Software Ontwikkeling: Een Slijtageslag

Jilles van Gurp & Jan Bosch

Samenvatting. Vaak wordt *verondersteld dat de opkomst van nieuwe technieken zoals bijvoorbeeld web services en object-orientatie, vereisen dat de ontwikkelmethodes die door software huizen gebruikt worden bij de ontwikkeling van systemen aangepast worden. In dit artikel wordt beargumenteerd dat veranderende techniek slechts een van de factoren is die bijdragen aan de geleidelijke erosie van software ontwerpen. Methodes moeten niet worden aangepast aan de nieuwste techniek maar aan het feit dat het vaststaat dat software ontwerpen onder voortdurende druk staan door verandering.*

Het hoofddoel van het gebruiken van een software ontwikkel methode is het bouwen van software systemen een voorspelbare en vooral ook herhaalbare activiteit te maken zodat bij aanvang van een software project met een redelijke zekerheid kan worden voorspeld wanneer het project klaar is. Hoe simpel dit ook klinkt, het bereiken van dit doel is al bijna veertig jaar het onderzoeksonderwerp van het onderzoeksgebied Software Engineering. Hoewel er in deze 40 jaar zeker vorderingen gemaakt zijn op het gebied van software ontwikkel methodes, wordt het merendeel van de software projecten nog steeds gekenmerkt door budget en tijds overschrijdingen. Dit ondanks de grote variatie aan methodes die worden toegepast in de automatiseringsindustrie. Het blijkt in de praktijk bijzonder lastig om op een gecontroleerde en gestructureerde manier software te ontwikkelen die aan vooraf opgestelde kwaliteitsvoorwaarden voldoet.

Onzekerheid veroorzakende factoren zoals veranderende klanteisen, nieuwe technieken, vertrekkend personeel en, ironisch genoeg, veranderende methodes gooien roet in het eten bij het uitvoeren van software projecten. Als gevolg van dit soort factoren is het erg lastig om realistische, lange termijn planningen te maken voor een software project. Noodzakelijkerwijs moeten er in een planning aannames gemaakt worden over productiviteit van werknemers, beschikbaarheid van kennis, de toegepaste technieken, de juistheid/volledigheid van de op dat moment beschikbare klanteisen en toekomstige technische veranderingen.

Met name het introduceren van nieuwe technieken brengt zowel de voorspelbaarheid als de herhaalbaarheid van het produceren van software in gevaar. In de eerste plaats is er bij de introductie van nieuwe technieken doorgaans een gebrek aan kennis over die technieken. In de tweede plaats zijn de gevolgen voor eventuele planningen slecht te overzien. Daarnaast is het maar de vraag in hoeverre de al bestaande software geschikt is voor het integreren van de nieuwe technieken.
Dit laatste is in toenemende mate een probleem aangezien er een verschuiving is van het ontwikkelen van nieuwe software naar het aanpassen en uitbreiden van bestaande software systemen. De reden hiervoor is dat software systemen steeds groter worden en een steeds grotere economische waarde hebben. Bovendien hebben veel bedrijven in de loop der jaren een hoop software verzameld waar zij in grote mate van afhankelijk zijn.

Het gevolg van deze verschuiving is dat er bij de ontwikkeling van software sterk rekening moet worden gehouden met de bestaande software.
De flexibiliteit en het ontwerp van de oude software bepalen wat er mogelijk is met betrekking tot eventuele uitbreidingen. Dit ontwerp kan door de aanhoudende stroom veranderingen en aanpassingen dusdanig onder druk komen te staan dat er een fenomeen optreed dat bekend staat als 'software aging' [Parnas 1994] of ook wel 'ontwerp erosie' [Van Gurp & Bosch 2002]. Dit houdt in dat de kwaliteit van de software afneemt doordat er voortdurend ontwerpveranderingen moeten worden doorgevoerd om nieuwe aanpassingen te kunnen doen. Aangezien niet al deze ontwerp veranderingen even optimaal zijn, treedt er een geleidelijk kwaliteitsverlies op.

