

In dit artikel zal nader worden ingegaan op wat requirements management tools zijn en op welke wijze ze helpen goede software te ontwikkelen. Om een goed beeld te krijgen van eisen zal daaraan voorafgaand worden ingegaan op wat eisen zijn, en op hoe requirements management in het software-ontwikkeltraject dient te worden gezien. Hierbij zal ook kort worden ingegaan op het Capability Maturity Model [HUMPHREY].

thema

Een systeem voor het beheer van eisen aan een software-product

Requirements management tools: bewakers van het ontwikkeltraject?

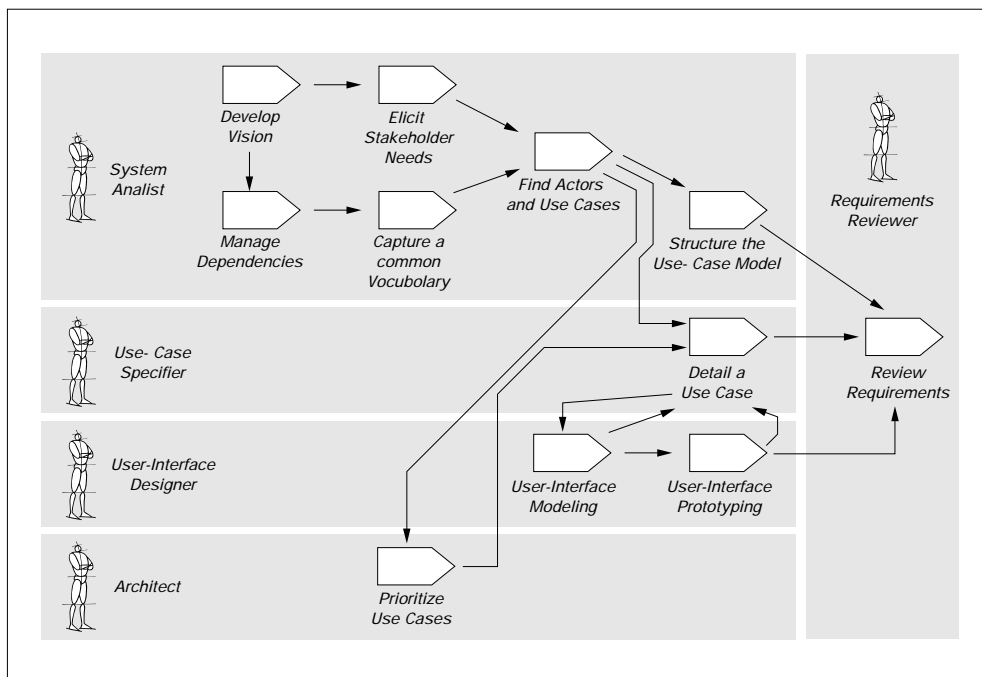
Software-ontwikkeling is een activiteit die reeds tientallen jaren wordt uitgevoerd, en is uitgegroeid tot een volwassen tak van engineering. Niet alleen weten we nu veel beter wat een goed software-product is; ook hebben we betere inzichten in het proces dat gebruikt zou moeten worden om tot een dergelijk product te komen. Des te verbazingwekkender is het dat de meerderheid van de software-ontwikkeltrajecten nog steeds faalt. Dit falen uit zich in het niet op tijd en binnen het budget opleveren van het door de klant gewenste software-product. Het belangrijkste hierbij is dat het product niet is wat de klant gewenst, of beter gezegd, geëist heeft. Blijkbaar is het een probleem software te laten voldoen aan de eisen (requirements) van de klant. Het op een structurele wijze beheer van eisen (requirements management) is het geëigende middel om dit probleem het hoofd te bieden. Het Software Engineering Institute heeft niet voor niets in het Capability Maturity Model (CMM), een model voor het bepalen van de volwassenheid van ontwikkelorganisaties, requirements management als belangrijk onderdeel onderkend. Requirements management kan worden ondersteund door gebruik te maken van tools, zogenaamde requirements management tools. Eenvoudig gezegd helpen dergelijke tools in het organiseren, structureren en relateren van eisen.

