Architectuur en dynamiek van productsoftware

Architectuur productsoftware ontwikkelt evolutionair

Productsoftwareontwikkeling kent een voorgeschreven dynamiek waarin product- en procesverbeteringen een bepaald patroon volgen. De rol van de architect is deze evolutie te sturen.

Ilja Heitlager, Remko Helms en Sjaak Brinkkemper

Het begrip 'architectuur' staat volop in de belangstelling binnen de diverse disciplines van IT- en softwareontwikkeling. Zo wordt gesteld dat 'werken onder architectuur' een bijdrage levert aan het succesvol opleveren van software binnen de gewenste tijd, met de gewenste kwaliteit en tegen acceptabele kosten. Het architectuurbegrip wordt daarbij opgerekt en omvat naast de applicatiearchitectuur ook het voorschrijven van de ontwikkelmethode en -processen. De resultaten zijn afhankelijk van het vooraf inrichten van een architectuur.
Productsoftwareontwikkeling kent daarentegen een evolutionaire invulling. Het ontwikkelen van productsoftware (ook bekend als pakketsoftware) is het maken van software met de intentie die meervoudig aan de man te brengen. Voorbeelden van productsoftware zijn desktopapplicaties, boekhoudpakketten die met enige parametrisatie kunnen worden toegepast, maar ook ordermanagementsystemen die een klantspecifieke inrichting tijdens implementatie bij de klant vereisen. Productsoftware wordt aangeboden als zilverschijfje, als lokaal te installeren software en in de vorm van een dienst, de *software-as-a-service*.

Productsoftwareontwikkeling

Er is een fundamenteel verschil tussen software ontwikkeld als product en software ontwikkeld in projectvorm. Softwareontwikkeling in een project is een eindige activiteit. Het basisprincipe van een project is dat het een keer zal eindigen. De software gaat daarna in onderhoud. Productsoftwareontwikkeling is een vooraf voorbestemde oneindige activiteit. Het product is een component in een niet-aflatende zoektocht naar economische groei. Deze groei betekent dat er meer krachten zijn om het product te laten evolueren, maar ook dat de organisatie en dus het proces meegroeien. Dit proces is per definitie evolutionair en heeft een eigen dynamiek.
De standaarddefinitie voor architectuur volgens IEEE 1471 is:'Architectuur is de beschrijving van de fundamentele opbouw van een systeem bestaande uit de componenten, de onderliggende relaties en die tot hun omgeving en de principes voor hun ontwerp en evolutie.' Net als bij bijvoorbeeld NORA en DYA zal ook binnen productsoftware dit begrip ruimer worden opgevat en zal ook de organisatorische context worden meegenomen.
De principes voor evolutie van zowel product als proces wijken namelijk bij productsoftwareontwikkeling af van de maatwerksoftwareontwikkeling. Traditioneel wordt het softwareontwikkelingsproces als een lineair proces geschetst waarin een idee ontstaat voor een product. Er wordt dan een ontwikkelproces gekozen en de sofware wordt ontwikkeld. Vervolgens wordt het marketingmateriaal geschreven en daarna wordt de software gelanceerd en verkocht. Dit beeld blijkt te simpel, want de dynamiek van softwareontwikkeling is meer divers.

Productsoftware komt in diverse vormen tot stand, bijvoorbeeld:

Samenvatting

Productsoftwareontwikkeling kent een eigen dynamiek. In aanvang is er behoefte aan flexibiliteit als klantwensen nog onderzocht worden. Naarmate de interesse van de markt toeneemt, zal de organisatie zich moeten voegen naar de stijgende vraag. Dit vergt aanpassingen in het ontwikkelproces en het product zelf. Er wordt onderzoek gedaan naar het verloop van deze ontwikkelingen binnen bestaande productsoftwarebedrijven.

• Een branchespecifiek orderafhandelingssysteem dat aanvankelijk voor intern gebruik is ontwikkeld door de interne IT-afdeling van een exportbedrijf, wordt nu aan andere partijen binnen de branche aangeboden.
• Een softwaredienst die ontwikkeld is voor een overheidsdepartement, is nu geïdentificeerd als breder inzetbaar en is kandidaat voor landelijke uitrol bij een veelvoud van overheidsinstellingen.
• Een ziekenhuisinformatiesysteem is door een externe partij op maat ontwikkeld voor een ziekenhuis. De externe ontwikkelaar ziet mogelijkheden en verkoopt het nu aan meerdere klanten.
• Twee studievrienden met een wens om ondernemer te worden, zijn nu succesvol met een online boekhoudpakket.

