Kwaliteit van software: niet langer een black box door gebruik van softwareanalyse

Dr. T. Kuipers en drs. C.M. Piek RE

Veel problemen bij de bouw, invoering en migratie van informatiesystemen kunnen worden voorkomen door de systemen voorafgaand aan de implementatie door te lichten. Door gebruik te maken van tooling is het mogelijk zelfs van heel grote systemen in weinig tijd een 'röntgenfoto' te maken. Deze foto laat mogelijke zwakke plekken en andere risico's in het systeem zien voordat het systeem operationeel is. Door dit zo vroeg mogelijk in het proces te doen kan een systeemontwikkelingsproject worden bijgestuurd voordat zich problemen voordoen. Ondanks de minder efficiënte wijze van het in een later stadium verbeteren van systemen, kan deze aanpak ook worden toegepast op legacy-systemen.

Inleiding

Het gebruik van informatiesystemen kan volgens de wervende artikelen en advertenties binnen een organisatie leiden tot substantiële voordelen: de arbeidsproductiviteit gaat omhoog, klanten worden beter en sneller geholpen, en de software kan verbanden onderkennen tussen bepaalde klantgroepen en productgroepen waar marketeers onmiddellijk op in kunnen springen. Kortom, het gebruik van op maat gemaakte, slimme en efficiënte software kan uw bedrijf een flinke voorsprong op de concurrentie geven.

Iedereen die iets te maken heeft met de IT in zijn organisatie, zal nu bedenken dat bovenstaande uitspraak de zoveelste herhaling van hetzelfde sprookje is, en dat in de praktijk informatiesystemen in veel gevallen tot ellende leiden. Dit blijkt ook uit diverse statistieken.

Het grote probleem bij de invoering en het onderhoud van informatiesystemen is de complexiteit van de onderliggende technologie. Het is onmogelijk een omvangrijk ontwikkelproject geheel te overzien en alle details in de gaten te houden, daarvoor ontbreekt doorgaans de technische kennis bij het management en zijn er simpelweg te veel details.

Door gebruik te maken van een geavanceerd geautomatiseerd systeem dat grote informatiesystemen integraal kan doormeten en door deze meetgegevens vervolgens te laten analyseren door software engineering experts kan op efficiënte wijze een objectief inzicht in de informatiesystemen worden verkregen. Dit artikel behandelt de Software Risk Assessment in meer detail.

Dr. T. Kuipers
is technisch directeur van de Software Improvement Group. De Software Improvement Group is een spin-off van het Centrum voor Wiskunde en Informatica (CWI). Hij promoveerde aan de Universiteit van Amsterdam op het terrein van de software engineering en meer specifiek de renovatie van oude informatiesystemen. Hij adviseert en publiceert over de renovatie en migratie van informatiesystemen. Hij doceert over software engineering en het management daarvan aan de Universiteit van Amsterdam en geeft regelmatig gastcolleges.

tobias.kuipers@software-improvers.com

Drs. C.M. Piek RE
is manager bij KPMG Information Risk Management in De Meern. Hij heeft zich gespecialiseerd in de kwaliteitsverbetering van ICT-toepassing en ICT-organisaties en begeleidt ICT-infrastructuur- en -uitbestedingstrajecten bij een aantal organisaties.

