Moet software gebouwd worden op basis van dikke documenten, overladen ontwikkelprocessen en overwerkte programmeurs? Of zijn er situaties waarbij op een meer menselijke manier kan worden samengewerkt, waarbij alleen activiteiten worden uitgevoerd die direct bijdragen aan de kwaliteit van de software en de productiviteit en het plezier van de programmeur? Extreme programming komt als ontwikkelproces aardig tegemoet aan deze wensen maar is het ook vernieuwend?

achtergrond

Gedisciplineerd ontwikkelen zonder overbodig papierwerk

Extreme programming: niet revolutionair, wel praktisch

Alhoewel de term programming anders doet vermoeden, richt XP zich niet alleen op de programmeerfase maar op het hele ontwikkelproces. Het programmeren wordt daarbij wel als een belangrijke ontwikkelactiviteit gezien die waarde genereert. XP is extreem in de zin dat er maar een beperkt aantal programmeer- en projectmanagementtechnieken wordt toegepast, maar dat die technieken wel ten volle gebruikt worden. De activiteiten die XP beschrijft, zijn (met tussen haakjes de oorspronkelijke Engelse termen): de planning game, kleine releases (small releases), metafoor (metaphor), het eenvoudig houden van ontwerp en code (simple design), testen (testing), refactoring, paarprogrammeren (pair programming), collectief eigendom van ontwerp en code (collective ownership), continu integreren (continuous integration), 40-urige werkweek (40-hour week), altijd aanwezige klant (on-site customer) en het toepassen van programmeerstandaarden (coding standards). We zullen deze activiteiten verderop in dit artikel beschrijven.

XP stelt een aantal randvoorwaarden aan de technologie en de ontwerptechnieken die gebruikt worden. Deze voorwaarden zijn onder andere:

- Sterke betrokkenheid van klant en gebruiker (klant en gebruiker mogen ook één en dezelfde persoon zijn). Deze moeten ook incrementele en iteratieve ontwikkeling willen steunen.
- Ontwikkelgereedschappen die iteratieve ontwikkeling gemakkelijk maken, zoals snelle compilers, goede editors en debuggers.
- De cultuur van de ontwikkelaars. Mensen die bijvoorbeeld het liefst alleen werken passen niet in een XP-team.

XP is pragmatisch en minimalistisch. Een van de basisideeën is het accepteren van verandering (bijvoorbeeld in eisen of in de technologie) in plaats van het bestrijden. Het is vaak economisch meer verantwoord om het huidige ontwerp zo eenvoudig mogelijk te houden, dan om in verwachte veranderingen te investeren. Een belangrijke aanname van XP is dat de kostencurve, die de kosten van veranderingen tegen de tijd weergeeft, niet langer exponentieel (zie afbeelding 1), maar afgevlakt is (zie afbeelding 2) door gebruik te maken van moderne tools en technologieën. Een gevolg hiervan is dat het niet nodig is om alle mogelijke veranderingen en variaties zo vroeg mogelijk in het project te voorspellen. Het aanbrengen

XP uit de praktijk geboren

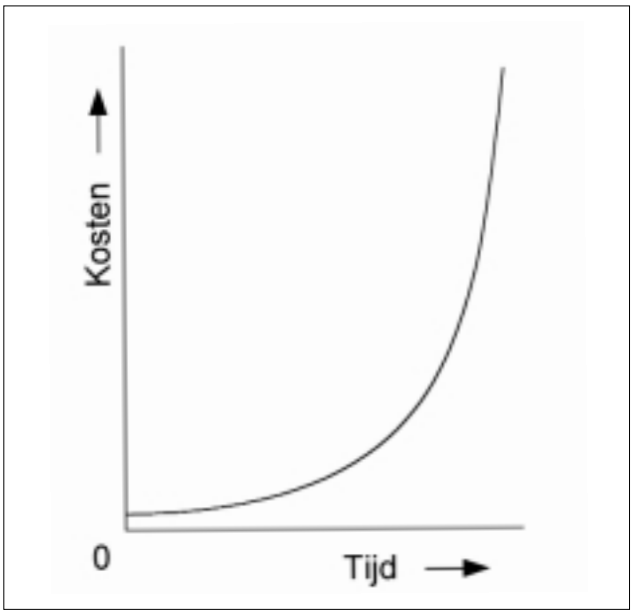
Extreme programming (XP), een lichtgewicht ontwikkelproces, biedt teams tot tien personen de mogelijkheid om met minimale documentatie productief te zijn. De communicatie en validatie, waar documentatie voor is bedoeld, worden vervangen door andere gebruiken, zoals intensieve samenwerking tussen de projectdeelnemers onderling en tussen projectdeelnemers en de klant. XP is geen academisch product, maar is uit praktijkervaring ontstaan. Het is door een groep zeer ervaren ontwikkelaars uitgeprobeerd en vastgelegd. Een van de belangrijke mensen achter XP is Kent Beck [Beck 2000], die mede de CRC-cardtechniek (Class-Responsibilities-Collaborators) heeft ontwikkeld en die al ruim 10 jaar betrokken is bij ontwerppatronen [Van Elswijk 1998].

