben van alle formele talen die er bestaan en gebruikt kunnen worden hierbij. Op hun eigen terrein zijn er onderzoekers die heel bekend zijn met formele talen en anderen met RE. Om verder onderzoek naar een geformaliseerde RE, is er kennis van beide kennisgebieden nodig om dit te realiseren. Daar blijkt een grote kloof te zijn [3]. Met mijn onderzoek wil ik een eerste steen zetten voor het verder onderzoek naar formele RE in de vorm van een overzicht van de verschillende formele talen en/of technieken en methodes die daarbij gebruikt worden, waarvan er beweerd wordt dat ze geschikt of bedoeld zijn voor een formele RE. Door mijn overzicht is de lezer(onderzoeker) zelf in staat de eigenschappen van de verschillende talen te vergelijken en/of een oordeel te vormen over het wel of niet geschikt zijn van de taal voor de taak (formele RE). Het is niet de bedoeling dat ik als onderzoeker zelf een oordeel geef over het wel of niet geschikt zijn van de talen, dat wordt aan de lezer gelaten. Dat er niet al een overzicht bestaat van deze talen en ik in mijn onderzoek op zoek ga naar die talen, betekent dat niet dat mijn overzicht volledig is aangezien er voortdurend onderzoek en ontwikkeling is met betrekking tot nieuwe talen.

Theoretisch kader

Dit onderzoek richt zich op voornamelijk op twee kennisgebieden binnen het vakgebied van de Informatiekunde, namelijk Requirements Engineering en Formele Talen. Het gaat namelijk om het onderzoeksthema formele RE dat op de raakvalk zit tussen Requirements Engineering en formele talen waarbij getracht wordt om

1. een overbrugging te maken tussen het informele RE(waarbij gebruikt wordt gemaakt van natuurlijke taal) en het formele van programmeertalen die gebruikt zullen worden verder in het systeem-ontwikkelproces
2. te kunnen redeneren over de specificatie(dus bij. tegenstrijdigheden uit de requirements te halen)

Voor dit allemaal zou een formele taal met zijn exacte opbouw goed van pas komen om het informele die er nu bestaat in RE, door meestal gebruik te maken van natuurlijke taal, te vertalen in een formele RE. Met de talen alleen zonder methodes en technieken (richtlijnen/processen en hulpmiddelen)om ermee te werken bereikt je niet zoveel. Er bestaan veel talen, die onderverdeeld kunnen worden in informeel, semi-formeel en (volledig) formele talen, waarbij de semantiek of de syntax formeel is of allebei. Bij een

informele taal zijn de semantiek en syntax niet precies beschreven, bij semi-formeel is alleen de syntax precies beschreven en bij formele talen zijn de syntax en semantiek allebei precies beschreven. In dit onderzoek worden semi-formele en formele talen meegenomen, doordat er ook methodes en technieken kunnen bestaan dat het geheel formeel maakt. Vandaar dat de methodes en technieken ook belangrijk kunnen zijn en in bijzondere gevallen zullen worden meegenomen in het overzicht.

Methode

Zoals al genoemd in het onderdeel hierboven zal het onderzoek richten op semi-formele en formele talen en/of technieken en methodes en vromen deze het domein van dit onderzoek.Verder heeft dit onderzoek een beschrijvende functie waarbij de talen, methodes en/of technieken die geschikt of bedoeld zijn voor formele RE in kaart gebracht zullen worden. De deelvragen die gebruikt zullen worden om een structuur aan het onderzoek te geven zijn:

- Wat houdt Requirements Engineering globaal in en hoe kunnen formele talen daarbij worden gebruikt?
- Wat zijn formele talen en wat zijn hun globale eigenschappen?
- Wat houdt de indeling informeel, semi-formeel en formeel in?
- Welke zijn de eigenschappen van een taal die belangrijk zijn bij hun geschiktheid voor RE en die ook gebruikt kunnen worden om talen te vergelijken?
- Welke talen bevatten die eigenschappen?

Dit allemaal ga ik d.m.v. literatuurstudie proberen zo volledig mogelijk te beantwoorden.

Begeleider

dr. S.J.B.A. (Stijn) Hoppenbrouwers
Faculty of Science
Toernooiveld 1
Nijmegen

Tevens wil ik hem ook alvast bedanken voor de begeleiding met betrekking tot dit traject.
.

Aanpak en structuur

Aanpak

Al bij het begin van dit traject, bij het formuleren van een onderzoeksvraag, werd het mij duidelijk dat er verschillende interpretaties bestonden van dingen waarvan ik dacht dat er alleen één betekenis van waren. Bij het zoeken en doornemen van literatuur hiervoor ben ik steeds verschillende definities tegengekomen van één bepaalde term zoals voor de term formeel. Om van het begin af een eenduidige betekenis te geven aan alle concepten binnen dit traject (onderzoeksvraag en context), vond ik het nodig om het domein waarin dit onderzoek zich afspeelt op mijn eigen manier te gaan modelleren, zodanig dat de betekenis van alle gebruikte concepten duidelijk werd voor mezelf en ook voor anderen die dit moeten lezen.
Behalve het beschrijven van concepten heb ik ook gekozen voor een top-down benadering van het geheel. Dit komt door het aantal verschillende benaderingen van Requirements Engineering dat ik ben tegengekomen in de literatuur in het begin fase. Hierdoor was het voor mij niet helemaal duidelijk meer wat waar hoorde in het geheel. Daarom ben ik begonnen van het hoogste niveau bij Requirements Engineering (RE) in zijn geheel. Eerst wordt het kennisgebied Requirements Engineering (RE) afgebakend en vervolgens daal ik stapsgewijs af tot op het niveau van de gewenste gebruikseigenschappen van een formele specificatietaal binnen RE.
Was het maar zó simpel! Er zijn verschillende aspecten dat dit traject toch lastig maken. Aangezien er geen methode is om een dergelijke literatuurstudie aan te pakken, is het heel moeilijk om jezelf een weg te banen door al die literatuur over dit onderwerp. Wat ik hier heb geleerd, is dat je bij een literatuurstudie geleid wordt door allerlei kleine vragen dat bij jezelf opkomt en waarop je daarna antwoorden gaat zoeken. Zo kom je uiteindelijk tot een structuur waarin alle concepten en ook hun relaties tot elkaar beschreven staan (domein model). Een ander probleem is het omgaan met die verschillende definities. De meeste auteurs uit de literatuur geven hun eigen invulling aan elke concept. Soms zijn ze heel verschillend en soms overlappen ze elkaar grotendeels en andere keren staat er nog dat éne aspect bij dat het anders maakt dan de vorige. De truc is nu om een midden weg te vinden tussen al die verschillende definities om zo tot eentje te komen dat het concept het beste omvat, zonder al die verschillenden aan elkaar te plakken en één geheel van te maken. Dat valt niet altijd mee! Tenslotte is er ook een neiging om van het onderwerp af te dwalen. Door dieper te gaan in de literatuur en steeds nieuwe concepten tegen te komen, ben je geneigd iets dieper te gaan in die concepten met de hoop meer licht te scheppen op het geheel. Zo ben ik afgedwaald op het pad van formele methodes en het heeft me veel tijd gekost.

Zoektocht

Bij mijn zoektocht naar die geschikte formele talen heb ik voornamelijk gebruikt gemaakt van zoekmachines en portals op het Internet. Dit heb ik gedaan middels zoekmachines zoals Google Scholar, Catalogus van het Radboud Universiteit, Google zelf, CiteSeer.IST[1] en zoek mogelijkheden op IEEE Computer Society's Digital Library[2], SpringerLink[3].
Voor formele definities zoals die van een formele taal heb ik gebruikt gemaakt van boeken. Zo heb ik ook informatie gehaald uit diverse boeken van verschillende vakken die ik heb

[1] http://citeseer.ist.psu.edu/cis
[2] http://www.computer.org/portal/site/ieeecs/
[3] http://www.springerlink.com/home/main.mpx

gevolgd hier op de universiteit. Op dit manier kon ik ook een link maken tussen verschillende onderwerpen dat ik tegen ben gekomen in mijn studie tot nu.
Een andere manier om bij literatuur te komen was de verwijzingen in artikels of boeken dat ik had doorgenomen. Zo kon ik meer uitgebreide informatie krijgen over bepaalde relevante onderwerpen.

