‘De Menselijke Maat’

Software ontwikkeling in breder perspectief

~Erik Philippus

Bij discussies over de ontwikkeling van software voeren technische en rationele zaken gewoonlijk de boventoon. Dit artikel is bedoeld om software ontwikkeling, in het bijzonder de relatie tussen software architectuur en de kwaliteit van software producten, in een breder kader te plaatsen – een kader waarin een plaats is ingeruimd voor de menselijke maat.

Egyptische godin ‘Maat’

Laten we om te beginnen het woord ‘maat’ eens wat nauwkeuriger bekijken. Hiervoor gaan we ver terug in de tijd, naar het oude Egypte. Vanaf de eerste dynastie tot ver na de Griekse en Romeinse invasies werd in Egypte de godin met de in onze oren bekende naam ‘Maat’ aanbeden.

In het Egyptisch betekent ’maat’ letterlijk ‘waarheid’. Maat gaf betekenis aan de wereld, en zij was de perfecte orde waar niet alleen de mens, maar zelfs de Farao respect aan verschuldigd was. Het werd gezien als uitermate belangrijk om te leven naar de universele principes van balans en gerechtigheid, anders zou het bestaan in gevaar komen. Zonder Maat zou het universum vervallen in chaos, zo geloofden de inwoners van het oude Egypte.

Maat droeg een struisvogelveer, die gebruikt werd om de harten van overledenen te wegen. Het wekt dan ook geen verwondering dat Maat een grote rol speelde in de rechtspraak in het oude Egypte: Maat is immers de standaard waarmee de mens wordt gemeten.
Eigenlijk was Maat niet alleen een godin, maar ook een concept. Zij was niet alleen de personificatie van waarheid, gerechtigheid en orde, maar zij vertegenwoordigde ook een universeel ethisch principe. Door de invloed van Maat zou alles in het universum zich volgens bepaalde patronen voltrekken: van de bewegingen van sterren en planeten tot de overstromingen van de Nijl.

Zij was de basis voor natuurkundige wetmatigheden, maar ook in de morele sfeer deed zij haar invloed gelden waardoor zuiverheid werd beloond en misdaad bestraft.

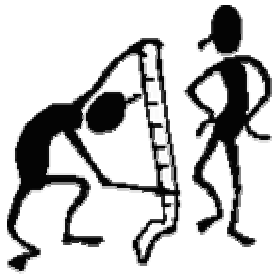
Maar wat heeft de godin Maat te maken met software ontwikkeltrajecten? De vraag die dan eigenlijk gesteld moet worden is: 'welke rol speelt de menselijke maat bij het ontwikkelen van software?' Het antwoord ligt besloten in het voornaamste doel van de geproduceerde software: om zoveel mogelijk tegemoet te komen aan de wensen van de opdrachtgever en de verwachtingen van de gebruikers.

Derhalve is het de vraag of de geleverde software de vereiste 'maat' heeft, en bovendien kan deze vraag niet los worden gezien van de aandacht die de menselijke maat krijgt tijdens de ontwikkeling van de software. Hiermee raken we de kern van dit artikel.

Betekenissen van 'maat'

Voordat we gaan kijken op welke wijze aandacht voor de menselijke maat wordt weerspiegeld in de kwaliteit van het ontwikkelde product, nemen eerst nog eens de term 'maat' onder de loep. Het woord 'maat' kent diverse betekenissen:

Maat als ***afmeting***: de term maat slaat hierbij op een hoeveelheid, of ook wel eenheid (zoals schoenmaat). In de context van software ontwikkeling refereert deze betekenis van maat aan de 'afmetingen' van toekomstige gebruikers. Sluit het software product aan bij de fysieke mogelijkheden, mentale capaciteiten en sociale en organisatorische context van de afnemers? Hierbij is de achterliggende vraag eigenlijk: 'hoe neem je de mens eigenlijk de maat'?