piek.christiaan@kpmg.nl

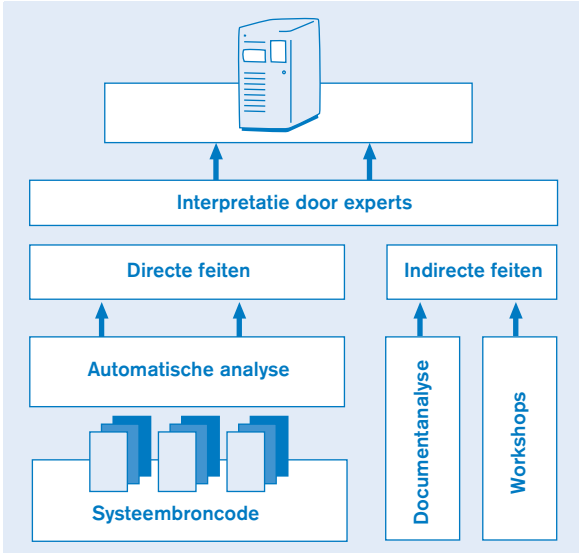
Software Risk Assessment

Een Software Risk Assessment is een onafhankelijke analyse van de risico's bij het bouwen, installeren, testen, beheren en onderhouden van een informatiesysteem. Een assessment wordt uitgevoerd om een specifieke reden op een specifiek systeem, en dient in een beperkte doorlooptijd (doorgaans binnen drie weken) te worden uitgevoerd.

Informatiesystemen bevatten risico's op het terrein van onderhoudbaarheid, performance, operationele kosten, schaalbaarheid, etc. Organisaties vragen zich af of het informatiesysteem dat nu met twintig gebruikers werkt op te schalen valt naar tweeduizend gebruikers, en of het systeem dat nu functioneert op een hardwaresysteem van één miljoen euro ook kan functioneren op een goedkopere computer, en in hoeverre het informatiesysteem over vijf jaar nog te onderhouden is door dan beschikbare programmeurs.

Dit is precies het soort vragen waar een Software Risk Assessment antwoord op geeft. Dat gebeurt door de broncode bij de risicoanalyses te betrekken. Doorgaans gaat het hierbij om honderdduizenden zo niet miljoenen regels Cobol, Java, C en diverse andere programmeertalen. Deze broncode wordt beschouwd als de directe bron voor de analyses. Informatie die direct uit de broncode kan worden afgeleid (zoals afhankelijkheden tussen componenten, de scheidingslijn tussen klantconfiguraties en pakketmodules, organisatie van de gegevens in het datamodel, ...), geeft het meest betrouwbare beeld over wat er daadwerkelijk aan de hand is met het onderliggende informatiesysteem.

Daarnaast worden allerhande nuttige indirecte bronnen gebruikt. Denk daarbij aan documentatie over de architectuur en over het gebruikte ontwikkelingsproces en aan bij betrokken medewerkers aanwezige kennis. De informatie uit deze indirecte bronnen heeft als voordeel het hogere abstractieniveau. Een groot probleem is echter de betrouwbaarheid: ontwerpdocumentatie is vaak verouderd en kennis van medewerkers is subjectief en vaak gericht op specifieke aspecten van het systeem. Een accuraat totaaloverzicht ontbreekt. Een Software Risk Assessment bestaat derhalve niet alleen uit het naar boven halen van informatie uit de directe en indirecte bronnen, maar juist ook uit het relateren van de subjectieve, mogelijk incorrecte indirecte informatie aan de objectieve werkelijkheid van de broncode.



Figuur 1. De stappen in het assessmentproces.

Beide vormen van informatie dienen ten slotte geïnterpreteerd en vertaald te worden naar risico's die samenhangen met het oorspronkelijke probleem. Deze stappen worden geïllustreerd in figuur 1.

Redenen voor Software Risk Assessments in de praktijk

Tot de voorbeelden van assessments zoals die in de praktijk zijn uitgevoerd, behoren onder meer (de bedrijfsnamen en probleemomschrijvingen zijn geanonimiseerd):