Structuur

Voor de presentatie van de resultaten van dit traject heb ik de volgende opbouw gekozen. In deel 1 geef ik het theoretische kader weer bij dit traject met in hoofdstuk 1 een beschrijving van RE, de rol van talen binnen RE en wat er precies met formele RE wordt bedoeld. Vervolgens wordt in hoofdstuk 2 uitgelegd wat formele talen zijn, wat de indeling informele, semi-formele en formele talen inhoudt en wordt als laatste ook de rol van die formele talen binnen RE omschreven. Tenslotte worden de gewenste gebruikseigenschappen van zo'n formele taal voor hun gebruik binnen RE in onderdeel 3 opgesomd. Het resultaat van dit onderzoek in de vorm van de gevonden talen wordt in onderdeel 4 van Deel 2 weergegeven met aan het einde een conclusie.
Het is uiteindelijk de bedoeling dat elk onderdeel een antwoord geeft op een deelvraag genoemd hierboven met het onderzoeksresultaat aan het einde.

Deel 1: Theoretische kader

In dit gedeelte wordt het theoretische kader beschreven waarin dit onderzoek zich afspeelt. Hier worden de verschillende termen en concepten van de onderzoeksvraag uitgelegd en hun onderlinge relaties beschreven.

Hoofdstuk 1 (Formele) Requirements Engineering

Dit hoofdstuk begint met een beschrijving van wat er in deze scriptie onder RE wordt verstaan en de belangrijkste aspecten daarvan worden op een rijtje gezet. De rol van talen binnen RE wordt ook duidelijk gemaakt mbv de bestaande literatuur over RE. Ten slotte wordt ook beschreven wat er met Formele RE wordt bedoeld. Maar nu eerst RE

1.1 Requirements Engineering

Over de jaren van onderzoek op dit gebied van software ontwikkeling zijn er verschillende definities van RE ontstaan in praktijk en literatuur. Bijvoorbeeld:

> "Het systematische gebruik van bewezen
> principes, technieken, talen en
> gereedschappen voor een kosten-effectieve
> analyse, documentatie, en voortschrijdende
> evolutie van gebruikersbehoeften (belanghebbenden – stakeholders)en de
> specificatie van het gedrag van een systeem
> afgestemd op die behoeften." [4]

> "Requirements engineering (RE) aims at developing
> methods and tools which support the construction of
> complete, consistent, and unambiguous specifications
> describing what a software system under development
> is supposed to do." [1]

> "the Art and Science of gathering and specifying what users and other relevant stakeholders demand and expect of a future software system" [7]

Het ene ziet het als een proces en het ander als een activiteit. Waar onderzoekers en wetenschappers op dit gebied het wel over eens zijn is het belang van deze fase voor het systeemontwikkeltraject.
Om duidelijk te maken waar we het over hebben als we het hebben over RE zonder verloren te raken in de zee van literatuur en verschillende benaderingen die er bestaan van RE, wordt er hier gebruikt gemaakt van de benadering van Pohl van het proces als geheel. Hier wordt er vanuit een abstract niveau gekeken naar RE als proces in tegenstelling tot de literatuur die in detail kijkt naar her RE-proces en de afzonderlijke activiteiten (elicitatie…verificatie) waaruit men denkt dat het bestaat.
Aangezien er geen standaard [5] pakket bestaat van activiteiten waaruit een RE-proces moet bestaan zullen er dus verschillende views zijn op het geheel aan activiteiten waaruit het moet bestaan, die al dan wel of niet overeenkomen.

Wat er onder Requirements Engineering wordt verstaan, wordt door Pohl [12] niet gekarakteriseerd door de gestructureerde verzameling activiteiten die men moet doen, maar aan de hand van een drietal hoofddoelen, dat in deze fase bereikt moeten worden.

Hier wordt er uitgegaan van RE als een proces, deel van het systeemontwikkeltraject, waarbij een bepaalde input hoort en een bepaalde **gewenste** output uitkomt, ongeacht de indeling van het proces, waarbij een drietal hoofddoelen bereikt moeten worden.

Deze doelen zijn:

- het verbeteren van een ondoorzichtig/onduidelijk systeembegrip in een volledige systeemspecificatie;
- het verstrekken van geïntegreerde representaties en het ondersteunen van de transformatie tussen hen;
- het bereiken van een gemeenschappelijke overeenkomst over de specificatie uit de verschillende persoonlijke views van betrokkenen;

Zo kan men uitgaan van de ondoorzichtige en onduidelijke beelden van alle betrokkenen (stakeholders) van het te bouwen systeem aan het begin van het systeemontwikkeltraject als input van het RE-proces, tot een compleet systeem specificatie aan het einde van het RE proces waar alle betrokkenen het over eens zijn.

Een ander punt waar onderzoekers op RE-gebied het over eens zijn is, dat er aan het einde van het RE-poces een requirements specificatie document uit komt rollen, dus een gewenste output.

De belangrijkste aspecten van het RE-proces kunnen volgens Pohl verdeeld worden in drie dimensies. Gebruikmakend van zijn framework en kijkend naar bestaand onderzoek naar RE, kunnen de verschillende benaderingen die er zijn van RE geclassificeerd worden aan de hand van deze drie dimensies.

Wat dit artikel heel goed bruikbaar maakt voor dit project is de representatie dimensie uit het framework, waarin expliciet aandacht wordt besteed aan de representaties die gebruikt worden om kennis over het systeem uit te drukken (en uit te wisselen).

De meeste literatuur over RE hebben het over representatie alleen aan het einde van het RE proces wanneer alles gespecificeerd moeten worden in de vorm van een requirements specificatie document, ze hebben het niet over de representaties binnen het hele proces waarbij de wensen van bijvoorbeeld de gebruikers in hun eigen taal worden uitgedrukt/ vastgelegd en dus in die vorm deel zal uitmaken van het gehele proces.
Aangezien het RE proces een interactief en tevens iteratief proces is waarbij er veel communicatie over en weer is tussen engineer en klant (gebruiker, domeindeskundige) om zo alle requirements voor het te bouwen systeem boven water te krijgen en zo precies mogelijk te specificeren, zal de documentatie van deze requirements en requirements specificaties een belangrijke rol spelen bij de communicatie tussen de verschillende partijen omtrent die requirements om deze zo goed mogelijk te definiëren of specificeren. Hier zien we alweer de rol van representatie binnen het RE proces als geheel en dat het zich niet alleen beperkt tot het maken van de uiteindelijke specificatie.

Voor dit project zijn de neven activiteiten waaruit het RE-proces kan bestaan niet zozeer van belang als de documentatie die geproduceerd en gebruikt wordt tijdens het proces, aangezien daarbij taal een belangrijke rol speelt.

1.2 Taal binnen RE

Zoals we in de vorige paragraaf hebben gezien speelt representatie een belangrijke rol in het RE-proces. Hier komt ook taal bij te pas.
Talen kunnen gebruikt worden in het RE-proces als een representatie middel om kennis van en de verschillende views op het systeem uit te drukken en uit te wisselen o.a. in de vorm van documenten die op hun beurt als communicatiemiddel gebruikt zullen worden in het gehele software ontwikkeltraject.
Aangezien dat behalve in geschriften, taal ook in gesproken vorm kan voorkomen, zullen ons slechts richten op taal in geschreven vorm.

