EEN METHODISCHE AANPAK

De waarde van **softwareproducten** bepalen

Sinds de kredietcrisis in 2008 wordt scherper gekeken naar de financiële kant van IT en is er een groeiende behoefte aan duidelijkheid over de waarde die IT-componenten als bezitting vertegenwoordigen. Dit artikel presenteert een methode die organisaties in staat stelt om op een systematische wijze tot een waardering van maatwerkwerksoftware te komen.

JELLE DE GROOT EN JOOST VISSER

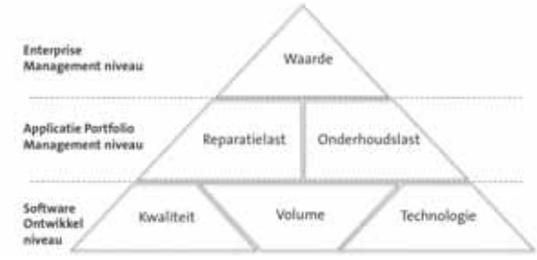
Je kunt op twee manieren tegen de waarde van IT-componenten aankijken. De eerste manier is om de kosten van IT-componenten te zien als investeringen die gedaan worden om de prestaties van de organisatie te verbeteren. Dit is de zienswijze die ten grondslag ligt aan kosten-batenalyses in business cases bij IT-investeringsbeslissingen. Het is de gebruikelijke benadering van IT-financiën in een *going concern* situatie.
De tweede manier is om na te gaan welke waarde IT-componenten vertegenwoordigen als bezitting van de organisatie. Deze zienswijze past bij due diligence-onderzoek in het geval van een faillissement of bedrijfsovername. In dit artikel bespreken we een waarderingsmethode die aansluit bij deze tweede zienswijze, waarbij we ons richten op software.

Een veelgebruikte maat voor de waarde van software als bezitting is de aanschafwaarde ervan, dat wil zeggen de kosten die gemaakt zijn om de software te verwerven. Bij maatwerksoftware zijn dat de ontwikkelkosten. Dit bedrag is weliswaar relatief eenvoudig te bepalen, maar is eigenlijk nauwelijks relevant als we willen weten welke waarde de maatwerksoftware als bezitting van de organisatie vertegenwoordigt. In dit artikel beschrijven we een methode die organisaties in staat stelt om op een systematische wijze tot een waardering van maatwerkwerksoftware te komen die wel relevante bedragen oplevert. Deze methode is gebaseerd op meetbare eigenschappen van het softwareproduct.

In de volgende paragrafen beschrijven we de meetbare eigenschappen van software waar we van uitgaan. Vervolgens schetsen we een drietal manieren om de meetwaarden te vertalen naar financiële waarden. We sluiten af met conclusies.

SOFTWAREWAARDERINGS-PIRAMIDE

Wil je de financiële waarde van software enigszins objectief kunnen bepalen, dan moet je een softwareproduct kunnen vergelijken met andere softwareproducten. In het geval van maatwerk lijkt dit op het eerste gezicht onmogelijk – maatwerksoftware is immers uniek en dus per definitie niet vergelijkbaar met andere software. Dat is waar, maar wat uniek is aan maatsoftware is het gebouwde product zelf. De programmeertalen die gebruikt zijn om de maatwerksoftware te bouwen, zijn niet uniek. Ze worden voor uiteenlopende softwareprojecten gebruikt,



Figuur 1: De softwarewaarderingspiramide

en het is op dit niveau dat we verschillende maatwerksoftwareproducten onderling op een zinvolle manier kunnen vergelijken. Als bijvoorbeeld een bepaald softwareproduct is gebouwd in Java, dan kunnen we het product vergelijken met andere programma's van vergelijkbare omvang die ook in Java zijn gebouwd.

In dit artikel introduceren we een raamwerk dat we de softwarewaarderingspiramide noemen, die is weergegeven in figuur 1. Dit raamwerk gebruiken we om te komen tot een waardering van software op basis van programmeertaal en meetbare eigenschappen zoals omvang en technische kwaliteit van de software. Hierdoor wordt het mogelijk om vrij objectief tot een relevante schatting van de waarde van maatwerksoftware te komen. De piramide is ontwikkeld in een afstudeeronderzoek bij de Universiteit Leiden, in samenwerking met de Software Improvement Group (SIG) [GROO11].