Het vinden van een antwoord op dit soort vragen is niet eenvoudig, mede doordat dit meer dan alleen technische vaardigheden vereist. Bovendien dient hierbij niet alleen het product, maar ook het toegepaste ontwikkelproces betrokken te worden.
Om zowel de visie van de opdrachtgever, als de eisen van toekomstige gebruiker boven tafel te krijgen, is meer dan een oppervlakkig ontwikkeld inlevingsvermogen vereist. Algemeen bekend is dat het ondubbelzinnig definiëren van gebruikerseisen een cruciaal, maar tevens één van de moeilijkste onderdelen is van het software ontwikkeltraject. Maar in deze fase wordt wel bepaald of de toekomstige gebruikers zich aan het product moeten aanpassen, in plaats van andersom.

Een andere betekenis van maat voert ons naar de wereld van de muziek: maat als de **tijdsduur** van een groep noten (b.v. ¾ maat), waarmee de basisstructuur van een muziekstuk wordt aangegeven. Het ritme is de variatie van de individuele noten en pauzen, zowel binnen een maat (locaal) als ook over het hele muziekstuk heen (globaal). Ritme speelt ook een belangrijke rol in het software ontwikkeltraject. Neem maar het iteratieve aspect van de object geörienteerde ontwikkel-methodiek: niet voor niets noemt OO godfather Grady Booch ritme een kritische succesfactor voor ontwikkeltrajecten. Een goed ritme in het ontwikkelteam zorgt volgens hem voor stabiliteit, regelmaat in de oplevering, coördinatie van ondersteunende activiteiten, en doelmatige communicatie.

Zonder het juiste ritme zal het team bovendien minder goed opgewassen zijn tegen allerlei onverwachte problemen. Hoe komt dit? Omdat net als in de muziek het ritme ervoor zorgt dat binnen de globale structuur (zoals b.v. het release ritme), individuele variaties mogelijk zijn. In die zin is ritme dus een belangrijk aspect van de menselijke maat, want zonder enige speelruimte voor het individu zal elk ontwikkelschema strak en onmenselijk worden.

Een laatste betekenis van maat in dit verband is die van ***partner***. Net als in elk team zal het succes van een software ontwikkelteam gebaseerd zijn op met elkaar samenwerkende 'ploegmaten'. Deze betekenis zie je overigens ook terug in het samenspel van individuele noten binnen een maat. Het hoeft geen betoog dat de effectiviteit van deze samenwerking in directe relatie staat met het bereiken van de gewenste kwaliteit van het software product. Dit aspect van de menselijke maat is daardoor een kritische factor: software ontwikkelaars die hun rationele en technische capaciteiten kunnen koppelen aan volwassenheid wat betreft de relationele aspecten van hun werk dragen in belangrijke mate bij tot het succes van een software project.

Kwaliteit

De verschillende betekenissen van maat wijzen op een relatie tussen de menselijke maat en kwaliteit. Hierbij wordt niet alleen gedoeld op het opgeleverde product, maar komen ook verschillende aspecten van het ontwikkeltraject in beeld. Dat er een relatie bestaat tussen de kwaliteit van het proces en het product is natuurlijk niet nieuw. Het uitgangspunt van verbeterinitiatieven zoals SPI is niet voor niets dat een goed software engineering proces een voorwaarde is voor een goed product.

Aangenomen dat het welslagen van een software project mede afhankelijk is van de menselijke maat in al zijn hierboven aangestipte betekenissen, hoe kunnen de daarin besloten factoren dan zichtbaar en concreet gemaakt worden? Met andere woorden: in hoeverre, en langs welke wegen vertaalt de menselijke maat zich in de kwaliteit van software producten en diensten?
Hoewel voor velen herkenbaar, blijkt dit in de praktijk toch moeilijk grijpbare materie. Uiteindelijk is de te beantwoorden vraag of, en zo ja op welke wijze inzicht in deze zaken een wezenlijke bijdrage kan leveren aan complexe software ontwikkel trajecten – en daarmee aan verhoging van de software kwaliteit.

