

# De planologie van software systemen

Erik Philippus

Mijn zoon Tim is 1<sup>e</sup> jaars student planologie, en hij vroeg me laatst om 'm te helpen met het tentamen "Inleiding in de Planologie". Hij had nog een weekend de tijd om een dikke reader met artikelen te lezen, en ik nam de taak op me om van de belangrijkste artikelen een mindmap te maken.

Zodoende zette ik mij aan een artikel uit 1959: "The Science of "Muddling Through" van de Charles Lindblom, een thans 90-jarige Amerikaanse econoom [1]. Lindblom had destijds kritiek op het economisch planbeleid, dat in de jaren '50 was gebaseerd op het idee van de maakbare samenleving. Hij refereerde aan de 'Rational Comprehension Method': beleidmakers maken fanatiek gebruik van rationele methoden om problemen tot de grond toe uit te kleden, alle relevante factoren en variabelen te beschouwen. Lindblom stelde dat deze aanpak niet werkt, omdat mensen beperkt rationeel zijn en de problemen daarvoor meestal veel te complex zijn. Dus hij stelde voor om voor planologisch beleid een andere aanpak te volgen, die hij de "Successive Limited Comparison Method" noemde. Belangrijke beslissingen worden hierbij niet in één klap genomen, maar uitgevoerd in een reeks kleine stapjes, met de mogelijkheid tot tussentijdse bijsturing. Kenmerkend is ook dat de probleem-analyse drastisch wordt beperkt: lang niet alle mogelijke uitkomsten en alternatieven worden a-priori in beschouwing genomen. In de woorden van Lindblom: "The wise planner proceeds through a succession of incremental changes, avoiding serious lasting mistakes in several ways".

Mijn zoon keek even op van z'n werk toen ik hardop te kennen gaf dat er sprake was van enige herkenning(...). Ik bladerde even terug naar de eerste pagina - en ja, ik was toch echt een artikel uit 1959 aan 't lezen... Maar mijn verbazing steeg naar nog grotere hoogte toen ik het volgende artikel uit 1967 onder ogen kreeg, met de titel: "Mixed Scanning: A Third Approach to Decision-making", van Amitai Etzioni [2]. Onze eigen Jan Peter Balkenende is overigens een fervent bewonderaar van deze Joods-Amerikaanse socioloog, evenals Bill Clinton en Tony Blair - maar dat terzijde.

Etzioni begint z'n relaas met een historisch overzicht van planning en besliskunde. In den beginne was er "The Rationalistic Approach', met als heilige graal het verkrijgen van rationele controle over het traject. Zoals we al zagen, is hierop forse kritiek gekomen van een door Lindblom geïnspireerde groep planologen. Inmiddels verenigd onder de naam "The Incrementalists", wezen zij erop dat er meestal geen relevante criteria beschikbaar zijn om alle alternatieven te evalueren. Een ander punt van kritiek betrof de aanname dat consequenties van keuzes vooraf goed zijn te overzien: "Rather than being confronted with a limited universe of relevant consequences, planners face an open system of variables, a world in which all consequences cannot be surveyed. A decision-maker, attempting to adhere to the tenets of a rationalistic model, will become frustrated, exhaust his resources without coming to a decision, and remain without an effective model to guide him".

Vervang "rationalistic model" en "decision-maker" door "waterval model" en "architect", en de overeenkomst met ons vakgebied is frappant. Verwachtingsvol las ik dus verder. Wat bleek? De incrementele aanpak bleek in de praktijk toch niet zo goed te werken voor fundamentele beslissingen. Met het nemen van uitsluitend kleine stapjes wordt werkelijke verandering uit de weg gegaan, hoogstens wordt een bestaande (mogelijk ongewenste) situatie geoptimaliseerd. Want wie zegt dat je niet stapje voor stapje op weg bent in een doodlopende straat?

Etzioni vat de kritiek als volgt samen: "Incrementalism would tend to neglect *basic* innovation, as it focuses on the short run and seeks no more than limited variations from past policies. While an accumulation of small steps could lead to a significant change, there is nothing in this approach to guide the accumulation; the steps may be circular - leading back to where they started, or dispersed - leading in many directions at once but leading nowhere".

