

# Afstudeerrapportage

## Ontwikkelmethoden onderzoek



Naam: Tom van Leijssen  
Studentnummer: 1113962

Datum: 27 februari 2008  
Versie: 1.1

Course: Werkend Leren 2  
Periode: Kwartaal 3 en 4  
Schooljaar: 2007/2008

Bedrijf: Sogeti Nederland B.V.  
Bedrijfsbegeleider: S. Bosma / C. van Aart

Docentbegeleider: G. Wagenaar

## Verbeteringen

Hieronder wordt per versie aangegeven wat er is veranderd ten opzichte van versie 1.0.

### ***Verbeteringen in versie 1.1***

- De conclusie is beter onderbouwd.

## Inhoudsopgave

1	Inleiding .....	4
2	Watervalmethode .....	5
3	Scrum .....	7
4	Rational Unified Process .....	8
5	Dynamic Systems Development Method .....	10
6	eXtreme Programming .....	12
	Conclusie .....	14
	Bronnenlijst .....	15

# 1 Inleiding

Op dit moment ben ik aan het afstuderen bij Sogeti Nederland BV. Sogeti Nederland BV maakt onderdeel uit van Sogeti SAS, een Franse onderneming, met haar hoofdkantoor in Parijs. Sogeti SAS maakt op haar beurt onderdeel uit van Cap Gemini SA, tevens gevestigd in Parijs. Cap Gemini is een wereldwijde organisatie die diensten biedt op gebied van ICT en consultancy, waarbij de "Sogeti organisatie" binnen de "Cap Gemini organisatie" het ICT-vakbedrijf is. Binnen de "Sogeti groep" is Sogeti Nederland BV met 3100 medewerkers één van de grotere spelers.

Voor de ontwikkeling van een toepassing, is het verstandig om een ontwikkelmethode te gebruiken. Maar er zijn zoveel ontwikkelmethodes dat het lastig is om hier een juiste uit te kiezen. Vandaar dit onderzoek, tijdens dit onderzoek zal ik de volgende ontwikkelmethodes kort beschrijven, Watervalmethode, Scrum, RUP, DSDM en XP. Vervolgens ga ik bepalen welke methode het beste te gebruiken is voor dit project. De klant (in dit geval C. van Aart van Sogeti Nederland BV) geeft de voorkeur aan een ontwikkelmethode die iteratief is en gebruik maakt van prototyping. Op deze manier krijgt hij snel inzicht in de resultaten en kan hij, als het eindresultaat van het project dreigt te mislukken, direct ingrijpen en waar nodig is bijsturen.

## 2 Watervalmethode

De watervalmethode is een methode, waarin de ontwikkelfasen vloeiend naar beneden lopen (als een waterval). Deze methode is afgeleid van de manier van werken in de constructiebouw. Je verdeelt het project in een aantal fasen en begint met fase 1. Pas als fase 1 is afgesloten ga je verder met fase 2 enz. Wanneer je in een van de fasen een fout ontdekt moet je terug om die fase te corrigeren en de daaropvolgende stappen opnieuw uit te voeren.

Het watervalmodel bestaat uit de volgende fasen:

- **Definitiestudie:** in deze wordt er gebrainstormd over het doel van de toepassing. Daarnaast wordt er ook een onderzoek gedaan om het doel van de toepassing helder te krijgen.
- **Basisontwerp:** de resultaten van de eerste fase worden duidelijker uitgewerkt. De eisen en wensen van de klant worden op papier gezet en er wordt al gedacht aan de vormgeving van de toepassing.
- **Technisch ontwerp:** aan de hand van het basisontwerp kan er een werkelijke toepassing uitgedacht worden. In deze fase wordt vastgelegd hoe de in het basisontwerp vastgelegde functionaliteit gerealiseerd gaat worden.
- **Bouwen:** in deze fase wordt de toepassing daadwerkelijk gerealiseerd. De eisen en wensen van de klant worden in het systeem geïntegreerd en het systeem moet testklaar zijn.
- **Testen:** er wordt gecontroleerd of de toepassing goed volgens de ontwerpen is gebouwd en of de beschreven functionaliteiten goed werken. Er kunnen in deze fase fouten boven water komen die al in eerdere stadia gemaakt zijn.
- **Integratie:** het systeem is klaar en getest. Tijdens deze fase wordt het systeem overgedragen aan het bedrijf en zal het bij het bedrijf gebruikt gaan worden.
- **Beheer en onderhoud:** om er voor te zorgen dat het systeem blijft werken en wordt voorzien van eventuele updates zal er onderhoud verricht moeten worden. Dit is een fase die door blijft lopen zolang de toepassing wordt gebruikt.