van veranderingen later in het project is betaalbaar geworden.

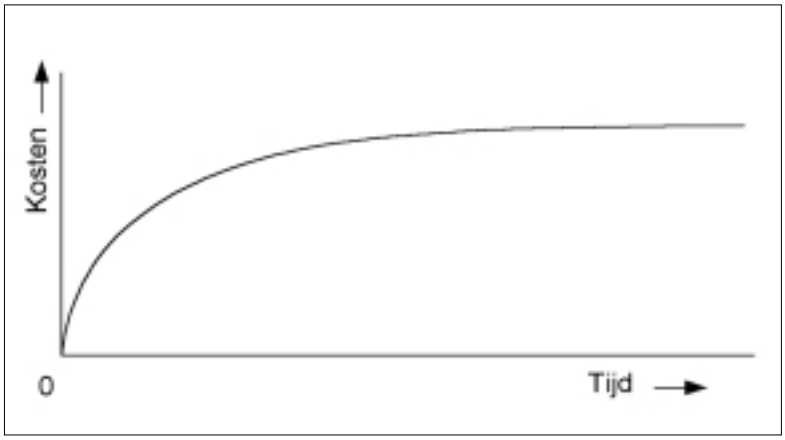
De menselijke factor speelt een centrale rol. Software ontwikkelen is mensenwerk. XP benadrukt dat ontwikkelaars in het algemeen normale werkweken dienen te maken. Structureel overwerk en hoge werkdruk gaan ten koste van de ontwikkelaars en ook ten koste van de kwaliteit van de software. XP reduceert de noodzaak tot het uitvoeren van onvoorzien werk door het ontwikkelproces beter beheersbaar te maken en door te concentreren op die functionaliteit die echt nodig is.

BEHEERSBAARHEID XP richt zich op het probleem van de gebrekkige beheersbaarheid van softwareontwikkelprojecten met betrekking tot de variabelen kosten, tijd, kwaliteit en omvang (hoeveelheid functionaliteit of features). Veel projecten zijn niet klaar binnen de gestelde tijd of komen zelfs geheel niet af. Projecten overschrijden het begrote budget fors en vaak blijkt dan ook niet voldaan aan de gestelde eisen [AG 7 april 2000]. De eisen zijn aan verandering onderhevig. Daarnaast leidt gebrek aan communicatie tot functionaliteit die niet echt nodig is, maar die wel een fors deel van het budget opslokt.

Voor de verschillende betrokkenen bij het project uiten deze problemen zich op verschillende wijzen. De klant moet aan het begin van het project bepalen wat hij/zij over een jaar of langer nodig denkt te hebben, maar kan dat allemaal nog niet precies bepalen. Als de software eenmaal ontwikkeld is, doet die niet wat de klant wil en als de klant om wijzigingen vraagt, zijn die duur. De projectmanager kan aan het begin van het project nog niet precies zeggen wat het team in een periode van een jaar of langer precies op kan leveren, terwijl de klant zo veel mogelijk wil voor zo min mogelijk geld. De ontwikkelaars worden geconfronteerd met een hoge werkdruk, een klant die nooit tevreden is en die steeds weer wat anders wil.



