

Legacy: lust en/of last?

De twee kanten van de legacymedaille

Oudere softwaresystemen zijn alom. Hoe groter en bedrijfskritischer een softwaresysteem, hoe langer het in gebruik blijft, soms wel tientallen jaren lang. Welke concrete maatregelen zijn nodig om de veroudering van software beheersbaar te houden?

Steven Klusener en Halbe Kuipers

Legacysystemen zijn softwaresystemen die als meervoudig verouderd worden beschouwd en zijn daarmee de zwarte schapen in organisaties. Ze missen ogenschijnlijk sexy features, zijn lastig aan te passen en zijn ontwikkeld in technologieën waarmee je je niet populair maakt op een verjaardagsfeestje. Ze horen echter ook bij de 'inboedel' die evolutionair gegroeid is, kunnen, mits van kwaliteit, inmiddels via tal van webinterfaces ontsloten worden, zijn stabiel en draaien enorme volumes tegen relatief lage kosten. Veroudering is ook bij software een gegeven. Hoe groter en bedrijfskritischer een softwaresysteem, hoe langer het in gebruik blijft, tot zelfs vele tientallen jaren lang. Deze systemen zijn veelal een erfenis van voorgangers van de huidige organisatie, bijvoorbeeld na een fusie of overname. Software is niet statisch, want bedrijfsfuncties, regels en voorschriften veranderen en die veranderingen moeten geabsorbeerd worden. Ook is de technische context continu aan verandering onderhevig. Dit vergt sturing, onderhoud en voortdurende specialistische arbeid. Prestaties uit het verleden zijn zonder het plegen van goed onderhoud geen garantie voor de toekomst. Welke concrete maatregelen zijn nodig om de veroudering van software beheersbaar te houden? Oudere systemen, vooral grote, zijn alom en de ervaringen ermee alsook de toekomstvastheid verschillen (zie de kaders).

Veroudering versus stabiliteit

Hoe kan software eigenlijk verouderen? Op het eerste gehoor klinkt het vreemd dat software verouderd: wanneer de broncode niet wordt gewijzigd, zal het systeem zich immers ook jaren later nog precies zo gedragen als op de eerste dag van oplevering. Dat klinkt als een auto die ook na vele malen de teller rond gereden te hebben nog precies zo functioneert als toen hij nieuw bij de dealer stond. Geen zuigerveren en rubbertje versleten, krukas, distributietandwielen en kopkapping zijn in functie, geen krasje erop, de ideale auto dus. Waarom spreken we dan toch van het verouderen van softwaresystemen?

Het antwoord is eenvoudig: de wereld waarin de softwaresystemen functioneren verandert, en daarmee veranderen de eisen die gesteld worden aan deze systemen. Dit fenomeen is al in 1974 geformuleerd door Lehman in zijn eerste wet van software-evolutie: 'E-type systems must be continually adapted or they become progressively less satisfactory'. De term 'E-type' staat hier voor evolutionaire systemen, in tegenstelling tot de S-type-systemen; dat zijn systemen die los van hun context met wiskundige nauwkeurigheid gedefinieerd kunnen worden en waarvan de functionaliteit voor de eeuwigheid vastligt. Een typisch voorbeeld is een sorteeralgoritme als Bubble-sort: over honderd jaar zullen de gewenste eigenschappen van het sorteeralgoritme nog

Samenvatting

Een legacysysteem is ouder of verouderd en daarom soms lastiger aan te passen aan veranderende situaties dan een 'modern' systeem, maar het heeft zich ook bewezen en is stabiel en relatief goedkoop. Men ziet vaak echter alleen de negatieve kant. Om de juiste keuze te maken tussen legacy dan wel nieuwbouw helpt het een softwarekwaliteitsonderzoek te doen en afspraken te maken over de softwarekwaliteit.