### Voordelen

- Wanneer in het begin van het project fouten worden ontdekt, kost het minder inspanning (en dus tijd en geld) om deze fout te herstellen. Bij de watervalmethode wil men alle fasen eerst goed afsluiten voordat men verder gaat met de volgende fase, om fouten te voorkomen.
- Bij de watervalmethode legt men de nadruk op documentatie. Als er nieuwe mensen in het project opgenomen worden of als er mensen weggaan is het minder lastig om de kennis over te dragen.
- Het is een strakke methode. De manier van werken zorgt ervoor dat er concrete fasen gevormd worden. Hierdoor is, samen met het gebruik van mijlpalen, de voortgang goed zichtbaar.
- De watervalmethode is erg bekend. Veel mensen hebben kennis van deze ontwikkelmethode en kunnen er gemakkelijk mee werken.

**Nadelen**

Er zijn een aantal nadelen van deze manier een toepassing te ontwikkelen.

- De eisen en wensen van de klant veranderen vaak gedurende het project. De watervalmethode gaat ervan uit dat de eisen en wensen niet veranderen tijdens het project.
- Het is erg moeilijk om de benodigde tijd en de kosten te schatten. De fasen zijn erg groot, daardoor is het lastig om in te schatten hoeveel tijd en geld iedere fase gaat kosten.
- Binnen het project zijn verschillende medewerkers en verschillende fasen. In de eerste fase houden de ontwerpers zich bezig met het ontwerp en de bouwers zijn alleen actief in de bouwfase. Dit kan leiden tot tijdverspilling.
- Bij deze methode is de toepassing pas op het einde klaar. Dan pas krijgt de klant het resultaat te zien. Wanneer de klant tussendoor resultaten ziet, geeft hem dit vaker meer vertrouwen en kan de klant tips en suggesties geven.
- Het testen gebeurt pas in één van de laatste fasen van het project. Als er dan een fout ontdekt wordt, kan het mogelijk zijn dat het gehele traject opnieuw doorlopen moet worden.

## 3 Scrum

Scrum is een variatie op de watervalmethode. Bij de watervalmethode was er teveel overlap bij de verschillende fases. Bij scrum is er nog maar 1 fase. Scrum is een term die afkomstig is uit de rugbysport, hierbij staan de spelers in een grote groep en proberen ze al duwend de bal naar de overkant van het veld te brengen. Er wordt dus niet afgewacht of de vorige fase afgelopen is maar er wordt tegelijkertijd gewerkt.

### Doelstellingen

- Verhogen van de effectiviteit van het team;
- Het bewaken van de vooruitgang van het team;
- Het oplossen van blokkades;
- Het bewaken van de projectvoortgang;
- In kaart brengen en minimaliseren van de risico's.

### Werkwijze

Bij de watervalmethode heeft iedere fase een expert op dat gebied. Die voert zijn taak uit met een team dat daar ook in gespecialiseerd is. Bij scrum haal je uit iedere fase de expert en plaats je deze 7 experts (van de 7 fasen) bij elkaar in één team. Het team wordt geleid door de "scrum-master" en houdt vrijwel dagelijks bij aanvang van de werkdag een zogenaamde "scrum-meeting". In deze meeting beantwoordt elk teamlid de volgende 3 vragen:

- Wat heb je gedaan?
- Wat ga je doen?
- Wat zijn je problemen?

Daarna gaat de expert weer aan het werk met zijn eigen team om de opdracht te volbrengen. De personen werken veel samen en pakken het project met zijn allen aan en zullen er voor zorgen dat het project ook slaagt.