Binnen RE zijn er verschillende categorieën van representaties en dus ook van talen (zie 2.1) die gebruikt kunnen worden [12]:

- *informeel*
- *semi-formeel*
- *formeel*

Men heeft het hier over het representeren en/of beschrijven van de verschillende specificaties met behulp van taal, van informeel tot formeel. Vandaar dat we het hier niet alleen zullen hebben over *representatietalen*, maar ook talen om te specificeren.
Een *specificatietaal* is taal die gebruikt kan worden om er specificaties mee te representeren, omdat die (compleet, ondubbelzinnig enz.) moeten zijn en omdat we in RE van een onduidelijke representatie vanuit een idee van een stakeholder uiteindelijk bij een specificatie van het systeem willen komen, moet alles vertaald worden in een specificatietaal waardoor het uiteindelijk ook gebruikt kan worden om later in het ontwikkeltraject te kunnen testen enz. en ook als basis voor het systeemontwerp. Specificatietaal is dus een verbijzondering van de representatietaal (deelverzameling) te gebruiken om te specificeren.

Een specificatie kan volgens het artikel van Pohl opaque zijn tot compleet. Dus als we het over een specificatie hebben in deze scriptie, betekent dat niet meteen dat het helemaal precies is, maar het kan ook opaque/onduidelijk zijn.

Waar wij in dit project geïnteresseerd zijn, zijn die formele of semi-formele talen die in het *software ontwikkeltraject in het bijzonder de RE-proces* gebruikt kunnen worden om de verschillende requirements specificaties te representeren van opaque tot complete (dus vanaf het begin tot het uiteindelijke product van de requirements specificatie document). Doordat we het hier hebben over *formele* talen om te gebruiken om specificaties te kunnen representeren (en/of specificeren) hebben we dan te maken met (dus) formele specificatietalen. Niet alle formele talen kunnen als specificatietaal gebruikt worden maar hierover meer in de volgende hoofdstukken.

1.3 Formele RE

Formele RE is RE waarbij gebruikt wordt gemaakt van formele methoden, technieken en de daarbij behorende formele taal(-en).
Iets dat heel belangrijk is om te melden is het feit dat in dit document er twee verschillende interpretaties van het woord 'formeel' worden gebruikt. Het gaat hier om de interpretatie van formeel in die zin dat er op een heel precieze en gestructureerde manier wordt omgegaan in het proces en de andere zoals bedoeld is in de theorie van formele talen (wiskundig) (zie 2.1.3).
Het gehele RE-proces kan onmogelijk formeel zijn, want men begint met de ideeën van de stakeholders die in een informele taal zijn uitgedrukt (gesproken of geschreven in natuurlijke taal) . Wat men wel kan doen is formele methoden [2] en talen gebruiken zodanig dat men aan het einde een formele specificatie als product krijgt. Hoewel de formele methoden een belangrijke rol spelen in formele RE, gaat de focus in dit project primair naar de formele talen en hun bijbehorende methoden en technieken en niet andersom. (niet al die methoden die bij de formele talen horen zijn formele methoden).

Waarom formaliteit nodig? [8,3]

Formaliteit is nodig om correctheid en compleetheid te waarborgen gedurende het gehele proces van systeem ontwikkeling en hier in het bijzonder voor het RE-fase.
Formaliteit brengt een serie voordelen met zich mee.Door informele formuleringen van klanten te vertalen naar formele constructies, duiken er vragen op bij de engineers waardoor het mogelijk wordt dat er fouten of tegenstellingen gedetecteerd kunnen worden. Een ander voordeel dat formaliteit met zich meebrengt is dat de semantiek van een gebruikte formele taal het mogelijk maakt dat er duidelijke interpretatieregels zijn, waardoor problemen van natuurlijke taal zoals dubbelzinnigheid omzeild kunnen worden. Meer over semantiek in paragraaf 2.1.

Hoofdstuk 2 Formele talen

In dit project hebben we steeds te maken met formele talen, want er wordt namelijk gezocht naar die formele talen en technieken waarvan beweerd wordt dat ze geschikt zijn voor een formele RE. Maar wat houden die formele talen nou eigenlijk in? Waardoor worden ze gekarakteriseerd? Wat zijn hun belangrijkste aspecten? In dit hoofdstuk wordt er een korte beschrijving gegeven van wat een taal inhoudt, welke de verschillende gradaties zijn van formaliteit van een taal, en wordt er getracht een relatie te plaatsen tussen formele talen en RE. We zeggen ook iets over de rol van formele talen binnen Formele RE.

2.1 Informele, semi-formele en formele talen

Voordat er een definitie en een beschrijving wordt gegeven van wat een formele taal is en wat dat inhoudt, moet eerst duidelijk worden weergegeven wat er onder een taal wordt verstaan. In dit project hebben we het namelijk niet alleen over formele talen, maar ook die semi-formele talen die ingezet kunnen worden in een formele Requirements Engineering proces. Vandaar dat we eerst een algemene definitie van taal neer moeten zetten.

Taal

Om het formeel te houden zullen we deze generieke definitie gebruiken [12]. Een **taal** wordt gedefinieerd als iedere verzameling zinnen over een *alfabet*, waarbij een alfabet uit iedere eindige verzameling *symbolen* bestaat en een *zin over een alfabet* iedere string van eindige lengte van symbolen uit het alfabet omvat. In dit opzicht zijn woorden zoals wij dat kennen en zinnen synoniemen van elkaar. Deze generieke definitie kunnen we ook toepassen bij gesproken taal waarbij geluidspatronen gebruikt worden als symbolen en gemakkelijk vertaald kunnen worden in geschreven symbolen met weinig verlies van informatie.
Elke taal heeft ook z'n *grammatica* die de structuren van zinnen en woorden uit die taal regelt. De grammatica van een taal is dan ook een systeem van regels en principes dat gebruikt worden bij het samenstellen van correcte woorden of zinnen in een taal.
Met behulp van deze karakteristieken van taal kunnen we dan overstappen naar twee aspecten van taal, die relevant zijn voor ons onderzoek, namelijk syntax en semantiek.

Syntax

Een aspect van taal dat hier van belang is, is het aspect van syntax. Zoals van Lamsweerde in z'n artikel [8] beweert, gaat het hier om de vorm van zinnen in een bepaalde taal. De syntax van een taal zijn regels om te bepalen of expressies die in een bepaalde taal zijn gemaakt de correcte vorm hebben conform de regels(syntax) van die taal en dus geldig zijn. Dus er moet precies vastliggen wat de geldige expressies in die taal zijn. Als we dit concept dan relateert aan de definitie van taal hierboven, dan heeft het te maken met de volgende karakteristiek van taal: grammatica. De syntax van een taal wordt gedefinieerd door de grammatica van die taal. Om preciezer te zijn gaat het hier om de descriptieve perspectief van de regels van grammatica. Dit betekent dat men meer gericht is op het bepalen of een expressie geldig/grammaticaal is in praktijk dan dat men gericht is op hoe die correct geconstrueerd moet worden volgens de theorie. Dus een geldige expressie in een taal, is een expressie waarin de taalconstructie (vorm) goed is, doordat de onderdelen in de goede volgorde (conform de syntax) zijn samengesteld.

Semantiek

Semantiek van een taal heeft te maken met de betekenis van de expressies uitgedrukt in die bepaalde taal. Het zijn regels om expressies/taalconstructies in een bepaalde taal, die gedefinieerd zijn aan de hand van een syntax, in een precieze en betekenisvolle manier te interpreteren [8,12]. De semantiek neemt in verschillende talen verschillende vormen aan, zoals we in de volgende paragraaf zullen zien. Hier wordt tevens het belang van deze twee concepten voor deze scriptie duidelijk.