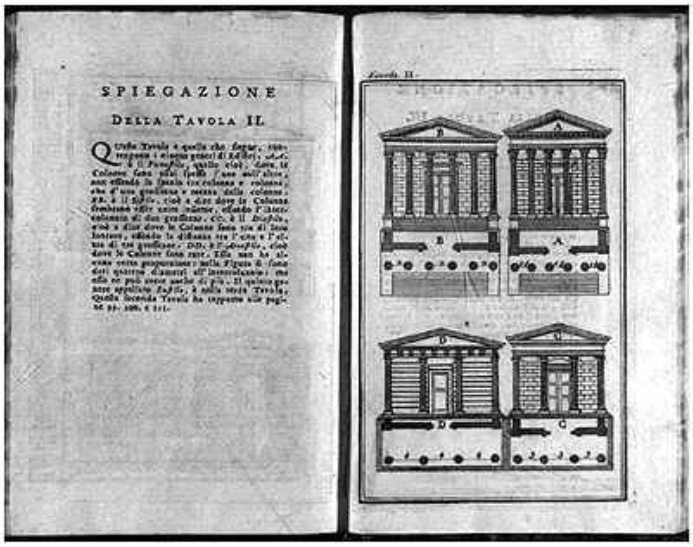
Kwaliteit is niet iets wat 'vanzelf' ontstaat omdat een bepaalde ontwikkel methodiek wordt gevolgd, of iets dat vlak voor een release in de software aangebracht kan worden. Uiteindelijk zijn het de mensen in het project die bepalend zijn voor de kwaliteit van het eindproduct – hun afwegingen en beslissingen, hun kennis, vakmanschap, en gereedschap, maar ook de wijze van samenwerking en de communicatie vormen de basis voor het behalen van de vereiste kwaliteit. De ervaring leert dat er klaarblijkelijk een scala van factoren is die invloed hebben op het welslagen van een software project, en het zijn zeker niet alleen rationele en technische factoren die hierbij maatgevend zijn.

Architectuur

In verhandelingen over software kwaliteit valt vaak de term *software architectuur*. Toch blijft meestal onduidelijk wat 'het bouwen van software onder architectuur' betekent, en in hoeverre dit dan bijdraagt aan het bereiken van de gewenste product kwaliteit. In analogie met de huizenmarkt, wordt al gauw verondersteld dat het kenmerk 'gebouwd onder architectuur' allereerst een effect zal hebben op de prijs. Toch kan software architectuur een bruikbare invalshoek zijn om de invloed van de menselijke maat te onderkennen en mee te nemen in het software ontwikkeltraject - met als doel tot een gunstige prijs/prestatie verhouding te komen.

Het vergelijken van software architectuur met het begrip architectuur uit de bouwwereld is weliswaar niet altijd terecht, maar kan ook hier verhelderend werken.

De betekenis van architectuur in de bouwwereld is reeds eeuwen geleden aangegeven door Marcus Pollio (70-25 BC), die in zijn standaardwerk '*Architectura Libris Decem*' stelde dat architectuur aan ten minste drie voorwaarden moest voldoen: *commoditas, firmitas en venustas*, ofwel het bouwwerk moet functioneel zijn, stevig van constructie zijn en een bepaalde schoonheid bezitten.



Pas veel later werd door Leon Alberti (1404-1472) het criterium '*concinnitas*' toegevoegd, waarmee orde en harmonie werden verbonden aan architectuur. In feite zegt deze voorwaarde dat architectuur pas dan tot z'n recht komt wanneer de samenstellende delen in de juiste verhouding tot elkaar staan, en tegelijkertijd het geheel in goede harmonie met de omgeving is.
Overigens zien we in dit laatste criterium een duidelijke overeenkomst met de principes van de Egyptische godin Maat.

