

Het ontwerpen van software in opdracht van een klant is tweerichtingsverkeer. Een klant en ontwikkelorganisatie leggen de eisen en voorwaarden vast in een overeenkomst. Het biedt voordeel als beide organisaties open staan voor tussentijdse wijzigingen. Een ontwerper die aldoende een slimmere oplossing vindt voor een complex vraagstuk, kan zijn klant daarmee voordeel bieden. En een klant moet nieuwe wensen en inzichten van zijn markt in het project kunnen inbrengen. In de ideale situatie is een klant zowel opdrachtgever als supporter van het project.

# Houding is bepalend voor robuuste software

**S**oftware ontwikkelen kan op vele manieren. Natuurlijk zijn er verschillende programmeertalen en platforms. Maar ook de methode kan sterk verschillen: formeel versus informeel, strikt versus flexibel enzovoorts. Over formele ontwikkelmethodes is al veel gepubliceerd. De belangstelling voor informele methodes groeit, zodat ook daarover steeds meer publicaties verschijnen. Dit artikel gaat over de combinatie van formele en informele methodes. Deze combinatie wordt vaak toegepast. Zakelijk en formeel in de contracten en opdrachtformulering, flexibel in de uitvoering en de omgang met wijzigende inzichten. Verder hangt het van de persoon en bedrijfscultuur af welke methode het meest geschikt is. De een heeft aan een half woord genoeg, de ander heeft liever alles van a tot z op papier. Naarmate klant en ontwikkelorganisatie beter met elkaar bekend raken, zal de methode verschuiven naar het informele. Er groeit wederzijds vertrouwen. Een ontwikkelaar voelt zich betrokken bij een opdrachtgever, en omgekeerd. Beide partijen voelen zich verantwoordelijk voor het project. Deze houding biedt een klant en ontwik-

kelaar voordelen. Door deze betrokkenheid is er de mogelijkheid te optimaliseren gedurende het ontwikkelproces. Een tussentijdse aanpassing of verandering van inzicht is zo snel door te voeren. Voor een ontwikkelaar is het ook prettiger om een betrokken klant te hebben, met wie hij contact kan zoeken. In de ideale situatie weten beide partijen het voortschrijdend inzicht dat ze gezamenlijk opdoen om te zetten in optimalisatie. Vooraf hebben partijen afgesproken hoe hier zakelijk mee wordt omgegaan.

## Combinatie van methodes

In de praktijk wordt vaak een mix van formele en informele methodes toegepast. Deze gecombineerde werkwijze verloopt beter als vooraf niet het hele ontwikkeltraject wordt vastgelegd op papier. Een opdrachtgever moet kunnen rekenen op de kwaliteiten van een bedrijf. Hij moet vertrouwen dat de ontwikkelaars zijn wensen begrijpen. In een nieuwe relatie is dat een grote stap. Hoe weet hij dat een bedrijf zijn probleem kan oplossen en dat ontwikkelaars aan een half woord genoeg hebben? Hoe krijgt hij een beeld van het vakmanschap van

*In de praktijk wordt vaak een mix van formele en informele methodes toegepast.*



1

1. Rollen en verantwoordelijkheden klant-, ontwikkel-, en projectorganisatie tijdens ontwikkeltraject.

een organisatie? Dit kan door bijvoorbeeld de projectreferenties van een bedrijf te bestuderen. Wat heeft het bedrijf reeds afgeleverd aan andere opdrachtgevers en wat vonden zij ervan? Naast projectreferenties kan een ontwikkelorganisatie haar specifieke vaardigheden bewijzen met demonstrators en proofs of concept. Dit kunnen interne projecten zijn, door het bedrijf zelf geïnitieerd en betaald, om kennis aantoonbaar te maken. Uit deze voorbeelden blijkt wat een bedrijf kan, maar ook hoe ze werken. Welke methodes en technieken worden gebruikt. Plus minder concrete zaken als houding en visie. Is het eindproduct helder, overzichtelijk en toegankelijk of is het een onlogische brei aan bijvoorbeeld menu's en functies? De keuze zal duidelijk zijn: eenvoud is de essentie van kwaliteit.