Legacy, nog geen tien jaar oud: **implosie?**

Een van de grote legacysystemen van een verzekeringsconcern (circa 20.000 functiepunten) functioneert problematisch. Het systeem is nog slechts tien jaar oud en werkende weg ontwikkeld. De functionele en technische documentatie is verre van compleet, het aantal wijzigingsverzoeken blijft jaar op jaar groot, ook omdat gewijzigd beleid, nieuwe producten en eisen van toezichhouders steeds verwerkt moeten worden. De gebruikte codegenerator is inmiddels een nicheproduct en wordt bovendien niet goed beheerst door de ontwikkelaars. De problemen nemen toe: veel wisselingen van programmeurs, slechte performance, 'verdwijnen' van gegevens, batchruns die uit de tijd lopen. Na een onafhankelijke, grondige analyse van de broncode blijkt de waarde van de erfenis: onherstelbaar complex en extreme codeduplicatie. Einde verhaal. Maar het systeem moet in exploitatie blijven tot er een vervangend systeem is. Een managementprobleem van de eerste orde.

Legacy met en zonder toekomst: **wie kent de feiten?**

Een groot internationaal opererend concern in de financiële sector heeft een geautomatiseerde informatiehuishouding van circa 800.000 functiepunten. Van de 1250 'systemen' zijn er 23 die qua grootte variëren van 10.000 tot 25.000 functiepunten. Van het aantal koppelingen bestaan slechts schattingen. Na enkele nieuwbouwcatastrofes werd het roer omgezet. De grote applicaties zijn onderworpen aan een softwarekwaliteitsonderzoek, tot op broncode-niveau. En wat bleek? Vier systemen hebben het predicaat 'legacy' in de vaak gebruikelijke negatieve connotatie. Deze vier systemen moeten worden vervangen door iets anders: nieuwbouw en/of pakketsoftware. Met de andere applicaties kan zeker meer dan tien jaar worden doorgewerkt, onder voorwaarde dat ze professioneel worden onderhouden. De eigen ontwikkelaars werden bevestigd in hun kwaliteiten. Het management moest erkennen dat de term 'legacy' niet in negatieve zin mag worden gebruikt zonder de feiten te kennen.

Erfenissen van meer dan dertig jaar oud: **waarde ongekend**

Het hart van de geautomatiseerde ondersteuning van een groot productiebedrijf (met fysieke producten) wordt gevormd door acht systemen, in grootte variërend van 8.000 tot 17.000 functiepunten. Zes van de acht systemen zijn (door)ontwikkeld vanaf eind jaren zeventig. De andere twee zijn eind jaren tachtig gebouwd. De systemen zijn alle dagen in productie, hebben veel satellieten en het aantal koppelingen van het hele systeem-complex is meer dan 500. Volgens de heersende mening van het bestuur en het topmanagement is hier sprake van oude meuk: opruimen die handel. Twee kostbare afgebroken nieuwbouwtrajecten verder wordt besloten tot gedetailleerd softwarekwaliteitsonderzoek van de bestaande systemen. De uitkomst: één systeemcomplex is gevaarlijk obsoleet, de andere krijgen mooie rapportcijfers en zijn goed onderhoudbaar op de lange termijn. Een erfenis van ongekende waarde dus.

implosie?

waarde ongekend

wie kent de feiten?



steeds zo zijn als vandaag of veertig jaar geleden toen Bubble-sort zijn intrede deed. Kortom, de wereld verandert continu en elk systeem moet daarin mee. Deze wijzigingen maken zo'n systeem echter weer complexer, waarmee we aankomen bij de tweede wet van Lehman: 'As an E-type system evolves its complexity increases unless work is done to maintain or reduce it'. In de loop van de tijd zal een wijziging dus steeds meer inspanning vergen, tot het punt dat de benodigde inspanningen niet meer in verhouding staan tot de gewenste wijziging. Dit fenomeen kan worden beperkt door de juiste preventieve maatregelen uit te voeren, zoals ook verwoord in de tweede wet van Lehman. Het lastige hierbij is om vooraf te bepalen welke preventieve maatregelen wel de moeite waard zijn en welke niet.