Maar incidenteel komt de software projectmatig en door een team met zeer veel ervaring tot stand. Productsoftware zal over het algemeen eerder tot stand komen in een vrij spontane omgeving en op vrij organische wijze (Livingstone, 2007). Afhankelijk van het investeringsklimaat kent de start van productsoftwareontwikkeling beperkingen en daarnaast ook meer onzekerheden dan projectgebaseerde software, in het bijzonder in de markt en het team. Voor een nieuw product is de markt en dus de eindgebruiker onbekend. In deze situatie is het minder vanzelfsprekend om *requirements* in beton te gieten en vanuit een bevroren situatie de software te gaan ontwikkelen. Papieren prototypes verkopen slecht en daardoor is de druk groter om werkende software op te leveren. Functionaliteit zal daarbij belangrijker zijn dan kwaliteit. Het vroeg introduceren van het product in de markt levert opvallend genoeg juist een positieve bijdrage aan de kwaliteit van software (MacCormack, 2001).
Terwijl in contractgebaseerde softwareontwikkeling een projectteam vaak een ruimere keuze heeft uit goed opgeleide specialisten, zal productsoftwareontwikkeling meer afhankelijk zijn van enkele, minder formeel opgeleide maar getalenteerde ontwikkelaars. Vanuit deze onzekerheden bekeken is een evolutionaire aanpak vanzelfsprekend. Deze situatie kan echter geen standhouden. Onafhankelijk van de totstandkoming van de eerste versie van de software krijgt elk succesvol softwarebedrijf te maken met schaalvergroting. Succes betekent meer klanten en meer klanten betekent onherroepelijk meer klantwensen, wat leidt tot aanpassingen in de software en dus ook tot meer personeel.

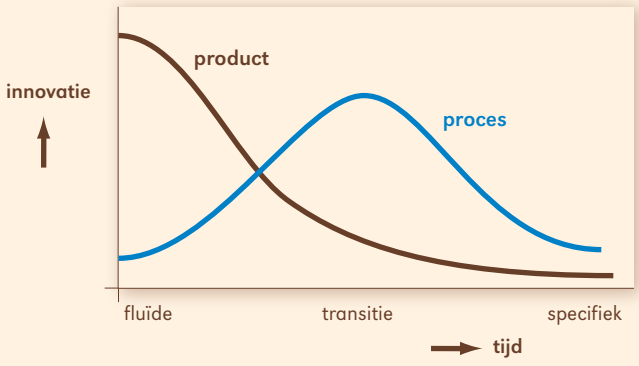
De productinnovatiedynamiek

De dynamiek in de ontwikkeling van productsoftware vertoont overeenkomsten met de dynamiek van reguliere tastbare producten zoals deze bekend is uit de ontwikkeling van fysieke producten zoals machines en vliegtuigen. Een vergelijking met het innovatiemodel van Utterback en Abernathy is daarmee snel gemaakt (Utterback, 1994). In dit model wordt uitgedrukt dat innovaties, gedefinieerd als verbeteringen in zowel product als proces, een voorbeschreven dynamiek volgen naarmate het bedrijf volwassen wordt en meer gaat verkopen.
Volgens Abernathy en Utterback doorloopt een productie-unit (dit kan zowel een bedrijf als een hele industrie zijn) drie belangrijke groeifasen. De productie-unit bevindt zich eerst in een fluïde fase. Hier worden voornamelijk verbeteringen in het product doorgevoerd. Via de transitiefase, waarin procesaanpassingen de overhand hebben, ontwikkelt de productie-unit zich door naar de specifieke fase. In de laatste fase gaan proces- en productinnovatie door, maar allebei op een lager niveau. Het model is weergegeven in figuur 1.
Bij elke fase hoort een andere vorm van organisatie. In aanvang zijn zowel de productie-unit als de markt nog onbekenden van elkaar. Generalisten werken in een informeel georganiseerde organisatie. Ze maken daarbij gebruik van algemene tools voor de ontwikkeling van het product. Alle aan-