AFBEELDING 1: Exponentiële curve van kosten per verandering tegen de tijd



AFBEELDING 2: Vlakke curve van kosten per verandering tegen de tijd

PROJECTORGANISATIE Het projectmanagement van XP is erop gericht om op een voorspelbare en economisch verantwoorde manier voor de klant nuttige software te maken. Keuzes in projectmanagement worden over het algemeen bepaald door de vier variabelen: kosten, tijd, kwaliteit en omvang van het systeem. Deze vier variabelen zijn gerelateerd, maar wel op een complexe wijze, zodat ze niet allemaal tegelijk geoptimaliseerd kunnen worden. Een klant die eist dat de software “gisteren af moet zijn, moet doen wat ik wil, niet vast mag lopen, en vooral niks mag kosten” plaatst zich duidelijk buiten de werkelijkheid.

Een XP-project begint met een planningsproces, genaamd planning game, gevolgd door een aantal korte iteraties. Elke iteratie duurt maximaal drie tot vier weken en resulteert in een werkend systeem. Dit betekent niet dat er na elke iteratie ook een release van een productieversie plaatsvindt. Door de korte iteraties krijgt de klant snelle feedback over hoe het systeem zich ontwikkelt en kan de klant regelmatig bepalen of de kwaliteit en omvang naar wens zijn, met andere woorden, of er nog steeds waarde wordt geleverd. Na elke iteratie kunnen de eisen worden aangepast. Eventueel kan de klant de ontwikkeling stopzetten als de verhouding tussen investeren in ontwikkelen van nieuwe features en de geleverde waarde niet meer klopt. De planning game is een dialoog tussen de klant en het ontwikkelteam. De klant bepaalt wat wenselijk is, en de ontwikkelaars bepalen wat mogelijk is. De klant wordt in dit planningsproces gedwongen prioriteiten te stellen. Beide partijen moeten risico’s identificeren en afwegen. En van beide wordt ook discipline vereist. Ontwikkelaars bouwen alleen door de klant verlangde functionaliteit, en geen overbodige franje. Ze geven aan welke features moeilijk te maken zijn en hoe lang het bouwen van een feature duurt. De klant stelt geen

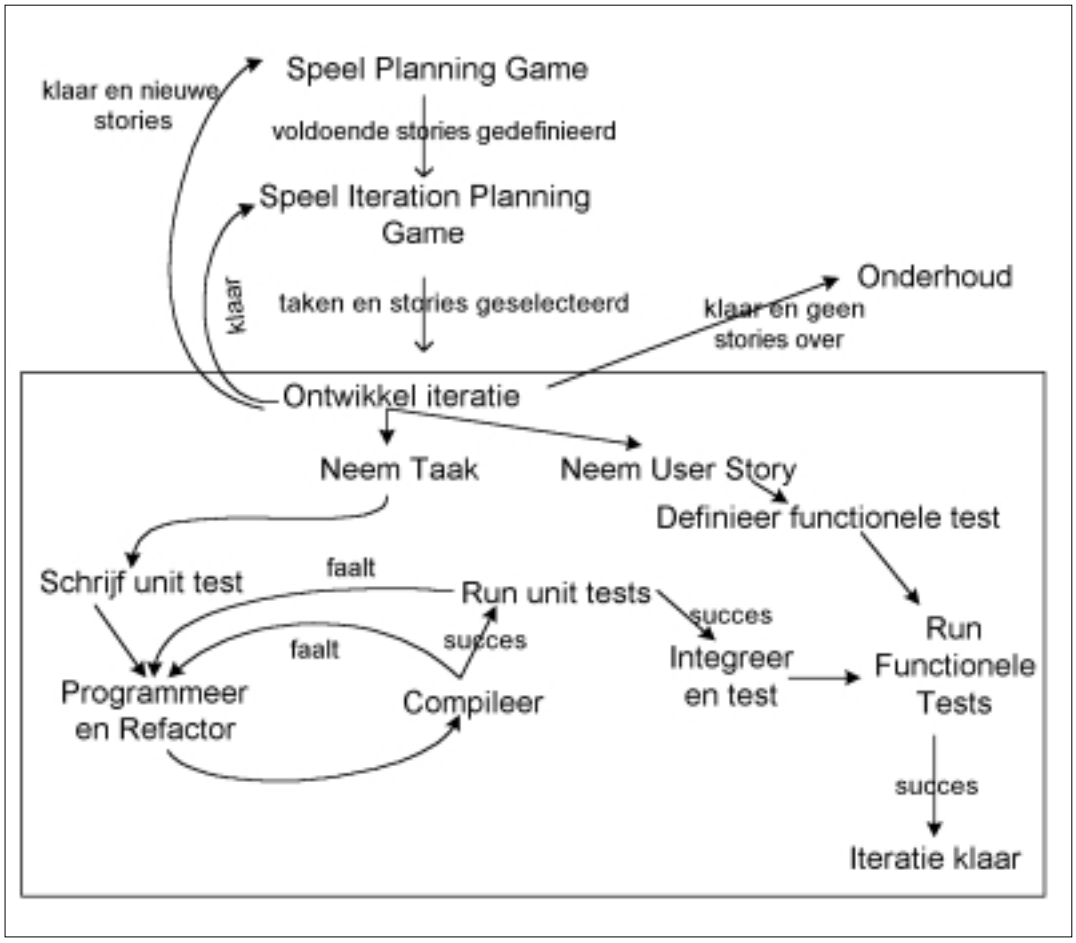
onmogelijke deadlines en bepaalt welke eigenschappen van het systeem het eerst moeten worden gebouwd.

