Scrum

ICT-projecten op tijd en binnen budget!

Een belangrijk kenmerk van veel IT-projecten is dat ze mislukken. Enerzijds is dat oud nieuws, aan de andere kant nog steeds een bijzonder actueel probleem. Geslaagde IT-projecten voldoen aan drie criteria: het systeem functioneert zoals de opdrachtgever en de leverancier dat overeen kwamen, het project wordt op tijd afgeleverd en de kosten blijven binnen het budget. Bij veel projecten is dat niet het geval. Vooral het overschrijden van tijd en budget is een veel voorkomend probleem. Niet voor niets worden dergelijke uitlopende projecten in het jargon runaway projects genoemd. Over waarom IT-projecten zo vaak mislukken zijn boekenkasten volgeschreven en menig consultant verdient er zijn dagelijkse boterham mee.

Scrum lost het op. Triodor hanteert bij de ontwikkeling van software de scrum-methodiek. In deze whitepaper lichten we toe hoe dat zit en hoe het werkt.

Triodor
software

INHOUDSOPGAVE

Table of Contents

Generieke valkuilen bij IT-projecten

Er zijn legio redenen waarom een IT-project mislukt. Denk daarbij aan een gebrekkig commitment van de leidinggevende of onvoldoende betrokkenheid van de eindgebruiker bij de ontwikkeling, een onrealistische inschatting van benodigde tijd of resources, onduidelijke doelen en doelstellingen die in de loop van het project veranderen en onvoldoende testen.

Weinig betrokken

Hoewel een gebrekkige betrokkenheid van de eindgebruiker bij de ontwikkeling fataal is voor een IT-project, komt dit nog steeds vaak voor. Het is een hardnekkige erfenis uit het recente verleden van de IT-functie waarbij software-ontwikkeling gekenmerkt werd door een sterke technologische en aanbodgestuurde top-downbenadering van de ontwikkeling. Het op afstand houden van de eindgebruiker voor en tijdens het ontwikkelproces kent echter verschillende gevaren. Niet alleen is de kans groot dat ontwerp, functionaliteit en interface van het eindproduct niet aansluiten op de wensen van de gebruiker en de eisen die de praktijk stelt aan een applicatie, er is evenmin een `buy-in` door de gebruiker. Het systeem wordt de gebruiker opgelegd en roept weerstand op.

Slechte planning

Veel projectgerelateerde problemen zijn voorts terug te voeren op *onrealistische inschattingen* van de vereiste tijd en resources. Zo begaat nog steeds menig project manager de fout de *time on task* gelijk te stellen aan *duration* (time on task is de *minimale* tijd die vereist is om een bepaalde taak te vervullen, terwijl duration de tijd is die een taak in beslag neemt *inclusief* de onvermijdelijke onderbrekingen). Kortom: projecten kennen vaak te weinig *slack*. Bij grotere projecten bestaat echter juist het gevaar dat er weer *te veel* tijd voor wordt uitgetrokken. Te lange tijdspannes kunnen ertoe leiden dat systemen geleverd worden voor producten of diensten die niet langer in gebruik zijn of prioriteit genieten. Doordat niets zo veranderlijk is als een business, is het niet realistisch om te verwachten dat requirements onveranderlijk blijven gedurende de loop van een project. Dit pleit dan ook voor kortere tijdspannes en een gefaseerde benadering van systeembouw – de kans dat verandering de ontwikkeling radicaal beïnvloedt, wordt daarmee kleiner.

Moving targets

Een ander onvermijdelijk aspect van een IT-project is het tussentijds veranderen van doelstellingen door de opdrachtgever. Waardoor het project vaak groter wordt, langer duurt en daarmee ook duurder wordt. Twee begrippen zijn in deze relevant: *scope creep* en *feature creep*. Scope creep refereert aan ongecontroleerde en onverwachte wijzigingen in gebruikersvereisten en *requirements* terwijl het project nog voortduurt. Een klassiek voorbeeld hiervan is het systeem dat klantdata moet opslaan, maar waarvan de opdrachtgever *en passant* besluit dat het systeem ook de klantfacturering moet bevatten en, o ja, die facturering ook nog eens online afgehandeld moet kunnen worden. Onnodig te zeggen dat het onmogelijk is om al functionaliteiten binnen de afgesproken tijd en budget af te leveren.