Veranderingen die gevolgen hebben voor software zijn aan de orde van de dag en kennen vele dimensies:

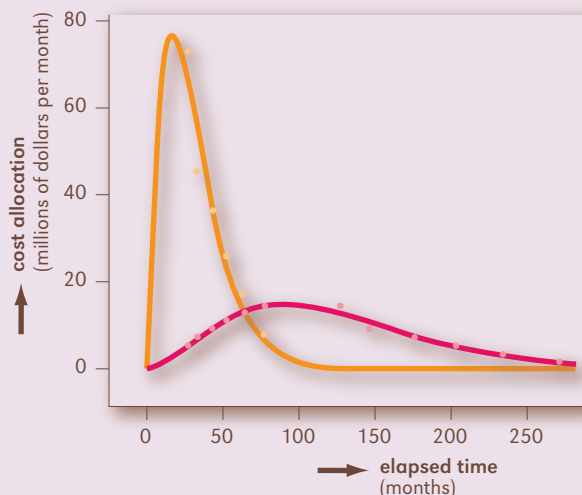
- Functionele eisen: de eisen en wensen van de gebruikersorganisatie vastgelegd in requirements en specificaties. Veranderingen zijn aan de orde van de dag, bijvoorbeeld als er een nieuwe feature aan een systeem voor internetbankieren moet worden toegevoegd omdat de concurrent ook al iets dergelijks biedt of omdat nieuwe wet- en regelgeving dit voorschrijft.
- Niet-functionele eisen: eisen die niet zichtbaar zijn voor de eindgebruikers. Ook deze eisen veranderen continu, bijvoorbeeld als vanuit auditopunt een hoger niveau van security vereist wordt of bij de vraag of het systeem opgeschaald kan worden om na een fusie twee keer zoveel klanten te bedienen.
- De architectuur: het systeem staat niet op zichzelf, maar zal over het algemeen met meerdere systemen samenwerken. Het 'landschap' van systemen verandert voortdurend, bijvoorbeeld als het systeem zijn informatie moet aanleveren aan een nieuw datawarehouse of als een systeem informatie moet gaan leveren aan een andere organisatie, zoals de Belastingdienst.
- De technologieën: de technologieën waarin het systeem is uitgevoerd, zoals de programmeertalen, databasesystemen, frameworks (dit zijn systemen

waarmee standaardtaken ingevuld kunnen worden, bijvoorbeeld J2EE voor webapplicaties), et cetera. Het spreekt voor zich dat deze technologieën sterk aan verandering onderhevig zijn; bij de overgang naar een nieuwe technologie of zelfs een nieuwe release van een bestaande technologie is het overigens wel zaak de 'halfwaardetijd' zo goed mogelijk vooraf te bepalen.

- De beschikbare kennis: de kennis die beschikbaar is in de arbeidsmarkt is een essentieel punt en is gerelateerd aan het voorgaande. Een belangrijk aspect van legacy is de mismatch tussen enerzijds de kennis die men nodig heeft om decennia oude systemen te onderhouden en anderzijds de kennis die men opdoet in het hoger en wetenschappelijk onderwijs. Zo is het mainframeplatform inmiddels totaal onbekend bij vele generaties afgestudeerden, terwijl er wereldwijd en in Nederland heel veel cruciale systemen op draaien. Dit fenomeen kan enigszins worden gecompenseerd door over de grens te kijken. Feit is dat jonge afgestudeerden uit Bangalore zich nog wel voor kortere tijd willen verdiepen in deze technologieën, maar of dit een oplossing is voor de langere termijn, valt te bezien. Het onderwijs zal zich moeten heroriënteren: grote applicaties met hoge verwerkingsvolumes en veel gelijktijdige gebruikers blijven altijd nodig, op het mainframe (of een de facto vergelijkbare configuratie) draait immers de kern van het applicatielandschap.

Elk van de genoemde dimensies is continu in beweging en het is duidelijk dat de softwaresystemen niet zonder meer met elke verandering kunnen meegaan, tenminste als men de budgetten redelijkerwijs beperkt houdt. De afstand tussen het werkelijke systeem en het 'ideale' systeem wordt groter naarmate de tijd verstrijkt. Ontwerpkeuzes die ooit een goed idee leken, blijken vandaag een obstakel voor aanpassingen. Dergelijke keuzes worden daarom ook wel aangeduid met de term 'software-asbest' (Klusener, Lämmel & Verhoef, 2005). Het is zaak om periodiek de vormen van software-asbest in kaart te brengen en zo nodig te verhelpen in preventief onderhoud. De term 'legacy' heeft dus een negatieve connotatie en richt zich vooral op het niet altijd op feiten gebaseerde en daarmee gepercipieerde onvermogen om het systeem met redelijke inspanningen aan te passen aan de wensen van vandaag. Het is echter zeker niet alleen kommer en kwel met deze legacysystemen. Onmiskienbaar vervullen zij veelvuldig een belangrijke taak, anders zouden zij al zijn uitgefaseerd. De bouwkosten zijn lang geleden