- Een bedrijf schaft een standaardpakket aan. Dit pakket voldoet niet aan specifieke extra eisen en wensen van het bedrijf. De leverancier van het pakket wordt gevraagd het pakket aan te passen aan deze extra vereisten. Na levering van het gewijzigde pakket ondervindt het bedrijf grote problemen bij de invoering, en vraagt zich af wat de risico's van het 'live gaan' met dit systeem inhouden, in ogenschouw nemende dat gegevens van miljoenen klanten door het pakket dienen te worden verwerkt.
- Een bedrijf koopt een standaard e-businesspakket voor de implementatie van een virtuele marktplaats in de energiehandel. Het onderhoud van dit pakket zal worden uitgevoerd door het bedrijf: de leverancier bouwt versie 1.0 en draagt daarna de broncode over. Het bedrijf wil een onderzoek van het pakket om inzicht te krijgen in de onderhoudsrisico's.
- Een bedrijf beschikt over een groot callcenter. De programmatuur die wordt gebruikt om de telefoontjes naar het centrum te administreren is al vijftien jaar uitbesteed. Om kosten te besparen wil het bedrijf het uitbestedingscontract beëindigen en zelf het onderhoud ter hand nemen. Hiertoe wordt een assessment uitgevoerd van de risico's die het dagelijks functioneren van het callcenter loopt bij deze inbestedingsoperatie.
- Een overheidsdienst heeft tien jaar geleden het beheer en onderhoud op een groot administratief informatiesysteem uitbesteed. De overheid heeft het gevoel dat de totaalkosten die zij doorberekend krijgt voor deze uitbesteding, te hoog zijn. Een benchmark wordt uitgevoerd die de kosten van het onderhoud op dit systeem relateert aan de kosten van onderhoud op vergelijkbare systemen. Als onderdeel van deze benchmark wordt een assessment uitgevoerd om risico's op het gebied van de broncode te koppelen aan de onderhoudskosten.

Softwareanalyse

De informatiesystemen worden geanalyseerd op de volgende acht aspecten:

- *Technische architectuur*. De analyse van de technische architectuur richt zich op het hoog-niveau technisch ontwerp van het systeem. De technische architectuur heeft een langetermijneffect op de levenscyclus van het systeem. Correcte architecturen verlengen het leven van een informatiesysteem doordat ze een solide en aanpasbare basis vormen.
- *Modulariteit*. De modulariteitsanalyse richt zich op de laag-niveau implementatie van het systeem. Een goed modulair ontwerp verhoogt de onderhoudbaarheid en maakt het mogelijk het systeem eenvoudig uit te breiden en aan te passen aan toekomstige eisen en wensen.
- *Technische complexiteit*. De complexiteitsanalyse richt zich op de manier waarop de code kan worden begrepen. Complexe systemen zijn moeilijker (en dus duurder) aan te passen. Het reduceren van complexiteit maakt het proces van aanpassen van code (zowel bij fouten als bij wijzigingen) eenvoudiger, sneller en minder foutgevoelig.
- *Omvang*. De grootte van het systeem geeft een kwantitatief inzicht. Grote systemen zijn relatief duurder in het onderhoud dan kleine systemen. De technische omvang van het systeem moet in overeenstemming zijn met de omvang van de geleverde functionaliteit.
- *Testen*. De testanalyse richt zich op de teststrategie die gebruikt wordt. Een solide testopzet vermindert het aantal fouten, en derhalve de kosten die gemaakt moeten worden. Herhaalbare en automatiseerbare tests zorgen ervoor dat verbeteringen aan de code meetbaar worden gemaakt.
- *Performance*. De performanceanalyse richt zich op de tijds- en resource-efficiëntie van het systeem. Performanceproblemen kunnen een drastisch effect hebben op de bruikbaarheid en de kosten van een systeem.
- *Documentatie*. Actuele en volledige documentatie verlengt de levensduur en verhoogt de overdraagbaarheid van een systeem. Documentatie die niet actueel is werkt contraproductief. Volledige documentatie is echter een moeilijk te realiseren ideaal en veelal inefficiënt.
- *Beheer en onderhoud*. Dit aspect omvat de manier waarop het systeem beheerd en onderhouden wordt of kan worden. Goed beheer en onderhoud zijn essentieel voor het dagelijks gebruik van een systeem.

Indirecte informatiebronnen

Elke Software Risk Assessment begint met de organisatie van een workshop waaraan de diverse belanghebbenden van het informatiesysteem deelnemen. Wie er voor een dergelijke workshop worden uitgenodigd, is weergegeven in kader 1.