### Voordelen

- Doordat er iedere dag een "scrum-meeting" is en er veel communicatie is zal de teamspirit hoog zijn en zal iedere persoon elkaar willen helpen.
- Doordat er veel tegelijk gebeurt, zal de tijd waarin het project zal worden afgerond korter zijn dan bij andere methodes.
- Omdat iedereen van elkaar weet waar zij mee bezig zijn, kan iedereen elkaar direct helpen als er een probleem is, zonder eerst te wachten op een andere fase.

## 4 Rational Unified Process

De bedenkers en ontwikkelaars van RUP hebben zich gefocust op het analyseren van de karakteristieken waardoor projecten mislukken. Hierdoor konden ze de hoofdproblemen herkennen waardoor projecten vaak mislukken. Op deze lijst staan bijvoorbeeld:

- Vage, dubbelzinnige en onduidelijke eisen;
- Overmatige complexiteit;
- Onvoldoende testen.

De uitkomst van hun onderzoek was een verzameling van best practices die ze hadden afgeleid van de projecten die wel slaagden. Door deze best practices samen te voegen is het Rational Unified Process ontstaan.

### Fasering

RUP verteld ons dat er vier hoofdfases zijn:

- **Inceptiefase** (Aanvang): de haalbaarheid en de scope van het project worden bepaald. Het idee wordt omgezet in een productvisie. De globale kostprijs en de verwachte baten van het project worden geschat en de belangrijkste risico's worden geïdentificeerd.  
Het uiteindelijke doel van deze fase is een haalbaarheidsstudie ('business case') voor het project, of anders gesteld een GO/NOGO voor het project.
- **Elaboratiefase** (Detailering): het merendeel van de functionele eisen wordt gespecificeerd en de systeemarchitectuur wordt ontworpen. De focus ligt op de technische haalbaarheid van het project. Het uiteindelijke doel van deze fase is een projectplan, waarin de gedetailleerde scope, tijdsplanning en kostenraming voor het project zijn opgenomen.
- **Constructiefase** (Bouw): tijdens deze fase wordt het product ontwikkeld vanaf de architectuur tot een systeem dat compleet genoeg is om te testen.
- **Transitiefase** (Overgang): aan de hand van de testen, die in deze fase worden gedaan, wordt het product gevalideerd door de belanghebbenden. Andere activiteiten in deze fase zijn: voorbereiding en in productiestelling, nazorg, en overdracht van de verantwoordelijkheden. Deze fase wordt afgesloten met een inventarisatie van de opgedane ervaringen ('lessons learned') voor volgende projecten.

### Ontwikkel incrementeel/iteratief

Gezien de tijd die het kost om grote softwaresystemen te bouwen, is het niet mogelijk om een heel systeem in één keer te maken. Eisen kunnen veranderen in de loop van het traject door technische beperkingen, veranderende eisen/wensen of een beter begrip van het daadwerkelijke probleem. Iteratief ontwikkelen stelt je in staat om het project op te delen in een aantal deelproducten en deze afzonderlijk van elkaar op te leveren. Dit verkleint het risico dat het eindproduct niet is wat de klant wil. Daarnaast kan de klant per deelproduct zijn feedback geven.

### Manage software eisen

Het managen van eisen heeft te maken met het achterhalen, vastleggen en beheren van de behoeften van de klant. Daaronder vallen de volgende activiteiten:

- Analyseer het probleem;
- Definieer wat de klant wil;
- Definieer het op te leveren systeem;
- Beheer de randvoorwaarden van het systeem;
- Verfijn de systeemeisen/systeemomschrijvingen;
- Beheer de wijzigende eisen.

**Maak prototypes**

Door de gebruiker een grafische voorstelling te geven van het product (prototyping), verkleint de faalkans van het project. Een globale, grafische oplossing voor het probleem is door de gebruiker beter te begrijpen dan pagina's vol broncode. Het is een versimpeling van de complexiteit.