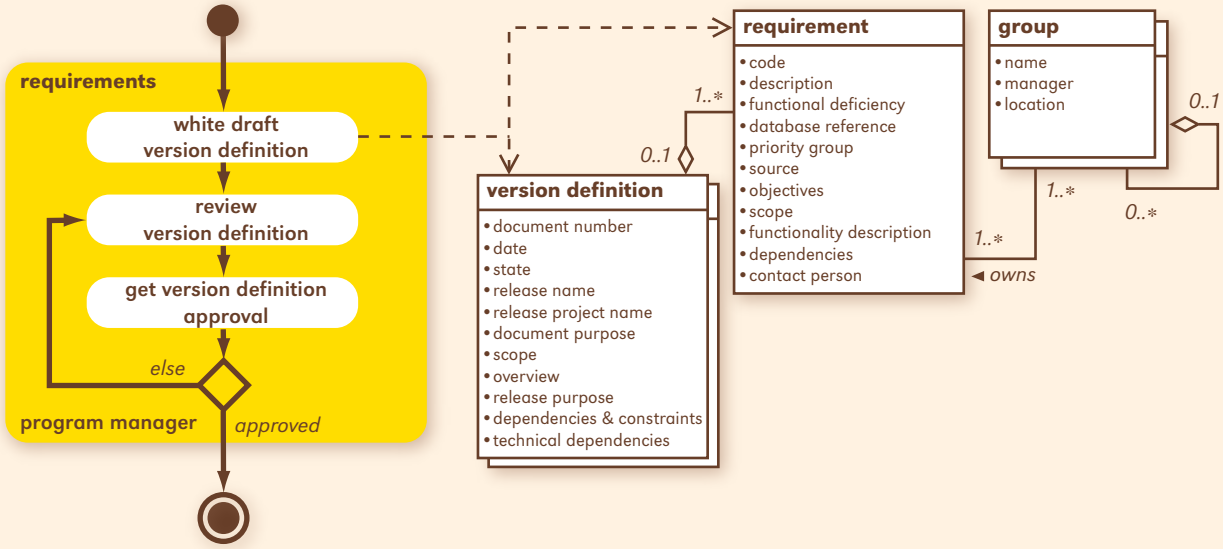
passingen die worden gedaan, betreffen voornamelijk het product en nauwelijks het proces. Het ontwikkelproces is flexibel, vaak ongedefinieerd, en niet noodzakelijkerwijs efficiënt van aard.
Op een gegeven, niet algemeen bepaalbaar en voorspelbaar, moment ontstaat er een ontwerp dat aantrekkelijk is voor de markt. De verkopen nemen toe: er is een dominant ontwerp tot stand gekomen. Utterback stelt heel duidelijk dat een dominant ontwerp niet specifiek is voorbestemd, maar een resultaat is van de interactie tussen technische keuzes en marktkeuzes op een gegeven moment.
De verkopen nemen toe en de organisatie gaat zich richten naar dit volume. De eerste vormen van organisatie volgen. De verbeteringen en innovaties vinden in deze fase voornamelijk plaats op procesniveau. De innovaties op productniveau gaan door, maar nemen af in intensiteit, frequentie en impact. Uiteindelijk zal de organisatie terechtkomen in een specifieke fase. Innovaties van zowel product als proces gaan door, waarbij de focus voornamelijk ligt op kostenbesparing. In plaats van generieke tooling wordt nu gebruikgemaakt van tooling die op maat is gemaakt voor het productieproces. Radicale wijzigingen zijn kostbaar en arbeidsintensief en worden daarom vermeden. Kortom, de flexibiliteit is ingeruild voor optimalisering van de productieprocessen.
Het model van Abernathy en Utterback is uitvoerig onderzocht in de wereld van productie, bijvoorbeeld de luchtvaartsector, kantoorautomatisering, fotografie en bulkproductie. Er zijn natuurlijk verschillen, want met de reproductie van productsoftware wordt niet de productie van installatie-cd's en boeken bedoeld, maar dat is van ondergeschikt belang. Er wordt geduid op de evolutie van het softwareontwikkelingsproces, vanaf requirements management tot aan software delivery aan klanten.
Wat het model van Abernathy en Utterback ons leert is dat er een duidelijk onderscheid is tussen product- en procesontwikkelingen. Tijdens het volwassen worden van de organisatie volgen deze een voorgeschreven dynamiek. Bewustwording van deze evolutie maakt het mogelijk tooling en technieken op het juiste moment te plaatsen.
Voor startende ondernemingen ligt de nadruk vooral op creativiteit en flexibiliteit (Bhidé, 2004). Deze omgevingen kennen vaak onervaren, maar zeer gemotiveerd talent, grote aanhangers van het *agile*-kamp (Beck, 2001). Structuur tussen de werknemers is nauwelijks nodig, code is belangrijker dan specificatie en veranderingen zijn eenvoudig door te voeren.
Hier wordt de aanname gedaan dat een softwareproduct verder evolueert. De klantwensen staan daarbij niet stil en daarom zal een productsoftwareorganisatie zich verder ontwikkelen. Haar ontwikkelproces hoort zich daarbij aan te passen. In het beginstadium zal een groepje generalistische programmeurs aan het product beginnen, dat zijn medewerkers die aan een half woord genoeg hebben. Naarmate de organisatie volwassener wordt, zal haar proces formeler worden. Deelstappen in het ontwikkel-, test-, package- en deploymentproces, die eerst handmatig werden uitgevoerd, zullen in deze overgang verregaand worden geautomatiseerd. Daarmee zal de organisatie onvermijdelijk flexibiliteit inruilen voor optimalisatie en dus doorgroeien van fluïde naar specifiek.
Naarmate een softwareproduct volwassener wordt, worden ook de producten in de periferie verder ontwikkeld. Heeft versie 1.0 alleen nog maar een binary om te distribueren (ook wel generiek product genoemd), latere versies kennen verschillende componenten waaruit kan worden gekozen en een heel scala aan ondersteunend materiaal zoals *getting started guides*, handleidingen en cursus- en marketingmateriaal. Hiermee is met het volwassen worden van de organisatie een geheel ontstaan, het heel product genoemd (Moore, 1991).

