Metaforen in requirements engineering

Metaforen hulpmiddel bij communicatie over it

Tijdens het requirements elicitation-proces voor de ontwikkeling van een informatiesysteem wordt vaak gebruikgemaakt van metaforen om de gezamenlijke beeldvorming van de betrokken partijen te verbeteren en de communicatie te versoepelen. Aan de hand van een veelgebruikte metafoor in de software-engineering zetten de auteurs uiteen wat de voor- en nadelen zijn van het gebruik van metaforen.

Willem Visscher, Jonne Kats, Bas Terwijn, Jeroen Arnoldus en Daan van den Berg

Het ontwikkelen van een informatiesysteem begint met het op schrift stellen van de klantspecifieke eisen, wensen, situaties en regels. Dit *requirements elicitation*-proces is veelal moeizaam, omdat er moet worden gepraat over specificaties van een informatiesysteem dat nog niet bestaat. Onderzoek wijst uit dat juist tijdens dit stadium de meest ingrijpende fouten worden geïntroduceerd, namelijk de zogenaamde conceptuele fouten (men is niet in staat een juist en overeenstemmend beeld van het probleem te vormen) (McConnell, 2004).
Het feit dat veel ontwikkelaars in een zeer technisch jargon communiceren over deze specificaties, bemoeilijkt het proces verder. Niet-technici (zoals gebruikers en opdrachtgevers) hebben geen inzicht in de abstracte kijk op zaken van de programmeurs, waardoor de communicatie tussen beide partijen vaak stroef verloopt. Zelfs als technici met elkaar over deze onderwerpen praten, gebruiken ze al gauw beeldspraak om sneller en duidelijker tot een gezamenlijke beeldvorming te komen.

Metaforen

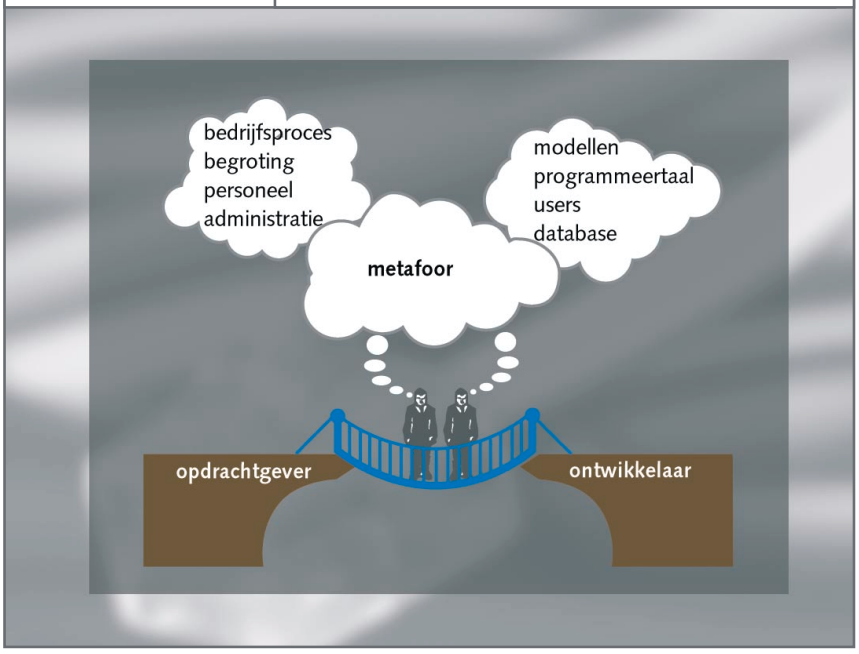
Om de gezamenlijke beeldvorming tijdens dit proces te verbeteren en de communicatie erover te versoepelen, wordt vaak gebruikgemaakt van metaforen. Het woord 'metafoor' komt uit het Grieks en betekent het overdragen van iets (Kopp, 1998). Een metafoor is een beeldspraak en suggereert een overeenkomst tussen twee objecten of ideeën die in de werkelijkheid ongelijk zijn. Metaforen bieden daarom een handvat om onbekende abstracties beter te kunnen begrijpen. Het vergelijken van een onderwerp dat men niet goed begrijpt met een ander onderwerp dat men wel goed begrijpt, kan leiden tot inzichten die resulteren in een beter begrip van het minder bekende onderwerp.
In het algemeen is de kracht van metaforen dat ze dynamisch zijn, een gezamenlijk conceptueel referentiemodel opleveren en eigenschappen en soms relaties suggereren. Metaforen zijn dynamisch omdat ze in verschillende contexten kunnen worden geplaatst en ze verder kunnen worden gedetailleerd om bepaalde aspecten toe te lichten.
Met 'conceptueel model' wordt bedoeld dat betrokken gesprekspartijen