Er verschijnt zelfs een artikel met de titel: "Muddling through - science or inertia?" De soep wordt niet zo heet gegeten, meent Etzioni. De geschiedenis leert dat een incrementele benadering vaak anticipeert op een fundamentele verandering, en de cumulatieve waarde van incrementele stappen kan een wezenlijke bijdrage leveren aan een werkelijke innovatie. De incrementalisten menen dat dit komt door een ingebakken eigenschap van hun methode: "Incremental decision tend to be remedial: small steps are taken in the "right" direction, or, when it is evident the direction is "wrong", the course is altered." En juist op deze zienswijze heeft Etzioni fundamentele kritiek. Ter evaluatie van elke tussenstap moet je beschikken over criteria. En om te bepalen of die criteria passen bij de globale koers moet je uit het incrementele model stappen. Met andere woorden: staande op de ladder, valt het niet mee om te bepalen of je ladder tegen het juiste huis staat.

Doordat de incrementalisten angstvallig binnen het eigen model blijven, hebben ze volgens Etzioni het kind met het badwater weggegooid. Hij stelt daarom de volgende synthese voor: "(a) high-order, fundamental policy-making processes which set basic directions and (b) incremental processes which prepare for fundamental decisions and work them out after they have been reached." De "Mixed-Scanning Approach" is geboren. De naamgeving van deze methode is afkomstig van het voorbeeld dat Etzioni zelf geeft. Om tijdig orkanen te kunnen waarnemen heb je twee camera's nodig: één camera met een groothoeklens die de gehele hemel bestrijkt, en een tweede camera met een zoomlens die aan de hand van de globale beelden, relevante details van bepaalde gebieden onthult.

Door op gezette tijden een allesomvattende scan te maken, kan worden gecontroleerd of een reeks van kleine stappen het gewenste resultaat heeft opgeleverd in een bredere context. De "high-coverage scan" helpt dus om blinde vlekken te ontdekken. En vanuit het incrementele model kan een stap als een verslechtering worden bestempeld, terwijl vanuit breder perspectief het juist een stap in de goede richting blijkt te zijn. Ter vergelijking: als we ziek zijn grijpen we al snel naar zelfmedicatie om de koorts te verlagen, ons daarbij niet realizerend dat een tijdelijke verhoging van de lichaamstemperatuur juist bedoeld is om het genezingsproces van de patient te bespoedigen - vanuit dat perspectief kan koortverlaging uiteindelijk een contraproductieve maatregel blijken te zijn.

Volgens Etzioni verenigt Mixed-Scanning het beste van twee werelden: fundamentele beslissingen worden genomen vanuit overzicht met weglating van allerlei details en specificaties, besluiten worden effectief voorbereid en uitgevoerd op incrementele wijze, waarbij de evaluatie van de afzonderlijke stappen plaatsvindt op basis van de globale context.

Opmerkelijk is verder hoe "goed" beleid word gedefinieerd. Beleidsmakers dienen waarden en doelen te rangschikken op een ordinale schaal, en de mate waarin het primaire doel wordt gerealiseerd is voor betrokkenen een maatstaf voor een "goed" ontwerp. Zoals Etzioni opmerkt: "As I care very much about one goal and little about the others, if the project does not serve the first goal, it is no good and I do not have to worry about measuring and totalling up whatever other gains it may be providing for my secondary values".