Deze aspecten van architectuur gelden in feite nog steeds in de bouwwereld, en ondanks het verschil in historisch perspectief is ook de vertaalslag van gebouwen naar software niet moeilijk te maken: *commoditas* heeft te maken met het voldoen aan de (functionele) eisen van de toekomstige gebruikers, *firmitas* zegt iets over de robuustheid en levensduur van de software, en verwijst *concinnitas* naar conceptuele integriteit, welke tot uitdrukking komt in zowel een doordachte partitionering van het systeem (interne harmonie), als ook naar de inbedding van het geleverde product in de omgeving, in de organisatie met de mensen die het dienen te gebruiken (externe harmonie). Net als in de kunst is 'schoonheid' een moeilijk grijpbaar begrip, maar ook *venustas* heeft wel degelijk een analogie met het bouwen van software. Je zou kunnen zeggen dat architecturele schoonheid ontstaat als de verschillende aspecten van het ontwerp in evenwicht zijn, maar bovendien een hoger ideaal of idee tot uitdrukking brengen.

Wat leert ons deze vergelijking? Kennelijk is er een zekere relatie tussen voorwaarden die software architectuur stelt, en de kwaliteit zoals die uiteindelijk door de afnemer van de software wordt ervaren. Immers, kijken we naar de ISO-norm 9126 voor de kwaliteit van software producten, dan herkennen we in diverse kwaliteits-attributen de hierboven aangehaalde normen voor architectuur.

Ook voor software architecturen zijn er methoden ontwikkeld (zoals b.v. ATAM) om tot een architectuur te komen waarin rekening wordt gehouden met deze kwaliteitsfactoren. Nadere beschouwing laat ook zien dat de menselijk maat niet alleen in de functionaliteit en het gedrag van het systeem naar voren komt, maar zeker ook tot uitdrukking komt in de hier geschetste relatie tussen software architectuur en kwaliteit.

Vertaalslag

Net zo min als het volgen van een formele ontwikkelmethodiek een garantie is voor het binnen budget en planning afronden van een software project, zien we in de praktijk dat ook een in technisch opzicht goede software architectuur niet de vereiste kwaliteit van de afgeleverde software garandeert.

Klaarblijkelijk kunnen er vele kinken in de kabel komen in de vertaalslag van software architectuur naar product kwaliteit, die - als er sprake is van embedded software – roet in het eten kunnen gooien in de totstandkoming van het volledige systeem.

Nog afgezien van de beruchte 'erosie' van software architectuur door aantasting van de conceptuele integriteit (concinnitas!), maken groeiende complexiteit en steeds kortere time-to-market van software-intensieve produkten dit probleem alleen maar nijpender. Daarbij komt dat zeker in multi-disciplinaire omgevingen het steeds duidelijker wordt dat naast het oplossen van technologische uitdagingen, het juist omgaan met organisatorische aspecten ook een cruciale rol speelt. Conway ontdekte in 1968 al een wetmatigheid die nog niets aan zeggingskracht heeft ingeboet: 'de structuur van een systeem is een weerspiegeling van de organisatie die het systeem heeft ontworpen'[1].

De Software Architect

Tegen deze achtergrond is het gerechtvaardigd de vraag te stellen wat in brede zin de invloed is van de menselijke maat in software ontwikkeltrajecten. Mede door de geschetste samenhang tussen architectuur en kwaliteit lijkt met name de rol van vooral de software architect bepalend voor het tot uitdrukking brengen van de menselijke maat in zowel het ontwikkeltraject als het product.
Dit vereist echter een bredere invulling van de functie van architect dan gebruikelijk. Zo zal een architect dienen te participeren op een nivo dat verder gaat dan alleen het 'uitvoerende' werk: net als in de bouwwereld zal de architect een andere rol vervullen dan de aannemer. Een architect werkt niet alleen met mensen, maar ook voor mensen.