De piramide bestaat uit drie lagen. De basis wordt gevormd door het Software-ontwikkelniveau. Op dit niveau worden meetbare eigenschappen zoals volume en compexiteit van de broncode vastgesteld.
Een niveau hoger is het niveau van Applicatie-portfoliomanagement,
waar de gemeten eigenschappen een bedrijfseconomische vertaling krijgen. Op dit niveau kunnen op basis van informatie uit het eerste niveau beslissingen genomen worden om bijvoorbeeld software eventueel uit te faseren, om een voorgesteld ontwikkelproject niet te starten of om bestaande software te verbeteren.
Het derde en hoogste niveau is dat van Enterprisemanagement. Op dit niveau kunnen we de waarde van maatwerksoftware inschatten op basis van informatie uit de twee lagen eronder, op een wijze die financiële waarden oplevert welke relevant zijn voor de business. De waarde-indicatoren op dit niveau zijn ontworpen om verschillende maatwerkcomponenten onderling te kunnen vergelijken. Een waarde-inschatting met gebruikmaking van de softwarewaarderingspiramide geeft een degelijk onderbouwd antwoord op de vraag wat een redelijke prijs is voor een gegeven stuk maatwerksoftware. Daarmee is het een instrument voor beslissers die geen IT-specialist zijn om gevoel te krijgen voor prijzen van softwareproducten.

In de volgende paragrafen lichten we elk van de bouwstenen toe en illustreren we het gebruik van de piramide aan de hand van een concrete casus. Daarbij hebben we de werkelijke getallen enigszins aangepast om de anonimiteit te waarborgen. De casus betreft een belangrijke, middelgrote maatwerk-webapplicatie, die ontwikkeld is voor een bedrijf binnen de telecommunicatiebranche.

BASISNIVEAU: SOFTWARE-ONTWIKKELING

De piramide omvat op het basisniveau drie bouwstenen die meetbare eigenschappen (Kwaliteit en Volume) en de onderliggende technologie (in het bijzonder de programmeertaal) beschrijven.

Kwaliteit

In de eerste bouwsteen op het basisniveau bepalen we de (technische) kwaliteit van softwareproducten. Dit kan op verschillende manieren gebeuren. In dit artikel gebruiken we het kwaliteitsmodel van de SIG. Dit model onderscheidt de zes volgende kwaliteitskenmerken van een softwareproduct:[1]

- Volume: de totale omvang van de applicatie, uitgedrukt in regels broncode, uitgesplitst naar programmeertaal.
- Duplicatie: de mate waarin dezelfde code verschillende keren voorkomt, wat vaak voortkomt uit overmatig gebruik van copy-paste.
- Unitcomplexiteit: het aantal mogelijke paden dat de computer kan doorlopen bij het uitvoeren van een code-element zoals een functie of methode.
- Unitgrootte: het aantal regels broncode van een code-element.
- Unit interfacing: het aantal interfaces per code-element.
- Module coupling: de mate waarin de verschillende softwaremodules van elkaar afhangen, bijvoorbeeld doordat ze elkaars data gebruiken.

Elk van deze zes karakteristieken kunnen we meten, waarna we de meetwaarden kunnen vergelijken met een benchmark. De benchmark van SIG is gebaseerd op een database waarin de resultaten zijn opgenomen van metingen op inmiddels meer dan 600 softwareproducten. De gemeten waarde van elke eigenschap van een te waarderen softwareproduct vergelijken we met de verzameling waarden van de corresponderende eigenschap van de softwareproducten in de database. Het resultaat van de vergelijking is een score op een schaal van 1 tot 5 sterren. De laagste kwaliteitswaarde is 1 ster en het hoogste 5 sterren. In dit kwaliteitsmodel behoort een 5-sterrenproduct tot de 5% beste softwareproducten, een 4-sterrenproduct tot de 30 procent daaronder, een 3-sterrenproduct tot de 30 procent daaronder, een 2-sterrenpro-

duct tot de 30 procent daaronder en een 1-sterproduct tot de 5 procent slechtste producten. Meer informatie over het model is te vinden in [BAGG10].

De kwaliteit van de applicatie uit ons voorbeeld is bepaald op 2.9 sterren (★★★☆☆) op de schaal van 1 tot 5.

Eigenschap	Waarde	
Java-volume	173.000	regels code
JSP-volume	49.000	regels code
JavaScript-volume	18.000	regels code
Herbouwinspanning	373,5	manmaanden

Tabel 1: Voorbeeld van volume per programmeertaal en de daaruit afgeleide totale herbouwinspanning

Volume

In de tweede bouwsteen op het basisniveau meten we het volume van softwareproducten. Hieronder verstaan we de totale omvang van een softwareproduct, uitgedrukt in regels broncode. Lege regels en commentaarregels tellen we natuurlijk niet mee. Het getelde aantal regels wegen we aan de hand van de marktgemiddelde productiviteit van ontwikkelaars die een bepaalde programmeertaal gebruiken [JONE11] – ter illustratie: voor Java bedraagt de marktgemiddelde productiviteit ongeveer 9.000 regels code per ontwikkelaar per jaar. Zo verkrijgen we een waarde die het volume van een applicatie uitdrukt in het aantal manmaanden ontwikkelinspanning dat gemiddeld nodig zou zijn om de applicatie opnieuw te bouwen, de zogenaamde herbouwinspanning.