### Refactoring

Het is echter hard werken om iets eenvoudig te houden. Bij softwareontwikkeling is de verleiding groot om elk probleem op te lossen met extra software. Terwijl het juist beter is een stap terug te gaan en het intrinsieke ontwerp aan te passen. Deze verbeteracties van het ontwerp worden ook wel 'refactoring' genoemd. Het is een vast onderdeel van een kwalitatief ontwikkeltraject. Het is een houding: kritisch zijn op het eigen werk. Het lef hebben om delen opnieuw te doen als het niet aan de vooraf gestelde criteria voldoet. Dat maakt het verschil tussen robuuste software en matige software. Refactoring is het integreren van voortschrijdend inzicht waarmee de eenvoud kan worden gehandhaafd.

### Incrementele oplevering

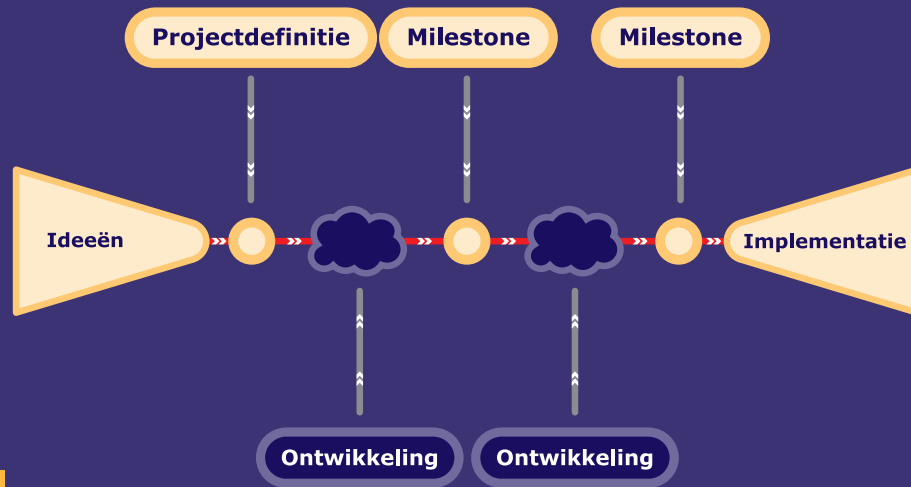
Een andere methode die van groot nut is bij het ontwikkelen van software is de 'incrementele oplevering'. Hierbij wordt het project intern in onderdelen opgeleverd. Soms komt het voor dat een klant deze delen wil gebruiken in het eigen

ontwikkeltraject. Bijvoorbeeld om de integratie in een groter systeem stapsgewijs te testen. Voor een ontwikkelaar is het een fundamentele manier van werken. Ook als een klant deze tussenproducten niet wil toetsen, wordt intern toch gewerkt met 'increments'. Een van de eerste increments kan een prototype zijn, maar dat hoeft niet. Een prototype dient om specificaties te verifiëren. Maar met een prototype kan ook worden gevalideerd of de wensen van de opdrachtgever goed zijn begrepen. Vooral als niet vooraf precies duidelijk was wat een klant wilde hebben: "Ik wil iets hebben met een uitstraling van ...". Dan wordt eerst een prototype gemaakt, een soort schets of een concept. Om te bekijken of het is wat een klant zoekt. Vaak is het een gebruikersinterface, waarbij smaak en gevoel een rol spelen. De functionaliteit die er uiteindelijk achter komt, is nog niet in het prototype verwerkt.

### Werkend systeem

In een project wordt zo snel mogelijk een werkend systeem gemaakt dat continu 'draaiend' wordt gehouden. Ondertussen gaat het ontwikkelen door en met grote regelmaat worden de nieuw ontworpen onderdelen aan het werkende systeem toegevoegd. Deze periodieke increments zijn voor intern gebruik. Een klant krijgt pas een incrementele oplevering als er een concreet stuk werk is afgerond. Met de periodieke increments is er continu een werkend systeem dat voortdurend wordt getest. Eventuele problemen zijn daarmee snel zichtbaar. Indien nodig worden bepaalde onderdelen gerefactored. Na de start van het project is het belangrijk dat de moeilijkste problemen het eerst worden aangepakt. Daarmee haalt een ontwikkelaar de risico's naar voren en zitten de verrassingen niet in de staart. Het oplossen van een groot probleem kan resulteren in een andere aanpak van het project of een aangepaste vorm van het product. Met een vroegtijdige signalering van knelpunten zijn de consequenties beter te overzien.

