(**Arjen Uittenbogaard** in: Automatiseringsgids, 8 november 2002)

Softwareontwikkeling vraagt improvisatietalent

Op het gebied van softwareontwikkeling vindt een cultuuromslag plaats. Steeds meer projecten werken op een incrementele, iteratieve en interactieve manier. Niet elk project is in staat dit gedachtegoed in de praktijk te brengen. Arjen Uittenbogaard laat aan de hand van acht stellingen zien wat een succesvolle route kan zijn.

Begin jaren negentig is RAD, Rapid Application Development, enige tijd populair geweest. De methode had als belangrijk kenmerk dat de gebruiker en bouwer samen op basis van prototypes incrementeel, dat wil zeggen stukje-bij-beetje, een systeem in elkaar zetten. Helaas was het denken over architectuur nog niet ver gevorderd. En ook de 4GL-tools hielpen nog niet mee om uitbreidbare systemen te bouwen. Incrementeel willen bouwen terwijl de applicaties slecht uitbreidbaar waren, dat moest wel botsen. En inderdaad: RAD verwerd tot ‘rapidly achieving disaster’ en was vervolgens enige jaren ‘not done’.
Gelukkig schreed het architectuurdenken voort en werden de tools beter. Techniek hoeft geen belemmering meer te zijn voor het succesvol uitvoeren van een RAD-project. En zo herleefde RAD in diverse gedaantes, waaronder DSDM (Dynamic System Development Method) en XP (eXtreme Programming). Prominente vertegenwoordigers van deze nieuwe methodes hebben in een ‘agile manifesto’ (www.agilealliance.org) geformuleerd wat hen bindt. Kort gezegd: moderne, RAD-achtige of agile softwareontwikkeling kenmerkt zich door een incrementele, iteratieve en interactieve benadering.
Meer en meer worden projecten op deze wijze uitgevoerd. In elk geval wordt meer en meer lippendienst bewezen aan de basisgedachten van incrementele, iteratieve en interactieve systeemontwikkeling. Helaas sneuvelen nogal wat van deze projecten omdat het in praktijk brengen van dit gedachtegoed ander gedrag vraagt van alle betrokkenen. Zonder overdrijving kan worden gesproken over de noodzaak van een cultuuromslag. Deze omslag kan in acht stellingen worden samengevat.

De watervalmethode heeft niet afgedaan
Voordat RAD ten tonele verscheen was het lineaire of watervalproces de enige formele projectaanpak. Het is niet zo dat met de komst van RAD de watervalmethode heeft afgedaan. Er zijn teams, afdelingen en bedrijven waar een lineaire aanpak met succes wordt toegepast. Het is dan ook goed om stil te staan bij de vraag waarom voor een andere benadering gekozen zou worden. Het klinkt zo vanzelfsprekend, maar RAD-achtig gaan werken alleen maar omdat het nieuw is, is onverstandig.
Het DSDM-consortium biedt een handvat met de formulering van kenmerken die projecten geschikter maken voor een DSDM-benadering. Projecten onder tijdsdruk, waarvan de eisen op voorhand niet duidelijk zijn of die aan verandering onderhevig zijn, lenen zich goed voor een moderne benadering. Ook zijn internetapplicaties in het algemeen geschikter om zo gebouwd te worden dan bijvoorbeeld batchprocessen.

Iteratief werken met activiteitenplanning mislukt
Een traditionele projectaanpak is gebaseerd op activiteiten die worden toegewezen aan personen. De werkzaamheden worden vooraf opgedeeld in kleinere activiteiten die zo goed mogelijk worden toebedeeld aan de teamleden. Elk teamlid is ervoor verantwoordelijk zijn activiteiten binnen de daarvoor gestelde tijd te verrichten. Om tal van redenen vraagt deze manier van werken om moeilijkheden. Zo vereist het a priori een diepgaande kennis van het te bouwen systeem om tot een adequate activiteitenopdeling te komen. Ook leidt het toewijzen van activiteiten aan mensen tot suboptimale oplossingen omdat een ieder zijn activiteit zo goed mogelijk uitvoert en daarbij maar al te gemakkelijk het grote geheel uit het oog verliest. Verder zal een project slechts bij hoge uitzondering eerder klaar zijn dan gepland: uitvoering van een taak duurt namelijk altijd minstens de tijd die ervoor beschikbaar is gesteld.
Een RAD-achtig project daarentegen, is productgebaseerd. Alles is erop gericht om producten op te leveren. Dat kunnen documenten zijn (een belangrijk product in het grootste deel van traditionele trajecten), maar liefst zijn het werkende deelsystemen. Het is aan het team om dat voor elkaar te krijgen. Het team verdeelt gaandeweg de taken die nodig zijn om de op te leveren functionaliteit te realiseren.