Gradaties van formaliteit

Het al geheel aan talen die er kunnen bestaan kunnen in drie categorieën ingedeeld worden naar hun gradatie van formaliteit [12]. Deze indeling wordt mogelijk door de syntax en semantiek van de taal. Het formele heeft te maken met de toepassing van wiskunde bij het vastleggen van de syntax en semantiek van de taal. Dus de taaldefinitie wordt dan wiskundig beschreven.

Om het duidelijk en begrijpelijk te houden in dit stadium, maak ik gebruik van de al eerder genoemde categorisering van representaties door Pohl om hier de verschillende gradaties van formaliteit duidelijk te maken. Aangezien er wetenschappelijke literatuur ontbreekt over het begrip semi-formele talen, leek het ook verstandig om deze beschrijving van Pohl te gebruiken voor een definiëring op gelijk niveau van de gradaties. Er wordt niet beweerd dat dit de exacte definities zijn van de termen, maar dekkend genoeg om te begrijpen wat er onder die termen wordt verstaan voor de rest van dit document.

2.1.1 Informele taal

Informele talen worden gekarakteriseerd door niet-wiskundig beschreven syntax en semantiek. Dit betekent dat er geen precieze regels zijn van welke constructies in de taal geldig en niet geldig zijn en ook geen regels voor de interpretaties daarvan. Een gevolg hiervan is dat de verzameling woorden (strings) van een dergelijk taal zo enorm groot is dat het als oneindig beschouwd kan worden. Verder kunnen de regels van de grammatica en dus ook de syntax te complex zijn om formeel te formuleren. Informele taal wordt ook gekenmerkt door ambiguïteit. Dit omdat de semantiek niet formeel is en dus geen eenduidige, precieze regels zijn om een constructie in een dergelijke taal te interpreteren. Dus van een constructie zijn er meerdere interpretaties mogelijk.

2.1.2 Semi-formele taal

Bij semi-formele talen is de syntax van de taal wel precies gedefinieerd, maar de semantiek niet. De expressies van deze talen worden vaak in structureel grafische vormen weergegeven. De verschillende mogelijke elementen en de mogelijke wijze van samenstellen hiervan worden expliciet gedefinieerd. Hierdoor zijn de diagrammen die geproduceerd worden wel duidelijk en wordt het systeem op een overzichtelijke manier weergegeven, maar toch blijft de precieze betekenis van wat er gemodelleerd wordt achterwege. Dit ook omdat er meerdere interpretaties mogelijk zijn bij één expressie/taalconstructie. Voorbeeld van een semi-formele taal is UML. Je weet wat de elementen van die diagrammen voorstaan, maar je kunt weinig zeggen over de betekenis van wat er precies gemodelleerd wordt en dus is redeneren daarover vrijwel onmogelijk.

2.1.3 Formele taal

Formele talen zijn talen waarbij beiden, de syntax en semantiek, formeel gedefinieerd zijn op een wiskundige manier en waarbij de betekenis van een constructie in die taal eenduidig vastligt. Een formeel gedefinieerde syntax geeft op een expliciete (wiskundige) manier de structuur van geldige expressies weer binnen die taal [12]. Een formeel gedefinieerde semantiek geeft een wiskundige betekenis aan de expressies uit de formele syntax, meestal d.m.v. van vertaling van deze expressies in wiskundige structuren. Dit betekent dat bij elke constructie het alleen op één bepaalde manier geïnterpreteerd kan worden en dus één betekenis kan hebben. Dit is dan ook één van de belangrijkste eigenschappen van formele talen die van belang is voor het gebruik in RE. Verder is ook redeneren mogelijk over het gerepresenteerde. Voorbeelden van formele talen zijn Eerste-Orde Predikatenlogica en programmeertalen.

2.2 Formele taal binnen RE

Zoals eerder vermeld in paragraaf 1.2 worden formele talen voornamelijk in de eindfase van RE-traject gebruikt om een specificatie van de requirements van het te bouwen systeem te produceren zodanig dat de ontwerpers daarmee meteen aan de slag kunnen. Dit betekent dus dat gedurende het RE-proces de vage ideeën van en verschillende views op het systeem die in de informele en semi-formele talen waren geformuleerd, vertaald moeten worden in een formele taal zodanig dat er een precieze, eenduidige specificatie ontstaat waarover iedereen het mee eens is. Een formele taal brengt o.a. de volgende voordelen met zich mee. Als belangrijkste voordeel, kunnen de geproduceerde formele specificaties door geautomatiseerde tools (indien beschikbaar voor betreffende taal) worden gebruikt oa [8]

- om logische gevolgen te trekken van de specificatie m.b.v. deductie, voor gebruikersbevestiging;
- om te bevestigen of dat een operationele specificatie aan meer abstracte specificaties voldoet, gedragstegenvoorbeelden te produceren als niet, door algoritmisch model dat technieken controleert;
- om tegenvoorbeelden van requirements over een bepaalde specificatie te produceren;
- om concrete scenario's te produceren die gewenste of ongewenste eigenschappen over de specificatie illustreren of, omgekeerd,
- om inductief te kijken of die ongewenste scenario’s uit de specificatie rollen;
- om bijvoorbeeld prototypes uit die specificatie te maken om zo te controleren op adequaatheid;
- om specifieke vormen van specificatie consistentie/volledigheid efficiënt te controleren;
- om te steunen bij het verbeteren en verfijnen van de specificatie en verplichte bewijzen te produceren;
- om testcases van de specificatie te produceren;
- om formeel hergebruik van componenten door specificatie aanpassing te steunen;
- om computerprogramma’s te produceren indien het executeerbaar is.

2.3 Methoden en technieken bij formele talen

Het is vaak zo dat een formele specificatietaal niet alleen op zichzelf staat, maar dat het een deel vormt van een (formele)methode [10]. Bij de meeste talen ontbreekt er een methode met aanwijzingen en hulpmiddelen voor het gebruik van deze talen [11]. Bij sommigen is het zo dat ze een deel vormen van een formele methode die naast een specificatietaal ook uit technieken en methodes bestaat. Die zijn gericht op de met deze talen geproduceerde specificaties. Zo kan een specificatie automatisch geverifieerd en gevalideerd worden met behulp van de tools, maar zijn er geen richtlijnen enz hoe je zo´n specificatie moet bouwen in zo'n formele methode. Doordat zulke talen meestal niet alleen staan en een vorm van methode daarbij hoort, zullen wij ook die meenemen bij het behandelen van die talen. Doordat de meeste van de specificatietalen van tegenwoordig zo ingewikkeld zijn dat er een vorm van methodologie daarbij nodig is om goede specificaties te produceren, is het belangrijk om ook de methodes en technieken die in welke vorm dan ook daarbij horen te betrekken bij de behandeling van deze talen.

Hoofdstuk 3 Gewenste gebruikseigenschappen van een formele taal voor RE

Uit de vorige hoofdstukken kunnen we concluderen dat we formele en semi-formele talen als specificatietalen kunnen gebruiken in RE. Bij hun gebruik als specificatietaal zijn er Om deze talen te gebruiken in RE zijn er een aantal eigenschappen dat gewenst worden geacht bij hun gebruik als specificatietaal. Hoewel er onderzoek is gedaan naar deze gewenste gebruikseigenschappen, bestaat er geen standaard lijst van deze eigenschappen. Zo hebben Tse en Pong(1991) zo´n lijst samengesteld door verschillende voorstellen van verscheidene auteurs samen te voegen. Verder komen er sommige eigenschappen steeds terug in literatuur over RE, over specificeren (hier in de vorm van eigenschappen die een specificatie moet hebben) en over specificatietalen zelf.
In de volgende paragrafen zullen een aantal eigenschappen besproken worden, samengesteld uit eigenschappen die worden genoemd in de literatuur [11,13]. Met gebruikers wordt er in dit gedeelte niet alleen die mensen bedoeld dat de taal gebruiken om specificaties te maken maar ook de stakeholders die deze specificaties gebruiken bij het nemen van beslissingen en dergelijke.