De wensen van de klant worden vastgelegd in zogeheten user stories. Elke user story beschrijft een voor de klant zichtbare eigenschap van het systeem, compleet met eventuele randvoorwaarden aan deze eigenschap, zoals prestatie-eisen.

Aan het begin van een iteratie wordt een iteration planning game gespeeld. De klant bepaalt daarin welke stories in die iteratie worden gerealiseerd en mag daarbij nieuwe stories toevoegen en bestaande verwijderen. De ontwikkelaars bepalen voor elke story welke ontwikkeltaken nodig zijn om deze story te implementeren. Ze bepalen tevens de tijd die een taak gaat kosten. De tijd die een story gaat kosten is dan de som van de taken. De klant hoeft niet te weten wat die taken precies zijn. Wat het ontwikkelteam wel duidelijk moet maken zijn de risico's die verbonden zijn aan bepaalde stories, zoals bijvoorbeeld het gebruik van een nieuw en onbekend databasesysteem. Soms is het nodig om eerst een technologieverkenning te doen. Dit wordt dan gewoon als een story ingepland, met toestemming van de klant uiteraard. Hoe de taken worden verdeeld in het team, en hoe ze gepland worden is een zaak van het ontwikkelteam. De klant kan prioriteiten in zijn stories aangeven door te bepalen welke nu en welke later worden uitgevoerd. De volgorde is uiteraard wel afhankelijk van de technische randvoorwaarden die door de ontwikkelaars worden vastgesteld. De levenscyclus van een XP-project wordt in afbeelding 3 schematisch weergegeven.

Bij een XP project is er altijd een (vertegenwoordiger van de) klant aanwezig als lid van het projectteam, overigens niet alleen tijdens de planning game. Het gaat hierbij om een of meer personen die het systeem daadwerkelijk gaan gebruiken. Deze klant schrijft de eisen in de vorm van user stories en functionele tests, daarbij geholpen door de rest van het projectteam. Door de aanwezigheid van de klant hebben de ontwikkelaars altijd een aanspreekpunt en wordt het risico verminderd dat het systeem uiteindelijk niet doet wat de klant wil. De altijd aanwezige klant moet gezien worden als een investering in de kwaliteit van het systeem en de beheersbaarheid van het proces.

De conceptuele architectuur van het systeem wordt vastgelegd met behulp van een metafoor. Het belangrijkste doel van de metafoor is communicatie. De metafoor is het gemeenschappelijk beeld dat zowel de ontwikkelaars als de klant hebben van het systeem. Een voorbeeld van een metafoor is de bureaublad-metafoor die we kennen van onder andere de Apple Macintosh en Microsoft Windows. Een metafoor hoeft niet grafisch te zijn, zo lang het maar een beschrijving is waarmee de ontwikkelaars belangrijke delen van het systeem kunnen structureren en waarmee de klanten hun werkzaamheden kunnen uitleggen. De metafoor kan zich ontwikkelen gedurende het project als het inzicht in het systeem toeneemt.