Technologie

In de derde bouwsteen op het basisniveau splitsen we het volume van een softwareproduct uit naar gebruikte technologie, waaronder we in dit artikel de programmeertaal verstaan. Bij de bouw van vrijwel alle maatwerksoftware wordt meer dan één programmeertaal gebruikt, en de productiviteit van ontwikkelaars varieert nu eenmaal tussen programmeertalen. Het volume van een softwareproduct splitsen we daarom uit naar programmeertaal.

Bij de bouw van de applicatie uit onze casus zijn drie programmeertalen gebruikt. De laag met business logica is gebouwd in Java en voor de presentatielaag zijn JSP en JavaScript gebruikt (zie tabel 1). Op basis van industrie-gemiddelde productiviteitscijfers hebben we de herbouwinspanning bepaald op 373,5 manmaanden.

MIDDENNIVEAU: APPLICATIEPORTFOLIOMANAGEMENT

Het middenniveau van het model bestaat uit twee bouwstenen, te weten Reparatielast en Onderhoudslast. Deze twee bouwstenen zijn te gebruiken in berekeningen voor business cases maar vormen ook de basis voor de bepaling van de financiële waarde van maatwerksoftwareniveau op het topniveau van het model. Om deze bouwstenen uit te leggen hebben we het begrip 'Technische schuld' nodig.

Het concept Technische schuld ('Technical Debt') is bedacht door Ward Cunningham [CUNN93]. Het idee is dat als je software oplevert die nog technische onvolkomenheden heeft, je in feite een schuld op je neemt. Deze schuld kun je eventueel later terugbetalen door dan de inspanning te leveren waarmee je het softwareproduct alsnog vervolmaakt. Het concept Technische schuld is een metafoor voor tekortschietende kwaliteit van een softwareproduct, wat allerlei oorzaken kan hebben. Een voorbeeld van een mogelijke reden waarom software met mindere kwaliteit opgeleverd wordt, is een snelle time-to-marketstrategie. Een korte oplevertermijn staat dan voorop en kwaliteitszorg is van minder belang. Maar het kan ook zijn dat de kwaliteit te lijden heeft onder slecht programmeren door onervaren ontwikkelaars. Welke oorzaken de in het ontwikkelproces opgebouwde technische schuld ook heeft, het gevolg is dat het ontwikkelen misschien goedkoper is doordat het snel en met goedkope programmeurs gebeurt, maar dat de opgeleverde software van mindere kwaliteit is. Hierdoor is het onderhoud en de eventuele doorontwikkeling van de opgeleverde software lastiger, dus duurder.

Maatwerksoftware wordt doorgaans regelmatig aangepast aan veranderende situaties en nieuwe wensen. De praktijk leert dat de onderhoudslast bij software die technisch van slechte kwaliteit is, in de loop van de jaren exponentieel groeit (zie figuur 2) [NUGR10]. Deze exponentiële stijging van de onderhoudslast bij slechte software, software met een technische schuld dus, is te vergelijken met het principe van 'rente op rente' bij een banklening.

In figuur 2 zijn de exponentiële groei van de technische schuld en de onderhoudskosten weergegeven. De optimale onderhoudslast wordt weergegeven door de onderste streepjeslijn, die over de tijd constant blijft wanneer het kwaliteitsniveau van de opgeleverde software op het optimale niveau is en er dus geen sprake is van technische schuld. Wanneer er door suboptimale kwaliteit wel een technische schuld is, is er 'technische rente' verschuldigd. Door het rente op renteprincipe stijgt de technische rente en daarmee de technische schuld exponentieel met de tijd. Dit betekent dat de inspanning die nodig is om de software alsnog op het optimale kwaliteitsniveau te krijgen, exponentieel stijgt.

De optimale kwaliteit is van essentieel belang voor het bepalen van de

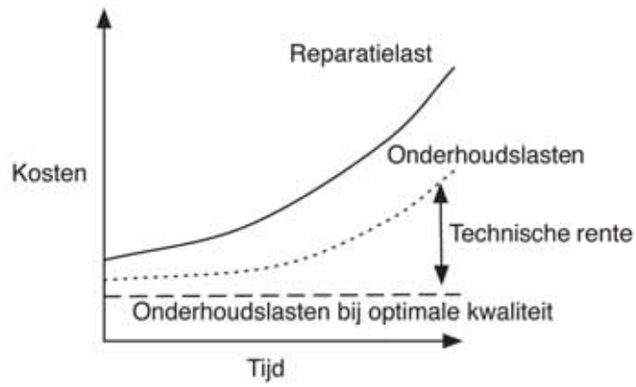
hoogte van de technische schuld en de daarbij horende cumulatieve rente die in de loop van de tijd betaald moet worden. Zoals gezegd, is de optimale kwaliteit de kwaliteit die correspondeert met een jaarlijkse onderhoudslast die over de jaren heen gelijk blijft. De optimale kwaliteit van softwareproducten stellen we vast per organisatie en per product. Voor de meeste software is een 4-sterrenniveau optimaal, maar voor software met zeer hoge betrouwbaarheidseisen kan het van belang zijn om naar 5 sterren te streven. Het optimale niveau van de applicatie uit ons voorbeeld hebben we bepaald op 4,0 sterren (★★★★☆).