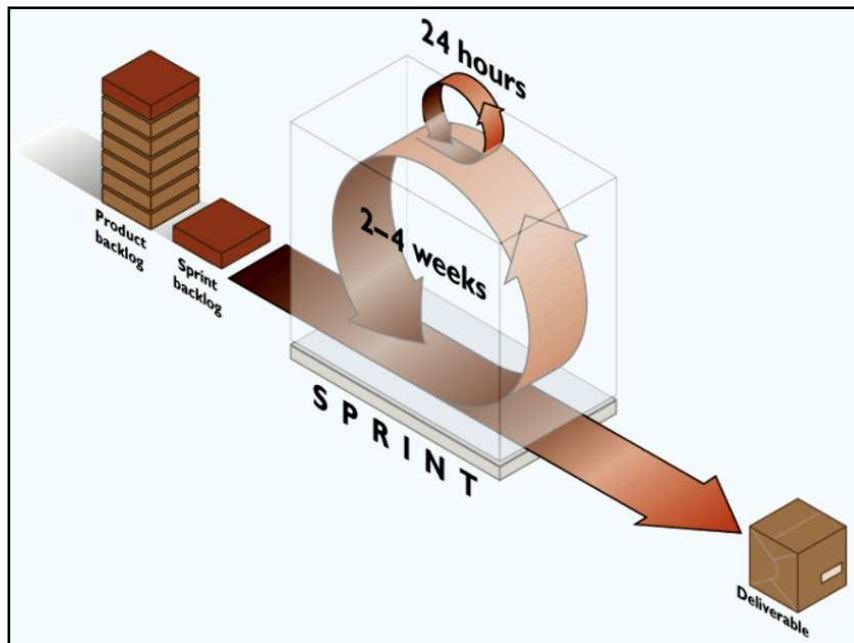
Etzioni geeft vervolgens een praktijkvoorbeeld van deze aanpak, waarmee hij definitief afstand neemt van het beeld van 'de wetenschap van het doormodderen'. Zijn 'Mixed-Scanning Approach' is tot de dag van vandaag vaak de basis voor het nemen van planologische beslissingen. En bij mij is inmiddels het kwartje gevallen dat we hier te maken hebben met het analogon van de Agile Scrum projectmanagement aanpak voor software ontwikkeltrajecten - maar daarover later meer.

De overeenkomst gaat overigens nog een stuk verder: de historische ontwikkeling van beide vakgebieden kent een op z'n minst opmerkelijke analogie. De Rational Comprehensive Methode begon in de naoorlogse jaren behoorlijke slijtplekken te vertonen, net zoals onze Waterval methodiek een tweetal decennia later. Vertrekpunt van beide methoden is de aanname dat je bij aanvang van een project alle relevante variabelen kent, de consequenties kunt overzien, zal (blijven) beschikken over de juiste informatie en middelen (mensen, tijd en geld), en dus de voortgang en de uitkomst van het traject nauwkeurig kan voorspellen. Zowel het ontwerpen van steden als software was in die jaren een rationele, analytische en rechtlijnige aangelegenheid, met als voornaamste doel het afleveren van een betrouwbare blauwdruk voor het nog uit te voeren project. Naast die allesomvattende analyse fase, duikt er trouwens nog een andere overeenkomst op: het betrekken van de toekomstige bewoners/gebruikers bij het maken van de plannen werd niet nodig geacht (...).

In beide domeinen kwam er op het ontstaan van de scheuren in het bastion van de maakbaarheid een opvallend gelijke reactie: de iteratieve werkwijze. Eind jaren 80 kwam Barry Boehm met zijn 'Spiral Model for Software Development': een evolutionaire ontwikkelmethode met als belangrijkste kenmerk de kleine (en dus overzichtelijke) incrementele stappen [3]. Bij voltooiing van elke increment wordt feedback verzameld, waarmee sturing wordt gegeven aan volgende incrementen. Net als bij de Incrementalisten kwam de ivoren toren van het "Grand Up-Front Design" onder vuur te liggen. De nieuwe aanpak werd in hoog tempo opgestuwd in de vaart der volkeren - niet in het minst gehinderd door bedenkers van ontwikkelmethoden (zoals RUP) en leveranciers van geavanceerde ontwikkelomgevingen. Toch bleven vele software ontwikkelaars (of liever gezegd hun managers) toch hardnekkig op zoek naar de heilige graal van de voorspelbaarheid. De onzekerheid veroorzaakt door inherente instabiliteit van requirements, werd massaal ingewisseld voor de schijnzekerheid van top-zware, bureaucratische ontwikkelmodellen. Door alle stroperige procedures viel het bovendien niet mee om nog echt gehoor te geven aan 'The Voice of the Customer'.