Feature creep heeft daarentegen betrekking op de toevoeging van features aan het systeem vanuit de foutieve aanname dat een kleine feature niet of nauwelijks van invloed is op de kosten of het tijdschema.

Acceptance tests

Onvoldoende testen ten slotte is de hekkensluiter van de faalfactoren. Weliswaar zullen ontwikkelaars gedurende een project veelvuldig hun producten en componenten ervan testen, de lakmoesproef wordt natuurlijk gevormd door de testen die de eindgebruikers moeten uitvoeren, de zogeheten *acceptance tests* die duidelijk moeten maken of een systeem wel aan de eisen van de business voldoet. Acceptance tests slagen er echter vaak niet in om fouten te identificeren voordat het systeem *live* gaat. Immers, het systeem is vaak niet eerder methodisch getest, het gaat meestal om moeilijk te testen requirements en er is te weinig tijd om grondig te testen doordat het systeem sowieso te laat is opgeleverd.

Kortom: veel falen is terug te voeren op communicatie, hoe het projectmanagement is vormgegeven, op het niet kunnen omgaan met onzekerheid en op inflexibiliteit. Logischerwijs moeten oplossingen dus in die richting gezocht worden.

Traditioneel IT-projectmanagement: watervalmethode

Een van de oudste en meest gebruikte methoden waarmee software ontwikkeld wordt is de zogeheten watervalmethode. Binnen dit proces loopt de ontwikkeling als een waterval regelmatig vloeiend naar beneden. De methode werd ontwikkeld door informaticabedrijven die daarmee meer grip hoopten te krijgen op grote, onoverzichtelijke projecten. Het model is geënt op de traditionele, gefaseerde manier waarop grootschalige constructiebouwprojecten hun beslag krijgen - eerst wordt fase 1 afgerond, pas daarna wordt met fase 2 begonnen. Het watervalmodel bestaat uit 7 fasen.

- **Definitiestudie/analyse**. Er wordt onderzoek gedaan naar en gebrainstormd over de software om helder te krijgen wat het doel is van de software;
- **Basisontwerp**. Er wordt duidelijker uitgewerkt wat er tijdens de eerste fase naar boven is gekomen. In deze fase worden de wensen van de klant op papier gezet en wordt al gedacht aan de vorm van het programma. In deze fase wordt ook vastgelegd *wat* het op te leveren systeem moet doen;
- **Technisch ontwerp/detailontwerp**. Aan de hand van het basisontwerp wordt er een werkelijk programma uitgedacht. In deze fase wordt vastgelegd *hoe* de in het basisontwerp vastgelegde functionaliteit gerealiseerd gaat worden. Nu vindt ook een onderverdeling plaats in technische eenheden zoals programma's, modules en functies;
- **Bouw/implementatie**. Hier wordt de broncode van de programma's geschreven;
- **Testen**. Er wordt gecontroleerd of de software conform de ontwerpen is gebouwd. Ook kunnen er in deze fase fouten boven water komen die al in eerdere stadia gemaakt zijn;
- **Integratie**. Het systeem is klaar en getest. Toch zal het nog in het bedrijf in gebruik genomen moeten worden. Ook dat wordt in deze fase gedaan;
- **Beheer en onderhoud**. Om ervoor te zorgen dat het systeem blijft functioneren, wordt er periodiek onderhoud verricht.

Voordelen

Het watervalmodel kent verschillende belangrijke voordelen. Zo kost het weinig tijd en inspanning om fouten die in het begin van het project ontdekt worden, te herstellen. Elke fase wordt eerst afgesloten voordat met een volgende wordt begonnen en er wordt ervan uitgegaan dat elke fase foutloos wordt afgesloten. Een ander pluspunt van de methode is dat er, in tegenstelling tot nieuwere ontwikkelmethoden, uitgebreid gedocumenteerd wordt - niet onbelangrijk bij de overdracht van het project aan nieuwe mensen en in het kader van kennismanagement. Door de gefaseerdheid kan eenvoudig van mijlpalen gebruik

gemaakt worden om de voortgang van het project in te schatten. Bovendien is de watervalmethode erg bekend en hebben veel mensen er ervaring mee. Ten slotte worden frequent gedeelten van het softwareproduct opgeleverd, iets dat zowel de klant als het ontwikkelteam vertrouwen geeft.