ADA Voordat een software-product wordt ontwikkeld, dient vast te worden gelegd wat het systeem moet gaan doen en welke beperkingen een rol spelen. Dergelijke informatie wordt dan ook in de eerste fase van een

software-ontwikkeltraject vastgelegd in een programma van eisen [SOMMERVILLE]. Een eis is hierbij een precies gedefinieerde eigenschap of beperking van een programmatuursysteem [YEH]. Merk hierbij op dat er een verschil is tussen wensen, eisen en doelstellingen van een systeem. Een klant kan wel een bepaalde wens hebben, maar slechts na analyse kan worden bepaald of het een eis aan het systeem is. Eisen zijn dus iets waaraan moet worden voldaan en die kunnen worden getest om te bepalen of eraan voldaan is. Doelstellingen tenslotte, zijn nog algemener van aard, en daardoor vaak moeilijk te testen.

Eisen kunnen op verschillende wijzen worden uitgedrukt. In veruit de meeste gevallen wordt natuurlijke taal gebruikt. Daarnaast is het ook mogelijk gestructureerde, of formele talen te gebruiken voor het definiëren van eisen. In [HILL] wordt bijvoorbeeld beschreven hoe een variant van de programmeertaal Ada gebruikt zou kunnen worden voor het specificeren van eisen. Tenslotte kunnen natuurlijk ook illustraties worden gebruikt om eisen te beschrijven.

INGEWIKKELD Het programma van eisen moet niet worden onderschat; het kan voor een ingewikkeld systeem bestaan uit duizenden eisen. Daarnaast moet het programma van eisen zelf ook aan bepaalde eisen voldoen om bruikbaar te zijn in het vervoltraject (de realisatie van het systeem). Zo dienen de eisen volledig en consistent te zijn. Alles wat het systeem moet doen, dient te worden vastgelegd en de eisen mogen uiteraard niet met elkaar in strijd zijn. In de praktijk is dit vaak moei-



FIGUUR 1: Requirements management workflow in het Rational Unified Process

lijk te controleren, vooral als de eisen in natuurlijke taal zijn geformuleerd. Naast dat het programma van eisen tot doel heeft eisen te beschrijven voor de te realiseren software, moet het ook kunnen dienen als naslagwerk later in het traject, zoals in de onderhoudsfase. Een standaardstructuur van een programma van eisen is dan ook aan te bevelen [ROBERTSON]. Een typisch programma van eisen bevat dan ook zeker de volgende onderdelen:

- Algemeen: In het kort dienen het probleem van de klant en de doelstelling van de software worden weergegeven. Daarnaast kan ook de context, zoals alle partijen, en aannames worden geschetst.
- Apparatuur: Indien er speciale apparatuur nodig is voor het systeem dan moet deze worden beschreven.

Requirements management blijft niet beperkt tot analyseren en documenteren

- Conceptueel model: Een conceptueel model beschrijft in grote lijnen de opdeling van het systeem in componenten en hun verantwoordelijkheden. Het is een aanzet tot de uiteindelijke architectuur van het systeem.
- Functionele eisen: De functionele eisen beschrijven de functionaliteit die het systeem moet gaan bieden. Hierbij kan gebruik gemaakt worden van use-case-diagrammen waarin de actoren en stukken functionaliteit zijn weergegeven.
- Gegevens-eisen: In een traditioneel systeem, of bij een object-georiënteerd systeem waarbij gebruik gemaakt moet worden van een bestaande (relationele) databa-

se, is een gegevensmodel nodig. Een gegevensmodel wordt hierbij typisch als ER (Entity-Relatie) model beschreven.

- Niet-functionele eisen: Deze eisen formuleren beperkingen waaraan het systeem moet voldoen. Kort samengevat zijn deze onder te verdelen in bruikbaarheids-, betrouwbaarheids-, snelheids- en onderhoudbaarheids-eisen. Voor een uitgebreider model wordt verwezen naar [ZEIST].
- Verklarende woordenlijst: Aangezien het programma van eisen door een niet-deskundig persoon moet kunnen worden gelezen dient alle vak- en systeemterminologie te worden verklaard.

WORKFLOWS Requirements management maakt deel uit van het software-ontwikkeltraject. Het doel van

requirements management is het bewerkstelligen en onderhouden van een wederzijds begrip omtrent de eisen die door het project geadresseerd zullen worden tussen de klant en het softwareproject. Het is niet beperkt tot het analyseren en documenteren van eisen in het begin van het traject. Ook gedurende het proces is het beheer van eisen, en wijzigingen van eisen van belang. Zo dienen initiële eisen gerelateerd te worden aan specifieke software-eisen en -producten zoals ontwerp, code en testspecificaties [RAMESH]. Door deze relaties te documenteren en te onderhouden is het mogelijk te controleren of eisen op alle niveaus worden gedekt. Daarnaast kunnen de relaties gebruikt worden om de impact van een wijziging te bepalen. Tenslotte kunnen eisen dienen als documentatie van het systeem in de onderhoudsfase.