De workshop heeft een aantal doelen. Ten eerste maakt zij iedereen binnen een project bewust van het feit dat er een assessment wordt uitgevoerd. De workshop wordt gebruikt om het doel van de assessment en de gebruikte assessmentmethode uit te leggen. Afhankelijk van het doel kan het zinvol zijn twee workshops te organiseren. Een voorbeeld betreft een softwareontwikkelingsteam voor bedrijf A en een implementatieteam voor bedrijf B, in een situatie waarin A en B niet even gelukkig zijn over hun relatie. Een belangrijk doel van de workshop is het wegnemen van angst en onzekerheid. Tijdens de workshop wordt expliciet uitgelegd dat de assessment beoogt risico's te identificeren en tot aanbevelingen te komen over hoe het systeem verbeterd kan worden, en dat het vinden van zondebokken niet tot de doelstellingen behoort en derhalve geen onderdeel uitmaakt van de methode of de resultaten.

Technische analyse

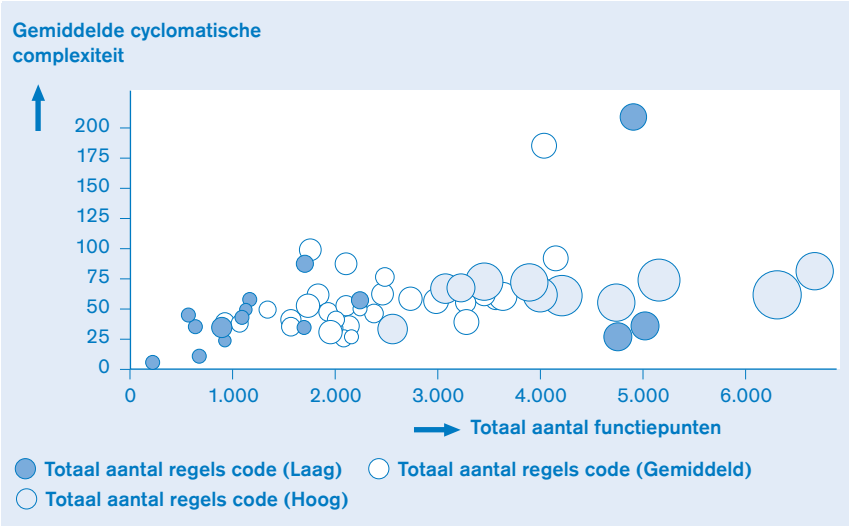
Voordat de workshop wordt georganiseerd, wordt eerst een analyse op de broncode uitgevoerd. Dit geeft een globale indicatie van de omvang en de architectuur van het systeem. Een belangrijke stap is het identificeren van uitzonderingsgevallen: modules of subsystemen die (sterk) afwijken van het gemiddelde. Dit kan gebeuren

Het uitnodigen van deelnemers aan de workshop kan een behoorlijk politiek getint proces zijn. Wie er worden uitgenodigd, hangt sterk af van wie de Software Risk Assessment initieert en waarom. Er zijn hoe dan ook belanghebbenden die altijd uitgenodigd zouden moeten worden. Afhankelijk van de politieke situatie worden deze mensen wel of niet uitgenodigd, en als ze worden uitgenodigd kan het gebeuren dat ze geen toestemming krijgen om mee te doen. In sommige gevallen is het door dit soort omstandigheden effectiever (en efficiënter) om twee workshops te organiseren.

Mensen die over het algemeen uitgenodigd dienen te worden, zijn:

- *Projectmanager:* degene die verantwoordelijk is voor de dagelijkse gang van zaken in het project en die uiteindelijk verantwoordelijk is voor het succes van het project.
- *Klant:* degene die het project financiert of de formele opdrachtgever is. De opdrachtgever kan overigens ook een stuurgroep zijn. Als deze persoon of groep geen of weinig directe bemoeienis met het project heeft, dan degene uit de klantorganisatie die het meest betrokken is bij het project. In sommige gevallen is het zinvol ze beiden uit te nodigen.
- *Architect:* de systeemarchitect(en), voorzover aanwezig. Dit is de persoon die het grootste aandeel heeft in het ontwerp van het systeem.
- *Hoofdontwikkelaar:* degene die de dagelijkse leiding heeft over de programmeeractiviteit, en bij voorkeur zelf mee programmeert. Iemand uit het middenmanagement die geen werkelijke kennis van het ontwikkelproces heeft, is hier onbruikbaar.
- *Hoofdtester:* degene die de testactiviteiten leidt.
- *IT-auditor:* een in- of externe auditor die het formele kwaliteitsonderzoek uitvoert, in een Quality Assurance-rol functioneert of het management adviseert over het systeem.
- *Diverse specialisten*, zoals de databasespecialist, de performance (tuning) specialist, de implementatiespecialist en de netwerkspecialist. Wie daadwerkelijk worden uitgenodigd, hangt af van de aard van de Software Risk Assessment.