Hedendaagse projecten eisen slagvaardige teams
Een gevolg van het bovenstaande is dat een iteratief, incrementeel en interactief project vraagt om een slagvaardig team. Dit moet bevoegdheden hebben om keuzes te maken. Het moet, zogezegd, ‘empowered’ zijn. De kortcyclische werkwijze in een project wordt gehinderd wanneer bij elke keuze toestemming nodig is vanuit de hiërarchische lijn in de organisatie.
Er zijn voorbeelden van projecten die mislukten omdat de vertegenwoordigers van de (eind)gebruikers geen mandaat hadden om prioriteiten te stellen. Ook zijn projecten mislukt waar de vertegenwoordigers dat mandaat op papier wel hadden, maar door hun achterban of management regelmatig werden teruggefloten. En ten slotte mislukken projecten vanwege gebrek aan daadkracht van de betrokkenen. Personen die een mandaat hebben gekregen, moeten ook knopen durven doorhakken.

Traditionele standaarden voldoen niet

Ontwikkelingsorganisaties hebben in de loop der jaren standaarden ingevoerd voor QA, projectmanagement, testen, configuratiebeheer et cetera. Deze standaarden bieden imposante archieven met raamwerkdocumenten die ‘zo uit de kast getrokken kunnen worden’. De gedachte is dat dit de kwaliteit ten goede komt. De hamvraag is echter: De kwaliteit van wat? Lang niet altijd komen de standaarden de kwaliteit van het op te leveren systeem ten goede. Een document kan bedoeld zijn om de kwaliteit van het onderhoud te verbeteren, om testen te ondersteunen of om het versiebeheer te verbeteren. Allemaal procesonderdelen die met standaarden kunnen worden verbeterd.
Een fundamenteel gebrek van veel standaardisatie-inspanningen is dat niet wordt onderkend dat verschillende projecten verschillende prioriteiten stellen. Het is dan ook een slecht idee om te proberen op voorhand voor alle varianten templates te maken. Een studieproject, bijvoorbeeld, van een kwaliteitsafdeling om het QA-systeem uit te breiden met documenten voor een agile ontwikkelingsstraat, heeft al snel een hoog ivorentorengehalte.
Het is aan te bevelen de eerste moderne projecten in isolatie uit te voeren. Het team bevindt zich als het ware op een eiland waarop het zich niets hoeft aan te trekken van de traditionele standaarden en zich volledig kan wijden aan de implementatie van de nieuwe methode. Methodes als DSDM en XP bieden uitgebreide werkwijzen om de kwaliteit van een systeem te garanderen (inclusief onderhoudbaarheid, testbaarheid enzovoort). Het kan dan ook geen kwaad om op het eiland te proberen roomser te zijn dan de paus. Een halfslachtige implementatie van een dergelijke methode is vragen om moeilijkheden.
Het is in dit verband goed erop te wijzen dat XP bijvoorbeeld van ontwikkelaars een hoge mate van discipline, onderlinge communicatie en vertrouwen vraagt. Het is dan ook een veeg teken als programmeurs XP iets te gretig lijken te omarmen. XP is allesbehalve een ‘license to hack’. Het vraagt meer discipline dan menig programmeur gewend is.

Te veel tijd en te veel geld zijn funest
Het lijkt een luxe, maar blijkt vaak een valkuil: een agile project waarvoor meer dan genoeg tijd en geld is vrijgemaakt. Dit soort projecten gedijt namelijk onder een zekere mate van druk. Ze vragen een ‘sense of urgency’, niet in het minst omdat zonder een gevoelde noodzaak het vrijmaken van gebruikers voor participatie in het project een lastige opgave is. Om het nog scherper te zeggen: dit soort projecten mislukt als de klant niet zit te springen om een oplossing.
Een van de basistechnieken van hedendaagse methodes is timeboxing: het periodiek vaststellen van een deadline waarop de klant gegarandeerd een nuttig (deel)systeem krijgt opgeleverd. Over de lengte van een timebox verschillen de methodes van mening. Maar dat het niet meer dan enkele maanden moet zijn, daarover is iedereen het eens. Te veel tijd maakt schatten, prioriteren en productgericht werken vrijwel onmogelijk.
Te veel geld kan ertoe leiden dat gepoogd wordt zekerheid te kopen: nog een studiegroep vrijmaken, nog een analyse uitvoeren, nog een adviseur inhuren. Terwijl een incrementele benadering probeert zekerheid te krijgen door zo snel mogelijk stukken systeem te bouwen. De stelregel is: ‘the proof of the pudding is in the eating’. Werkt een gekozen oplossing niet dan blijkt dat snel genoeg en is er des te meer tijd om die keuze terug te draaien.