Organisaties kennen niet alleen een intern netwerk, maar hebben meestal ook te maken met een breed palet van stakeholders. Belangrijke afnemers kunnen met onverwachte eisen komen, en het verwezenlijken van nieuwe, vaak niet voorziene features vraagt soms om een offer wat betreft kwaliteit. Aangezien zij als geen ander de impact van onverwachte wijzigingen of nieuwe eisen kunnen inschatten, behoren software architecten betrokken te zijn bij het maken van dergelijke afwegingen. Daarbij is het niet uitgesloten dat men met een op zich technisch uitstekende oplossing toch de plank mis kan slaan om commerciële of bedrijfs-politieke redenen.

Hoe het ook zij, het is duidelijk dat dit aspect van het werk een beroep doet op de communicatieve vaardigheden van de software architect. Zeker een senior architect zal in staat moeten zijn om – in een constructieve relatie met alle belanghebbenden - allerlei organisatorische en procesmatige aspecten mee te wegen in belangrijke architecturele beslissingen. Dit doet een beroep op de kunst van het luisteren en de dialoog.

VRAPS-model

De invloed van allerlei niet-technische patronen op software architectuur is uitgebreid in kaart gebracht door een aantal vooraanstaande software architecten in hun boek: "Software Architecture – Organizational Principles and Patterns", verschenen in 2001 [2]. Het in dit boek gepresenteerde 'VRAPS-model' is geinspireerd door vijf groepen van 'organizational patterns' (best-practices) die in het software ontwikkeltraject een belangrijke rol spelen: **V**ision, **R**hythm, **A**nticipation, **P**artnering en **S**implification.

Het VRAPS-model bevat met elkaar samenhangende principes, die een indicatie geven of een organisatie in staat is om onder architectuur software te kunnen bouwen. Deze principes worden duidelijk gemaakt door middel van een groot aantal herkenbare patterns en anti-patterns. Het model geeft een kapstok voor het zichtbaar maken van de interaktie tussen een technische architectuur, de processen gebruikt voor de ontwikkeling en het onderhoud ervan, en de invloed van allerlei eigenschappen van het ontwikkel-team en de organisatie. Deze interaktie maakt nog eens te meer duidelijk, dat een architect in staat moet zijn om zowel technische als sociale complexiteit te hanteren.

In de context van de menselijke maat is het niet verwonderlijk dat we de diverse betekenissen van 'maat' terugzien in de principes van het VRAPS-model. Het VRAPS-model blijkt dan ook een practisch hulpmiddel te zijn om de invloed van de menselijke maat op software ontwikkeltrajecten in kaart te brengen.

Philippe Kruchten van Rational Software wijst op de relatie van het VRAPS-model met de uitgangspunten van het Rational Unified Process. Hij besluit zijn beschrijving van het VRAPS-model [3] met de woorden: "As I used to tell the architecture team on an air-traffic control project I worked on: Fifty percent of the job of an architect is communications -- communications with other teams, internally, with the customers, with suppliers, with other parts of the company -- twenty percent is planning; and the rest is technical stuff."

Verbetertrajecten

Hoe kan het VRAPS-model ons helpen grip te krijgen op de factoren die hun stempel drukken op het succes van een software project? Technisch vakmanschap vormt uiteraard de basis van dit succes, maar met name in een technologisch gedreven omgeving is men vaak niet gewend om ook eens langs een andere invalshoek naar software ontwikkeltrajecten te kijken. Het VRAPS-model biedt echter een concreet handvat om ook de niet-technische factoren boven tafel te krijgen. De vijf principes bieden niet alleen de helpende hand bij het stellen van een 'diagnose', maar geven ook aan welke mogelijke oorzaken en krachten aan de basis liggen van bepaalde (niet-effectieve) patronen, en wat de wegen zijn om weer uit een valkuil te komen.