Kader 1. Deelnemers aan de workshop.



Figuur 2. Functiepunten versus complexiteit per module en regels code.

door middel van volumemetrieken (zoals het aantal regels code), maar ook door het aantal databasebenaderingen, de cyclomatische complexiteit, het aantal 'sockets' dat wordt geopend voor lezen, enz.

Gebaseerd op ervaring en op internationale literatuur is een vragenlijst opgesteld die een groot aantal onderwerpen op het terrein van informatiesysteemontwerp, -bouw, -implementatie, -tests en dergelijke behandelt. Deze vragenlijst wordt gebruikt om de workshop te structureren. Afhankelijk van het zwaartepunt van de Software Risk Assessment worden specifieke vragen al dan niet aan de deelnemers van de workshop gesteld. Na elke assessment wordt de vragenlijst geëvalueerd en zo mogelijk verbeterd.

Tijdens de workshop (en überhaupt in deze fase van de Software Risk Assessment) worden de problemen zoals de deelnemers die zien, in kaart gebracht. In de hiernavolgende fase worden deze problemen vergeleken met de werkelijke toestand van de broncode.

Software Analysis Toolkit

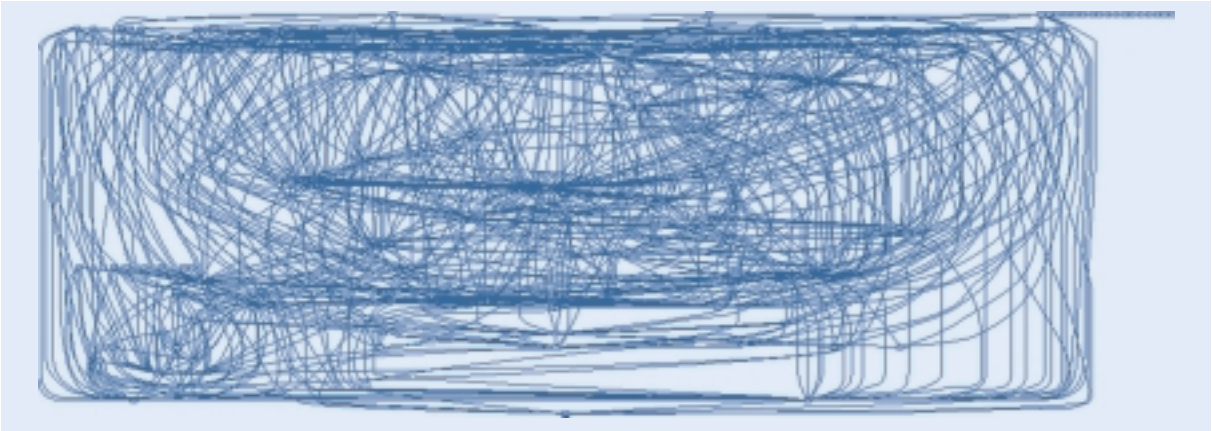
Voor het analyseren van (zeer) grote hoeveelheden broncode in verschillende programmeertalen is een raamwerk ontwikkeld. Dit raamwerk heet de Software Analysis Toolkit (SAT). Hiermee worden bijvoorbeeld analyses uitgevoerd zoals in figuur 2: van een aantal systemen, in totaal zo'n zestig miljoen regels Cobol-code, wordt de omvang zowel in functiepunten als in regels code vergeleken met de technische complexiteit.