Nadelen

De watervalmethode heeft echter ook evidente nadelen. Een van die nadelen is inflexibiliteit. Zo komt het regelmatig voor dat tijdens het ontwikkeltraject de opdrachtgever iets anders wil. En dat betekent dat de vereisten gewijzigd moeten worden. De watervalmethode kent de daarvoor benodigde flexibiliteit echter niet: de methode gaat immers uit van ongewijzigde vereisten. Verandert er in een bouwfase iets aan de vereisten, dan zullen verschillende voorgaande fasen opnieuw doorlopen moeten worden. Een tweede nadeel is dat het moeilijk is om de benodigde tijd en kosten in te schatten doordat de fasen in kwestie vaak zeer omvangrijk zijn. De waterval is voorts een sterk tijdconsumerende methode. Binnen het project zijn de verschillende teamleden namelijk vaak gespecialiseerd en *dedicated* aan een bepaalde activiteit of een fase. Zijn bijvoorbeeld ontwerpers, die alleen in de eerste fase actief zijn, bezig met het *fine tunen* van het ontwerp, dan zullen de bouwers moeten wachten tot de eerste fase geheel is afgesloten. Een andere handicap is dat het testen pas in één van de laatste fasen van het project plaatsvindt. Bij veel andere software-ontwikkelingsmethoden wordt getest zodra er één bepaald deelproduct klaar is en ook wordt op het laatst een integratietest gedaan. Doordat er ten slotte zo veel nadruk wordt gelegd op documentatie, is de watervalmethode niet efficiënt voor kleinere projecten. Qua documentatie zit er dan te veel werk om het project zelf heen.

Behendig

De inherente beperkingen van ‘zwaargewicht’-methoden zoals detailgestuurde en sterk gereguleerde watervalmodellen en de toenemende vraag naar flexibiliteit in software-ontwikkeling als gevolg van een steeds dynamischer en volatieler wordende businessomgeving, leidde in de mid-jaren negentig tot de komst van een nieuwe manier van ontwikkelen: *agile.* Agile is het Engelse woord voor ‘behendig’ of ‘lenig’ en agile-methoden gelden als een aantrekkelijk alternatief voor traditionele starre praktijken.

Agile – het antwoord op inflexibiliteit

De meeste agile-methoden proberen risico's te verkleinen door software te ontwikkelen in korte, overzichtelijke perioden, de zogeheten *timeboxes of iteraties*. Elke iteratie is als het ware een soort miniatuurproject en omvat alle noodzakelijke componenten die voor een project noodzakelijk zijn: analyse, ontwerp, testen en documentatie. En hoewel niet elke iteratie genoeg oplevert om een eindproduct vrij te geven, is het wel de bedoeling van een agile-project om na iedere iteratie iets bruikbaars voor de markt af te leveren.

Agile-projecten berusten op vier principes en waarden waarmee ze zich onderscheiden van de conventionele ontwikkelbenaderingen:

- **Interacties en individuen** in plaats van processen en tools;
- **Werkende software** in plaats van duidelijke documentatie;
- **Samenwerking met de klant** in plaats van contractuele onderhandelingen;
- **Reageren op veranderingen** in plaats van het volgen van een dichtgetimmerd plan

Communicatie

De nadruk bij agile-methoden ligt dus op directe communicatie, bij voorkeur persoonlijk contact, in plaats van geschreven verslaglegging. Verreweg de meeste agile-teams zijn gehuisvest op één locatie, een zogenaamde bullpen, om de lijnen zo kort mogelijk te houden en de communicatie verder te stroomlijnen. Waar mogelijk zijn alle mensen die voor een project relevant zijn in zo`n team ondergebracht.