3.1 Semi-formeel/Formeel

Doordat we in dit onderzoek te maken hebben met twee categorieën van talen waarbij het verschil tussen die twee invloed heeft op de keuze bij gebruik daarvan voor het specificeren van requirements, is het van belang om ook deze eigenschap mee te nemen in deze lijst. Het heeft met verschillende dingen te maken. Zo heeft het onder andere te maken met gebruikersgericht of systeemgericht. Een semi-formele taal leunt meer naar de kant van de gebruiker toe met grafische visualisatie van het geheel en bij een formele is men meer geïnteresseerd op o.a. de mogelijkheid van automatisch code generatie. Daarnaast speelt deze eigenschap ook een rol bij andere eigenschappen die later behandeld zullen en tezijnertijd genoemd zullen worden. Verder speelt de stadia waarin de taal wordt gebruikt ook een rol. Een formele taal wordt vaker pas voor de design fase gebruikt om zo een formele specificatie te produceren voor de designers en tevens brengt dit ons bij de tweede eigenschap.

3.2 Ondersteuning voor verschillende situaties in het traject

Hiermee wordt bedoeld dat zo´n taal een gebruiker moet kunnen ondersteunen in het maken van de verschillende specificaties dat die moet produceren in al die fasen van softwareontwikkeling. Doordat er verschillende specificaties voor verschillende doeleinden worden geproduceerd gedurende het traject en om vertaal problemen tussen deze specificaties te zoveel mogelijk voorkomen is deze eigenschap zeer gewenst onder de gebruikers. Bijvoorbeeld een specificatie met algoritmische details is handig bij implementatie, maar eentje dat de hiërarchische structuur weergeeft van hetzelfde systeem in z´n geheel is gewenst bij een andere fase. Om deze vertaalslag te vermijden is zo´n taal dat de productie van beide kan ondersteunen ideaal.

3.3 Taal transparantie

Bij de ondersteuning van hierboven moet het wel duidelijk zijn hoe die één-op-één relatie tussen die verschillende soorten specificaties in elkaar steekt. De taal moet zodanig transparant zijn dat de formele aspecten van die taal begrijpbaar moeten zijn voor de stakeholders indien ze de specificatie willen analyseren. Er moet een wiskundig raamwerk aanwezig zijn in de taal zodat het mogelijk wordt een één-op-één relatie te garanderen tussen de formele zijde en gebruiksvriendelijke zijde van de taal. Er moet gegarandeerd worden dat zo´n specificatie met hiërarchische structuur van een systeem hetzelfde is als die met de algoritmische details. Hier speelt het verschil tussen semi-formele en formele talen ook een rol omdat een formele taal dit meer kan garanderen dan een semi-formele taal, aangezien de semantiek van deze niet formeel is beschreven.

3.4 Gebruikers vertrouwdheid met de taal

Een (nieuwe) taal dat niet bekend is bij de gebruikers daarvan en leidinggevenden in zo´n project van softwareontwikkeling, wordt vaak niet gebruikt in zo´n project. Dit heeft te maken met hun vertrouwen in de prestaties van zo´n nieuwe taal. Men is meer geneigd zo´n bestaande taal te gebruiken waarvan, door ervaring, bewezen is dat het een goede taal is in gebruik, dat ook over de meeste gewenste eigenschappen beschikt en populair is onder gebruikers daarvan.

3.5 Taalstijl

Hier gaat het om de representatie vormen die mogelijk zijn binnen een bepaalde taal. De drie verschillende stijlen kunnen als volgt worden omschreven

- *Tekstueel*
  Hier zijn er twee keuzes mogelijk: natuurlijke taal en eentje dat meer lijkt op een programmeertaal. In sommige situaties is het handiger om een taal te gebruiken in een vorm dat gemakkelijk te volgen is voor alle gebruikers en op andere momenten is een meer precieze beschrijving gewenst.

- *Grafisch*
  Grafische representaties worden gemaakt van meestal complexe onderwerpen waardoor het vaak begrijpelijker wordt voor de gebruiker, dan bij het gebruik van een tekstuele stijl. Zo kan een hiërarchische structuur zoals genoemd hierboven beter weergegeven worden m.b.v grafische representatie om een overzicht te geven van de verschillende lagen of componenten en relaties daartussen.

- *Hybride*
  De laatste en belangrijkste stijl vind ik, is de hybride stijl. Hiermee wordt een taal bedoeld dat in meerdere vormen bestaat. Dus een specificatie kan dan in meerdere vormen weergegeven worden, waarbij het mogelijk wordt dat ieder stakeholder een formaat kan kiezen dat het dichtst bij hem/haar ligt. Máár er moet wel een één-op-één relatie zijn tussen de verschillende syntaxen van die taal. Bijvoorbeeld hoewel formele talen een

precies gedefinieerde semantiek hebben, de betekenis daarvan wordt vaak in een natuurlijke taal uitgelegd.

3.6 Complexiteit reductie

Binnen zo´n taal moet er een mechanisme ingebouwd zijn om complexiteit te reduceren om zo het probleem in kwestie meer begrijpelijk kunnen maken voor de verschillende betrokkenen (eind gebruikers, analisten, designers, programmeurs). Dit kan op twee manieren

- *Scheiding tussen logische en fysieke karakteristieken*
  Logische karakteristieken: de essentiële eigenschappen dat het systeem moet hebben, wil men aan de requirements van de eindgebruiker voldoen.
  Fysieke karakteristieken: de manier waarop het systeem uiteindelijk functioneert.
  Het moet mogelijk zijn om m.b.v. zo´n taal gescheiden modellen te maken voor het logische en fysieke gebeuren, om zo een onderscheid te kunnen maken tussen het essentiële en niet belangrijke aspecten. Bijvoorbeeld bij het maken van een GUI moeten we het logische gebeuren "achter het scherm" kunnen scheiden van de keuze van kleuren en vorm van buttons.

- *Multi-level abstractie mogelijk*
  Een hierarchisch raamwerk is gewenst waardoor een gebruiker het te bouwen systeem makkelijk in z´n geheel kan visualiseren en ook makkelijk kan navigeren naar voor hen relevante aspecten van de specificatie. Door zo´n raamwerk wordt het ook mogelijk dat er op dezelfde tijd op verschillende subonderdelen gewerkt kan worden door verschillende mensen.

3.7 Aanpasbaar

Specificaties in die taal moeten zo gestructureerd zijn, dat wijzigingen makkelijk aangebracht kunnen worden door een willekeurig iemand dat bekend is met de taal. Dit omdat het een feit is dat requirements steeds veranderen en nieuwe technologieën moeten kunnen worden toegepast indien gewenst.

3.8 Tools

Zoals genoemd in 2.2, brengt het aanwezig zijn van geautomatiseerde tools dat om kan gaan met specificaties van de taal een scala aan voordelen met zich mee. Dus het is van belang dat er tools aanwezig zijn die behalve gebruikers helpen bij het maken van specificaties, ook ingezet kan worden bij de verificatie van het geheel.

3.9 Formele methode

Formele methodes[4] kunnen handig zijn bij het maken en verifiëren van specificaties met behulp van bijbehorende technieken [14]. Maar niet alle specificatietalen hebben een

[4] Lezer wordt geraadpleegd J.Bowen´s webpagina (http://vl.fmnet.info/)over Formele methodes te bezoeken voor meer info over het onderwerp.

formele methode waarbij ze toebehoren. Het hebben van en bijbehorende formele methode is ook een gewenste eigenschap van een formele taal voor RE.