Uit deze analyse blijkt onmiddellijk dat er twee systemen zijn die significant complexer zijn dan de andere, en verder blijkt dat de relatie tussen hoeveelheid functiepunten (hoe verder naar rechts op de x-as, hoe meer functiepunten) niet één-op-één relateert aan de omvang in regels code (aangegeven door het verschil in raster). Rond de vijfduizend functiepunten blijken systemen te zitten die relatief klein zijn in aantal regels code, en systemen die relatief groot zijn. Een mogelijk risico is hier de onderhoudbaarheid: als twee systemen een even grote functiepuntenwaarde hebben, dan is het systeem dat uit meer regels code bestaat waarschijnlijk duurder in het onderhoud.

De analysetoolkit maakt het mogelijk 'in te zoomen' op de systemen met een verhoogd risico. Zo kunnen bijvoorbeeld op de twee meest complexe systemen gedetailleerde analyses worden uitgevoerd die laten zien waardoor die complexiteit precies ontstaat.

Een andere 'view' is mogelijk door de afhankelijkheden tussen de systemen in kaart te brengen. Dit kan letterlijk (er wordt een tekening gemaakt van de afhankelijkheden) zoals in figuur 3. Vaak is het echter nuttiger om niet alle afhankelijkheden in één keer te tonen, omdat het er te veel zijn. Afhankelijk van de zwaartepunten van de Software Risk Assessment worden specifieke afhankelijkheden geïdentificeerd. De analysesoftware is in te stellen zodat specifieke afhankelijkheden zichtbaar worden gemaakt.

Het centrale gedeelte van het raamwerk is een database waarin constructies uit diverse programmeersystemen kunnen worden opgeslagen. Er zijn meerdere overlappende modellen, voor bijvoorbeeld procedurele talen (zoals Cobol), vierdegeneratietalen (zoals Uniface, Powerbuilder en vele andere) en objectgeoriënteerde talen (zoals Java en C++).



Figuur 3. Een deel van de afhankelijkheden tussen de systemen.

Op basis van de gegevens in de database kunnen metrieken worden uitgerekend die relevant zijn voor de huidige Software Risk Assessment. Deze metrieken kunnen met grafieken worden weergegeven, om ze beter inzichtelijk te maken of om trends in de metrieken te identificeren. Zoals hierboven al geïllustreerd kunnen zo ook de uitzonderingsgevallen worden geïdentificeerd: welke module leest het meest uit een database, welke module schrijft het meest naar bestanden, welke module roept de meeste andere modules aan. Hiermee kan rekening worden gehouden bij correctief en adaptief onderhoud van het informatiesysteem.

Benchmarking

De resultaten van de diverse Software Risk Assessments worden bewaard. Dat geeft de mogelijkheid om analyses uit te voeren met behulp van benchmarkinggegevens. Zo kunnen de resultaten van een systeem worden vergeleken met de resultaten van een functioneel equivalent systeem dat door een andere organisatie is gebouwd. Op dit moment bevat deze benchmarkingdatabase gegevens die zijn afgeleid van ongeveer 45 gigabyte aan broncode, van zowel commerciële als overheidsorganisaties.

De volgende metrieken worden standaard berekend:
- het aantal modules in een systeem;
- het aantal regels code, het aantal programmeertalen;
- de cyclomatische complexiteit van McCabe (zie [McCa89]);
- de onderhoudbaarheidsindex van Oman en Hagemeister (zie [Oman94]);
- het geschatte aantal functiepunten met de backfiringmethode;
- de fan-in en fan-out van de modules;
- het aantal databasetabellen dat door een module wordt gebruikt;
- het percentage gedupliceerde (of gekloonde) code;
- het percentage ongebruikte code;
- het aantal parameters per module/procedure;
- het aantal velden per databasetabel;
- het aantal 'goto-statements'.