Nu deze achtergrondinformatie gegeven is, kunnen we de bouwstenen 'Reparatielast' en 'Onderhoudslast' bespreken.

Reparatielast

In de eerste bouwsteen op het middenniveau van de softwarewaarderingspiramide bepalen we de reparatielast. Deze is gelijk aan de technische schuld in figuur 2 en komt overeen met de inspanning die nodig is om de software kwalitatief te verbeteren tot het optimale niveau. De reparatielast is in het geval van ons voorbeeld het verschil tussen 2,9 sterren (het feitelijke kwaliteitsniveau) en 4,0 sterren (het optimale kwaliteitsniveau). Dit verschil bedraagt 1,1 ster. Nu de reparatielast is geïdentificeerd, kan het in een geldbedrag worden uitgedrukt. Hiervoor hebben we nog drie andere gegevens nodig, namelijk Reparatiefractie, Vervangingswaarde en Refactoring-aanpassing. Deze begrippen leggen we hierna uit.

De Reparatiefractie is het percentage regels code dat veranderd moet worden om aan het optimale kwaliteitsniveau van 4 sterren te voldoen. Dit getal wordt afgeleid uit de eerste bouwsteen van het basisniveau, de bouwsteen Kwaliteit. In ons voorbeeld is de kwaliteit 2,9 sterren en de optimale kwaliteit 4 sterren. Deze kwaliteitssprong van 1,1 ster vergt volgens de data in de benchmarkdatabase[2] gemiddeld een aanpassing van 39 procent van het codevolume.



Figuur 2: Technische schuld, onderhoudslast en reparatielast

De Vervangingswaarde van het softwareproduct wordt berekend met de tweede en derde bouwsteen van het basisniveau (Volume en Technologie). Zoals gezegd, wordt het volume van software vastgesteld in aantallen regels code uitgesplitst naar programmeertaal, en vervolgens omgerekend naar de gemiddelde inspanning in manmaanden die nodig is om een soortgelijk softwareproduct te ontwikkelen. Ten slotte wordt deze inspanningslast in financiële termen uitgedrukt. De Vervangingswaarde bestaat dus uit de gemiddelde kosten die gemaakt moeten worden om een soortgelijk softwareproduct te ontwikkelen. In ons voorbeeld is dit een bedrag van € 3.114.000.

De Refactoring-aanpassing houdt rekening met zachtere aspecten die de efficiëntie van het ontwikkelproces kunnen beïnvloeden. Zo kan de productiviteit van een ontwikkelteam naar boven of beneden afwijken van het gemiddelde, bijvoorbeeld door organisatorische en technische omgevingsfactoren, door verschillen in de omvang van teams en door de gebruikte programmeertools. Zo kunnen wij ons voorstellen dat een ervaren team, dat een bonus in het vooruitzicht gesteld heeft gekregen, productiever zal zijn dan een ongemotiveerd offshore team dat de bouw van het product ziet als de zoveelste routineklus. De Refactoring-aanpassing is een percentage dat wordt gebaseerd op een expertopinie. In ons voorbeeld werd de refactoring-aanpassing op 20 procent gesteld omdat het ontwikkelteam ruime ervaring bezat.

De formule voor de berekening van de reparatielast is: Reparatielast = Reparatiefractie * Vervangingswaarde * (1 – Refactoring-aanpassing).
Wanneer we de waarden voor Reparatiefractie, Vervangingswaarde en Refactoring-aanpassing invullen in deze formule, krijgen we:
Reparatielast = 39 procent * € 3.114.000 * (1 - 20%) = € 970.000.
De reparatielast voor de applicatie uit onze casus schatten we dus op ongeveer 970 duizend euro.

Extra onderhoudslast

In de tweede bouwsteen op het middenniveau bepalen we de extra onderhoudslast. Dit zijn extra onderhoudskosten aan een softwareproduct, als gevolg van de lagere technische kwaliteit ervan. De Extra onderhoudslast is gelijk aan de Technische rente in figuur 2.

De Extra onderhoudslast wordt bepaald door de Herbouwinspanning, de Onderhoudsfractie en de Kwaliteitsfactor. De Herbouwinspanning voor onze casus is 373,5 manmaanden zoals uitgerekend in ▸

de bouwsteen 'Technologie' uit het basisniveau van de piramide. De Onderhoudsfractie staat voor het volume (regels code) aan onderhoud dat per jaar voor het softwareproduct nodig is in het kader van de natuurlijke groei van het product. Dit betreft 'standaard' onderhoud, dus zonder grote wijzigingen. Uit onderzoek blijkt dat organisaties hiervoor gemiddeld per jaar 15 procent van het totale volume aan code wijzigen. Vervolgens komt daar de Kwaliteitsfactor bij, dat is een percentage dat bovenop de natuurlijke groei van een applicatie komt doordat de software van een lagere kwaliteit is. De plausibele aanname hierbij is dat bij slechtere kwaliteit ook meer nieuwe code nodig is om nieuwe onderdelen aan de software te kunnen toevoegen.[3] Tevens wordt er in het vervolgonderhoud doorontwikkeld op een kwalitatief slechtere code, wat ook weer extra onderhoudslast met zich meebrengt. Voor onze casus is deze factor 0,97.