In principe zou ontwerp erosie te voorkomen zijn door voor iedere veranderingen de tijd te nemen om weloverwogen ontwerpbeslissingen te nemen en geen compromissen te sluiten ten koste van de kwaliteit van het ontwerp. Echter, alleen al om economische reden is dit onmogelijk. Het is onvermijdelijk dat dit soort compromissen af en toe gesloten moeten worden. Daarnaast is het zo dat iedere beslissing plaatsvindt in de context van allerlei toekomstverwachtingen die later onjuist kunnen blijken te zijn.
Als gevolg hiervan is het onvermijdelijk dat er in de loop der tijd een hele serie niet optimale wijzigingen op de software gemaakt worden die een cumulatief effect hebben. Door het cumulatieve karakter van ontwerpbeslissingen is het doorgaans slecht mogelijk om eerder genomen ontwerpbeslissingen terug te draaien. Nieuwere ontwerpbeslissingen vinden plaats in de context van alle voorgaande beslissingen en vaak zijn er allerlei afhankelijkheden tussen beslissingen.

Een stelling die we in [Van Gurp & Bosch 2002] onderbouwen is dat ontwerp erosie onvermijdelijk ieder software systeem aantast en op de lange termijn tot gevolg zal hebben dat bepaalde wijzigingen niet langer op een economisch verantwoorde wijze kunnen worden gemaakt. De gevolgen kunnen verstrekkend zijn voor zowel bedrijven die de software produceren als voor de klanten van deze bedrijven. Het vervangen van de software (als dit economisch überhaupt haalbaar is) kost vaak jaren. Tegelijkertijd verslechtert de concurrentiepositie van betrokken bedrijven ten opzichte van bedrijven die wel in staat zijn zich aan te passen aan nieuwe markteisen en aan nieuwe technieken. Als deze situatie te lang niet onderkent wordt (waardoor de vereiste koerswijzigingen te lang uitblijven), kan dit fataal zijn voor een bedrijf.

Een goed voorbeeld van een bedrijf dat dit is overkomen is Netscape. In 1997 was dit bedrijf verwikkeld in een felle concurrentiestrijd met Microsoft. De ene feature na de andere werd toegevoegd aan hun Netscape Communicator product. Eind 1997 werd het duidelijk dat Netscape het onderspit ging delven en uiteindelijk werd er besloten de source code vrij te geven in de hoop dat de open source gemeenschap zou slagen waar Netscape gefaald had. Het bleek steeds lastiger om aanpassingen te maken. Een paar maanden na het vrijgeven van de source code besloten de coordinatoren van wat inmiddels het Mozilla project was gaan heten, geheel overnieuw te beginnen. Het daarop volgende ontwikkeltraject heeft drie jaar geduurd (tot de 1.0 versie). Hoewel de daaruit resulterende browser best een aardig product is, heeft het Netscape weinig opgeleverd. Netscape is een paar jaar opgekocht door media gigant America Online (AOL). In juli 2003 is uiteindelijk het bedrijf Netscape ontmanteld. AOL (en diverse andere IT

bedrijven) sponsort overigens nog steeds de Mozilla Foundation, een stichting die is opgericht om Mozilla verder te ontwikkelen.
Dit voorbeeld illustreert hoe het te laat signaleren van ontwerp erosie verstrekkende gevolgen kan hebben. Toen eenmaal besloten werd dat de oude Netscape 4.x code niet meer te repareren viel, had Netscape de concurrentiestrijd al lang verloren. Het daarop volgende herontwikkel traject is illustratief door zijn lengte en kosten.
Het bedrijf Netscape is geen uitzondering en ontwerp erosie is voor veel bedrijven een reële bedreiging. In de afgelopen tien jaar is er veel geinvesteerd in IT infrastructuur met als gevolg dat een doorsnee bedrijf in het bezit is, en afhankelijk is van enorme hoeveelheden software. Nieuwe software projecten bij dit soort bedrijven vinden plaats in de context van deze software. Ontwerp erosie is voor dit soort bedrijven erg bedreigend omdat het nieuwe software projecten duurder maakt en tegelijkertijd de concurrentie positie ondermijnd.