Vertalen van cijfers naar risico's

Het resultaat van de Software Risk Assessment is een omschrijving en classificatie van de risico's zoals die zijn geïdentificeerd in de workshop, gebaseerd op de directe feiten zoals die uit de broncode zijn afgeleid. Om deze analyse bruikbaar te laten zijn voor de opdrachtgever en andere belanghebbenden bevat het rapport een interpretatie van de waarnemingen in de broncode. Deze interpretatie maakt duidelijk waarom de waarnemingen relevant zijn in relatie tot de geïdentificeerde risico's en hoe deze waarnemingen gebruikt kunnen worden om oplossingen te vinden voor de geïdentificeerde problemen.

De Software Risk Assessment-rapportage bestaat uit de volgende onderdelen:
- een objectieve beschrijving van de problemen zoals ze gezien worden door de belanghebbenden, hetgeen in de workshop naar voren is gekomen;
- een objectieve beschrijving van de waarnemingen die gedaan zijn in of op basis van de broncode, gerelateerd aan de geïdentificeerde problemen;
- een objectieve inventarisatie van de potentiële risico's voor elk van de technische waarnemingen;
- een subjectieve evaluatie, gebaseerd op de ervaring van de betrokken experts, van de risico's, die leidt tot aanbevelingen om de risico's te verkleinen en de voordelen uit te buiten;
- een score voor elk van de acht genoemde aandachtsgebieden in het informatiesysteem. De score per gebied is er één van de trits slecht, onvoldoende, neutraal, voldoende, goed.

De op de broncode gebaseerde waarnemingen vormen de onderbouwing, het 'bewijs' van de bevindingen en conclusies in de rapportage.

De resultaten van deze assessment kunnen ook worden gebruikt voor reverse engineering van een bestaand, maar slecht gedocumenteerd informatiesysteem.

Conclusies

Dit artikel beschrijft een systematische methode om op basis van een broncodeanalyse een evaluatie van de risico's uit te voeren. Het management kan de uitkomsten benutten om beslissingen te nemen over de toekomst van bestaande informatiesystemen. Beslissingen over bijvoorbeeld vervanging, integratie met andere systemen of uitbesteding van systemen. De softwareanalyses worden gekarakteriseerd door:
- de betrokkenheid van alle belanghebbenden;
- volledige analyse van de broncode van het informatiesysteem;
- benchmarking met andere informatiesystemen;
- interpretatie van de beschikbare gegevens door experts.

Literatuur

[McCa89] Th. J. McCabe and C.W. Butler, *Design Complexity Measurement and Testing, Communications of the ACM 32*, nr. 123, December 1989.

[Oman94] P. Oman and J. Hagemeister, *Construction and testing of polynominals predicting in maintainability*, Journal of Systems and Software, volume 24, issue 3, maart 1994.

Bijlage: Functionele en technische kwaliteit van applicatieportfolio

De methode in dit artikel richt zich op enkele informatiesystemen waarin met een geautomatiseerde analyse inzicht valt te verkrijgen. Op het niveau van de applicatieportfolio is ook inzicht te verkrijgen in de functionele en technische kwaliteit met de volgende aanpak, zoals ontwikkeld door en in de praktijk toegepast bij KPMG.

Nadat een applicatie door een organisatie in gebruik is genomen, ondergaat zij continu veranderingen als gevolg van interne en externe factoren (bijvoorbeeld wijzigingen in wet- en regelgeving, een reorganisatie en de introductie van een nieuw IT-platform). Diverse onderzoeken wijzen telkenmale uit dat de kwaliteit van applicaties gedurende de 'productiefase' voortdurend afneemt. In feite is er sprake van een 'halfwaardetijd' van applicaties. Voor een organisatie kan het nuttig zijn om (periodiek) inzicht te hebben in de kwaliteit van de in gebruik zijnde applicaties. Een zogenaamde applicatieportfolio assessment kan dit inzicht op een pragmatische en eenvoudige manier bieden.