De zwaar opgetuigde methodieken bleken in de praktijk steeds minder te voldoen aan de eisen voor het ontwikkelen van de sterk in complexiteit toenemende software-intensieve, multidisciplinaire systemen. De roep om een lichtere en effectievere aanpak werd luider, en mede geïnspireerd door het succes van Lean Manufacturing [4] kwam halverwege de jaren 90 de Agile beweging opzetten [5]. De observatie dat het succes of falen van software projecten wordt bepaald door subjectief, chaotisch en *ogenschiijnlijk* irrationeel gedrag van betrokkenen, werd eindelijk vertaald in een doelmatige, no-nonsense aanpak van ontwikkeltrajecten. De droom van voorspelbaarheid werd omgeruild voor de realiteit van adaptiviteit.

Als we de opkomst van het Agile gedachtengoed leggen naast de tijdlijn van de planologische besluitvorming, dan lijkt de Agile methode te liggen ergens tussen de 'Incrementalist Approach' en 'Mixed-Screening Approach'. Deze vergelijking zich het beste illustreren door de eind jaren '90 in zwang gekomen Agile Scrum methode van project management [6] eens nader onder de loupe te nemen.



Figuur 1: De Agile Scrum project management methode

Wat is er nu kenmerkend en vernieuwend aan Scrum? Om te beginnen wordt ervoor gezorgd dat het project niet meteen op z'n gat ligt als de requirements (ingrijpend) veranderen. Want dat meer dan de helft zal veranderen gedurende de looptijd van het project, daar kun je gif op innemen - dus je kunt maar beter voorbereid zijn. In plaats van proberen de requirements te bevriezen, wordt juist een mechanisme ingebouwd om optimaal te kunnen reageren op de onvermijdelijke veranderingen.

Centraal in dit mechanisme staat de *product backlog*: een lijst met alle zaken die gerealiseerd moeten worden, gerangschikt naar belangrijkheid. Cruciaal is echter dat alleen de user-requirements met de hoogste prioriteit nader worden uitgewerkt tot concrete taken voor het ontwikkelteam. De rest blijft tot nader order even in de koelkast. Dit heeft ingrijpende gevolgen voor de manier van werken. In plaats van *alle* requirements bij aanvang van het project uit te werken in een gedetailleerd, up-front ontwerp, wordt nu een lijst gemaakt met 'user stories', waarvan alleen die met de hoogste prioriteit nader worden gespecificeerd en aansluitend worden gerealiseerd in een time-boxed *sprint*. Gedurende deze periode wordt het team niet lastiggevallen met veranderingen: de overeengekomen requirements voor de lopende sprint zijn fixed! In de tussentijd worden onder de hoede van de *product owner*, gekeken welke zaken er nu bovenaan de product backlog moeten komen te staan, zodat die vervolgens kunnen worden uitgewerkt ten behoeve van de eerstvolgende sprint. Dit zijn meestal user stories die al ergens op de backlog stonden, maar ook nieuwe requirements kunnen het label "high priority" krijgen. In plaats van urgente zaken over de muur te gooien bij het ontwikkelteam, dienen stakeholders zich nu te vervoegen bij de product owner om aanvullende of nieuwe eisen gerealiseerd te krijgen. De wachttijd hiervoor bedraagt maximaal de afgesproken duur van de sprint, dus nooit meer dan 4 weken.

De product backlog fungeert derhalve als een 'brandgang' rond het ontwikkelteam, dat ongestoord en met volle concentratie onder supervisie van de *scrum master* kan werken aan de oplevering van hetgeen is afgesproken. Naast het intensieve onderlinge contact dat elke dag begint met de *scrum meeting*, is het mechanisme van de geprioritiseerde product backlog de voornaamste reden voor de toename in doelmatigheid van teams na de overschakeling op de Scrum methode.