Instrumenten voor productsoftware

Zoals gezegd is het vinden van een dominant ontwerp een samenspel tussen technische keuzes en marktkeuzes. Een dominant ontwerp kan alleen



Figuur 1. **Het model van Abernathy en Utterback over de dynamiek van innovatie in productbedrijven**



Figuur 2. Voorbeeld van methodefragment voor requirements management, waarin de samenhang tussen requirementsactiviteiten en requirements deliverables wordt aangegeven

ontstaan als de organisatie flexibel is en snel kan inspelen op veranderingen. Als het dominante ontwerp eenmaal staat en de markt het product afneemt, zal de groei inzetten.
Maar in deze onvoorspelbare en evolutionaire ontwikkeling blijven deze stappen bewuste keuzes (Bhidé, 2003). Een architect of lead engineer binnen een productorganisatie is zich bewust van de evolutie en zal de samenhang van product en proces sturen. In de ogenschijnlijke oneindigheid van de levensduur van het product zullen beide hand in hand doorevolueren. Het instrumentarium dat de architect daarbij tot zijn beschikking heeft, is *method engineering*.
Method engineering is het modelleren van het ontwikkelproces (Van de Weerd & Brinkkemper, 2007). Activiteiten worden onderverdeeld in processtappen en bijbehorende artefacten. Gedreven door noodzaak zal het proces zich incrementeel aanpassen: extra processtappen en methodefragmenten worden ingevoerd, extra documenten en deliverablefragmenten worden bijgevoegd. Een voorbeeld is gegeven in figuur 2.