Software werkt

Bij agile-methoden wordt de voortgang afgemeten aan de hand van werkende producten of prototypes en wordt er, vergeleken met andere methoden, erg weinig geschreven documentatie geproduceerd. Dit heeft geleid tot kritiek als zouden agile-methoden ongedisciplineerd zijn. Agile-methoden worden dan ook nog steeds gekarakteriseerd als de tegenpool van *geplande* en *gedisciplineerde* methoden. Het is echter beter om te spreken van *adaptieve* en *voorschrijvende* methoden. Adaptieve methoden zijn erop gericht om zich snel aan te passen aan een veranderende werkelijkheid. Een adaptief team past zich dus snel aan, maar is niet in staat om exact te beschrijven wat er in de toekomst staat te gebeuren en wat het daarmee gaat doen. Adaptieve (agile) methoden staan tegenover voorschrijvende methoden. Deze zijn er immers op gericht om een toekomst tot in detail te plannen.

Teams die deze methode hanteren kunnen precies aangeven welke resultaten en taken gepland staan voor de gehele periode van de ontwikkeling. Dergelijk teams hebben daarentegen grote moeite om tussentijds de koers te verleggen.

In tegenstelling tot de loggere watervalmethode leveren agile-methoden elke zoveel weken uitontwikkelde en geteste onderdelen, alhoewel dit telkens kleine delen van het geheel zijn. Het accent ligt erop om zo snel mogelijk de kleinst mogelijke functionele onderdelen te leveren, en die voortdurend te verbeteren en uit te breiden. Er bestaan uiteenlopende Agile-methoden en hoewel ze in de praktijk verschillen, hebben ze een aantal karakteristieken gemeenschappelijk zoals iteratieve ontwikkeling, een focus op interactie, communicatie en een beperkt gebruik van procedures en hulpmiddelen die een beroep doen op resources.

Succesfactoren

In hoeverre de toepassing van een agile-methode succesvol is, hangt af van verschillende factoren. Vanuit een productperspectief zijn agile-methoden geschikter naarmate eisen nog niet vast omschreven en veranderlijk zijn, ze zijn in principe minder geschikt voor systemen die aan kritische eisen moeten voldoen zoals betrouwbaarheid en veiligheid. Vanuit een organisatorische invalshoek kan de geschiktheid van een agile-traject afgemeten worden aan de hand van cultuur, mensen en communicatie. In deze gelden er vijf succesfactoren: zo moet de cultuur van de organisatie openstaan voor discussie en onderhandeling; moeten mensen vertrouwd worden; gaat het om minder, maar competentere mensen; moeten organisaties beslissingen accepteren die de ontwikkelaars nemen en moeten de organisaties een omgeving hebben waarin snelle communicatie tussen teamleden mogelijk is. De belangrijkste factor evenwel is de omvang van een project. Immers, de persoonlijke communicatie neemt af naarmate een project groter wordt. Agile-methoden zijn dan ook geschikter voor kleinere projecten van hooguit twintig tot veertig mensen.

Scrum: iteratief zoeken naar de best practice

Een veel gebruikte agile-methode is Scrum. Scrum is een term die ontleend is aan rugby. Een scrum is een manier van spelhervatting na een kleine overtreding, waarbij een groep spelers de tegenpartij probeert weg te duwen om de bal te verkrijgen en deze al duwend naar de overkant van het veld te krijgen. De term werd als eerste gebruikt door de Japanse onderzoekers Nonaka en Takeuchi die aantoonden dat projecten met kleine cross-functionele teams historisch gezien de beste resultaten opleveren.

Waar bij de watervalmethode elke fase zijn experts heeft die hun taken uitvoeren en het resultaat overdragen naar de experts in een volgende fase, daar worden bij Scrum de experts uit de verschillende fasen in één team bij elkaar gezet. Het doel van de Scrum-methode is vijfledig:

- **Verhogen van de effectiviteit van het team;**
- **Het bewaken van de vooruitgang van het team;**
- **Het oplossen van blokkades;**
- **Het bewaken van de projectvoortgang;**
- **Het in kaart brengen en minimaliseren van risico`s**

Kort samengevat komt Scrum erop neer dat de wensen van de opdrachtgever in kleine modules wordt onderverdeeld die binnen een redelijk welomschreven tijdsbestek een voor de opdrachtgever tastbaar resultaat opleveren. Een dergelijke set requirements wordt dan vastgelegd in een sprint, een tijdsbestek van twee tot maximaal vier weken. Binnen zo`n sprint wordt een product of productmodule ontworpen, gecodeerd en getest.

Karakteristieken van Scrum

De Scrum-methode heeft verschillende karakteristieken. Zo wordt het project uitgevoerd door zelforganiserende teams die bestaan uit een Product Owner, de ScrumMaster en het team.