Samenvatting

Het gebruik van metaforen kan de communicatie tussen partijen verbeteren en kan helpen problemen inzichtelijker te maken. Een metafoor moet echter wel voldoende overlap hebben met het bedoelde onderwerp en alle partijen moeten een goed beeld hebben van de metafoor. Concepten in een metafoor kunnen echter ook onovereenkomsten hebben, waardoor de beeldspraak voor sommige aspecten niet opgaat.

zich een bepaald beeld vormen van het te bouwen systeem en dat wanneer dit beeld dat de partijen hebben overeenkomt, er tijdens verdere communicatie aan kan worden gerefereerd. Een goede metafoor is simpel en is goed gerelateerd aan andere relevante metaforen binnen dezelfde context (McConnell, 2004).
Zo wordt het internet soms vergeleken met een bibliotheek, terwijl dit in werkelijkheid twee totaal verschillende objecten zijn; de vergelijking dient slechts om de communicatie erover te vergemakkelijken. Omdat iedereen een mentale voorstelling heeft van het concept 'bibliotheek' en zijn eigenschappen (dit geldt dan als het gezamenlijk conceptueel referentiemodel), kan het handig zijn de informatierijke eigenschap van het internet uit te leggen aan de hand van deze metafoor. Ze suggereert overeenkomstige eigenschappen die kunnen worden gerelateerd aan het concept 'internet' (sites representeren boeken en zoekmachines representeren bibliothecarissen) en die derhalve een werkbaar model opleveren.

Metafoor als brug 1



Omdat het te bouwen systeem nog niet bestaat, nog niet tastbaar is, en het dus abstracte materie is om over te redeneren, worden tijdens requirements elicitation vaak metaforen gebruikt. Het systeem wordt concreter voor zowel de klant, de programmeur als de overige stakeholders doordat het kan worden vergeleken met zaken uit de dagelijkse omgeving waarvan men een goede voorstelling heeft. Zo kan beter inzicht worden verschaft in de structuur, het gedrag en de werking ervan.
Er zijn drie categorieën metaforen die vaak worden gebruikt tijdens de requirements engineering: vergelijking met een fysiek object (prullenbak, firewall, controlpanel), vergelijking met een ruimtelijke plaats/route (container, state space, tijdsas) en vergelijking met een mentaal of communicatief proces (listener, searchbot, verkenner, manager). Onze taal en ons vermogen om dingen te onderscheiden aan de hand van waarneming, verplichten ons gewenst gedrag en gewenste eigenschappen te beschrijven in object-, plaats-, route- en uiteindelijk gedachteachtige termen (Potts, 2001). Deze categorieën metaforen stellen ons in staat softwaresystemen te beschrijven met een kwaliteit die niet te evenaren is met een abstracte (logische of taalkundige) beschrijving. Met 'kwaliteit' wordt hier bedoeld: de mate waarin de toehoorder het systeem intuïtief begrijpt en de tijdsduur die nodig is om dit begrip over te brengen.
Dit geldt vooral voor de vergelijking met mentale en communicatieve processen, waarbij menselijke eigenschappen (zoals kennis, intelligentie, wensen en verantwoordelijkheden) worden toegedicht aan componenten van softwaresystemen. Deze eigenschappen lenen zich niet om abstract te worden beschreven en ons begrip van deze eigenschappen komt dan ook voort uit praktische ervaring. Er is eigenlijk geen goed alternatief voor het gebruik van deze categorie metaforen. Meestal gebruiken we ze zelfs zonder het te beseffen. Zo zou de betekenis van de zin 'Het systeem is verantwoordelijk voor het bezorgen van e-mail binnen de gestelde tijd' of de zin 'Gebruikers navigeren door een boomstructuur van mappen' niet gemakkelijk zonder deze metaforen kunnen worden uitgedrukt. Ook zal volgens McConnell (2004) een programmeur die metaforen gebruikt om het ontwikkelproces te belichten, worden beschouwd als iemand die een beter begrip heeft van programmeren en sneller en beter code schrijft dan iemand die geen gebruikmaakt van metaforen.