**Test het systeem**

Het bepalen van de kwaliteit van een systeem gebeurt op basis van testen. Dit is een van de punten waarop projecten vaak falen omdat het testen vaak aan het einde van het project wordt gedaan, soms helemaal niet en soms door andere teams. RUP vangt dit probleem af door het testen in het gehele proces terug te laten komen en daarbij alle belanghebbenden te betrekken (zowel programmeurs als klanten). RUP gaat er vanuit dat elke belanghebbenden verantwoordelijk is voor de kwaliteit gedurende het gehele project.

## 5 Dynamic Systems Development Method

DSDM is een methode die de ontwikkeling van een toepassing vastlegt in een raamwerk van tijdsplanning (timeboxing). De duur van het project en de te gebruiken resources worden vastgelegd, maar de eisen kunnen in het verloop van het project variëren.

In het begin van het project worden op globaal niveau de functionele- en niet-functionele eisen vastgesteld en geordend op prioriteit (MoSCoW). Tijdens de ontwikkeling worden deze eisen steeds gedetailleerder en worden ze opnieuw op basis van prioriteit ingedeeld. Binnen de tijdsplanning worden eerst de zaken opgeleverd, die de hoogste prioriteit hebben. Hierdoor maakt DSDM een ICT project flexibeler.

### Basisprincipes

DSDM is gebaseerd op ervaringen van organisaties met versnelde systeemontwikkeling. Het belangrijkste van al deze ervaringen heeft geleid tot de theorie van DSDM. De basis hiervan wordt gelegd door 9 principes die in de hele theorie terug te vinden zijn.

1. Klant en eindgebruikers moeten actief betrokken worden.
2. Teams moeten bevoegd zijn om beslissingen te nemen.
3. Producten moeten op te delen zijn in deelproducten.
4. De applicatie moet uiteindelijk voldoen aan de gestelde eisen. Het moet geschikt zijn voor de bedrijfsdoeleinden.
5. Iteraties en incrementen zijn nodig om samen tot één juiste bedrijfsoplossing te komen.
6. Als er wijzigingen zijn gemaakt, moeten deze gemakkelijk terug te draaien zijn.
7. Eisen worden op hoog niveau omschreven, afgebakend en geordend op prioriteit.
8. Er wordt tijdens de ontwikkeling direct getest.
9. Alle belanghebbenden moeten samen willen werken.

### DSDM fasen

- **Feasibility Study:** in deze fase wordt bepaald of het systeem, met de gekozen technische realisatie, kosten en tijdsduur, geschikt is voor het bedrijf.
- **Business Study:** tijdens deze fase worden de primaire functies van het systeem bepaald en wordt er nagedacht over de gewenste betrouwbaarheid en prestaties.
- **Functional model iteration:** met behulp van prototyping is het mogelijk om gebruikerseisen te bepalen, informatie te verzamelen, de functionaliteit te tonen en niet-functionele eisen te achterhalen. Deze fase wordt net zo vaak herhaald als nodig.
- **System Design/Build Iteration:** deze fase kan al beginnen voordat de FMI fase voorbij is, om het functionele prototype verder uit te werken, met het doel om een design prototype op te leveren dat voldoet aan de functionele en niet-functionele eisen. Ook deze fase kan vaker uitgevoerd worden.
- **Implementation:** hier wordt het systeem overgedragen naar de eindgebruikers voor evaluatie en beoordeling. Aan de hand van de uitkomst van deze evaluatie wordt het systeem ofwel definitief opgeleverd of wordt de ontwikkeling teruggezet naar een eerdere fase.

### Timeboxing

Timeboxing is een techniek die zorgt voor de tijdige oplevering en realisatie van het project. Binnen DSDM projecten is de opleverdatum bekend en de productiecapaciteit is daardoor duidelijk te benoemen (op basis van beschikbare tijd en resources).

Timeboxing zorgt ervoor dat tijd en geld worden vastgesteld, en dat de functionaliteit variabel is. Het managen van functionaliteit gebeurt door middel van prioriteitstelling (MoSCoW).

### MoSCoW

Er wordt een lijst met eisen opgesteld waaraan vervolgens prioriteiten worden gekoppeld. De afkorting MoSCoW staat voor de relatieve gewenstheid van de diverse onderdelen van het systeem:

- **Must have:** deze eisen moeten gerealiseerd worden.
- **Should have:** deze eisen zouden eigenlijk gerealiseerd moeten worden.
- **Could have:** deze eisen kunnen gerealiseerd worden (als de must have's en should have's eerder zijn gerealiseerd dan verwacht).
- **Would have next time:** deze eisen worden waarschijnlijk niet gerealiseerd binnen het betreffende project.