De formule voor de berekening van de onderhoudslast is: Onderhoudslast jaar 1 = Onderhoudsfractie * Herbouwinspanning / Kwaliteitsfactor. Als we de cijfers invullen voor onze casus krijgen we:
Onderhoudslast jaar 1 = 15 procent * 373,5 / 0,97 = 57,8 manmaanden.

Op basis van de onderhoudslast kunnen we berekenen welke *extra* onderhoudslasten zullen voortvloeien uit de lagere kwaliteit van het softwareproduct. Voor de applicatie uit het voorbeeld zijn de extra onderhoudslasten weergegeven in tabel 2. Dit komt uit op een totaal van € 2,2 miljoen aan extra onderhoudslasten over een levensduur van 7 jaar van onze casusapplicatie met een kwaliteit van 2,9 sterren. Zie het tekstkader voor een toelichting op de berekeningswijze.

Berekening extra onderhoudslasten
Als we 1 fte stellen op € 100.000 per jaar, bedragen de kosten van de 57,8 manmaanden van jaar 1 € 481.000. Dit is de totale onderhoudslast bij een kwaliteitsniveau van 2,9 sterren. Als we dezelfde formule gebruiken bij een kwaliteitsniveau van 4-sterren zal de kwaliteitsfactor veranderen naar 1,4. Met deze hogere kwaliteitsfactor zou de onderhoudslast € 335.000 zijn wat betekent dat er € 150.000 aan extra onderhoudslast is door de lagere kwaliteit bij 2,9 sterren (dit getal vinden we terug in tabel 2 bij jaar 1). Dit is alleen nog voor het eerste jaar uitgerekend. Doordat er in de daarop volgende onderhoudsjaren op kwalitatief lagere software doorontwikkeld wordt, zal de onderhoudslast ook toenemen. Om deze lasten in te schatten, moeten we rekening houden met de groei van het totale volume van het systeem en extra opgelopen technische schuld. Zie voor verdere details [NUGR10].

TOPNIVEAU: ENTERPRISEMANAGEMENT

Het model kent op het topniveau één bouwsteen: 'Waarde'. Hiermee aggregeren we alle metingen en andere informatie uit de onderliggende niveaus tot één getal dat de financiële waarde van de maatwerksoftware voor de organisatie uitdrukt. We gebruiken hiervoor drie verschillende modellen. Uitgangspunt is steeds de vervangingswaarde, dus de gemiddelde kosten die gemaakt moeten worden om een soortgelijke applicatie te ontwikkelen. Elk model corrigeert vervolgens op een andere manier voor suboptimale kwaliteit van de software. Terwijl de informatie op het middenniveau vooral relevant is voor de CIO, ontstaat op het topniveau de financiële informatie voor de CFO.

Jaar	Extra onderhoudslasten
1	€ 150.000
2	€ 190.000
3	€ 240.000
4	€ 300.000
5	€ 370.000
6	€ 440.000
7	€ 510.000
Totaal	€ 2.200.000

Tabel 2: Extra onderhoudslasten over de jaren door lagere kwaliteit

MODEL 1: CORRECTIE VOOR REPARATIELAST

Het eerste model voor waardebepaling van maatwerksoftware corrigeert voor suboptimale kwaliteit door de vervangingswaarde van de software te verminderen met de Reparatielast, dat wil zeggen de reparatiekosten die gemaakt moeten worden om de kwaliteitsgebreken te verhelpen: Softwarewaarde = Vervangingswaarde – Reparatielast. Wanneer we hierin de getallen voor Vervangingswaarde en Reparatielast uit het middenniveau van de softwarewaarderingspiramide invullen, krijgen we:
Softwarewaarde (model 1) = € 3.114.000 - € 970.000 = € 2.144.000.

De waardebepaling volgens model 1 geeft dus een softwarewaarde van 2,1 miljoen euro, berekend als de vervangingswaarde, verminderd met de benodigde reparatiekosten om de software op het optimale niveau te krijgen.