Het VRAPS-model geeft aan dat het opzetten van een puur technische bril vaak het zicht ontneemt aan een werkelijk afdoende oplossing voor ontstane problematiek. De ervaring leert b.v. dat legacy problemen meestal niet alleen te wijten zijn aan technische factoren. Om die reden hebben migratie-trajecten een veel grotere kans van slagen als ook aandacht wordt besteed aan de diverse organisatorische en procesmatige factoren die hebben geleid tot het onstaan van het legacy systeem. Naast een op maat gesneden, moderne technologische oplossing dient de menselijke maat een rol te spelen in elk verbetertraject. Het VRAPS-model kan hierbij de helpende hand bieden om de gewenste kwaliteitsverbetering te realiseren.

Opleiding

Het VRAPS-model is niet alleen een eerste-hulp kit bij het ontsporen van de architectuur, maar door de beschrijving van veel-voorkomende patronen ook een indicatie van competenties waarover senior architecten dienen te beschikken. Helaas voorziet het reguliere opleidingstraject van software engineers hierin maar mondjesmaat, en dienen software architecten in de dop de broodnodige ervaring en kennis op dit gebied in de praktijk op te doen, aangevuld met algemene trainingen op gebied van persoonlijke vaardigheden.

Het VRAPS-model verschaft de opleiding van architecten in dit opzicht een samenhangend, op de praktijk afgestemd kader. Gebaseerd op duidelijk herkenbare gevallen uit de praktijk komen de noodzakelijke persoonlijke vaardigheden duidelijk naar voren. Hierdoor is het VRAPS-model bij uitstek geschikt om de invloed van de menselijke maat op software ontwikkeltrajecten niet alleen inzichtelijk te maken, maar ook te vertalen naar een gerichte training van (aankomende) software architecten.

Conclusie

Alhoewel het begrip ‘menselijke maat’ op verschillende manieren kan worden uitgelegd, is haar invloed op software projecten onmiskenbaar. Niet alleen de eindgebruikers, maar ook ontwikkelteams hebben te maken met de menselijke maat. Natuurlijk is de basis voor een software product of dienst altijd het vakmanschap van de makers, maar de praktijk leert dat een uitstekende technische oplossing geen garantie is voor een succesvol ontwikkeltraject.
Uiteraard zal het onderkennen van de relevantie van de menselijke maat die garantie ook niet geven, maar het zal het succesvol afronden van een software project zeker dichterbij brengen.

De menselijke maat blijkt vooral naar voren te komen in de samenhang tussen software architectuur en product kwaliteit. De software architect speelt een belangrijke rol in het ‘vertalen’ van de menselijke maat in belangrijke architecturele beslissingen (en vice versa), en heeft daarnaast ook een invloedrijke positie in de aansturing van het ontwikkelteam en de kontakten met de diverse stakeholders. Overigens is daarmee niet gezegd dat het vraagstuk van de menselijke maat alleen op het bord van de architect zou liggen.

Het hier besproken VRAPS-model slaat een brug tussen de wereld van de techniek en invloedrijke procesmatige en organisatorische aspecten. Het model geeft een practisch handvat om de menselijke maat een plek te kunnen geven in het ontwikkelproces. Een software ontwikkeltraject dat de menselijke aspecten in acht neemt is een belangrijke voorwaarde om tot een eindproduct te komen waarvan de gebruikers het gevoel hebben dat het op de menselijke maat geschoeid is.

Referenties

[1] http://info/astrian.net/jargon/terms/c/Conway_s_Law.html

[2] David Dikel, David Kane and James Wilson, Software Architecture: Organizational Principles and Patterns, Prentice-Hall, 2001

[3] http://www.therationaledge.com/content/feb_02/r_softwarch_pk.jsp

Noot:
De auteur bedankt prof. Dieter Hammer voor zijn waardevolle opmerkingen en positieve feedback.