Waarde creëren met softwarearchitectuur

Periodiek code-units rangschikken naar genericiteit

Stakeholders verwachten dat it waarde creëert. Ook architectuur moet hier volgens de auteurs aan bijdragen. Aan de hand van een praktijkvoorbeeld laten zij zien dat met softwarearchitectuur jaarlijks aanzienlijke directe kostenbesparingen zijn te bereiken.

Ton Tijdink en Chris Verhoef

Twee methodieken die bijdragen aan waardecreatie met it zijn het ontwerpen van softwareproductlijnen en kwantitatieve it-portfolioanalyse. Deze methoden sluiten goed aan bij de wat beter bekende maar evengoed nog weinig[1] toegepaste methoden als roi-bepaling en het opstellen van business cases, omdat ze deze van munitie voorzien.[2] Softwareproductlijnen kunnen een directe bijdrage leveren aan waardecreatie, de andere instrumenten zijn diagnostisch van aard en kunnen zowel potentiële als gerealiseerde waardecreatie aantoonbaar maken.

Varianten

Geografische spreiding van de organisatie, fusies, lokaal beheer of aanpassing aan de wensen van gebruikersgroepen (*customization*) werken het ontstaan van meerdere versies van softwaresystemen in de hand. Naarmate zulke varianten meer waarde creëren, worden de onderliggende systemen steeds specifieker en voor je het weet is er niet één systeem maar zijn er vele tientallen. De hiervoor benodigde wijzigingen worden ofwel lokaal aangebracht ofwel centraal op maat vervaardigd voor de betreffende locatie. Zonder een geëigende centrale aansturing vermenigvuldigt het oorspronkelijke systeem zich als een levend organisme en produceert daarbij een groot aantal varianten. Het in stand houden van dit complex van varianten heeft kostbare consequenties en roept onvermijdelijk tegenkrachten op.

Specifiek en generiek

Software is onderhevig aan krachten die specificiteit nastreven, afgewisseld door krachten gericht op genericiteit. Variabiliteit van softwaresystemen hoeft niet te worden beschouwd als een slecht fenomeen. Het vergroot de gebruiksmogelijkheden en de waarde voor de direct betrokkenen in de business. Business en softwareontwikkelaars kunnen in onderlinge samenwerking opportunistisch inspelen op veranderde omstandigheden en bedrijfsdoelen. Op korte termijn is de druk om in te gaan op specifieke eisen groot.

Op langere termijn, zeg twee tot drie jaar, komt de softwareontwikkeling in een onhoudbare situatie. De kosten van onderhoud groeien disproportioneel met de toenemende variatie, tot het moment aanbreekt dat aanpassingen praktisch gesproken niet meer haalbaar zijn. De roep om een generiek systeem met gestandaardiseerde gebruiksmogelijkheden wordt steeds luider. Na een aanzienlijke consolidatie-inspanning staat er een generiek systeem dat makkelijker is in te zetten en te onderhouden. De gebruikersgroep groeit, hetzij autonoom, hetzij door fusies en

1. Het AG-rapport 'ICT-investeringen in de spotlights' van april 2002 laat zien dat het met de kennis van financiële tools in de it slecht en met de toepassing ervan nog slechter is gesteld.
2. Het artikel van Tijdink (2001) betoogt dat het zwaaien met financiële formules als roi, roa, irr, npv en pbp alleen niet genoeg is. Je moet ook nog aan gegevens zien te komen om erin te stoppen. Veel van de voor it bedachte financiële methodieken laten in het midden hoe je aan valide en betrouwbare kwantitatieve gegevens komt. Kijk ook bij www.cs.vu.nl/~x/val/val.pdf. Hier is te zien hoe gegevens zijn te verkrijgen voor een realistische economische analyse.