MODEL 2: CORRECTIE VOOR SLECHT GEPROGRAMMEERDE BRONCODE

Wie een huis koopt met een lekkend dak zal de kosten voor reparatie van dit gebrek aftrekken van de prijs. Hetzelfde principe kan worden toegepast op software. Als software kwaliteitsgebreken vertoont, zoals hoge unitcomplexiteit of veelvuldige duplicatie van codefragmenten, dan moet de waarde van de software verminderd worden met de kosten die gemoeid zijn met het verhelpen van deze gebreken. Het tweede model

voor waardebepaling wordt daarmee: de vervangingswaarde van software, gecorrigeerd voor het percentage aan kwalitatief slecht geprogrammeerde regels code (te complex, redundant, et cetera): Softwarewaarde = Vervangingswaarde * (1 – Reparatiefractie). Wanneer we hierin de getallen voor Vervangingswaarde en Reparatiefractie uit het middenniveau van de softwarewaarderingspiramide invullen, krijgen we:
Softwarewaarde (model 2) =
€ 3.114.000 * (1 – 39 procent) =
€ 1.899.540.

De waardebepaling volgens model 2 geeft dus een softwarewaarde van bijna 1,9 miljoen euro, berekend als de vervangingswaarde van de software verminderd met het percentage slecht geprogrammeerde regels code.

MODEL 3: CORRECTIE VOOR VERHOOGDE ONDERHOUDSLAST

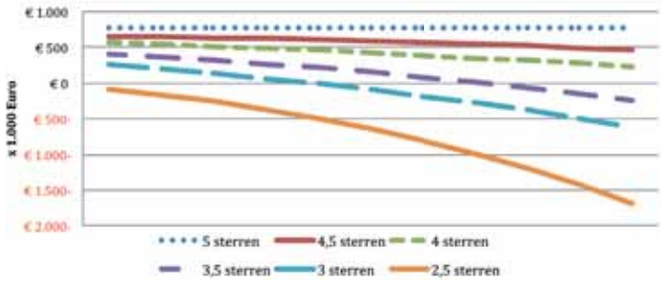
In model 1 en 2 brengen we de reparatiekosten in de vorm van een bedrag of een percentage in mindering op de vervangingswaarde. Deze twee benaderingen gaan ervan uit dat de kwaliteitsproblemen verholpen moeten worden. Maar men kan er ook voor kiezen om de kwaliteitsproblemen te laten voor wat ze zijn en de bijbehorende extra onderhoudslasten op de koop toe te nemen. In onze toelichting op het begrip 'technische schuld' hebben we laten zien dat wanneer we de kwaliteitsgebreken niet verhelpen, de kosten voor het onderhouden en verder ontwikkelen van de software hoger zullen zijn dan wanneer we dat wel doen. Ook deze verhoging van de onderhoudslast kan in financiële termen worden uitgedrukt. Hierbij zijn aannames nodig over de verwachte levensduur van de software en over de natuurlijke groei in termen van de hoeveelheid programmacode die per jaar moet worden aangepast. Bij hoge kwaliteit van de software zal de aanpassing van deze hoeveelheid code minder inspanning kosten dan bij lage kwaliteit. In de praktijk blijkt dat het productiviteitsverschil tussen lage kwaliteit (2 sterren op de schaal van 1 tot 5) en hoge kwaliteit (4 sterren) ongeveer een factor 2 bedraagt. Vermenigvuldiging van levensduur, te onderhouden codevolume en productiviteitsverschil, levert een indicatie op van de verhoogde onderhoudslast. Het derde waarderingsmodel wordt hiermee: de vervangingswaarde van de software, verminderd met de kosten van de verhoogde onderhoudslast. Dit model gaat dus niet uit van daadwerkelijke reparatie van kwaliteitsgebreken, maar van terugkerende kosten als gevolg van blijvende kwaliteitsgebreken: Softwarewaarde = Vervangingswaarde – Totale extra onderhoudslast over levensduur. Wanneer we hierin de getallen voor Vervangingswaarde en Totale extra onderhoudslast over levensduur uit het middenniveau van de piramide invullen, krijgen we:
Softwarewaarde (model 3) =
€ 3.114.000 - € 2.200.000 = € 914.000.

De waardebepaling volgens model 3 geeft dus een softwarewaarde van bijna 1 miljoen euro, berekend als de vervangingswaarde verminderd met de verwachte extra onderhoudskosten doordat de software lagere kwaliteit heeft dan het optimale niveau.

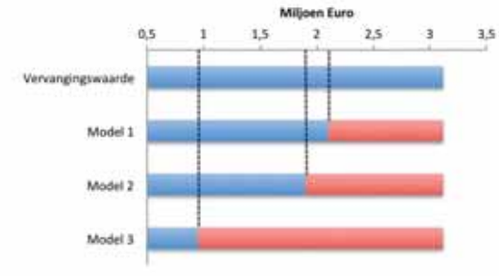
Het is ook mogelijk om met de formule van Model 3 de technische levensduur van maatwerksoftware te bepalen op basis van de vastgestelde kwaliteit van de broncode. We illustreren dit aan de hand van figuur 3, waarin te zien is hoe de financiële waarde van een applicatie afhangt van het kwaliteitsniveau ervan bij oplevering. De lijnen geven voor een aantal kwaliteitsniveaus weer hoe de financiële waarde van de software verandert in de tijd. De bovenste lijn geldt in het geval dat de applicatie een kwaliteitsniveau van 5 sterren bezit (en behoudt). Bij dit hoogste kwaliteitsniveau is er geen sprake van technische schuld, zodat de waarde van de applicatie constant blijft. Als het kwaliteitsniveau lager is dan 5, is er wel een technische schuld en zal de waarde van de applicatie met de jaren (steeds sneller) dalen. Vroeg of laat, afhankelijk van de kwaliteit, passeren alle curves die corresponderen met een kwaliteitsniveau lager dan 5 de nullijn en wordt de waarde van de applicatie negatief. De technische levensduur is dan bereikt. De onderste lijn geeft als startpunt al een negatieve waarde voor de applicatie. Gelet op de technische kwaliteit bij oplevering zou een organisatie in dit geval de applicatie niet moeten aanvaarden.