3.10 Functiegericht

Specificaties in RE gaan over "wat" het te bouwen systeem allemaal moet kunnen in plaats van het "hoe" hij het doet [8]. Vandaar dat de specificatietaal een functiegerichte taal moet zijn in plaats van een procedurele taal, waarmee het "hoe" beschreven wordt. Hierdoor wordt het voor ontwerpers mogelijk dat ze een vrije keuze hebben in het kiezen van een ontwerp voor het systeem, onafhankelijk van de gemaakte specificatie.

3.11 Traceability

Er moet enig geautomatiseerde mechanisme aanwezig zijn bij de taal dat het mogelijk maakt dat requirements gespecificeerd in de specificatie getraceerd kunnen worden vanuit de geïmplementeerde versie en vice versa. Dit is nodig voor de verificatie van de implementatie tegen de requirements specificatie om te kijken of ze consistent zijn met elkaar.

3.12 Expressiviteit van de taal

Expressiviteit speelt een belangrijke rol als het gaat om het vastleggen van de requirements [12,9,8]. De requirements van een bepaalde te bouwen systeem moeten wel op een eenduidige manier vastgestel worden, maar daarnaast moet het ook zo compleet en specifiek mogelijk vastgelegd worden in een document dat als basis dient voor het verdere ontwikkelingstraject. Voor de gebruikers is natuurlijke taal de ideale taal waarin men zich kan uitdrukken. De uitdrukkingskracht daarvan is daardoor ook hoog. Daarom zijn er ook mensen die vinden dat hoe dichter en bepaalde taal bij natuurlijke taal ligt, hoe expressiever die taal.De uitdrukkingskracht van een taal kan op een empirisch niveau vanuit verschillende hoeken bekeken worden. Zo kan men de expressiviteit van een taal bepalen door te kijken naar de twee vragen: "Wat wil men zeggen?" en "Kan men dat ook zeggen in die taal?". Een andere optie is om dan te kijken naar bijvoorbeeld de verschil in lengte van specificaties in verschillende talen, over één bepaald probleem. Dan kun je zeggen dat de taal van de kortste specificatie expressiever is dan de andere taal met de lange specificatie(wat je wilt zeggen wordt met minder). Er bestaat dus geen standaard manier waarop je de expressiviteit van een taal kan meten en het is dus een kwestie van ervaring of kennis binnen het bedrijf te hebben over zoveel talen mogelijk. Dit zodat men de meest expressieve taal kan kiezen voor een bepaalde situatie in ontwikkeltraject.

Door het beschikken van deze gewenste eigenschappen, zullen de geproduceerde requirements specificaties ook voldoen aan de eisen voor goede specificaties genoemd in de IEEE standaard 830-1948.[5]

---

[5] Dorfman, M.; Thayer, R.H.. *Standards, Guidelines and Examples on System and Software Requirements Engineering*. IEEE Computer Society Press – Tutorial, 1990.

Deel 2: Resultaat

In de volgende deel worden de resultaten van dit onderzoek beschreven in de vorm van een verzameling van formele en semi-formele talen en technieken die geschikt kunnen zijn voor gebruik in Requirements Engineering.

Hoofdstuk 4. Formele talen en technieken/methodes geschikt voor RE

Bij het opzoeken en doornemen van de literatuur voor al die onderdelen hierboven ben ik een heleboel talen (formeel en semi-formeel) tegengekomen dat door auteurs genoemd worden als voorbeeld specificatietalen voor RE. Een aantal van die talen ben ik meerdere keren tegengekomen en worden ook als gewenst geacht bij het specificeren van requirements. Om het betrouwbaar te maken heb ik hier gekozen voor deze methode, dus het samenstellen van een lijst van die talen die aanbevolen of naar verwezen worden in de literatuur door de auteurs. Van deze lijst zal ik een selectie maken van de meest aangehaalde talen in de literatuur en zal ik een korte beschrijving geven van die talen. Hier een lijst van de talen met de bijbehorende referenties.

Wat er wel bij vermeld moet worden, is dat dit lijst alleen een *deel*verzameling vormt van al de talen die gebruikt kunnen worden bij RE. Dit is omdat men voortdurend bezig is met het zoeken, onderzoeken en/of ontwikkelen van nieuwe geschikte talen voor RE. Bijvoorbeeld AFSL (Another Formal Specification Language) [11]. Daarnaast zijn er ook talen die specifiek zijn ontwikkeld voor bijvoorbeeld een bepaald bedrijf of bepaald soort applicaties (CCS). Deze talen kunnen op een of andere manier wel gebruikt worden in de RE-fase, maar die vallen buiten de scope van dit onderzoek. Het gaat hier om de meer bekende talen uit de praktijk en waarvan beweerd wordt in de literatuur dat ze populair zijn voor gebruik in projecten.

Talen:
- Z-notation [8,12,14,16,21]
- VDM-SL (Vienna Development Method Specification Language) [8,12,14,16,21]
- RSL (RAISE Specification Language) [14,13,21]
- LOTOS (Language Of Temporal Ordering Specification) [21]
- AMN (Abstract Machina Notation) [14,**wiki**[6]]
- ORM (Object Role Modeling) [7]
- COLD (Common Object-oriented Language for Design)[11]
- PSL (Problem Statement Language)[13]
- SADT (Structured Analysis and Design Technique) [13]
- EDDA [13]
- SAMM (Systematic Activity Modeling Method) [13]
- HOS (Higher Order Software) [13]
- X-notation [2]
- CSP (Communicating Sequential Processes)[21]
- Eerste-orde predikatenlogica [14]
- CCS (Calculus of Communicating Systems) [14]
- PAISley [8,12]
- Albert|| [22]
- **Voorgesteld**: Albert [23]
- **Voorgesteld**: AFSL (Another Formal Specification language)[11]

Van deze talen wordt een korte beschrijving gegeven van de volgende talen: Z-notation, VDM-SL, RSL en ORM. Er wordt gekeken of de gewenste karakteristieken genoemd in de vorige hoofdstuk van toepassing zijn voor die talen. Verder zal er aan het einde een overzichtelijke tabel (Tabel 1.) weergegeven worden aan de hand van de resultaten van Tse

[6] Staat expliciet bij B-Method http://en.wikipedia.org/wiki/B-Method (23-01-2007) .

en Pong [13], waarin gekeken wordt naar die eigenschappen van de talen: EDDA, SAMM, RSL, PSL, SADT en HOS.

Z-notation

Z-notation is een formele specificatietaal bedoel voor het specificeren van functionele aspecten van computer systemen[14,15]. Gebaseerd op de verzamelingenleer en predikatenlogica is deze taal een van de meest gebruikte specificatietalen in software ontwikkeling. Door het gebruiken van modules en schema´s maakt het overzichtelijker en makkelijker te volgen voor gebruikers bij het analyseren van meestal lange specificaties van systemen. Zoals genoemd hierboven heeft deze taal een formele wiskundige basis en kunnen abstracte specificaties in deze taal stap voor stap verfijnd worden tot specificaties geschikt voor implementatie. Die verfijningstappen tussen de verschillende verfijningen zijn wiskundig onderbouwd. Hierdoor is Z transparant en kan bovendien verschillende situaties in het ontwikkeltraject ondersteunen. Een hierarchische structuur binnen een specificatie is mogelijk, waardoor het ook mogelijk maakt om een specificatie te maken van een onderdeel van het systeem en deze later andere deelspecificaties samen te voegen tot een geheel. Qua taalstijl is deze taal een hybride taal waarbij natuurlijke taal wordt gebruikt bij het uitleggen van schema's. Bij Z staan de logische karakteristieken centraal en door de opdeling van specificaties in modules zijn ze makkelijk aanpasbaar.
Wat ik niet kon vinden in de literatuur is een raamwerk of mechanisme om requirments te kunnen traceren. Overigens is het niet duidelijk uit de literatuur of dit een formele methode is of een specificatietaal. De termen worden door elkaar gebruikt en wat ik er uit kon halen is dat het alleen een formele specificatietaal is met technieken(o.a. om spec. te verfijnen). Voor meer informatie over Z wordt u geraadpleegd de Z website te bezoeken.[7] Lijst van beschikbare tools is ook hier te vinden.