Samenvatting

De auteurs beschrijven een in de praktijk toegepaste methode om de variatie van softwaresystemen in de hand te houden zonder gezonde groei te remmen. Eerst worden redundant aanwezige code-units verwijderd. Vervolgens wordt bepaald tot welke laag van genericiteit welke code-units behoren, waarna softwareontwikkelaars de units naar de passende lagen migreren. Deze operatie wordt periodiek herhaald.

overnames, en daarmee ook de druk om tegemoet te komen aan de eveneens steeds meer uiteenlopende wensen.
De volledig generieke code maakt het echter moeilijk om opportunistisch tegemoet te komen aan veranderde omstandigheden en wensen.
De druk op het vervaardigen van lokale varianten neemt weer toe en de cyclus begint opnieuw. De organisatie heeft intussen geen enkel zicht meer op de aard en omvang van het aantal softwarevarianten. Bijvoorbeeld telefooncentrales, medische systemen en *global-transaction-* and *settlement*systemen zijn vatbaar voor de eb- en vloedwerking van specificiteit en genericiteit.

Beheersbaar

Het is de kunst om de variatie van softwaresystemen en de kostbare consequenties ervan in de hand te houden zonder gezonde groei te remmen. De business kan zo geld blijven verdienen met behulp van it, zonder dat de kosten ervan prohibitief worden. De oplossing is het zichtbaar maken van de varianten en het omzetten van de grootste gemene deler in een overkoepelend systeem dat samen met de punten van verschil migreert naar een softwareproductlijn. Voor het zichtbaar maken en het beheersbaar houden van softwarevarianten staat ons kwantitatief it-portfoliomanagement ten dienste, voor het opzetten van een softwareproductlijn de snoei- en groeimethode.

»Software is afwisselend onderhevig aan krachten die specificiteit nastreven en krachten gericht op genericiteit.«

Kwantitatief it-portfoliomanagement

Kwantitatief it-portfoliomanagement is een instrument waarmee de *corporate-supportstaf* van de cio organisatiebreed een indruk verkrijgt van de stand van zaken van de totale it-portfolio en hoe de vandaag gemaakte kosten uitwerken in de budgetten van morgen. Ook maakt deze methode de risico's duidelijk die in de it-portfolio besloten liggen, en de *what-if*scenario's van toekomstige investeringen in it.
De methode gebruikt in sommige gevallen een wiskundige opzet. De corporate-supportstaf rekent aan de it-portfolio op basis van publieke of bedrijfsbenchmarks. Ook organisaties zonder historische informatie van hun it-portfolio kunnen zo inzicht verkrijgen. In andere gevallen is de corporate-supportstaf daadwerkelijk in staat operationele code te analyseren voor het beantwoorden van vragen over de portfolio. Een uitgebreide beschrijving van kwantitatief it-portfoliomanagement is te lezen in Verhoef (2002).

De corporate-supportstaf kan de in negatieve zin afwijkende kosten- en risicopatronen in de portfolio ontdekken om er vervolgens softwaresystemen mee op te sporen die in aanmerking komen voor de groei- en snoeimethode. Hier komt de softwarearchitect in beeld. Deze werkwijze lijkt misschien vanzelfsprekend. Dat is hij niet, omdat in de praktijk blijkt dat vooral het succes van een softwaresysteem het zoeken naar en managen van ongebreidelde groei ervan in de weg staat.

Groei en snoei

Om de werking van de snoei- en groeimethode toe te lichten, bewijst de metafoor van de softwarearchitect als fruitteler een goede dienst. Met kennis van zaken snoeien is al eeuwen het recept voor zo groot mogelijke waardecreatie in de fruitteelt. Jonge fruitbomen moeten uitgroeien om voldoende vrucht te dragen, maar ongebreidelde groei is contraproductief. Een boom heeft