Het project bestaat uit een verzameling Sprints en de requirements voor elke Sprint worden vastgelegd in ene zogeheten Product Backlog. De methode is niet voorschrijvend: er wordt geen gebruik gemaakt van specifieke engineering practices.

In plaats daarvan worden generatieve regels gehanteerd die een agile omgeving creëren waarbinnen projecten worden afgeleverd.

Product Owner

Een typisch Scrum-team bestaat uit drie partijen: de Product Owner, de ScrumMaster en het team. De Product Owner is degene die de productfeatures omschrijft en die besluit over de product release en de inhoud van het product. Hij is degene die verantwoordelijk is voor de Return on Investment van het product. Deze marktgedreven benadering betekent dat de Product Owner de productfeatures prioriteert aan de hand van hun marktwaarde. Bij elke iteratie, na afloop van elke Sprint, is hij het die features en prioriteiten aanpast en uiteindelijk het geleverde product goedkeurt dan wel afkeurt.

ScrumMaster

De ScrumMaster daarentegen vertegenwoordigt het management van het project. Hij (of zij) is verantwoordelijk voor het implementeren, toepassen en bewaken van de Scrumwaarden, -principes en –praktijk. Een belangrijke functie is het wegnemen van obstakels en ervoor zorgen dat het team optimaal kan functioneren en maximaal productief is. Dit betekent niet alleen dat hij het team afschermt van externe interventies – niet in de laatste plaats door de Product Owner – maar ook dat het team intern goed functioneert. Het team bestaat immers uit meerdere deelnemers met allemaal een eigen expertise, afkomstig uit verschillende disciplines en dus vaak met uiteenlopende zienswijzen. Naast inhoudelijke kennis is *people management* dan ook en niet onbelangrijke competentie van de ScrumMaster.

Team

Dan het team. Een typisch Scrum-team bestaat uit 5 tot 9 teamleden. Zoals gezegd is het een crossfunctioneel team waarbij de teamleden uit verschillende disciplines afkomstig zijn: ontwerpers, programmeurs, testers, experts op het gebied van gebruikerservaring etcetera. Het is in de regel een dedicated team waarbij de leden fulltime op een klus zitten hoewel er uitzonderingen zijn – bijvoorbeeld de database administrator.

Een ander kenmerk van de teams is dat ze zelforganiserend zijn en dat het de competenties zijn die tellen en niet de titels of de organisationele hiërarchie. Een laatste eigenschap is dat het teamlidmaatschap niet verandert *tijdens* een Sprint, maar alleen *tussen* de Sprints door.

De Sprint

Elke Sprint wordt voorafgegaan door een Sprint planning meeting. Z`n meeting valt uiteen in twee delen: de Sprint prioritization en de Sprint planning.

De prioritization op zijn beurt valt ook weer uiteen in twee activiteiten: het bepalen van het doel van de eerstvolgende Sprint en de analyse en evaluatie van de product backlog uit de voorgaande Sprint. De backlog bestaat uit de requirements zoals de Product Owner die heeft opgesteld en bevat een lijst met alle gewenste werkzaamheden van het project. De prioritization genereert uiteindelijk het doel van de eerstvolgende Sprint (Sprint Goal).

Tijdens de Sprint Planning wordt besloten hoe het Sprint Goal wordt gerealiseerd. De teamleden selecteren hier items uit het Product backlog die ze gaan afwerken. Daartoe wordt een weer een backlog gemaakt, ditmaal een Sprint backlog. Dit is een verzameling taken die op hun beurt weer voortvloeien uit de Product backlog waarin de user stories en de features zijn verwerkt. Bij de identificatie van de taken is het hele tam betrokken – toewijzing ervan wordt niet alleen door de ScrumMaster gedaan. Tenslotte wordt een inschatting gemaakt van de hoeveelheid tijd die vereist is om de Sprint backlog te realiseren.