## 6 eXtreme Programming

XP is een methodiek die vooral geschikt is voor projecten waarbij de exacte applicatie-eisen niet bij voorbaat vastliggen. EXtreme Programming dankt zijn naam aan het feit dat een aantal beproefde ontwikkelprincipes tot in het extreme worden door gevoerd. Voorbeelden zijn:

- Als code reviewen goed is, doe het dan voortdurend: ontwikkel alle code in koppels. Met andere woorden, twee mensen achter één computer.
- Als testen goed is, schrijf dan 'unit tests' en voer ze iedere keer uit als er ook maar een regel code is veranderd.
- Als eenvoud goed is, houdt het ontwerp dan zo simpel mogelijk. XP werkt veel met het KISS principe (Keep It Simple Stupid!).
- Als korte iteratieslagen goed zijn, maak ze dan ook heel erg kort.
- Als ontwerpen goed is, maak het dan onderdeel van ieders dagelijks werk: verbeter het ontwerp stapsgewijs, zodra de noodzaak zich voordoet.
- Als architectuur zo belangrijk is, laat dan iedereen werken aan het ontwikkelen van de architectuur.
- Als integratietesten belangrijk is, integreer de code dan zo vaak mogelijk, liefst meerdere keren per dag.

Geen van de onderstaande 'best practices' zijn uniek voor XP. Stuk voor stuk zijn ze decennia geleden al vaak toegepast.

### Best practices

De optimale kracht van XP komt voort uit het in samenhang toepassen van twaalf 'best practices', zodat deze elkaar versterken. Elke andere software ontwikkelmethode zal ook voordeel hebben bij het toepassen van één of meerdere genoemde 'best practices':

1. Gebruik een zo simpel mogelijk ontwerp dat toch voldoet aan de eisen.
2. Schrijf eerst de testcode ('unit test') voordat je functionaliteit codeert.
3. Continue herstructurering ('refactoring') van de programmacode.
4. Continue integratie van alle programmacode.
5. Iedere ontwikkelaar heeft gelijke rechten over alle programmacode.
6. Iedere ontwikkelaar gebruikt dezelfde codeerstandaard.
7. Programmeurs werken in paren ('pair programming').
8. De klant is onderdeel van het ontwikkelteam en moet dus continu voor vragen beschikbaar zijn.
9. De toepassing wordt in een vaste regelmaat in releases van beperkte omvang aan de klant opgeleverd voor beoordeling.
10. Ontwikkelaars plannen aan de hand van kleine brokken functionaliteit ('user stories').
11. Alle teamleden (ontwikkelaar en klant) delen een gemeenschappelijk beeld van het systeem ('metafoor').
12. Overwerken is een uitzondering.

### Feedback

XP ervan uit dat bij het maken van een toepassing alles om code draait. XP gaat er vanuit dat deze code dusdanig klantspecifiek is, dat de klant zo dicht mogelijk bij het ontwikkeltraject dient te staan. Bij XP wordt ontwikkeld op basis van testcases die de klant samenstelt. Geen functionele specificaties; er zal enkel worden gecodeerd wat de klant wil dat er uit de testcases moet komen. Niet meer en ook niet minder. Bij voorkeur dient een toekomstige gebruiker te allen tijde aanwezig te zijn, zodat er voor gezorgd kan worden dat de applicatie exact dat wordt wat de klant wenst. Om dit proces zo goed mogelijk te laten verlopen, werkt XP met korte ontwikkelcycli, waarin steeds een aantal geselecteerde testcases wordt geïmplementeerd tot een werkend systeem. Aan het eind van iedere cyclus kan de klant het systeem beoordelen en desnoods de ontwikkeling bijsturen.