een stam, takken en twijgen die in een bepaalde vorm en robuustheid aanwezig moeten zijn om het gestelde doel te bereiken in een efficiënt verlopend bedrijfsproces. Wat efficiënt is, wordt bepaald door lokale omstandigheden, in combinatie met een bepaalde stand van kennis en techniek. Zo heeft de opkomst van de mechanisatie in de fruitteelt op veel plaatsen hoogstamfruit verdrongen ten gunste van laagstamteelten.
De onderverdeling in geledingen geldt ook voor software. De stam staat voor de organisatiebrede laag, de takken voor de regionale of *line-of-business*-laag en de twijgen en bladeren voor de lagen met de meeste specifiteit. Overdadige snoei remt de groei en kan tot afsterving van de boom leiden. Hetzelfde geldt voor software wanneer de krachten die genericiteit nastreven te veel de overhand hebben. Te weinig snoeien leidt uiteindelijk tot ontoegankelijkheid van de boomgaard voor machines. Als in de softwareontwikkeling de krachten die specificiteit bewerkstelligen te lang werkzaam zijn, komt hier uiteindelijk de afweging tussen groot onderhoud of rooien nabij. Welke keuze de cio ook maakt, kostbaar wordt het zeker.
Technisch gesproken is het lastig om de groei- en snoeimethode in softwareontwikkeling in te voeren. De softwarearchitect treft meestal geen nette laagstamteelt aan maar, vooral door het ontbreken van de regionale laag, een grote berg stammen, twijgen en bladeren. Gezien de vette kernen en de opgeblazen *clients* blijken projecten die beter regionaal waren uitgevoerd, bedrijfsbreed of juist lokaal te zijn uitgevoerd. Ongeveer zoals een knotwilg eruit ziet: een dikke stam met dunne twijgen en geen takken met een dikte ertussenin. De oplossing is het rangschikken (realignment) van de code-units in de juiste mix van genericiteit en specificiteit, in halfjaarlijkse afwisselende groei- en snoeiseizoenen.

Softwareproductlijn

Een softwareproductlijn kent verschillende lagen van genericiteit. Een sprekende term voor een softwarearchitectuur die deze gelaagdheid volgt, is een federatieve architectuur. Deze representeert de mate waarin de business impact heeft op het systeem. Als de impact niet verder reikt dan lokaal benodigde functionaliteit, dan moet ook de impact van techniek en management beperkt blijven tot dat niveau. Een federatieve architectuur zorgt voor alignment van genericiteit tussen business en techniek. Softwareproductlijnen hebben minimaal drie lagen van genericiteit: organisatiebreed, regionaal en lokaal. De groei- en snoeimethode werd met succes toegepast bij een wereldwijd opererende bank. Hij werd gefaseerd uitgevoerd.
De eerste fase is een grote verjongingssnoei van het systeem. Op aanwijzing van de softwarearchitecten van de bank worden overduidelijk redundant aanwezige code-units verwijderd en geschoonde releases uitgegeven. In de tweede fase passen de architecten zes maatstaven toe, om vast te stellen welke code-units tot welke laag van genericiteit behoren. De maatstaven omvatten de mate van gebruik, frequentie van onderhoud, omvang, specificiteit, complexiteit en variabiliteit van code-units. Ten slotte stellen de softwarearchitecten in de derde fase beslisregels op, op grond van de in de fase daarvoor gevonden waarden. Hiermee migreren softwareontwikkelaars de code-units naar de betreffende lagen. Het resultaat is de release van een softwareproductlijn. Deze omvat alle varianten van het systeem, geordend in een federatieve architectuur van drie lagen.

De baten

De besparingen in het voorbeeld liggen in de orde van 3 tot 7% van het jaarlijkse budget voor het onderhanden genomen systeem, in absolute termen een miljoenenbedrag dat jaarlijks vrijkomt. Alignment van code-units in de regionale laag was cruciaal voor het bereiken van de baten. Tegenover de kosten van de groei- en snoeioperatie staan de baten van reductie van code-unitredundantie, de gemiddelde bouwkosten per code-unit en de kosten van *release digestion*. Verder zijn er minder herstelkosten door constructiefouten en lagere testkosten.
De reductie van de kosten van code-unitredundantie en release digestion dragen voor 80% bij aan de netto baten. De kosten van de realignmentoperatie bedragen nog geen 3% daarvan.