Het managen van de Sprint backlog is een dynamisch geheel waarbij alle teamleden betrokken zijn. Zo kan bijvoorbeeld elk individueel teamlid de backlog wijzigen, veranderen of verwijderen wanneer daar een goede reden voor is. Naarmate het werk vordert, wordt duidelijk hoeveel en welke taken blijven liggen – hiervan wordt een dagelijkse update gemaakt. Wanneer taken onduidelijk zijn, wordt de Sprint backlog opnieuw gedefinieerd en wordt meer tijd ingeruimd voor de te realiseren taken: die tijdspannen kan later weer worden verkleind of opgedeeld in nieuwe Sprints.

Daily Scrum

De dynamiek en de flexibiliteit vereisen een bijzondere werkwijze waarin een hoofdrol is weggelegd voor continue communicatie en feedback: de Daily Scrum, ook wel de dagelijkse Scrum-meeting genoemd. Deze meetings vinden dagelijks plaats bij aanvang van de werkdag. Tijdens dit overleg, dat meestal vijftien minuten duurt, beantwoordt elk teamlid drie vragen:

- **Wat heb je gedaan?**
- **Wat ga je doen?**
- **Tegen welke problemen loop je aan of ben je aangelopen?**

Niet alleen blijven de lijnen tussen de teamleden, ScrumMaster en Product Owner door deze meetings extreem kort en is een heldere communicatie verzekerd, Daily Scrum voorkomt ook onnodige en langdradige vergaderingen. Om te garanderen dat een meeting niet verzandt en te veel tijd in beslag neemt, wordt vaak staand vergaderd. Vandaar dat Scrum-meetings ook wel Standup-meetings genoemd worden.

Elke Sprint wordt afgesloten met een Sprint Review. Tijdens deze informele review presenteert het gehele team wat het gedurende de Sprint heeft bereikt. In de regel neemt zo`n review de vorm van van een demo waarin de nieuwe features of de onderliggende architectuur wordt gepresenteerd. Het is een pragmatisch gebeuren waarbij gekeken wordt wat wel en wat niet werkt.

Evidente voordelen

Ten opzichte van conventionele projectmethoden heeft Scrum evidente voordelen:

- Doordat er veel tegelijkertijd gebeurt, kan de doorloop van een project sterk ingekort worden;
- Er is veel contact en communicatie waardoor problemen bij iedereen bekend zijn, hetgeen het probleemoplossende vermogen van het team vergroot;
- Scrum stelt ontwikkelaars in staat om snel, real time en herhaaldelijk (elke twee weken) de software te evalueren die op dat moment gebruikt wordt;
- De business stelt de prioriteiten – er wordt dus geen software ontwikkeld die obsoleet is of die geen marktrelevantie heeft;
- Het project wordt op een integrale manier benaderd en niet als de som van verschillende deelprojecten (het eilandsyndroom);
- Het veelvuldige contact en communicatie leiden tot een verhoogde teamgeest;
- Veranderingen die de opdrachtgever wil doorvoeren, hebben geen al te grote impact op het werk, maar worden meegenomen in de workflow (de volgende Sprint);

- Het is snel duidelijk wat wel en wat niet werkt;
- De methode gedijt goed in chaos en binnen sterk dynamische omgevingen

Scrum bij Triodor Software

Gezien het voorgaande wekt het geen verbazing dat steeds meer ontwikkelaars de Scrum-methodiek omarmen. Het aantal toepassingen is immers indrukwekkend groot. Scrum is namelijk gebruikt bij de ontwikkeling van commerciële software, in-house software-ontwikkeling, contract development, fixed price projects, financiële applicaties, embedded systemen, de ontwikkeling van video games, software voor satellieten, websites, handheld software, mobiele telefonie, ISV applicaties en software voor medische monitoringsystemen en militaire toepassingen zoals de Joint Strike Fighter.

Ook Triodor Software hanteert de Scrum-methode bij haar software-ontwikkeltrajecten. Triodor is een *Microsoft Gold Certified* ontwikkelaar van software die met behulp van de nieuwste Microsoft technologie in verschillende Europese landen hoogwaardige producten en diensten levert. Het hoofdkantoor, waar de frontoffice-activiteiten plaatsvinden, staat in Amsterdam, terwijl de ontwikkelteams vanuit Istanbul werken.