AFBEELDING 3: Levenscyclus van een XP-project

ONTWERP, IMPLEMENTATIE EN TESTEN Belangrijke activiteiten van XP bij het uitvoeren van ontwikkeltaken zijn onder meer het eenvoudig houden van het ontwerp en de code, testen en refactoring. Deze activiteiten vinden continu plaats. Volgens XP moet het ontwerp en de code zo eenvoudig mogelijk gehouden worden. De intentie van een stuk code moet steeds duidelijk zijn, bij voorkeur middels een duidelijke naam. Code moet in het algemeen zonder commentaar en uitleg begrijpelijk zijn. Daarnaast wordt duplicatie van code

actief bestreden. Duplicatie van code is funest voor de aanpasbaarheid en vormt een groot onderhoudsrisico. Moderne programmeertalen bieden vele abstractiemogelijkheden waarmee eenmaal geschreven code op een toegankelijke manier opnieuw gebruikt kan worden.

Testen speelt op alle niveaus een cruciale rol in XP. XP onderscheidt functionele tests en unit tests. De functionele tests of acceptatietests worden door de klant geschreven en zijn een operationalisering van de eisen en user stories. Het bedenken van deze tests dwingt de klant en ontwikkelaars steeds om eisen concreet te maken, zodat ze verwezenlijkt kunnen worden in het systeem. Voordat code geschreven wordt, wordt eerst een geautomatiseerde unit test voor die code geschreven. Zo'n unit test probeert alles te testen wat mis zou kunnen gaan met de code. Een unit test kan gezien worden als een poging tot het weerleggen van de correctheid van een stuk code. Als een nieuwe bug wordt gevonden in code, wordt de bijbehorende unit test uitgebreid met een testgeval voor die bug. Op deze wijze combineert XP unit tests met automatische regressietests [Binder 1999]. Doordat de unit tests geautomatiseerd zijn, is het mogelijk ze herhaaldelijk zonder veel moeite uit te laten voeren, bijvoorbeeld na het aanbrengen van een wijziging in de code. Geautomatiseerde unit tests zijn een basis voor vertrouwen in de code. Ze maken het ontwikkelen en debuggen beter voorspelbaar. Ze zijn essentieel voor het beheersbaar houden van veranderingen en zijn een vangnet bij het uitvoeren van refactorings.

De testaanpak van XP heeft een aantal positieve bijeffecten. Doordat ontwikkelaars eerst unit tests schrijven, worden ze als het ware gedwongen om eerst goed na te denken alvorens te programmeren. Unit tests hebben ook een positieve invloed op het eenvoudig houden van het ontwerp: als het lastig is om unit tests te schrijven, is dit een indicatie dat het huidige ontwerp te complex is. Verder wordt het testen als activiteit geïntegreerd met het ontwerpen en programmeren. Het uitvoeren van geautomatiseerde tests kost weinig moeite en geeft directe feedback over de code. Dit werkt motivatieverhogend. Het voorkomt ook dat er bezuinigd wordt op het testen als het project ten einde loopt en men in tijdnood komt.

Refactoring is het op een systematische manier veranderen van de interne structuur van software zonder het observeerbare gedrag van de software te veranderen [Van den Ende 2000], [Fowler 1999]. Met refactoring worden de producten van het ontwikkelproces in goede conditie gehouden, zodat het steeds mogelijk blijft om efficiënt en effectief andere wijzigingen aan te brengen. Het is een belangrijke activiteit om het ontwerp en de code eenvoudig te houden. Het wordt onder meer toegepast om duplicatie van code te bestrijden. De achterliggende gedachte van refactoring is dat het moeilijk of niet te voorspellen is welke specifieke veranderingen en uitbreidingen er nodig zullen zijn. In een omgeving die sterk aan verandering onderhevig is levert een investering in refactoring in het algemeen meer op dan een investering in het anticiperen van specifieke veranderingen.