afgeschreven, de systemen zijn stabiel en hebben zich bewezen. Nieuwbouw van systemen vergt naast de kosten van het nieuwbouwproject ook nog eens kosten van onderhoud om het systeem stabiel te krijgen en in de lucht te houden. Deze onderhoudskosten zijn gemiddeld genomen van dezelfde orde als het nieuwbouwtraject, maar worden veelal niet vooraf meegenomen en komen als een soort van schokgolf achteraf. Dit fenomeen is duidelijk te zien in figuur 1. De piek betreft hier de totale kosten voor een portfolio aan projecten voor nieuwbouw en adaptief onderhoud, de kosten voor onderhoud vormen een tweede schokgolf die daar in verloop van tijd op volgt.



Figuur 1. Initiële ontwikkelkosten (eerste piek) en de bijbehorende onderhoudskosten (tweede golf) (Verhoef, 2002)

De legacymedaille heeft dus twee kanten. Enerzijds is een legacystelsel ouder of verouderd en daarom soms lastiger aan te passen aan wijzigende situaties dan een 'modern' systeem, anderzijds heeft het systeem zich bewezen en vervult het zijn taak stabiel tegen relatief lage kosten. Bij een nieuw systeem is het precies omgekeerd. Dat ligt nog op de tekentafel, dus alles is mogelijk met de charme van de belofte; de keerzijde is dat de totale kosten moeilijk zijn in te schatten en het lang duurt voordat het systeem stabiel zal zijn. Nieuwbouwtrajecten zijn echter interessanter en sexier, voor het management en de uitvoerende staf. Bij legacysystemen heeft de negatieve kijk, vaak niet feitelijk onderbouwd, veelal de overhand, en bij nieuwbouwprojecten juist de positieve kant. Veel organisaties hebben hierin al aanzienlijke sommen leergeld betaald.

Toekomstvastheid, een duurzame kijk op softwaresystemen

Zoals hierboven beschreven is veroudering inherent aan softwaresystemen die niet los van hun context gezien kunnen worden.¹ De term 'legacy' is daarbij niet voorbehouden aan de Cobol-, PL/I- of RPG-systemen van vijftien of zelfs veertig jaar geleden. Ook systemen die recent in Java, C# of Visual Basic geschreven zijn, zullen vroeg of laat kenmerken van een legacystelsel gaan vertonen, als dat niet al het geval is. Sterker nog, hoe groter de vooruitgang in de diverse software-technologieën, hoe groter de veroudering van hun voorgangers. Daarbij komt nog dat niet elke vernieuwing ook stand blijkt te houden. Zo zijn er diverse systemen die gebaseerd zijn op bepaalde codegeneratoren of regelgebaseerde systemen van

minder dan tien jaar geleden, met meer legacy-kenmerken dan de systemen die gebaseerd zijn op meer standaard derdegeneratietechnologieën als Cobol. Bij overgang op een nieuwe technologie is pas na jaren duidelijk of op het juiste paard gewed is. Het idee dat 'de legacyproblematiek' kan worden opgelost door alle oude systemen te vervangen door nieuwbouw is veel te simplistisch. Bovendien, zonder de juiste maatregelen is de nieuwbouw van vandaag de legacy van morgen. Waar het om gaat is om de veroudering de baas te blijven, dat wil zeggen dat het systeem in een 'gezonde' conditie moet komen en blijven, zodat ook in de toekomst de noodzakelijke wijzigingen tegen beheersbare kosten kunnen worden doorgevoerd. De ervaring leert dat veel organisaties het beheer van grote legacysystemen onvoldoende onder controle hebben. Om een nieuwe, op feiten gebaseerde basis te creëren is een volledig onderzoek naar de kwaliteit van de software noodzakelijk.