Agile Scrum is hiermee een incrementele werkwijze in optima forma. Er is geen sprake van een uitputtende, rationele a-priori analyse zoals bij de 'Rational Comprehension Method'. En ook de mensen waarvoor je het product maakt, zijn volop betrokken zijn bij de planning en de uitvoering: de product owner staat dicht bij de klant, en soms maakt de opdrachtgever of eindgebruiker zelfs deel uit van het *scrum team*. De aanhangers van 'The Incrementalist Approach' zouden daarover denk ik zeer te spreken zijn, zeker als ze zouden horen dat na afronding van elke increment voor alle betrokkenen een demonstratie wordt gegeven van de door het team gerealiseerde functionaliteit - dus aan feedback en tussentijdse bijsturing geen gebrek!

Ondanks de stap voorwaarts, heb ik een kritische kanttekening bij Scrum. En dit is in de kern vergelijkbaar met de kritiek van Etzioni op het incrementalisme. Een Scrum team is weliswaar in staat om met hoog rendement werkende software op te leveren, maar passen al die deelproducten wel in het grotere geheel, in lijn met de langere termijn business doelstellingen en product roadmap van de stakeholders? Zelfs als de product backlog met veel zorg en vakmanschap wordt samengesteld, geprioritiseerd en beheerd, kunnen er belangrijke zaken zijn die niet als zodanig in de product backlog tot uiting komen. Ik heb het dan over 'cross-cutting concerns' zoals algemene eisen ten aanzien van product-kwaliteit, een volgorde van taken die noodzakelijkerwijze afwijkt van de toegekende prioriteiten, globale eisen ten aanzien van product-line ontwikkeling en/of het effectief omgaan met legacy problematiek.

Het advies van Etzioni aan planologen is om een camera met groothoeklens te gebruiken ter uitbreiding van de incrementele methode met een hogere-orde proces. Om vergelijkbare redenen wil ik een lans breken voor een huwelijk tussen Scrum en Architectuur. Met een relatief simpel, overzichtelijk project zal je prima uit de voeten kunnen met Scrum, maar het is mijn overtuiging dat er meer nodig is als we kijken naar de gestaag toenemende complexiteit en omvang van software systemen zoals je die b.v. aantreft in de apparaten voor de fabricage van halfgeleiders, medische toepassingen of nanotechnologie. In een dergelijke omgeving zal je toch moeten beschikken over een 'kapstok' op een hoger niveau van abstractie, die de Agile Scrum methodiek stevig positioneert op de brug tussen het probeem-domein en het oplossings-domein.

De klaarblijkelijke overeenkomsten tussen het ontwerpen van gebouwen en softwaresystemen hebben al jaren geleden geleid tot het ontstaan van de term 'software architectuur'. Er is zelfs al een heuse polemieek ontstaan tussen de traditionele en 'digitale' architecten over het al dan niet rechtmatig gebruik van de titel 'architect' [7]. Inderdaad vind ik de gangbare functie-omschrijving van de software architect op z'n minst discutabel, en wellicht kan ik een bijdrage aan deze discussie leveren door te stellen dat een architect in ons domein zich eerder zou moeten vergelijken met de planoloog dan met een bouwkundig architect. In ieder geval zou de software architect zich dan niet langer kunnen beperken tot het oplossings domein, en gedwongen worden zich te verplaatsen in de wereld van de toekomstige gebruikers. Leve de groothoeklens!

Terug naar de mogelijke rol van architectuur in Agile Scrum. Deze relatie is op dit moment op z'n minst problematisch te noemen. Misschien terecht, maar weinig flateus is de opmerking dat Scrum de tekortkomingen in de architectuur onbarmharig zal blootleggen [8]. Sommige criticasters verwarren software architectuur nog steeds met een in steen uitgehakte structuur (wellicht vanwege een ongelukkige vergelijking met de bouwwereld), en verwijzen software architectuur naar de reserve-bank [9]. Desondanks denk ik dat Scrum een uitgelezen mogelijkheid is om de meerwaarde van software architectuur te verzilveren, en dat we in ons vakgebied toe moeten groeien naar een 'Mixed Approach' - net zoals de planologen een paar decennia geleden al hebben gedaan. In dit opzicht kunnen we zeker wel wat leren van andere vakgebieden...