Laten we eens kijken naar een aantal voorbeelden van methode-evolutie.
1. In de evolutie van productsoftwarebedrijven zullen diverse processen zich verder ontwikkelen. Een startende, zich tot het agile-kamp aangetrokken voelende onderneming zal vrij direct aan de start gaan met het schrijven van code: een werkend product is het doel, niet uitgebreide documentatie. Gestuurd door niet meer dan *user stories* gaan de 'codekloppers' aan de slag. Na de markt-introductie van het product, de release van de eerste officiële versie, zal versiebeheer zijn intrede doen. Wanneer de klantwensen toenemen, ontstaat er meer behoefte aan bespreking, inschatting en prioritisering vooraf, waardoor er een noodzaak ontstaat tot geschreven functionele ontwerpen. Het requirementsproces ontwikkelt zich zo verder.
2. Een software-as-a-service-leverancier in de supply-chain-managementmarkt. Dit product wordt geleverd als add-on product voor bestaande ERP-infrastructuren in diverse markten. Zagen de eerste versies en implementaties voor een klant er nog uit als veredeld maatwerk, gaandeweg ontstond de behoefte aan een standaardarchitectuur: een standaardkern voor alle implementaties en een aanpassingslaag specifiek voor de klant. Deze aanpassingslaag kreeg ook haar eigen taal waardoor er een enkelvoudige barrière ontstond tussen het productontwikkelteam en het klantimplementatieteam. Beide teams kennen hun eigen dynamiek, de organisatorische scheiding wordt gespiegeld in de applicatiearchitectuur.
3. Weinig softwareproducten worden in aanvang generiek opgezet. Pas bij een toenemende vraag in de markt of bij een strategische keuze wordt software geport naar andere omgevingen. Neem bijvoorbeeld BusinessObjects, dat in het begin meeliftte met het succes van Oracle. Later heeft Oracle besloten het ook voor andere platforms uit te brengen. Dit heeft consequenties voor de productarchitectuur maar ook voor het ontwikkel- en testproces. Het vraagt bijvoorbeeld voorheen niet noodzakelijke, extra expertise van diverse platforms. Als hier vooraf rekening mee wordt

gehouden, kan dat vaak juist belemmerend en vertragend werken. Dit is ook mede afhankelijk van de strategische allianties die een organisatie kan of wil aangaan (Cusumano, 2004).

Een architect in een productsoftwarebedrijf denkt evolutionair. Het softwareproduct krijgt vorm in interactie met de markt. Naarmate de belangstelling voor het product groter is, zullen er meer wensen komen. Het proces, de organisatie en de applicatiearchitectuur zullen zich daaraan aanpassen. Niet noodzakelijkerwijs zijn processen in een later stadium van een productsoftwarebedrijf al geschikt bij de start van een bedrijf.
Binnen de Informatie & Organisatie-groep aan de Universiteit Utrecht wordt, behalve naar method engineering, onderzoek gedaan naar reference frameworks voor productsoftware, bijvoorbeeld voor requirements management en software delivery. Deze referentiemodellen leveren een bijdrage aan het volwassen worden van de productsoftware-industrie en zijn vooral van toepassing op reeds gevestigde bedrijven. Verder wordt er onderzoek gedaan naar hoe startende organisaties de overgang maken naar de specifieke fase en hoe de architect dit overgangsproces kan ondersteunen.

Literatuur

Beck, K. e.a. (2001), Manifesto for Agile Software Development, www.agilemanifesto.org.

Bhidé, A.V. (2003). *The Origin and Evolution of New Businesses*. Oxford University Press USA.

Cusumano, M.A. (2004). *The Business of Software*. Free Press.

Livingston, J. (2007). *Founders at Work: Stories of Startup's Early Days*. Apress.

MacCormack, A. (2001). Product-Development Practices that Work: How Internet Companies Build Software. *MIT Sloan Management Review* 42-2, pp. 75-84.

Moore, G.A. (1991). *Crossing the Chasm*. HarperBusiness.

Utterback, J. (1994). *Mastering the Dynamics of Innovation*. HBS Press.

Weerd, I. van de & S. Brinkkemper (2007). Meta-modeling for situational analysis and design methods. In: *Handbook of Research on Modern Systems Analysis and Design Technologies and Applications*. Hershey: Idea Group Publishing.

Ilja Heitlager
is werkzaam bij de Software Improvement Group en is verbonden aan de Universiteit Utrecht als extern onderzoeker. E-mail: i.heitlager@sig.nl.

Remko Helms
is docent/onderzoeker Knowledge Management bij het Instituut voor Informatica en Informatiekunde van de Universiteit Utrecht. E-mail: r.w.helms@cs.uu.nl.

Sjaak Brinkkemper
is hoogleraar bij het Instituut voor Informatica en Informatiekunde van de Universiteit Utrecht. E-mail: s.brinkkemper@cs.uu.nl.