Communicatiekloof

Tijdens requirements elicitation komen opdrachtgevers en ontwikkelaars vaak voor het probleem te staan dat ze elkaar niet goed begrijpen. De opdrachtgever denkt in organisatorische en bedrijfskundige termen, de ontwikkelaar in technische termen. Het begrip van de opdrachtgever in deze technische zaken reikt maar tot een bepaald punt. Zo ook houdt het inzicht van de ontwikkelaar in de situatie van de opdrachtgever op een bepaald punt op. De problemen ontstaan als deze twee punten elkaar niet overlappen: er bestaat een kloof tussen de beide werelden.
Om deze kloof te dichten en tot gemeenschappelijk begrip te komen kunnen metaforen een handig hulpmiddel zijn. Er wordt een vergelijking getrokken waar zowel de ontwikkelaar als de opdrachtgever zich een beeld van kan vormen. Op deze manier kan de communicatie tussen beide partijen toch tot gemeenschappelijke afstemming leiden. Met andere woorden: met behulp van een metafoor dient er een brug te worden geslagen tussen de beide belevingswerelden.

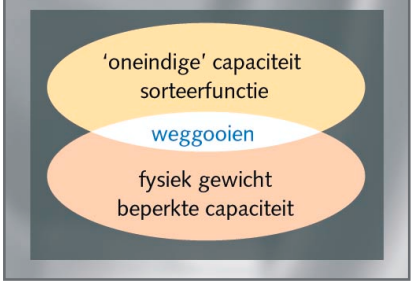
Voordelen van metaforen

Omdat metaforen een concept beschrijven aan de hand van een al bekend concept, kan in korte tijd veel informatie worden overgedragen. Daarnaast schept een metafoor bij de ontvanger een denkkader (mentaal model) waarbinnen deze verder kan denken over het onderwerp. Mede hierdoor zijn metaforen, als een van de weinige communicatietechnieken, geschikt om complexe abstracte concepten gedeeltelijk voor leken begrijpelijk te maken.
Als niet actief een denkkader wordt geschapen, zullen leken zelf een – mogelijk ongepast – denkkader opbouwen dat als basis fungeert voor verdere kennisverwerving (Blackwell, 1999). De observatie dat metaforen een sterk denkkader kunnen creëren, wordt onderbouwd door onderzoek dat aannemelijk maakt dat juiste metaforen in meerdere mate, de basis zijn voor menselijk redeneren. Een metafoor zou een denkkader scheppen waarbinnen kan worden geredeneerd met een symbolische taal (Potts, 2001).

Mogelijke problemen

Het gebruik van metaforen kan echter ook problemen opleveren doordat een toehoorder de metafoor onjuist interpreteert. Dit kan doordat een toehoorder niet bekend genoeg is met de metafoor omdat deze bijvoorbeeld cultureel- of bedrijfstakgebonden is. Problemen door culturele verschillen komen, door de toenemende globalisering, steeds vaker voor (Mahemoff & Johnston, 1998).
Een andere oorzaak van het onjuist interpreteren van een metafoor is het te ver doorvoeren van de beeldspraak. Ter illustratie: veel besturingssystemen hebben faciliteiten om met uiteenlopende doeleinden documenten op te slaan. Voorbeelden hiervan zijn mappen, zipbestanden en de prullenbak. De bijbehorende metaforen (respectievelijk 'ordenen', 'inpakken' en 'weggooien') doen over het algemeen goed dienst, maar met name beginnende computergebruikers kunnen erdoor in de war raken omdat ze de metaforen te ver doorvoeren. In de prullenbak lijken oneindig veel documenten te passen en het slepen met het dichtgeritste zipbestand wordt niet beïnvloed door het formaat.
Het zoeken naar de juiste metafoor is een moeilijk proces dat kennis over de toehoorders vereist (Lovgren, 1994). Voor sommige concepten bestaan geen metaforen die een duidelijk beeld scheppen. Als dan toch wordt gekozen voor een beschrijving met een metafoor, leidt dit tot een oversimplificatie of bemoeilijkt de metafoor juist de kennisverwerving doordat ze verwarring schept.