Het tot nu toe geschetste beeld is niet al te vrolijk. Ontwerp erosie is onafwendbaar en bedrijven zijn in grote mate afhankelijk van hun steeds grotere, eroderende, onvervangbare systemen. Echter, het probleem is niet zozeer dat software aan erosie onderhevig is (iets dat wij als een gegeven beschouwen) maar dat veel bedrijven zich hier veelal niet of nauwelijks van bewust zijn. Een bedrijf dat zich bewust is van het feit dat software slijt, kan daar rekening mee houden en zijn ontwikkelmethode aanpassen aan dit gegeven.

In de eerste plaats is het belangrijk met een kritische blik te kijken naar de kwaliteit van de software en alert te zijn op erosie symptomen. Ontwerp erosie is een heel geleidelijk proces, ontwikkelaars hebben vaak niet door dat hun software die ooit ontwikkeld werd op basis van een gedegen ontwerp, door de aanhoudende stroom van veranderingen steeds verder verwijderd is geraakt van de oorspronkelijke doelstellingen van dit ontwerp. Echter, ontwerp erosie heeft een heel duidelijk effect op software: aanpassingen worden duurder en gaan gepaard met relatief veel defecten. Het simpelweg in de gaten houden van metrieken op dit gebied (bijvoorbeeld het aantal defecten dat per tijdseenheid wordt afgehandeld, het aantal defecten dat gemeld wordt, kosten van aanpassingen, etc.) levert hele bruikbare informatie op over de toestand van een systeem.
Recent hebben wij een case studie gedaan bij een groot software bedrijf (helaas kunnen wij op dit moment nog niet vermelden om welk bedrijf het gaat) dat op deze wijze de kwaliteit van de software in de gaten houdt. We hebben bij dit bedrijf twee software componenten onderzocht die recent herontwikkeld zijn naar aanleiding van op deze wijze geconstateerde kwaliteitsproblemen. In een geval ging het om een component die al tien jaar in gebruik was en jarenlang prima zijn werk had gedaan. De reden om over te gaan tot herontwikkeling van deze component was dat de geprojecteerde kosten hiervoor minder waren dan het doorvoeren van de circa honderd noodzakelijke veranderingen die al gepland waren. Deze kostenschattingen konden worden gemaakt omdat dit bedrijf beschikte over de bovenstaande metrieken.

In de tweede plaats is het ook belangrijk om voorruit te kijken. Het is steeds vaker het geval dat software na de oplevering verder wordt ontwikkeld (hetzij in vervolg projecten, hetzij op een permanente basis). Het hebben van een technologie roadmap en het in acht nemen daarvan bij het maken van aanpassingen kan een hoop ellende voorkomen. Bij het

genoemde voorbeeld was dit een belangrijke factor in de beslissing over te gaan tot herontwikkeling van de component.
Verondersteld wordt vaak dat bestaande ontwikkelmethodes moeten worden aangepast wegens specifieke technische veranderingen. Echter, zoals we hierboven hebben beargumenteerd, is veranderende techniek slechts een van de onvoorspelbare factoren. Veel belangrijker is het om in te zien dat dit soort factoren bijdragen aan de erosie van software ontwerpen. Ontwikkelmethodes moeten worden aangepast aan het feit dat software erodeert. Door de kwaliteit van de software in de gaten te houden en door bij veranderingen rekening te houden met een lange termijn technologie roadmap kan worden voorkomen dat bedrijven in de problemen komen doordat hun software verouderd is en hun in staat stellen op tijd passende maatregelen te nemen (bijvoorbeeld herontwikkelen van een aangetaste component, refactoring [Fowler 1999]).

[Fowler 1999] M. Fowler, "Refactoring: Improving the Design of Existing Code", Addison-Wesley, 1999.
[Van Gurp & Bosch 2002] J. van Gurp, J. Bosch, “Design Erosion: Problems & Causes”, Journal of Systems & Software, 61(2), pp. 105-119, Elsevier, March 2002.
[Parnas 1994] D. L. Parnas, "Software Aging", in proceedings of ICSE 1994, pp. 279-287, 1994.