Ter illustratie zal hier worden beschreven hoe requirements management is ingevuld in het Rational Unified Process [KRUCHTEN]. Het Rational Unified Process is onderverdeeld in een aantal workflows die bestaan uit gerelateerde activiteiten. Een van de kernworkflows is requirements management (zie figuur 1).

De nadruk in het Rational Unified Process ligt op de initiële fase van het software-ontwikkeltraject. De activiteiten zijn grofweg onder te verdelen in een aantal stappen:

- Analyseren van het probleem: hierbij moeten de partijen en hun problemen in kaart worden gebracht. Tezamen met de hoog-niveau eigenschappen aan het systeem worden deze gedocumenteerd in een visiedocument.

- Bepalen van eisen: Door interviews en workshops moe-

ten de eisen gedetailleerd in kaart worden gebracht. Aan de eisen worden prioriteiten toegekend.

- Definieren van het systeem: Door use-cases wordt de functionaliteit van het systeem beschreven.
- Beheren van de scope van het project: Er wordt bepaald welke eisen in de huidige iteratie zullen worden gerealiseerd en welke eigenschappen van het systeem en use-cases hiervoor nodig zijn.
- Detailleren van het systeem: De use-cases worden in gedetailleerde scenario's uitgewerkt. User-interface prototypes worden ontwikkeld om de gebruiker zo snel mogelijk een gevoel te geven voor het systeem.
- Beheren van wijzigende eisen: Relaties tussen eisen en extra attributen worden gebruikt om wijzigingen in eisen te beoordelen. Review-sessies spelen hierbij een belangrijke rol.

ONDERSCHIED Het Capability Maturity Model [HUMPHREY], oftewel CMM, is een model voor het beschrijven van de volwassenheid van software-ontwikkelorganisaties. Daarnaast kan het dienen als middel voor het verbeteren van ontwikkelorganisaties. Het model is ontstaan door bestaande ontwikkelorganisaties te bestuderen en deze in categorieën in te delen. Het CMM verdeelt ontwikkelorganisaties in vijf niveaus van volwassenheid (zie figuur 2).

Software-ontwikkeling in organisaties die zich op het eerste niveau (initial) bevinden is ad hoc en wordt niet expliciet gemanaged. Bedenk echter dat ongeveer 75% van de software-ontwikkelorganisaties zich op dit niveau bevindt. Organisaties op niveau 2 (repeatable) beheersen het proces waardoor de meeste projecten slagen. Op niveau 3 (defined) is er een standaardproces gedefinieerd, dat door de gehele organisatie wordt gebruikt. Op niveau 4 (managed) wordt dit proces gekwantificeerd zodat het wordt begrepen en kan worden bijgesteld. Op het hoogste niveau (optimizing) is software-ontwikkeling een routine geworden. Merk echter op dat ook op dit niveau er continu aan het proces gewerkt dient te worden om niet terug te vallen in lagere niveaus.

Naast deze gelaagde structuur bestaat het CMM uit een aantal belangrijke onderdelen. In de eerste plaats zijn er per niveau een aantal Key Process Areas (KPAs) aangegeven. Dit zijn gebieden die van primair belang zijn voor het niveau. Per KPA zijn er verder een aantal doelen die gerealiseerd dienen te worden en key practices die invulling geven aan deze doelen. Hierbij wordt er onderscheid gemaakt tussen de implementatie-activiteiten en de institutionaliserende (of andere-) activiteiten.

Requirements management is de eerste, en tevens één van de belangrijkste Key Process Areas van CMM niveau 2. Hierbij zijn de doelen het beheersen van de eisen om te komen tot een baseline van eisen, en het consistent houden van alle producten en activiteiten met deze base-

line. Om deze doelen te verwezenlijken is de uitvoering nodig van een aantal activiteiten. Op de eerste plaats dient de software engineering groep de eisen te inspecteren voordat ze worden opgenomen in het proces. Hierbij vindt er een controle van de eisen plaats op onder meer volledigheid, consistentie, haalbaarheid en testbaarheid.