VDM-SL

VDM-SL(Vienna Development Method Specification Language) wordt samen met de formele methode VDM en beschikbare tools[8] gebruikt om systemen op een formele manier te ontwikkelen [17]. Het is gebaseerd op propositie- en predikantenlogica en ook de verzamelingenleer. Net zoals de net genoemde Z-notation wordt deze taal gebruikt om functionele requirements van systemen te specificeren op basis van een wiskundig model dat gemaakt wordt van het te bouwen systeem.
Door de grote overeenkomsten tussen de twee [16] zullen we hier een paar eigenschappen van de taal noemen samen met het verschil tussen die twee. Functionaliteit wordt in de specificaties in deze taal weergegeven door het systeem als een toestand te modelleren samen met de operaties daarop, waardoor een systeem van de ene toestand in de andere toestand terecht komt. Specificaties in VDM-SL zijn ook gestructureerd in modules waarbij het mogelijk wordt om verschillende onderdelen door verschillende gebruikers van de taal te laten specificeren. Verfijningstappen zijn ook hier van toepassing om van een initiële specificatie tot een code te komen met behulp van VDM. Behalve VDM kan VDM-SL ook gebruikt worden bij FRSM [21]. Het grootste en bijna enige verschil tussen VDM-SL en Z-notation is de notatie, waarbij de notatie bij Z (m.b.v. schema's) makkelijker te interpreteren is dan die bij VDM-SL (Tekstuele taalstijl met heleboel keywords o.a: inv,ex,pre.)

[7] Prof. Jonathan Bowen's The WWW Virtual Libraray: *The Z Notation* (http://vl.zuser.org/)
[8] Overzicht van de laatste beschikbare tools kun je vinden op de VDM Portal (http://www.vdmportal.org/twiki/bin/view) of VDM Information Website (http://www.vdmtools.jp/en/)

RAISE Specification Language (RSL)

RSL is een formele specificatietaal dat hoort bij het RAISE (Rigorous Approach to Software Engineering) methode voor softwareontwikkeling [18]. RAISE is een uitbreiding en verbetering van de hierboven genoemde VDM. Met deze taal is de ondersteuning van verschillende situaties mogelijk door verschillende abstractie niveau´s waarop gespecificeerd kan worden en dus ook complexiteit reducerend. Transparantie is gewaarborgd door één-op-één wiskundige relaties tussen die verschillende abstractie lagen.
RSL is functiegericht en maakt gebruikt van een tekstuele taalstijl dat lijkt op een programmeertaal. Verschillende onderdelen van de specificatie kunnen in latere stadia aangepast worden bij het veranderen van de requiremnts.
Behalve een specificatietaal heeft RAISE ook een methode en tools voor de taal en methode, waarmee het mogelijk wordt requirements op te slaan en te associëren met respectieve delen van de specificatie.

Behalve talen om de uiteindelijke specificatie mee te maken, worden ook andere talen gebruikt gedurende het RE-fase. Zo is het ideaal om een taal met grafische elementen te gebruiken in de beginfase van RE om zo de communicatie met de eindgebruikers te vergemakkelijken [13]. Hieronder volgt een voorbeeld van zo'n taal dat gebruikt maakt van diagrammen als notatie.

ORM

ORM is een voorbeeld van een semi-formele taal dat als specificatietaal kan dienen. Het heeft een syntax dat precies beschreven is maar geen formele semantiek. Geïdentificeerde objecten (entiteiten, waardes enz) uit het domein en de rollen die ze spelen kunnen m.b.v. ORM-diagrammen gemodelleerd/gespecificeerd worden. Dit is tevens de notatievorm van deze taal. Hoe completer en vollediger een ORM-diagram wordt gespecificeerd, hoe formeler het wordt. Door zijn grafische taalstijl aangevuld met natuurlijke taal, maakt het een ideaal middel bij het maken van een conceptuele model van het betreffende domein voor communicatie met domeinexpert aan het begin van de RE-fase. Verschillende lagen van abstractie is mogelijk in ORM door voor objecten verschillende entiteiten te gebruiken. Zo kan bijvoorbeeld een object in ORM refereren naar een fysiek object maar het kan ook refereren naar een bepaalde waarde van een variabele(feiten). Door zo te detailleren kan zo'n specificatie een goeie basis zijn voor technische specificatie van een systeem. Er zijn tools beschikbaar die het mogelijk maken om zulke modellen te maken (met nodige constraints), maar kunnen er automatisch ook regels gegenereerd worden dat gebruikt kunnen worden ter validatie door de klant (eindgebruiker). Meer over ORM kun je in dit boek vinden [19] of hier[9]

[9] www.orm.net van Terry Halpin zelf.

Eigenschap	EDDA	SAMM	RSL	SADT	HOS	PSL
Semi-formeel/ Formeel	Formeel	Formeel	Formeel	Semi-formeel	Formeel	Semi-formeel
Ondersteuning	√	√	√	√x	√	√
Transparantie	√	√	√	√x	√x	X
Vertrouwdheid	√x	X	X	√x	X	X
Taalstijl	√ √ √	X √ √x	√ √ √	X √ X	√ √ √	√ √x √
Complexiteit reductie	√	√	√	√	√	√
Aanpasbaar	√	√	√	√	√	√
Tools			√			
Formele methode	√	√	√	√x	√	√
Functiegericht	√x	√	√	√x	√	√
Traceability	√	√	√	√	√	√

Tabel 1. Tabel met gewenste karakteristieken van de volgende talen: EDDA, SAMM, RSL, PSL, SADT en HOS. [13] (Doordat expressiviteit op dit moment moeilijk te bepalen is, door gebrek aan een eenduidige rekenmethode, is het in dit tabel achterwege gelaten).

Met een "√" wordt aangegeven dat de betreffende eigenschap van toepassing is bij die taal. Een "√x" betekent dat de taal dat eigenschap gedeeltelijk ondersteunt en "X" betekent dat die eigenschap helemaal niet van toepassing is bij die taal.
(Bij de eigenschap taalstijl staan de drie tekens voor tekstuele, grafische en hybride stijl)

Conclusie

Er is getracht antwoord te geven op de verschillende deelvragen om zo het antwoord op de hoofdvraag te bepalen. De deelvragen worden in de verschillende onderdelen beantwoord. Zo wordt de eerste deelvraag in Hoofdstuk 1 beantwoord, deelvraag 2 in Hoofdstuk 2 , deelvraag 3 ook in Hoofdstuk 2, deelvraag 4 in hoofdstuk 3 en Hoofdstuk 4 is een poging om de laatste deelvraag te beantwoorden. Tevens wordt er in Hoofdstuk 4 een deelverzameling voorgedragen ter beantwoording van de hoofdvraag.