Standaarden verplicht

XP schrijft het gebruik van programmeerstandaarden voor om het collectieve eigendom van code te ondersteunen. Dit voorkomt ook zinloze discussies over bijvoorbeeld de lay-out van de code. Het projectteam is vrij om voor bepaalde standaarden te kiezen, zolang de projectleden maar in staat zijn elkaars code te begrijpen en aan te passen.

VERANTWOORDELIJKHEID In een XP-project zijn de code en andere producten van het ontwikkelproces collectief eigendom van de ontwikkelaars. Hierbij draagt elk projectlid medeverantwoordelijkheid voor de gehele code en mag elk projectlid overal wijzigingen aanbrengen. Dit vereist wel een zekere discipline van de ontwikkelaars. Het is zeker niet gelijk aan 'no ownership', waar iedereen naar eigen goeddunken wijzigingen kan aanbrengen. Als niemand verantwoordelijk is, leidt dit al snel tot chaos.

In de huidige praktijk wordt vaak individueel eigendom toegepast, waarbij ieder stuk code precies één verantwoordelijk projectlid heeft. Dit heeft ten opzichte van collectief eigendom een aantal belangrijke nadelen. Als er veranderingen nodig zijn in code van iemand anders, zullen deze minder snel worden aangebracht. Veranderingen die meerdere stukken code overspannen zijn ook problematisch, omdat niet één persoon daar de verantwoordelijkheid voor heeft. Dit leidt vaak tot onnodige duplicatie van code. Een ander probleem is het verdwijnen van kennis van een stuk code als de eigenaar ervan ziek wordt of vertrekt.

Verder wordt nieuwe en gewijzigde code continu, dat wil zeggen steeds na enkele uren of na hooguit een dag, geïntegreerd en getest. Voorwaarde voor integratie is dat alle unit tests voor 100% slagen met de gewijzigde code. Eventuele conflicten en problemen worden op deze manier direct gedetecteerd en zullen een beperkte impact hebben op de rest van de code. Een voorwaarde voor continue-integratie is dat de gebruikte tools dit moeten ondersteunen.

RUP EN CMM Hoe verhoudt XP zich tot het Rational Unified Process (RUP) en het Capability Maturity Model (CMM)? RUP beschrijft voornamelijk welke typen producten er kunnen worden opgeleverd, en in welke stappen dit moet gebeuren. Bij RUP ligt de nadruk veel meer op de methodologische aspecten en op welke processen, regels en artefacten gebruikt kunnen worden. XP is voornamelijk een proces en schrijft een aantal algemene ontwerptechnieken voor. XP kan een bijzondere

‘Programmeren met z’n tweeën’

Elke ontwikkeltaak wordt uitgevoerd door een paar van ontwikkelaars achter één werkstation (paarprogrammeren). Paren zijn niet vast, maar kunnen van dag tot dag verschillen. Voor elke nieuwe taak zoekt een ontwikkelaar opnieuw een projectlid om mee samen te werken. Paarprogrammeren heeft een aantal voordelen. Ontwikkelaars zullen minder snel het ontwerp van de code moeilijker maken dan strikt nodig is. Daarnaast zullen paren in het algemeen minder fouten maken. Er vindt continu een review van de code plaats evenals een uitwisseling van kennis over het systeem. Uit empirisch onderzoek [Cockburn et al. 2000] blijkt dat twee ontwikkelaars die als paar programmeren iets minder productief zijn (in termen van hoeveelheid code) dan twee afzonderlijk werkende ontwikkelaars. Daar staat tegenover dat ze code produceren van betere kwaliteit (in termen van minder fouten). De extra kosten van het programmeren in paren worden gecompenseerd doordat tijd bespaard kan worden op het debuggen. Daarnaast maakt paarprogrammeren het mogelijk om het aantal mensen in een team te vergroten, zonder de communicatieoverhead te vergroten. De grens waar men met meerdere teams moet gaan werken wordt verlegd.