De prullenbakmetafoor 2



De metafoor 'het bouwen van een huis'

We willen deze eigenschappen van metaforen toetsen aan de hand van een veelgebruikte metafoor in de softwareengineering. Het ontwikkelen van software wordt vaak vergeleken met het ontwerpen en bouwen van een huis (McConnell, 2004). Het doel van deze metafoor is om het proces van softwareontwikkeling te verduidelijken; de nadruk ervan ligt dan ook op het proces van het creëren van een object. Deze metafoor geeft ondeskundigen een goed beeld van wat het ontwikkelen van een groot softwaresysteem inhoudt. De metafoor laat zien welke uitdagingen, beperkingen en problemen er zijn. Ook voor de mensen die zelf software ontwikkelen geeft deze metafoor een goede omschrijving van hun activiteiten en ze kan worden gebruikt om sneller te communiceren. Voor ondeskundigen is het vergelijken van het ontwikkelen van software met het bouwen van een huis een pakkende metafoor, die de volgende gemeenschappelijke eigenschappen suggereert. Allereerst wordt een globaal ontwerp geschetst dat kan worden beschouwd als architectuur van het te ontwikkelen gebouw of softwaresysteem. Meerdere partijen hebben belang bij het eindproduct en iedere stakeholder heeft zijn eigen requirements. Bij een kantoorgebouw heeft elke afdeling specifieke eisen en wensen voor de indeling en de beschikbare faciliteiten. Tijdens de bouw zijn er veel verschillende partijen die allemaal hun ingangseisen opleggen: de loodgieter kan zijn werk pas beginnen als de metselaar klaar is, de stukadoor kan pas aan de slag als het dak af is. Ook bij software zijn er veel stakeholders met ieder hun eigen eisen en wensen: managers, gebruikers, ontwikkelaars,

onderhoudsteam, enzovoort.
Voordat de daadwerkelijke bouw kan beginnen, moet er goed zijn nagedacht over het onderhoud en de eventuele uitbreidingsmogelijkheden van het eindproduct. Als deze zaken vooraf niet goed zijn geregeld, kan dat in de toekomst tot hoge kosten leiden. Denk aan de onderhoudskosten van houten kozijnen (kunststof vereist geen onderhoud) of aan het toevoegen van een nieuwe module waarbij blijkt dat overal binnen het bestaande systeem wijzigingen moeten worden doorgevoerd. Aangezien iedereen een duidelijke mentale voorstelling heeft van de bouw van een gebouw, schept de metafoor ook voor ondeskundigen een duidelijk denkkader om te redeneren over de bouw van een softwaresysteem. Het is gemakkelijker om zich in te beelden welke problemen van toepassing kunnen zijn bij de ontwikkeling van een systeem.

Deze metafoor laat echter toch een aantal belangrijke zaken on(der-) belicht. Zo zegt het bouwen van een huis niets over het testen ervan: een huis hoeft niet zo grondig te worden getest als een softwaresysteem. Het testen van een huis is een klein gedeelte van de totale inspanning, terwijl bij software het testen tot veertig procent van het totale budget kan oplopen. Ook is het testen van software een ingrijpende en terugkerende activiteit binnen het ontwikkelproces. Dit verschil ontstaat doordat tijdens de bouw van een huis vrijwel uitsluitend gecertificeerde bouwstenen (bakstenen, gasleidingen, dakpannen) worden gebruikt, terwijl in de software-industrie vooral wordt gewerkt met klantspecifieke bouwstenen (maatwerkcomponenten). De relatief veel hogere testkosten worden geheel buiten beschouwing gelaten door de bouw-van-een huis-metafoor.
Tevens komt het onderhoud van software niet genoeg naar voren; onderhoud neemt in de praktijk een prominente plaats in. Gebouwen worden veel minder beïnvloed door veranderingen in de omgeving of door de gebruikers ervan: een gezinsuitbreiding resulteert niet meteen in de noodzaak het huis te vergroten. Softwaresystemen zijn echter wel sterk onderhevig aan de veranderingen in de omgeving waarin ze functioneren. De metafoor suggereert dat het softwaresysteem volledig klaar is wanneer het wordt opgeleverd en dat onderhoud relatief weinig kosten met zich meebrengt, zoals bij een huis. In werkelijkheid begint een softwaresysteem echter met evolueren na de oplevering ervan (Bennet & Rajlich, 2000).