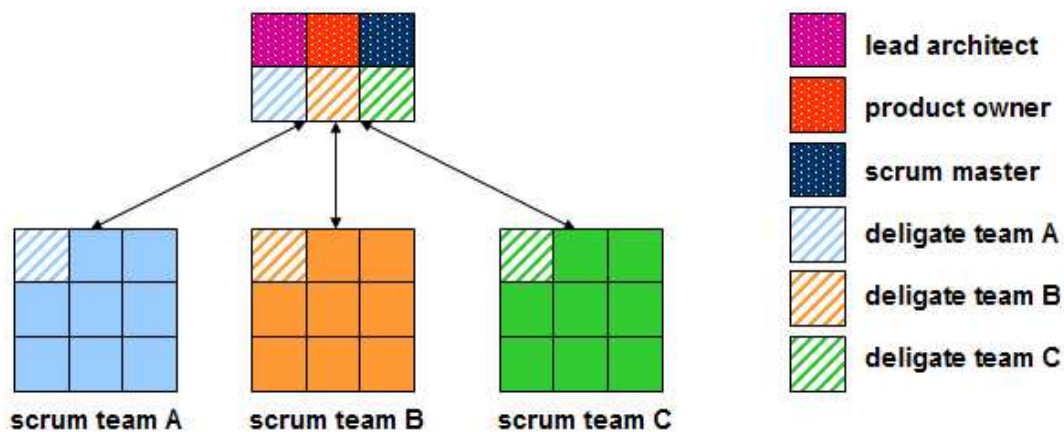
Om nog maar eens een knuppel in het hoenderhok te gooien - een belangrijke, in sommige gevallen zelfs belangrijkste drijfveer voor software architectuur is de set van requirements ten aanzien van product-kwaliteit. Het is onmogelijk, althans zeer kostbaar om kwaliteit in een laat stadium toe te voegen aan een software systeem - dus zal de architect in dialoog met opdrachtgevers en eindgebruikers bij aanvang van het project moeten vaststellen wat de meest belangrijke kwaliteits eisen zijn van de producten die op basis van de architectuur zullen worden ontwikkeld. Een voorbeeld: nog steeds worden de hoge ontwikkelkosten van software veroorzaakt door een tekort aan onderhoudbaarheid, en dat is een eigenschap van software die voor een groot deel wordt bepaald door architecturele beslissingen en mechanismen. Gaat het b.v. om uitbreidbaarheid, dan zal de architect - met het oog op de product roadmap en de ontwikkelingen in de markt - bepalen welke onderdelen van het systeem in hun ontwerp rekening moeten houden met 'veranderbaarheid'. Want uitbreidbaarheid opleggen als generieke eis aan alle onderdelen van een systeem, is vrijwel altijd overbodig en bovendien een bron van fouten en tekortschietende performance. De product backlog geeft in dit voorbeeld een uitgelezen kans voor de architect om in samenspraak met de product owner aan te geven, voor welke backlog items een hoge uitbreidbaarheid is vereist (en voor welke dat helemaal geen eis is). Vervolgens wordt deze eis meegenomen naar de sprint, waar de uitbreidbaarheid als meetbaar criterium onderdeel wordt van de evaluatie van volbrachte taken. Heel praktisch en direct: het is bij het inchecken dan niet meer voldoende dat de module compileert, maar dat de source code ook voldoet aan vooraf gestelde eisen wat betreft uitbreidbaarheid. Hiervoor kan de build-omgeving van het scrum team worden uitgebreid met (automatische) statische en dynamische analyse tools.

Uiteraard kan dit voorbeeld worden uitgebreid met legio ander eisen aangaande product-kwaliteit. Het punt is dat de Scrum methode een uitstekende ingang biedt om deze kwaliteits eisen op een directe en concrete manier in het ontwikkeltraject te introduceren (en zelfs af te dwingen). Door deze 'Mixed Approach' wordt het risico afgewenteld dat een Scrum team weliswaar in kleine effectieve stappen, uiteindelijk toch blijkt af te stevenen op een systeem dat bij aflevering al kampt met legacy problematiek.