1. Menig softwarestelsel heeft al vele computerinfrastructuren overleefd. De 'software' is dus veelal duurzamer, soms tegen wil en dank, dan de vervangbaar gebleken 'hardware'. Achteraf bekeken hadden de termen wellicht dus andersom gekozen moeten worden, de programmatuur als hardware en de computerinfrastructuur als software.

»De ervaring leert dat veel organisaties het beheer van grote legacysystemen onvoldoende onder controle hebben«

Sturen op kwaliteit

Op het niveau van de raad van bestuur heeft de IT-portefeuillehouder het vaak moeilijk. Voor de systemen die adequaat functioneren is geen belangstelling anders dan als kostenpost: hoe lager hoe beter. Over de problematische systemen



hebben alle leden een scherp oordeel en de legacy moet opgeruimd worden, nietwaar?

Het IT-management van een organisatie stuurt de uitvoerende organisatie, intern dan wel extern, veelal aan op kosten en datum van oplevering en minder op kwaliteit. De prijs hiervoor wordt op de lange termijn betaald, zoals blijkt uit het reeds aangehaalde feit dat een nieuw gebouwde systeem gemiddeld na oplevering nog vergelijkbare kosten vereist om het stabiel te krijgen. Na oplevering is het point of no return lang gepasseerd en kan er niet meer gewisseld worden van leverancier (alleen tegen zeer hoge kosten). 'Onderhoud' is hier eigenlijk een eufemisme, het betreft immers de kosten om de functionaliteit goed te krijgen; het opdrachtgevende management is vaak niet bij machte om deze kosten bij de leverancier te leggen dan wel de eisen van de eigen organisatie onder controle te houden.

Concrete maatregelen

Zoals gezegd zijn er twee kanten aan de legacy-medaille. Het managen van een softwareportfolio houdt onder meer in dat de negatieve kant van legacy moet worden beheerst en de positieve kant moet worden versterkt. Dit klinkt als een open deur, maar het kan worden ingevuld door maatregelen die helaas nog niet algemeen gangbaar zijn. De kern van de zaak is dat het management 'in control' moet zijn. Op dit moment ontbeert het management veelal de kennis om weerwoord te geven aan de uitvoerende organisatie en bieden de vooraf gemaakte afspraken geen soelaas om bij oplevering kanttekeningen te plaatsen bij de kwaliteit. Het gevolg is dat men te veel betaalt voor wat men krijgt – een lage prijs-kwaliteitverhouding – maar ook dat overgang naar een andere leverancier lastig blijkt. Om deze tendens te keren zijn twee maatregelen essentieel:

1. het uitvoeren van software-assessments waarin de software op kwaliteit wordt doorgelicht, door een gespecialiseerde en onafhankelijke partij, met het topmanagement en/of het bestuur als opdrachtgever;
2. het maken van een aantal afspraken over softwarekwaliteit tussen opdrachtgever en opdrachtnemer (in- en extern).

De internationale standaard ISO 9126 voor softwarekwaliteit schrijft een aantal aandachtsgebieden voor, te weten functionaliteit, betrouwbaarheid, bruikbaarheid, doelmatigheid, onderhoudbaarheid en overdraagbaarheid. De standaard beschrijft deze aandachtsgebieden in algemene zin, per softwaretechnologie en architectuur moeten zij nader worden ingevuld. Software is een complexe materie en het onderzoeken van deze aandachtsgebieden is dan ook veel meer dan het bepalen van een aantal standaardmetrieken en het nalopen van een checklist. Nadat het feitenmateriaal verzameld is, volgt nog een interpretatie waarbij ook de organisatorische context en strategische aspecten worden meegenomen, waarna een weloverwogen oordeel kan worden geveld.

Een belangrijk onderdeel van een assessment van de kwaliteit van de software betreft de gedetailleerde review van de broncode en aanverwante (meta)data zoals het datamodel, het versiemangementsysteem en de incidentendatabase. Dit is werk van specialistische aard dat schaars op de markt voorhanden is. Bovendien is het voor een opdrachtgever heel belangrijk dat dergelijk onderzoek wordt uitgevoerd in volledige onafhankelijkheid. De partij die het onderzoek uitvoert, mag dus niet in aanmerking komen voor het werk dat mogelijk anderszins voortvloeit uit het advies, zoals herorganiseren, herprogrammeren of converteren naar andere talen. Een belangrijk gedeelte van het onderzoek betreft het uitpluizen van de bestaande code; dit is een vorm van softwarearcheologie die nader wordt beschreven door Goes & Verhoef (ook in dit nummer van *Informatie*).