De kostenbesparingen werden in eerste instantie geraamd en konden later worden bevestigd, waaruit bleek dat de methode van raming voorspellende waarde had. Reductie van code-unit-redundantie leidt tot minder ontwikkelkosten. Overtollige code-units die worden vernietigd, vragen geen aandacht meer.
De kosten werden geraamd door uit te gaan van de gemiddelde kosten per code-unit voor de opruiming van de overbodige exemplaren. De gemiddelde bouwkosten per code-unit nemen af doordat door de realignment van code-units de gemiddelde omvang van nieuwe (releases van) code-units afneemt en daarmee de gemiddelde ontwikkelinspanning.
De winst zit vooral in de verminderde inspanning voor releasemanagement. Reductie van redundantie en afname van de omvang van code-units leidt tot minder constructiefouten en dus tot minder herstelwerkzaamheden. Opgeruimde code-units hoeven niet te worden getest. Ook de stroomlijning van het systeem in een softwareproductlijn leidt tot lagere kosten van testen.
Er zijn hoge kosten verbonden aan release digestion, het accepteren en opnemen van een nieuwe release door de gebruikersorganisatie. Dit geldt zeker als er lokaal bedoelde maar organisatiebreed ingevoerde wijzigingen moeten worden doorgevoerd die voor de meeste locaties irrelevant zijn. Naast overbodig werk brengt release digestion ook extra risico's op storingen met zich mee. Realignment van de lagen van genericiteit minimaliseert deze kosten en risico's, met als extra effect dat releases kleiner en minder complex worden. Ook vermindert de kans op *release indigestion*, het niet tijdig kunnen verwerken van nieuwe releases met alle risico's die daarbij horen.

»De winst zit vooral in de verminderde inspanning voor releasemanagement.«

Naast de directe kostenbesparingen zijn er indirecte besparingen die evenwel lastiger zijn te berekenen en moeilijk zijn toe te schrijven aan de groei- en snoeioperaties. Denk aan betere kennis van het systeem, snellere probleemdiagnose en grotere klanttevredenheid door minder frequente en complexe releases. Harde argumenten zijn er wel voor baten in de opbrengstensfeer, omdat met een softwareproductlijn die is opgezet volgens een federatieve architectuur de business flexibeler en dus sneller en effectiever wordt ondersteund in het benutten van kansen.

Met kwantitatieve gegevens zoals die in het praktijkvoorbeeld zijn verzameld, kan het it-management eenvoudig de roi bepalen of een business case bepalen voor deze softwarearchitectuuractiviteiten. Het kwantitatief aantoonbaar maken van de toegevoegde waarde van softwarearchitectuur is een effectieve methode om te valideren dat investeringen zich terugbetalen. Waardecreatie is een effectief criterium voor de beoordeling van it-architectuurbenaderingen. Waardecreatie als focus van architectuur is niet zozeer een nieuw kunstje als wel een vrucht van een fundamentele omslag in het denken over it, zoals in de Renaissance. De oude denkwijze, *sum ergo credit*, ik besta dus ik creëer waarde met it, is een algemeen voorkomende maar niettemin onhoudbare stelling. De pluk-de-dag-architectuurbenaderingen die daaruit voortkomen zijn dus onder de maat. Wat overblijft is de nieuwe denkwijze van *credit ergo sum*, ik creëer waarde met it dus ik besta. De nieuwe, nog weinig voorkomende denkwijze richt zich op het aantoonbaar inlossen van de beloften van it.

Literatuur

Diest, J.C.W. van, & R.W. de Graaf (2002). *ICT-investeringen in de spotlights*. Den Haag: Ten Hagen & Stam Uitgevers.

Tijdink, A.J. (2001). Het meten en realiseren van de baten van ICT-investeringen. In: *Business Process Magazine*, juni 2001, nr. 4, pp. 37-40.

Verhoef, C. (2002). Quantitative IT portfolio management. *Science of Computer Programming* 45 (2002) 1-96. Elsevier Science B.V.

Verhoef, C., & D. Faust (2003). Software product line migration and deployment. *Software – Practice & Experience*. (in press) (DOI: 10.1002/spe.530) John Wiley & Sons, Ltd.

Ir. Ton Tijdink MSc
is *senior adviseur bij Cibit adviseurs | opleiders*. E-mail: ttijdink@cibit.nl

Prof.dr. Chris Verhoef
werkt aan het instituut voor Informatie Management en Software Engineering van de Faculteit der Exacte Wetenschappen, Vrije Universiteit.
E-mail: x@cs.vu.nl.