Een ander voorbeeld waarbij er sprake is van een omvattend en sturend architectuur principe vinden we in de ontwikkeling van een product line [10]. Elk software platform bevat uiteraard gemeenschappelijke bouwstenen ("common assets"), maar het idee van een schoenendoos met legostukjes die je naar hartelust kunt combineren tot producten blijkt in de praktijk toch een enigszins naïeve gedachte. Voor de realisatie van een portfolio van producten zie je dat het platform een (hopelijk) groot aantal 'basale', universeel bruikbare componenten ter beschikking stelt, maar ook een aantal componenten die bedoeld zijn voor slechts een deel van de producten (of zelfs maar voor één specifiek product), met daartussen een reeks 'parametriseerbare' componenten die configureerbaar zijn voor bepaalde productvarianten.

In welke categorie een nog te ontwikkelen component valt, hangt - uiteraard naast software-technische redenen - in belangrijke mate af van de te ondersteunen portfolio, oftewel de relatie van het software platform met de product roadmap en business cases voor productvarianten. Het moge duidelijk zijn dat dergelijke afwegingen het niveau van de realisatie van afzonderlijke modules te boven gaat. De bewaking van de integriteit van een platform (voornamelijk door strenge controle op afgesproken interfaces) kan niet van binnenuit: hiervoor is een architect nodig die buiten het directe productieproces staat, en de coherentie en 'composability' van het geheel nauwkeurig in de gaten houdt. Hoewel de ontwikkeling van de product line 'assets' uitstekend met behulp van de Agile Scrum methode kan worden verricht, zal er toch behoefte zijn aan een hogere-orde richtlijn om uiteindelijk tot een goed werkende product line te komen. Zomaar ergens beginnen met sprints zou anders wel eens kunnen uitmonden in een lemmingen race.

Hoe ziet een succesvolle combinatie van architectuur en scrum er in de praktijk uit? Om te beginnen dient de software architect nauw samen te werken met de product owner om hogere-orde afwegingen terug te laten komen in de prioritisering van de product backlog. Hiermee wordt een brug geslagen naar de business doelstellingen van zowel de ontwikkelorganisatie als haar klanten. Daarnaast dient de architect invloed uit te oefenen op de uitvoering van elke sprint, b.v. bij de implementatie van overeengekomen eisen ten aanzien van product kwaliteit, of om ervoor te zorgen dat de gemaakte software naadloos aansluit bij aanpalende trajecten, zoals test & integratie, configuration management, etc. Hiertoe werkt de architect nauw samen met de scrum master (en eventueel een afgevaardigde uit het scrum team). Ten slotte is de architect aanwezig bij de scrum meetings zoals sprint review, sprint planning en retrospective, om gerichte feedback en input te kunnen geven voor toekomstige sprint(s). In onderstaand schema is aangegeven hoe een dergelijke samenwerking zou kunnen worden georganiseerd:



Figuur 2: Samenwerking tussen architect en scrum teams

Eigenlijk vormt de architect samen met product owner, scrum master en afgevaardigden uit elk scrum team zelf ook weer een scrum team op een hoger niveau van abstractie en aansturing. Relevante architecturele eisen en beperkingen kunnen worden opgenomen in een *architectuur backlog*, die in dezelfde 'heartbeat' als de scrum teams worden geaddresserd. Voor een project met enige omvang of complexiteit is het sterk aan te raden dat dit 'architectuur scrum team' begint een 'nul-sprint' waarin allerlei belangrijke architecturele zaken worden uitgewerkt voordat de scrum teams van start gaan.

Bij het opschalen van scrum trajecten wordt gebruik gemaakt van '*Scrum of Scrums*', waarbij een 'meta' scrum team de onderlinge coördinatie regelt [11]. Op deze wijze zijn grote ontwikkelprojecten met over de honderd engineers succesvol uitgevoerd met Scrum als project management methode. Uiteraard is deze wijze van opschalen uitstekend te combineren met de hierboven aangeduide inbreng van architecten in het gehele project.