Partnerships

Triodor streeft ernaar een integrale ketenpartner van haar klanten te zijn en gaat lange-termijn-relaties aan, die in sommige gevallen uitmonden in strategische samenwerkingsverbanden en joint ventures. De Scrum-methode sluit naadloos aan bij de manier waarop Triodor haar ontwikkeltrajecten vorm geeft. Triodors projecten kenmerken zich door korte lijnen en een heldere communicatie met zowel de opdrachtgever als de eindgebruiker. Immers, alleen in nauw overleg is het mogelijk het maatwerk te leveren dat maximaal inspeelt op de informatiseringsbehoefte van klanten.

Dedicated teams

De Scrum-methodiek wordt veelvuldig toegepast bij een van Triodors meest gebruikte werkwijze: die van het *dedicated team*. Het dedicated team is een

professioneel opererende unit van analisten, ontwikkelaars, consultants en projectmanagers dat exclusief aan één bepaalde klant wordt toegewezen en dat volledig geïntegreerd is in diens bedrijfsprocessen. Het dedicated team werkt op basis van *on-site sourcing*. Dit wil zeggen dat teamleden voor kortere of langere tijd bij een klant op locatie komen werken. Op die manier is continue feedback, overleg en voortgangsrapportage mogelijk en raken klant en teamleden goed op elkaar ingespeeld. Bovendien leert het team op deze manier de ins en outs kennen van de klants business en diens bedrijfsprocessen.

Lely Industries

Triodor brengt de Scrum-methodiek bij verschillende klanten en partnerorganisaties met succes in de praktijk. Een daarvan is Lely Industries, een agro-onderneming die innovatieve producten, diensten en oplossingen voor veehouderijen in de markt zet. Daarbij moet niet alleen gedacht worden aan landbouwmachines, maar ook aan gerobotiseerde apparatuur voor het volautomatisch melken van de veestapel of het schoonhouden en optimaal inrichten van de stallen. Lely is actief in meer dan 60 landen en heeft een jaarlijkse omzet van 340 miljoen euro. Sinds enkele jaren zijn Lely en Triodor partners in de joint venture Lely-Triodor. Lely maakt gebruik van een dedicated team van 23 software-ontwikkelaars bij Lely-Triodor in Istanbul. Dankzij deze samenwerking heeft Lely een kortere time-to-market dan voorheen gerealiseerd voor haar producten en diensten en kan Lely sneller inspelen op veranderende marktomstandigheden.

Het Lely ontwikkelteam gebruikt de nieuwste technologie voor het ontwerp en de ontwikkeling van software en softwarecomponenten voor de agro-business. Het team maakt onder andere gebruik van ASP.NET, JQUERY, C# en Microsoft SQL Server om een systeem te ontwikkelen dat bestaat uit webapplicaties, windows services, web services en setup modules. De software is geïnstalleerd in zo`n 4000 rundveebedrijven verspreid over 4 continenten.

Klantcases

Dezelfde of vergelijkbare methodes worden toegepast bij andere klanten zoals Qredits, Nederlandse Dakdekkers Associatie, Exact Software, MD Info, Flucon, Ovotrack, iScreen, MyWise Online, Havi Logistics, Novulo, Tema, SDL Tridion Development Lab, Intraworks, Icron Technology, Stichting Helpdesk Zaanstreek, Logistics Software Solutions/Skynet, Vonk Competentie Expertise, Adfocom, DIN Direct, A&B Software Solutions, B&A Groep Beleidsrealisatie, Sezer Consult, Campus Nieuw West, Stichting Microkredieten Nederland, Kamera Express, Bitbrains Nederland en Ondernemershuis Amsterdam Groot Oost.

Gratis business analyse

Maakt uw onderneming gebruik van maatwerk-ICT? Is ontwikkeling van software binnen gestelde termijnen en budgetten meestal een probleem? Voelt u voor de Scrum-benadering? Als u ons een grove schets geeft van uw bedrijfssituatie, dan investeren wij een dagdeel om samen met u te kijken of software-ontwikkeling via Triodors Srum-benadering voor uw bedrijf interessant is. Geheel vrijblijvend! U kunt daarvoor het formulier op onze site invullen. Wij nemen dan op korte termijn contact met u op.

Hoofdkantoor: Burg. Stramanweg 108N
1101 AA Amsterdam
T +31 (0)20 - 4511450
www.triodor.nl

Turkey Office: Baglarbasi Mah. Kumru Sokak
Cizgi Plaza No: 10/2
34844 Maltepe - Istanbul