Mijn conclusie is dat de verzameling van talen dat geschikt zijn en dus gebruikt kunnen worden in RE, heel groot is. Door de verschillende gedaantes dat een requirements specificatie kan hebben gedurende de RE-fase en ook de verschillende mogelijke combinaties van talen, is het dus mogelijk om een aantal talen te gebruiken in RE. Sommige talen kunnen meerdere verschijningsvormen hebben en is dus ideaal voor meerdere stadia in de RE-fase, aangezien sommige eigenschappen meer gewenst zijn in de ene stadia dan in de andere (grafische overzicht is beter van toepassing in communicatie met eindgebruiker of domeinexpert, dan als basis voor design).
Andere talen maken gebruik van één bepaalde taalstijl en kan het dus minder aantrekkelijk worden als je de specificatie van een heel systeem voor gaat leggen aan de klant. In dat geval is het dan prettig als er misschien andere talen zijn waarin de specificatie indien nodig vertaald kan worden voor gebruik in dat stadium. Er moet wel een wiskundig bewijsbare één-op-één relatie zijn tussen die twee talen, zodat de semantiek niet onder de vertaalslag kan lijden.
Gezien het feit dat er nog steeds gezocht wordt naar die ideale taal voor RE en men steeds met voorstellen komt voor nieuwe "geschikte" talen, is het duidelijk dat het aantal formele en semi-formele *specificatie*talen voor RE in de toekomst alleen maar zal stijgen. Bovendien komt men ook met talen die voortborduren op bestaande talen om zo te voldoen aan specificatie eisen van specifieke soorten systemen.

Wat betreft het beantwoorden van de hoofdvraag, is het niet zo makkelijk als ik had verwacht. Zonder de kennis over RE en de verschillende mogelijke talen zou je denken dat dit een makkelijk klusje is en dat je alleen "ff kan Google-en!" voor het antwoord. Achteraf is dat helemaal niet zo. Uit dit onderzoek is gebleken dat formele talen inderdaad gewenst zijn bij RE, maar dat niet al die talen geschikt zijn voor de taak. Bij RE zijn er specificaties nodig die uiteindelijk aan het einde van de RE-fase als basis kunnen dienen voor de daarop volgende fase in het ontwikkeltraject. Niet alle talen kunnen gebruikt worden om te specificeren, daarvoor heb je geschikte specificatietalen voor nodig. Uiteindelijk was de zoektocht niet naar gewone formele en semi-formele talen, maar naar die talen die ook als specificatie taal kunnen dienen bij het vastleggen en specificeren van requirements. Overigens is het ook gewenst dus dat die talen bepaalde gewenste gebruikseigenschappen moeten hebben, wil men op een optimale en formele manier omgaan met de requirements van de klant in de RE-fase en dus ook de fasen daarna.

Concluderend is het aan te raden alvorens keuzes te maken wat betreft de te gebruiken talen (formeel, semi-formeel,natuurlijke of combinaties), omtrent het specificeren van requirements, eerst de rij gewenste gebruikseigenschappen langs te gaan voor die talen. Zo kan men een bewuste keuze maken welke het beste zou uitpakken tijdens het project.

Al bij al vond ik het een zeer leerzaam traject, i.h.b. als voorbereiding op mijn afstudeerscriptie!

Referenties

[1] Beeck, M. von der, Margaria, T. , Steffen, B. : *A Formal Requirements Engineering Method for Specification, Synthesis, and Verification*.
Proceedings ot the 8th IEEE Conference on Software Engineering Environments (SEE'97), Cottbus ,Germany. pp: 131--144. IEEE Computer Society Press, April 8-9 (1997)

[2] Bos, V.; Kleijn, J.J.T. : *Formal specification and analysis of industrial systems.* Proefschrift ter verkrijging van de graad van doctor aan de Technische Universiteit Eindhoven, op gezag van de Rector Magnificus, prof.dr. R.A. van Santen, voor een commissie aangewezen door het College voor Promoties in het openbaar te verdedigen op donderdag 7 maart 2002 om 15.00 uur(2002)

[3] Dubois, E.; Huyts, S.; Wieringa, R.J. : *Integrating semi-formal and formal requirements.* Lecture Notes in Computer Science; Vol. 1250. In A. Oliv´e and J.A. Pastor, editors, Advanced Information Systems Engineering. pp 19–32. Springer (1997)

[4] Eekelen, M. :Slides College 2: *Requirements-Engineering,* Vak: Software Engineering, RU, Nijmegen. sl. 16. (14/02/2006)

[5] Herlea D. E. ; Jonker C. M.; Treur J. ; Wijngaards N. J. E. :*A formal knowledge level process model of Requirements Engineering,* Proceedings of the 12th international conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems: Multiple approaches to Intelligent Systems. Cairo, Eygpt. pp: 869–878 (1999)

[6] Hopcroft, J.E.; Ullman, J.D.: *Formal languages and their relationship to automata* Addison-Wesley Pub. Co (cop. 1969)

[7] Hoppenbrouwers, S.J.B.A. : Diktaat *Requirements Engeneering,* Vak: Requirements Engineering, RU, Nijmegen. pp: 1. (2006)

[8] Lamsweerde, A. van : *Formal Specification:A roadmap,* Proceedings of the Conference on The Future of Software Engineering. Limerick, Ireland. pp: 147 - 159 (2000)

[9] Lamsweerde, A. van **:** *Requirements engineering in the year 00: a research perspective,* Proceedings of the 22nd international conference on Software engineering. Limerick, Ireland. pp:5 – 19. (2000)

[10] Saeki, M. : *Software Specification & Design Methods and Method Engineering*, International Journal of Software Engineering and Knowledge Engineering; Vol. 0, No. 0 (1994)

[11] Saaman, E.H. : *Another Formal Specification Language*
Proefschrift ter verkrijging van het doctoraat in de Wiskunde en Natuurwetenschappen

aan de Rijksuniversiteit Groningen op gezag van de Rector Magnificus, dr. D.F.J. Bosscher, in het openbaar te verdedigen op vrijdag 24 november 2000 om 14.15 uur Samenvatting; pp:193 – 196.(2000)

[12] Pohl, K. : *The three dimensions of Requirements Engineering*, Lecture Notes In Computer Science; Vol. 685. Proceedings of Advanced Information Systems Engineering. pp: 275 - 292 (1993)

[13] Pong, L.; Tse, T.H. : *An Examination of Requirements Specification Languages*, The Computer Journal Volume 34 , Issue 2 (April 1991) Special issue on methodologies (systems and software) pp: 143 – 152. (1991)

[14] O´Regan, G. : *A Practical Approach to Software Quality,* Springer-Verlag New York Inc. pp: 256 - 277(2002)

[15] Spivey, J.M. : *The Z Notation: A Reference Manual*, Prentice Hall, International Series in Computer Science, (1988, 1992, 2001)

[16] McGibbon, T. : *An Analysis of Two Formal Methods: VDM and Z,* Paper Contract Number F30602-89-C-0082 (Data & Analysis Center for Software)
Prepared for: Air Force Research Laboratory -Information Directorate (AFRL/IF), Rome, NY.(20 August 1997)

[17] Wikipedia, The Free Encyclopedia
http://en.wikipedia.org/wiki/VDM_specification_language gekeken op 12-01-2007

[18] Nielsen, M.; Havelund, K.; Wagner, R.; George, C. :*The RAISE language, methods and tools.* Formal Aspects of Computing archive Vol. 1 , Issue 1 (Jan.-March 1989) pp: 85 – 114 (1989)

[19] Halpin, T. :Information Modeling and Relational Databases - From Conceptual Analysis to Logical Design, Morgan Kaufmann Publishers,Inc., (2002)

[20] W. Trattnig and H. Kerner, *EDDA: a very-high-level programming and specification language in the style of SADT*, in Proceedings of the 4th Annual International Computer Software and Applications Conference (*COMPSAC '80*), IEEE Computer Society Press, New York, pp.436−443 (1980).

[21] Liu, S. :*A Formal Definition of FRSM and Applications,* International Journal of Software Engineering and Knowledge Engineering Vol. 8 Issue 2 pp: 253 – 281 (1998)

[22] Du Bois, E.; Dubois, P.; Zeippen, J.:*On the Use of a Formal RE Language - The Generalized Railroad Crossing Problem*, Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (RE'97) pp:128-137 (1997)

[23] Dubois, E.: *ALBERT: A Formal Language and Its Supporting Tools for Requirements Engineering* , Lecture Notes in Computer Science, Vol. 1382, pp 322. Springer (1998)