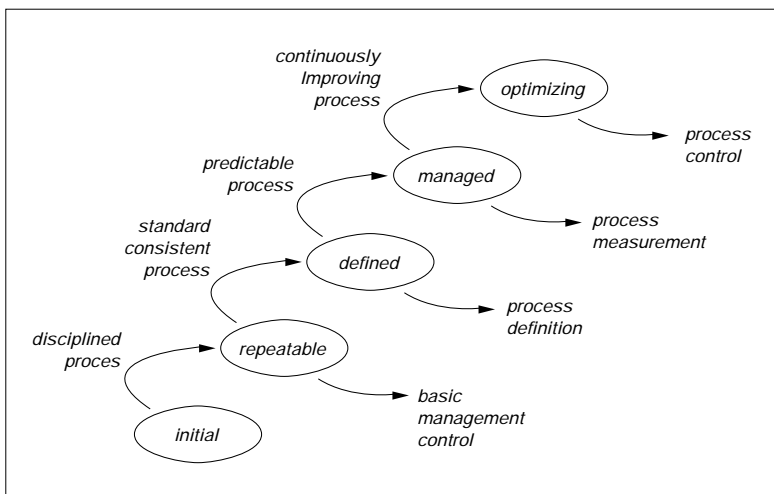
Op de tweede plaats gaat de software engineering groep met de eisen aan de slag als basis voor plannen, producten en activiteiten. Als laatste moeten alle wijzigingen aan de eisen worden geïnspecteerd en opgenomen in het project. Hierbij kan men uiteraard kijken naar de impact van wijzigingen op het project. Ook van belang is dat wijzigingen worden overlegd met alle betrokken partijen.

De andere activiteiten zijn algemener van aard en hebben ook betrekking op andere KPA's. Zo dienen eisen bijvoorbeeld te worden gedocumenteerd, dienen er adequate middelen aanwezig te zijn en dienen mensen te worden getraind in het beheren van eisen.

HISTORIE Requirements management tools bieden ondersteuning bij het requirements management proces. De precieze functionaliteit verschilt uiteraard per tool. De INCOSE Requirements working group [INCOSEa] heeft de volgende minimale eisen voor een requirements management tool gedefinieerd:

- Het kunnen identificeren van individuele eisen
- Het kunnen sorteren en toekennen aan een bestemming van eisen
- Het kunnen versioneren van een groep van eisen
- Het bieden van een gegevensinterface

Eisen dienen natuurlijk ook opgeslagen en ontsloten te kunnen worden. Naast deze minimale functionaliteit worden vaak veel andere functies geboden. Zo kunnen eisen aan elkaar gerelateerd worden, en worden getoond in verschillende overzichten. In deze overzichten kunnen de gerelateerde eisen en relaties via bijvoorbeeld matrices of bomen worden weergegeven. Hierbij kan er een selectie gemaakt worden uit eisen op basis van een aantal karakteristieken. Extra karakteristieken of attributen kunnen



FIGUUR 2: CMM-niveaus van volwassenheid

hierbij worden toegevoegd om extra (project- of organisatiespecifieke) informatie toe te voegen. Impact-analyse van wijzigingen is mogelijk door de invloeden van eisen op elkaar te definiëren en weer te geven.

De INCOSE Requirements Working Group [INCOSEb] benadrukt dat het kunnen koppelen van requirements management tools met andere tools erg belangrijk is omdat tools vaak hun eigen sterke punten hebben, in de tijd kunnen wijzigen en tools van betrokken partijen vaak verschillen. Niet alleen moeten eisen dus kunnen worden gesynchroniseerd met andere requirements management tools, ook integratie met andere tools zoals analyse, ontwerp, test en configuratie management tools is van belang.

Interessant is het om te zien in hoeverre tools de activiteiten zoals benoemd in CMM ondersteunen. De eerste activiteit behelst het inspecteren van eisen door de software engineering groep voordat ze worden opgenomen in het proces. Om deze activiteit te ondersteunen bieden tools mogelijkheden voor gebruik door meerdere personen, op potentieel verschillende locaties. Daarnaast kunnen er vaak discussies over eisen worden gevoerd via e-mail of het web. Idealiter zou er een workflow gedefinieerd kunnen worden die aangeeft welke personen nieuwe eisen zouden moeten inspecteren, waarna deze automatisch notificatie ontvangen, uiteindelijk leidend tot het wel of niet accepteren van eisen.