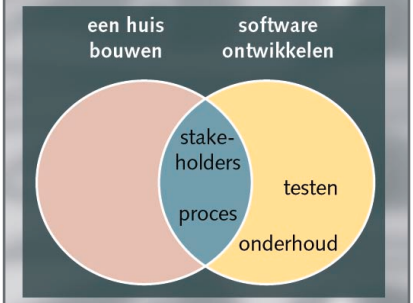
Het bouwen van een huis heeft dus veel overeenkomsten met het ontwikkelen van software, maar er zijn enkele aspecten waarvoor deze metafoor niet geheel dekkend is. Daardoor zal de metafoor misleidende suggesties doen als deze te ver wordt doorgevoerd. De metafoor is vooral geschikt om ondeskundigen inzicht te verschaffen in het globale proces van het ontwikkelen van software (McConnell, 2004), maar verschilt op enkele cruciale punten.

Conclusie

Over het algemeen kunnen we stellen dat het gebruik van metaforen de communicatie tussen partijen kan verbeteren. Een metafoor moet echter wel voldoende overlap hebben met het bedoelde onderwerp. Ook moeten alle partijen een goed beeld hebben van de metafoor die wordt gebruikt. Als metaforen goed worden gebruikt, kunnen ze helpen problemen inzichtelijker te maken. Vooral bij het verklaren van moeilijke materie kunnen metaforen die materie concreter maken voor de ander. Zonder het gebruik van metaforen zou de communicatie tussen mensen een stuk moeilijker en tijdrovender zijn. Wanneer we bewust gebruikmaken van metaforen, is dit omdat we denken dat datgene wat we trachten over te brengen, zonder metaforen niet of nauwelijks over te brengen is. Niettemin moeten we er altijd voor waken, en misschien zelfs benadrukken, dat concepten in een gebruikte metafoor ook onovereenkomsten hebben en dat daarom de beeldspraak voor sommige aspecten niet opgaat.

Dit artikel was niet totstandgekomen zonder de intensieve begeleiding en feedback van Hans Dekkers (Kaizen Advisering).

De metafoor 'het bouwen van een huis' 3



Literatuur

Bennet, K. & V. Rajlich (2000). Software Maintenance and Evolution: A Roadmap. *The Future of Software Engineering*. ICSE.

Blackwell, A.F. & T.R.G. Green (1999). Does Metaphor Increase Visual Language Usability? *Proceedings 1999 IEEE Symposium on Visual Languages (VL'99)*.

Kopp, B.M. (1998). Using Metaphors in Creative Writing. OWL at Purdue University, owl.english.purdue.edu/handouts/general/gl_metaphor.html.

Mahemoff, M.J. & L.J. Johnston (1998). Software Internationalisation: Implications for Requirements Engineering. In D. Fowler & L. Dawson (red.), *Proceedings of the Third Australian Workshop on Requirements Engineering*. Geelong: Deakin University, pp. 83-90.

McConnell, S. (2004). *Code Complete*. Microsoft Press.

Laney, R. e.a. (2004). Composing Requirements Using Problem Frames. 12th IEEE International Requirements Engineering Conference, www.re04.org.

Lovgren, J. (1994). How to Choose Good Metaphors. *IEEE Software*, mei 1994.

Potts, C. (2001). Metaphors of Intent. *Proceedings Fifth IEEE International Symposium on Requirements Engineering (RE'01)*. Atlanta: IEEE Computer Society Press, www.cc.gatech.edu/fac/Colin.Potts/pubs/2001/re01/metaphor.pdf.

Links

www.cc.gatech.edu/fac/Colin.Potts/pubs/2001/re01/metaphor.pdf

Willem Visscher
is master Software Engineering UHA. E-mail: wcmvisscher@hotmail.com.

Jonne Kats
is master Software Engineering UHA. E-mail: jonnekats@gmail.com.

Bas Terwijn
is master Software Engineering UHA. E-mail: b.i.terwijn@student.uva.nl.

Jeroen Arnoldus
is master Software Engineering UHA. E-mail: jeroen@bja-electronics.nl.

Daan van den Berg
is docent aan de masteropleiding Software Engineering UHA. E-mail: d.van.den.berg@hva.nl.

De masteropleiding Software Engineering is een samenwerkingsverband tussen de Universiteit van Amsterdam, de Vrije Universtiteit en de Hogeschool van Amsterdam.