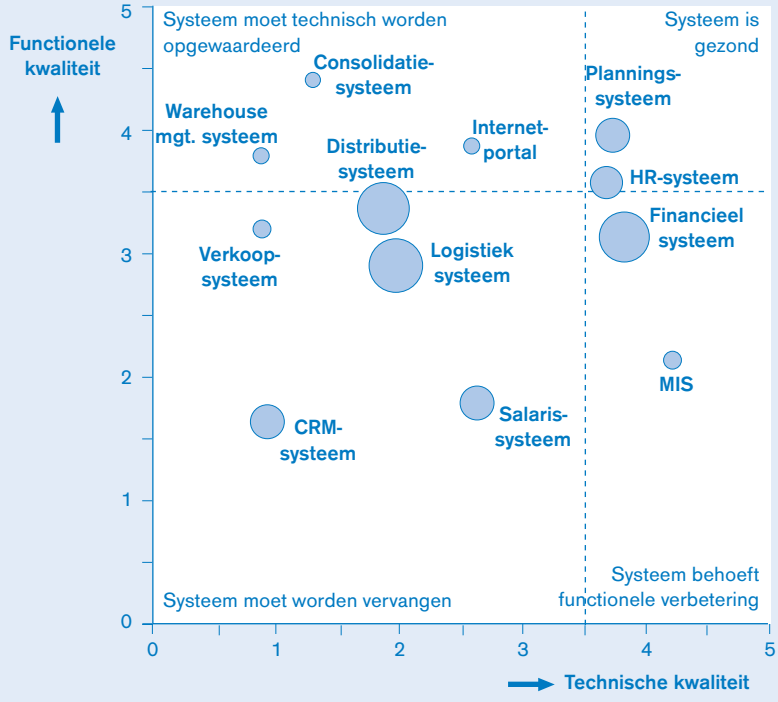
De primaire doelstelling van een applicatieportfolio assessment is het identificeren van aandachtspunten die betrekking hebben op zowel de functionele als technische kwaliteit van een applicatieportfolio (hieronder vallen zowel standaard- als maatwerksoftwarepakketten). Het onderzoeken van de functionele kwaliteit geeft de mate aan waarin de functionaliteiten van een applicatie de betreffende bedrijfsprocessen effectief ondersteunen; de technische kwaliteit geeft inzicht in de technische aspecten van de architectuur en programmatuur. Voor het uitvoeren van een dergelijke assessment is het nodig dat de organisatie (vooraf) inzicht heeft in bijvoorbeeld de ouderdom, de grootte, het platform en de kosten van de te onderzoeken applicaties.

De kwaliteit van de applicaties kan worden onderzocht door gebruik te maken van gestandaardiseerde vragenlijsten. Eén vragenlijst voor de functionele en één voor de technische kwaliteit. Kwaliteitsaspecten die in deze vragenlijsten aan bod komen zijn bijvoorbeeld bruikbaarheid, onderhoudbaarheid, portabiliteit en betrouwbaarheid[1]. De vragenlijsten worden door het onderzoeksteam gebruikt als richtlijn tijdens het houden van de interviews en voor het gestructureerd kunnen verzamelen van functionele en technische aspecten van de te analyseren applicaties. De functionele vragenlijst (per applicatie één) wordt ingevuld met het management en de gebruikers die bekend en vertrouwd zijn met de bedrijfsprocessen die door de betreffende applicatie(s) worden ondersteund. De technische vragenlijst wordt ingevuld met de ontwikkelaars en/of de betrokken applicatiebeheerders/databasebeheerders.

De verzamelde informatie uit de vragenlijsten wordt enerzijds gebruikt om de automatiseringsgraad van de processen vast te stellen. Door gebruik te maken van een zogenaamd businessmodel kan inzicht worden verkregen in de mate waarin de bedrijfsprocessen geautomatiseerd worden ondersteund en wat de kwaliteit van deze ondersteuning is.

De verkregen informatie kan worden ingevoerd in een geautomatiseerd tool waarmee een matrix kan worden gegenereerd (zie figuur 4). Deze matrix wordt als basis gebruikt om per applicatie te bepalen welke acties nodig zijn om de kwaliteit weer op het gewenste niveau te brengen.

1) Afgeleid uit de internationale ISO9126-standaard voor softwarekwaliteit.



Figuur 4. Inventarisatie van de kwaliteit van informatiesystemen.