ONDERLINGE VERGELIJKING VAN DE MODELLEN

Het derde en hoogste niveau van de piramide biedt dus verschillende waarderingsmodellen. Elk van deze modellen belicht een ander perspectief van softwarewaardering, waardoor ze elk leiden tot een ander bedrag voor de financiële waarde van een softwareproduct (zie figuur 4).



Figuur 3: Softwarewaarde en resterende technische levensduur (zie tekstkader)



Figuur 4: Waarde van het software product in miljoenen Euro, berekend via elk van de drie modellen

Criteria	Model 1	Model 2	Model 3
Begrijpelijkheid model	+	++	-
Benodigde expertopinie	+	++	0
Hoeveelheid benodigde data	0	++	0
Precisie	++	+	++
Bruikbaarheid in business case audit	+	0	++

Tabel 3: De drie modellen gescoord op vijf bruikbaarheidscriteria

In combinatie met elkaar en door resultaten van de modellen onderling te vergelijken, ontstaat een goed inzicht in de waarde van maatwerksoftware. Welk model het meest geschikt is hangt af van de situatie.
Model 1 is vooral geschikt in acquisitiescenario's, waar een transactieprijs bepaald moet worden. Dit model bevat als subjectief element de expertopinie bij het bepalen van de refactoring-aanpassing.
Model 2 is eenvoudiger en maakt geen gebruik van expertopinie. Hierdoor is het model objectiever dan Model 1, maar is het ook minder verfijnd. Dit model is vooral geschikt voor een snelle waardebepaling waar minder precisie vereist is.
Model 3 is geschikt bij het nemen van investeringsbeslissingen, doordat de lange termijnonderhoudslast ook in ogenschouw genomen wordt. Ook dit model bevat als subjectief element een expertopinie, namelijk de inschatting van de levensduur van de software.

De geschiktheid van de modellen is geëvalueerd in interviews met acht experts, die executive posities bekleden (CIO, CFO). De resultaten zijn weergegeven in Tabel 3, waarin de bruikbaarheid van drie modellen is aangegeven op een schaal met vijf waarden van slecht (--) tot goed (++). We hebben de modellen gescoord op de volgende vijf criteria voor bruikbaarheid:

- Begrijpelijkheid van het model: is het model eenvoudig uit te leggen aan verschillende stakeholders?
- Benodigde expertopinie: bevat de berekening volgens het model een element waarvan de waarde moet worden ingeschat door een expert?
- Hoeveelheid benodigde data: hoeveel variabelen bevat het model?
- Precisie: hoe precies kan de softwarewaarde worden vastgesteld?
- Bruikbaarheid in business case audit: is het model bruikbaar om het realiteitsgehalte van business cases te toetsen?

NUT VAN HET GEBRUIK VAN DE SOFTWAREWAARDERINGS-PIRAMIDE

Met de voorgestelde piramide voor waardering van maatwerksoftware kunnen beleidsmakers hun voordeel doen:

- Bij het samengaan van organisaties vormen de eigenschappen van de softwareproducten die bij die organisaties in gebruik zijn een belangrijke factor voor snelle en succesvolle integratie. Softwarewaardering op basis van vervangingswaarde en reparatiekosten die bepaald worden door metingen aan de software maakt een objectieve weging mogelijk tussen verschillende integratiescenario's.
- Bij het rationaliseren van een landschap van softwareproducten kan waardering van software leidend zijn bij het stellen van prioriteiten en het maken van investeringsbeslissingen. In plaats van te kijken naar gemaakte ontwikkelkosten kunnen softwareproducten immers geselecteerd worden op basis van benodigde reparatiekosten en toekomstige onderhoudslast.
- De voorgestelde softwarewaarderingspiramide leent zich uitstekend voor het vergelijken van softwareportfolio's van verschillende organisaties. Dit maakt het mogelijk voor executives om hun softwareportfolio te laten benchmarken op intrinsieke waarde in plaats van op historische kosten.
- De piramide is een belangrijk instrument voor waardebepaling tijdens due diligence. De aankopende partij kan snel en op basis van betrekkelijk beperkte informatie een goede inschatting maken van de waarde van de software assets.