manier zijn om aan het Capability Maturity Model te voldoen. CMM niveau 2 bevat zes aandachtspunten. Voor vijf van deze aandachtspunten heeft XP activiteiten en rollen gedefinieerd. Software Requirements Management wordt gedekt door user stories en de planning game. Software Project Planning wordt gedaan in de planning game en in het definiëren van taken. Software Tracking and Oversight wordt gedaan door de ‘Tracker’, een rol die de voortgang van het project bijhoudt en dit naar de andere projectleden communiceert. Software Quality Assurance vindt plaats doordat de altijd aanwezige klant waakt over de bruikbaarheid voor gebruikers, terwijl de functionele en unit-tests voor robuustheid van de software zorgen. Software Configuration Management wordt toegepast bij continue-integratie. Voor het zesde aandachtspunt, Subcontract Management, is niets beschreven, omdat XP alles voor één team definieert dat niets uitbesteedt. Als een organisatie dit bij het uitrollen van XP regelt en haar wijze van gebruik van XP vastlegt, is het bereiken van hogere CMM-niveaus mogelijk. Het bijzondere aan CMM door XP is dat er relatief weinig op papier wordt vastgelegd. CMM schrijft in feite alleen een aantal activiteiten voor. De nadruk op documentatie komt vooral naar voren bij een CMM-assessment, omdat een papieren proces voor auditors eenvoudiger is.

GOUDEN BERGEN Extreme programming is niet de zoveelste methodiek die de gebruikelijke gouden bergen belooft. Het is niet revolutionair vernieuwend, maar verenigt een aantal activiteiten die zich in de praktijk bewezen hebben. Sommige van deze activiteiten, zoals paarprogrammeren, wekken bij veel mensen in eerste instantie weerstand op. De activiteiten tonen hun sterke kanten vooral als ze in combinatie gebruikt worden, zodat sterke en zwakke punten elkaar compenseren. Elk project zal zijn eigen variaties op onderdelen van het proces hebben, afhankelijk van de behoeften.

XP is nog volop in ontwikkeling. In juni 2000 is de eerste XP-conferentie. Verder zijn enkele boeken in voorbereiding over hoe XP toegepast kan worden [Andersen et al. 2000]. XP kan worden gezien als een stap op weg naar professionalisering van software engineering. Professionals leggen discipline aan de dag als het gaat om het uitvoeren van hun taken. Daarbij zijn ze pragmatisch, economisch verantwoord en houden ze rekening met de mensen in hun omgeving. Meer informatie over XP kan gevonden worden in het boek van Kent Beck [Beck 2000] en op de XP website [XP website].

Willem van den Ende
is onderzoeker bij het Software Engineering Research Centre (SERC) en bereikbaar via ende@serc.nl

Marc Evers
is medewerker onderzoek bij de leerstoel Taal Kennis en Interactie van de Faculteit der Informatica, Universiteit Twente en bereikbaar via evers@cs.utwente.nl

Literatuur

[AG 7 april 2000] “Meeste IT-projecten te duur en te laat klaar”, Automatiseringgids, vrijdag 7 april 2000, p. 17

[Andersen et al. 2000] Ann Anderson, Chet Hendrickson, Ronald E. Jeffries, Extreme Programming Installed, Addison-Wesley, verwacht in augustus 2000

[Beck 2000] Kent Beck, Extreme programming explained: embrace change, Addison-Wesley, 2000

[Binder 1999] Robert V. Binder, Testing Object Oriented Sytems, Addison Wesley, 1999

[Cockburn et. al. 2000] Alistair Cockburn, Laurie Williams. “The Costs and Benefits of Pair Programming”, te verschijnen in eXtreme Programming and Flexible Processes in Software Engineering, proceedings van de XP2000 conferentie (21-23 Juni 2000, Cagliari, Italië)

[Van Elswijk 1998] Mark van Elswijk, “Software-patronen”, Handboek Database Systemen, Array Publications, juni 1998

[Van den Ende 2000] Willem van den Ende, “Refactoring – ontwerpverbetering in bestaande code”, Software Release Magazine, nummer 3, mei 2000

[Fowler 1999] Martin Fowler, ed., Refactoring: improving the design of existing code, Addison-Wesley, 1999

[XP website] Extreme Programming: A gentle introduction, http://www.extremeprogramming.org