Louter aansturen op kosten en opleverdatum is niet voldoende, kwaliteit dient ook centraal te staan in de contractuele afspraken en de aansturing. Het management en de uitvoerende organisatie moeten daarom overeenstemming krijgen over onder meer de volgende punten:

- Een budget voor preventief onderhoud, bijvoorbeeld als percentage van de totale contractwaarde, en de wijze waarop dit budget wordt ingezet.
- De kwaliteitseisen die worden gesteld aan het opleveren. Deze kunnen betrekking hebben op de programmatuur, de documentatie, de architectuur et cetera.
- De indicatoren waarop gestuurd wordt, zoals het aantal nog toegestane overtredingen van de kwaliteitseisen, het aantal incidenten gerelateerd aan de programmatuur et cetera.

- Het proces waarmee de opleveringen via deze indicatoren worden getoetst op de afgesproken kwaliteitseisen en de consequenties die eraan worden verbonden.
- De informatie die over het softwareproces moet worden bijgehouden om de controle efficiënt en structureel te kunnen uitvoeren.
- De eisen die worden gesteld aan de ontwikkelaars die aan de programmatuur werken. Hierbij valt te denken aan het opleidingsniveau, de kennis van het businessdomein en van het systeem en de architectuur.
- De voorgaande punten dienen direct of indirect deel uit te maken van de vooraf gemaakte contractuele afspraken; het nader invullen van deze punten gaandeweg een project of uitbestedingscontract kost veel energie en heeft weinig resultaat.

De veroudering van software is een weerbarstige problematiek die geen pasklare antwoorden kent, elke situatie is immers weer uniek. Wel komen telkens dezelfde thema's terug, zoals de schaalproblematiek, de beschikbaarheid van kennis en vendor lock-in. In de wetenschappelijke literatuur wordt een aantal van deze thema's in isolement en in bepaalde varianten beschreven, een compleet overzicht valt buiten de scope van dit artikel. Het is echter nog geen sinecure om deze resultaten te combineren en in de industrie toe te passen.

Conclusie

Legacy, last en/of lust? De term 'legacy' kent geen harde definitie. Wat legacy lijkt kan na een grondige studie een essentieel en waardevol systeem blijken; andersom kunnen nieuwe systemen legacykenmerken hebben in negatieve zin. Negatieve connotaties die niet op feiten zijn gebaseerd, zijn weinig constructief, er kan geen beleid op gebaseerd worden. De juiste feiten moeten bovenkomen om tot een visie te komen voor de lange termijn. Software veroudert, of je dat nu wilt of niet; preventieve en correctieve maatregelen blijven nodig, ook om te voorkomen dat de software van vandaag de legacy van morgen wordt.

Literatuur

- Goes, W. & C. Verhoef (2010). Softwarearcheologie als basis voor strategie. *Informatie* 52/4 (mei).
- Groot, R., M. Smits & H. Kuipers (2005). A Method to Redesign the IS Portfolios in Large Organisations. *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, vol. 08, 3-6 January, p. 223a.
- Klusener, S., R. Lämmel & C. Verhoef (2005). Architectural Modifications to Deployed Software. *Science of Computer Programming* 54, pp. 143-211.
- Lehman, M. & J. Ramil (2003). Software evolution: background, theory, practice. *Information Processing Letters*, vol. 88, issue 1-2, October, pp. 33-44.
- Verhoef, C. (2002). Quantitative IT Portfolio Management. *Science of Computer Programming* 45, pp. 1-96.

Steven Klusener

is zelfstandig adviseur, is als gastonderzoeker en gastdocent verbonden aan de VU en is als consultant to the firm verbonden aan PricewaterhouseCoopers Advisory. E-mail: steven@klusener.com.

Halbe Kuipers

was directeur van de Information Management Group van PricewaterhouseCoopers Advisory en is docent aan de Universiteit van Tilburg – TIAS Business School. E-mail: halbe.kuipers@xs4all.nl.