Niemand blijft buiten schot
Moderne softwareontwikkeling is niet een speeltje van de ontwikkelaars, maar vraagt een andere houding van alle betrokkenen. Incrementeel werken impliceert dat niet alleen het te bouwen systeem incrementeel wordt ontwikkeld. Als incrementeel wordt ontwikkeld, wordt de ‘business case’ van het project een levend document. Elke (deel)oplevering biedt de gelegenheid om de kosten-batenanalyse te herzien en, indien nodig, bij te stellen.
Als de software wordt ontwikkeld door een externe partij, wordt ook het contract een incrementele aangelegenheid. Wellicht wordt voor het project een raamwerkcontract afgesloten dat per timebox wordt verlengd. In dit kader kan worden nagedacht over bijvoorbeeld bonus- malusafspraken. In de DSDM- wereld staat dit idee van incrementele contracten sterk in de belangstelling. Juridische afdelingen van grote bedrijven moeten erg wennen aan dit idee. Verkopers van de aanbiedende partij trouwens ook: nu kan niet meer in een keer het hele project worden verkocht, maar is er na elke timebox het risico dat de klant besluit het project te stoppen. In een agile project is dit een realistisch scenario. Als de klant na, zeg, een halfjaar besluit het project te stoppen dan is er, dankzij de productgerichte werkwijze, een werkend (deel)systeem. In een traditioneel project waren op dat moment hoogstens wat documenten geproduceerd.
Ten slotte wordt ook de planning incrementeel. Vooraf wordt weliswaar zo goed mogelijk ingeschat hoe deze eruit gaat zien, maar elke timebox weer wordt gekeken of deze schatting nog realistisch is.

Open communicatie is geen open deur
DSDM formuleert negen principes of succesfactoren voor projecten. De negende vraagt van alle betrokkenen in het project een coöperatieve houding. Alistair Cockburn vergelijkt in zijn boek ‘Agile Software Development’ een modern project met een coöperatief spel: iedereen moet samenwerken om het team te laten slagen. Dit in tegenstelling tot competitieve spelen met een winnaar ten koste van verliezers. Een voorbeeld van een dergelijk coöperatief spel is improvisatietoneel. Improviseren is zelfs meer dan een metafoor, het kan gebruikt worden als werkvorm. Het zou goed zijn om bij de kick-off van een project een improvisatieworkshop te organiseren. Op deze wijze wordt geoefend met vaardigheden als samenwerken,

fouten durven maken en omgaan met veranderende en onverwachte situaties: kenmerkende vaardigheden die van de teamleden worden gevraagd.
Het is verrassend moeilijk een cultuur te realiseren waarin samenwerking en open communicatie hoog in het vaandel staan. Dit zou wel eens de grootste uitdaging kunnen zijn voor een hedendaagse projectleider. Ter illustratie twee voorbeelden van gangbare praktijken die hiermee op gespannen voet staan.
Ten eerste: documentatie voor planning, voortgangsmonitoring, risicomanagement enzovoort, elektronisch opgeslagen in een database onder zorgvuldig versiebeheer.
Stel, een programmeur loopt tegen een probleem aan. Hem is op het hart gedrukt dit direct te rapporteren in het risicodocument. Hij weet dat hij op zoek moet naar de database dat het document bevat, de goede versie moet vinden en deze moet uitchecken uit het systeem. Geen grote barrières. Maar toch. Het is maar een klein probleem. Is het gek dat hij besluit het te onthouden tot de projectleider langskomt? Is het de programmeur te verwijten als hij het tegen die tijd is vergeten? Vergelijk deze situatie met een XP-project. Hier hangen flip-overvellen aan de muur met daarop de planning, voortgang en risico's. Iedereen kan de informatie zonder enige drempel tot zich nemen en nieuwe punten toevoegen.
Ten tweede: tweewekelijks voortgangsoverleg tussen projectleider en architect. Dit heeft twee negatieve kanten. Twee weken is in een project met korte timeboxes een relatief lange periode waarin veel kan veranderen. Voortgangsoverleg moet frequenter plaatsvinden. Bovendien: een overleg waarbij maar een paar teamleden aanwezig zijn, sluit de andere teamleden buiten en mist hun inbreng. Agileprojecten kennen een dagelijks teamoverleg. Niet een urendurende vergadering, maar een korte sessie waarin iedereen vertelt wat hij gisteren heeft gedaan, of zich problemen hebben voorgedaan, en wat hij vandaag gaat doen. Door dit overleg consequent elke dag te voeren wordt de noodzaak voor lange bijeenkomsten vanzelf minder.

De waterval kruipt waar hij niet gaan kan
Succesvolle moderne projecten werden vrijwel zonder uitzondering begeleid door een ervaren mentor. Een belangrijke taak van de mentor is om het team iteratief te leren werken. Afstappen van de lineaire werkwijze is een lastig proces dat een lange adem vraagt. De voordelen van iteratief werken zijn rationeel goed te bevatten. Het is echter meer dan eens voorgekomen dat teams die onder begeleiding al maanden iteratief en incrementeel aan de slag waren, terugvielen in oude patronen zodra de mentor niet meer elke dag ter plekke was. De ontwerpers ontwierpen weer dagenlang zonder een analist, klant of programmeur te zien. Evenzo voor de andere groepen. Hoe dit komt is een boeiende vraag. Is het inherent menselijk om je terug te trekken op je oude terrein? Is het vanwege de soms tientallen jaren ervaring die is opgedaan met een watervalaanpak? Hoe dan ook: de projectleider en de mentor moeten voor dit probleem op hun hoede zijn.

Arjen Uittenbogaard is agile teambuilder en verzorgt workshops en presentaties over het gebruik van improvisatietoneel in softwareontwikkelingsprojecten.(http://www. uitje.org)