De hier geschetste 'Mixed Approach' met een architectuur team lijkt mij zelfs een noodzaak voor afschoring van ontwikkelactiviteiten [12]. Een vruchtbare multi-site samenwerking volgens de Agile Scrum aanpak houdt in dat ervaren leden van de teams van de verschillende sites deelnemen aan het meta-team, samen met lead-architecten en product owner(s). Dit architectuur team is regelmatig aanwezig op de verschillende sites, en onderhoudt nauw contact met belangrijke stakeholders: te weten opdrachtgever, ontwikkelorganisatie, en toekomstige gebruikers.

Volgens eenzelfde topologie zou multi-disciplinaire samenwerking vormgegeven kunnen worden. De betrokken disciplines (electronica, optica, mechatronica, hardware, etc.) worden dan georganiseerd in scrum teams, waarbij een meta-team onder aanvoering van systeem architecten en product-owners, de afstemming tussen de disciplines regelt. Bij mijn weten is dit nog nooit gedaan, maar het zou wel eens een grote stap voorwaarts kunnen zijn om de inmiddels beruchte integratieproblematiek het hoofd te bieden.

Om diverse redenen ben ik een voorstander van het Agile gedachtegoed, en heb als 'Certified Scrum Master' en bijdrage geleverd aan succesvolle Scrum ontwikkeltrajecten. Hoewel ik vele jaren als architect werkzaam ben geweest in omvangrijke en complexe ontwikkeltrajecten, is het geenszins mijn bedoeling om de slanke en wendbare Scrum methode weer vet te mesten met 'topzware architectuur artefacten'. Integendeel: ik zie een kans om Scrum zoals het is, tot grotere volwassenheid en acceptatie te brengen, en tegelijkertijd een meer Agile software/systeem architectuur proces te introduceren. Juist door de combinatie met een lichtgewicht hogere-orde architectuur proces zie ik een uitgelezen mogelijkheid om een methode zoals Agile Scrum te laten landen in grote ontwikkelprojecten voor complexe, software-intensieve multidisciplinaire systemen.

Erik Philippus

Versie 1.0 - December 2007

## Referenties

- [1] Charles Lindblom, The Science of "Muddling Through", Public Administration Review, Spring 1959
- [2] Amitai Etzioni, Mixed-scanning: "A "Third" Approach to Decision-Making, Public Administration Review, December 1967
- [3] Barry Boehm, "A Spiral Model of Software Development and Enhancement", ACM SIGSOFT Software Engineering Notes, Augustus 1986.
- [4] Takeuchi, H. and I. Nonaka, The New New Product Development Game. Harvard Business Review, 1986
- [5] Mary Poppendieck & Tom Poppendieck, Lean Software Development: An Agile Toolkit, Addison-Wesley Professional, 2003
- [6] Schwaber, K., Scrum Development Process, in OOPSLA Business Object Design and Implementation Workshop, J. Sutherland, et al., Editors. 1997, Springer: London.
- [7] Titel architect taboe in ict - Bouwkundig architecten hebben genoeg van onduidelijkheid, <http://www.computable.nl/artikel.jsp?id=1326925>
- [8] Scrum highlights poor software architecture, Stephan Schwab <http://www.stephan-schwab.com/2007/09/09/1189370079326.html>
- [9] Just-in-Time Architecture: Planning Software in an Uncertain World, William G. Griswold, <http://citeseer.ist.psu.edu/118888.html>
- [10] Software Product Lines, Software Engineering Institute, <http://www.sei.cmu.edu/productlines/>
- [11] Jeff Sutherland, Future of Scrum: Parallel Pipelining of Sprints in Complex Projects, <http://jeffsutherland.com/scrum/RP10-SutherlandFutureOfScrumAgile2005.pdf>
- [12] Interview with Jeff Sutherland: Offshore outsourcing with Scrum, [http://www.softhouse.se/Uploades/Intervju\\_070206.pdf](http://www.softhouse.se/Uploades/Intervju_070206.pdf)