CONCLUSIES

De beschreven aanpak op basis van de softwarekwaliteitspiramide brengt organisaties een stap dichter bij een objectieve waardebepaling van software door uit te gaan van reparatiekosten, onderhoudslast en vervangingswaarde. De resultaten van deze aanpak kunnen executives gebruiken om investeringsbeslissingen en financiële rapportages over software portfolio's te onderbouwen met relevante kwantitatieve gegevens.

IT-auditors kunnen de voorgestelde modellen gebruiken wanneer zij voor de taak staan om het realiteitsgehalte van business cases voor IT-investeringen te bepalen.

De voorgestelde modellen nemen alleen de kwaliteit en het volume van software in beschouwing. Uit de validatie-interviews, uitgevoerd tijdens het master scriptieonderzoek waar dit artikel op gebaseerd is, blijkt dat het wenselijk is om, naast onderhoudbaarheid, ook andere aspecten in ogenschouw te nemen zoals functionaliteit, robuustheid en schaalbaarheid. Dit zijn onderwerpen voor verder onderzoek. ■

Noten

1. Ditzelfde kwaliteitsmodel wordt gehanteerd bij de software-productcertificering door het bedrijfsonderdeel Informationstechnik van TÜV, de Duitse tegenhanger van TNO.
2. De benchmarkdatabase houdt onder meer gegevens bij over kwaliteitsverbeteringen die in het verleden zijn aangebracht op softwareproducten in de database en het aantal wijzigingen in de broncode dat

daarvoor nodig was. Hierdoor valt voor elke kwaliteitsprong de gemiddeld benodigde reparatiefractie vast te stellen op basis van ervaringsgegevens.

3. In het geval van identieke code die op verschillende plekken in de broncode voorkomt bijvoorbeeld (we spreken dan van codeduplicatie), moet bij een wijziging van de gedupliceerde code de wijziging overal worden aangebracht waar deze code voorkomt.

Literatuur

[BAGG10] Baggen, R. (TÜViT), K. Schill, (TÜViT) en J. Visser (SIG). 'Standardized Code Quality Benchmarking for Improving Software Maintainability', in *Proceedings of the 4th International Workshop on Software Quality and Maintainability* (SQM 2010, http://sqm2010.sig.eu/), March 15, 2010, Madrid, Spain, 2010 (http://www.sig.eu/blobs/Research/Scientific%20publication/SQM2010-StandardizedCodeQuality.pdf).

[CUNN93] Cunningham, W. (1993). The WyCash portfolio management system. *ACM SIGPLAN OOPS Messenger*, 4 (2), 47-52.

[GR0011] Het onderzoek wordt volledig beschreven in '*Incorporating Software Quality in the Capitalization of Software as an Asset*', M.Sc. Thesis, Augustus 2011, Leiden.

[JONE11] Jones, C. (2011). *SPR Programming Languages Table™*. Verkrijgbaar via http://www.spr.com/.

[NUGR10] Nugroho, A., J. Visser, en T. Kuipers. (2010). An Emperical Model of Technical Debt and Interest. *Proceedings of the Second International Workshop on Managing Technical Debt.*

[TOCK97] Tockey, S. (1997, 12). A Missing Link in Software Engineering. IEEE Software, 31-36.

J. (Jelle) de Groot is managementconsultant bij de Software Improvement Group en studeerde aan de Universiteit van Leiden waar hij de masteropleiding 'ICT in Business' succesvol afrondde. Zijn afstudeerscriptie vormde de basis voor dit artikel.

J. (Joost) Visser is Hoofd Research bij de Software Improvement Group waar hij het onderzoek leidt naar methoden voor het bepalen van softwarekwaliteit en de impact daarvan op bedrijfsvoering.

Bijlage: Overzicht van meetgegevens en afgeleide gegevens voor de casus-software

Eigenschap	Waarde	
Niveau: Softwareontwikkeling		
Java	173.000	regels code
JSP	49.000	regels code
JavaScript	18.000	regels code
Herbouwinspanning	373,5	manmaanden
SIG kwaliteit	2,9	Sterren (★★★☆☆)
Niveau: Applicatie-portfoliomanagement		
Optimale kwaliteitsniveau	4,0	Sterren (★★★★☆)
Reparatiefractie	39%	
Vervangingswaarde	€ 3.114.000	
Refactoring-aanpassing	20%	
Reparatielast	€ 970.000	
Jaarlijkse onderhoudslast bij optimale kwaliteit	€ 330.000	
Extra onderhoudslast in jaar 1 door lagere kwaliteit	€ 150.000	
Niveau: Enterprisemanagement		
Levensduur software	7	jaar
Totale onderhoudskosten over levensduur	€ 2.200.000	
Financiële waarde		
Volgens model 1	€ 2.144.000	
Volgens model 2	€ 1.899.540	
Volgens model 3	€ 914.000	