*Bij software-ontwikkeling is de verleiding groot om elk probleem op te lossen met extra software.*



2. Proces robuuste softwareontwikkeling.

## Cirkel

Het ontwikkeltraject is een continue cirkel van ontwikkelen, integreren en testen. Incrementele oplevering is daar een natuurlijk onderdeel van. Als een klant het wil, doet hij mee in het testen van de diverse onderdelen. Zo niet, dan speelt een van de ontwikkelaars de rol van testende klant. Dit is per definitie niet degene die het product maakt. Tester en ontwikkelaar zijn twee verschillende personen. Bij continue integratie is er altijd een werkend product, ook bij de oplevering. Dit resultaat, werkende robuuste software, is belangrijker dan de voorwaarden waaronder dit resultaat tot stand is gekomen. Deze werkwijze vraagt vertrouwen en betrokkenheid van ontwikkelaar én klant. Niet elke klant durft dat aan. Achterdochtig geworden door slechte ervaringen uit het verleden, stellen ze een fors pakket aan eisen en voorwaarden. Zo proberen ze hun leverancier te binden met zekerheden, die schijnzekerheden kunnen blijken. Een klant heeft het gevoel te zijn overgeleverd aan een leverancier, zonder zelf nog invloed te hebben.

## Vertrouwen

Een potentiële klant kan twijfelen bij welke organisatie hij zijn opdracht zal uitbesteden. Men wil een systeem met een bepaalde basisfunctionaliteit dat op de toekomst is voorbereid. De specificaties liggen nog niet volledig vast. Als het project mislukt, heeft dat grote gevolgen voor zijn bedrijf. Een ontwikkelorganisatie moet dan echte gemoedsrust kunnen bieden. Door met projectreferenties het aanwezige vakmanschap te bewijzen. Door te vertellen hoe het bedrijf werkt, hoe een toekomstbestendige architectuur wordt ontworpen en wat het bedrijf belangrijk vindt. Door een heldere projectbeschrijving te bieden, die echter nog niet volledig waterdicht zal en kan zijn. Maar wel een beschrijving die aangeeft: "Wij begrijpen de doelstelling van de klant". Als het project tot in de kleinste details wordt vastgelegd, is de impact

van een verandering zeer groot. Bij de informelere werkwijze staan afspraken zo compact mogelijk op papier. Kwaliteit en talent van de medewerkers zorgen ervoor dat er een kwalitatief product wordt gerealiseerd. Vaak blijkt het succes van een project omgekeerd evenredig te zijn met de dikte van de overeenkomst.

## Voortschrijdend inzicht

Reageren op verandering is belangrijk. Zeker als een ontwikkelaar langere tijd voor een opdrachtgever werkt. Die krijgt gedurende de looptijd andere ideeën en inzichten, of hij krijgt op zijn beurt weer vragen van zijn klanten. De opdrachtgever kan niet van tevoren alles overzien wat hij eigenlijk nodig heeft. Daar moeten beide partijen samen uit zien te komen. Dat kan alleen maar als ze elkaar echt willen begrijpen. Het is verstandig om vooraf overeen te komen hoe met veranderingen in het projectplan wordt omgegaan. Hoe wordt extra werk verrekend met niet uitgevoerd werk? Hoe wordt de begroting aangepast? Om misverstanden te voorkomen legt de ontwikkelaar tussentijdse aanpassingen vast. Een klant krijgt het wijzigingsvoorstel ter controle en geeft aan of dit is wat hij bedoelt. Voor de continuïteit van een bedrijf zijn langdurige relaties met klanten belangrijk. Dit belang is wederzijds. In een langdurige samenwerking leert het bedrijf zijn klanten en diens business beter begrijpen en kennen. Ervaring en kennis die keer op keer wordt hergebruikt. Daarmee is het risico voor beiden kleiner.

In de ideale situatie is een klant zowel opdrachtgever als toegewijde supporter van het project. Over het gehele traject werken klant en ontwikkelaar nauw samen. Beiden kunnen bijsturen indien nodig. Het is deze houding - een combinatie van formeel en informeel - die tot robuuste software leidt. Niet alleen de houding van een ontwikkelorganisatie is daarbij bepalend, maar ook die van een klant.

*De opdrachtgever kan niet van tevoren alles overzien wat hij eigenlijk nodig heeft. Daar moeten beide partijen samen uit zien te komen.*