In de tweede plaats dienen eisen gebruikt te worden als basis voor plannen, producten en activiteiten. Tools komen hieraan tegemoet door te integreren met tekstverwerkingspakketten waardoor eisen deel uitmaken van documenten. Het automatisch genereren van andere ontwikkelproducten zoals analyse, ontwerp, implementatie en testen wordt in mindere mate ondersteunt. Hierbij moet voornamelijk gedacht worden aan het doorsluizen van use-cases en test-specificaties richting CASE- en testtools.

De derde activiteit omvat het inspecteren van wijzigingen in eisen voordat ze worden opgenomen in het proces. Hierbij geldt een soortgelijke ondersteuning als voor de eerste activiteit. Daarnaast kunnen tools de historie en motivatie van wijzigingen in eisen bijhouden. Idealiter zouden tools eisen automatisch kunnen analyseren en controleren op volledigheid en consistentie.

Tenslotte geldt dat het verzamelen van metrieken bijvoorbeeld met betrekking tot het totale aantal en gewijzigde aantal eisen niet of nauwelijks wordt ondersteund, terwijl deze gegevens wel beschikbaar zijn. Dit is duidelijk een gemiste kans voor de requirements management tools.

Evaluatieproject

Momenteel is er een requirements management tools evaluatieproject in voorbereiding bij het SERC. Dit project zal eind 1999 worden gehouden en is voor iedereen toegankelijk. Voor meer informatie zie <http://www.serc.nl>.

ONDERSTEUNING Samengevat kunnen we stellen dat requirements management tools voornamelijk de basisfunctionaliteiten voor requirements management ondersteunen. Het team-denken is duidelijk in opkomst via bijvoorbeeld web- en e-mail-integratie. Actieve ondersteuning voor en integratie met het software-ontwikkelproces ontbreekt echter nog.

Drs. Danny Greefhorst

is senior onderzoeker bij het Software Engineering Research Centre.

Literatuur

- [ANDRIOLE] S. Andriole, The politics of requirements management, In IEEE Software, 11-12 1998, 82-84.
- [HILL] A. Hill, Towards an Ada-based specification and design language, ADA UK News, April, 1983.
- [HUMPRHEY] W.S. Humphrey, Managing the Software Process, SEI Series in Software Engineering, Addison-Wesley, Augustus, 1990.
- [INCOSEa] INCOSE Requirements Working Group, Factors influencing requirements management toolset selection, In Proceedings of the fifth annual symposium of the National Council on Systems Engineering, Volume II, July, 1995.
- [INCOSEb] INCOSE Requirements Working Group, Interfacing Requirements Management Tools In The Requirements Management Process – A First Look, In Proceedings of the 7th annual symposium of the INCOSE, Volume II, August, 1997.
- [JACOBSON] I. Jacobson, The Unified Software Development Process, Addison-Wesley, 1999.
- [KRUCHTEN] P. Kruchten, The Rational Unified Process, an introduction, Addison-Wesley, 1998.
- [PAULK] M. C. Paulk et al, Key Practices of the Capability Maturity Model – Version 1.1, Software Engineering Institute – Carnegie Mellon University.
- [RAMESH] B. Ramesh, Factors influencing requirements traceability practice, In Communications of the ACM, 12-1998, 37-44.
- [RATIONAL] Rational Software Corporation, Reaching CMM Levels 2 and 3 with the Rational Unified Process, 1998.
- [ROBERTSON] J. Robertson, S. Robertson, Requirements Specification Template, Atlantic Systems Guild, London, Aachen & New York, 1996, 1998.
- [SOMMERVILLE] I. Sommerville, Software Engineering, Addison-Wesley, 1982, 1985.
- [YEH] R.T. Yeh, P. Zave, Specifying software requirements, In Proceedings IEEE, 68(9), 1077-85.
- [YOURDON] E. Yourdon, When good enough is best, In Byte, 09-1996.
- [ZEIST] R.H.J. van Zeist, P.R.H. Hendriks, R.M.C. Paulussem, J.J.M. Trienekens, Kwaliteit van softwareproducten; praktijkervaringen met een kwaliteitsmodel, Kluwer Bedrijfswetenschappen, Deventer, mei 1996.