**Refactoring**

Een belangrijke techniek waarmee XP zich verder onderscheidt van traditionele ontwikkelmethodieken is refactoring: het herschrijven van code zonder dat daarbij de zichtbare functionaliteit wordt aangetast. Refactoring voegt kortom niets aan de functionaliteit toe, maar het verbetert de eenvoud van het ontwerp. De randvoorwaarde voor het succesvol toepassen van refactoring is dat er 'unit tests' voorhanden zijn, die automatisch uitgevoerd kunnen worden na iedere herschrijfstep om zeker te stellen dat de functionaliteit niet is veranderd.

**Eenvoud van ontwerp**

Moet een systeem gemakkelijk te veranderen zijn, dan dient het ontwerp zo eenvoudig mogelijk te zijn. Dit is makkelijker gezegd dan gedaan. Veel ontwikkelmethoden hebben ons geleerd vooruit te denken en bij het ontwerp steeds na te denken over functionaliteit die misschien in de toekomst moet worden gerealiseerd. Daarom hamert XP er op steeds het meest eenvoudige ontwerp te kiezen om de functionaliteit, die nu gerealiseerd moet worden, mogelijk te maken. Tevens blijkt dat bij de realisatie van een doordacht ontwerp maar al te vaak dat het eigenlijk niet (meer) voldoet aan de gestelde eisen. Dit kan enerzijds gebeuren doordat tijdens de analyse en het ontwerpen bepaalde details over het hoofd zijn gezien of anderzijds doordat de eisen zijn bijgesteld. Bij XP loopt het ontwerp niet voorop, maar volgt deze de code.

**Pair programming**

Één van de meest extreme aspecten van XP is het voorschrift dat alle ontwikkeling gedaan wordt in koppels: twee mensen achter een computer. Onderzoek heeft aangetoond dat 'peer-review' en 'code-inspectie' de krachtigste wapens zijn tegen bugs, veel krachtiger zelfs dan systematisch testen. Toch worden deze technieken maar mondjesmaat toegepast en roepen ze vaak grote weerstand op bij de programmeurs zelf, en bij managers die bang zijn voor een stijgend aantal arbeidsuren. Door af te dwingen dat alle softwareontwikkeling in koppels wordt uitgevoerd, die bovendien regelmatig van samenstelling wisselen, ontstaat er een collectief 'eigendomsgevoel' en worden 'peer-review' en 'code-inspectie' vanzelf onderdeel van het normale software proces.

## Conclusie

*Door dit onderzoek ben ik wat meer te weten gekomen over een aantal ontwikkelmethodes, welke ik nog niet (vaak) heb toegepast. Hierdoor kan ik een ontwikkelmethode kiezen die het beste past bij deze afstudeeropdracht.*

De klant (in dit geval C. van Aart van Sogeti Nederland BV) geeft de voorkeur aan een ontwikkelmethode die iteratief is en gebruik maakt van prototyping. Op deze manier krijgt hij snel inzicht in de resultaten en kan hij, als het eindresultaat van het project dreigt te mislukken, direct ingrijpen en waar nodig is bijsturen.

Hierdoor vallen Scrum en de Watervalmethode af. Daarnaast gaan beide methodes uit van een zevental fases met ieder hun eigen expert. Bij Scrum gaat het zelfs nog verder door deze zeven experts naast elkaar te laten werken (alle fases tegelijk starten) in plaats van achter elkaar (fase na fase). Het is voor mij niet haalbaar om mijzelf met zeven fasen tegelijk bezig te houden.

Dan blijven RUP, DSDM en XP over. Alle drie de methoden zijn iteratief, wat de klant noodzakelijk vindt voor deze opdracht. Van deze drie valt eXtreme Programming af. Bij deze methode wordt er te veel uitgegaan van 'pair-programming'. Dit gaat niet goed werken, het is immers een individueel project. Daarnaast zou de klant deel uit moeten maken van het ontwikkelteam.

Uiteraard is het belangrijk dat de klant betrokken wordt bij het project en beschikbaar is voor vragen. Maar het is voor dit project niet haalbaar om te zorgen dat de klant 40 uur per week bereikbaar is en dan klaar staat om in het ontwikkelteam plaats te nemen.

RUP en DSDM zijn allebei interessant en leerzaam genoeg om toe te passen op dit project. Daarnaast komen de fasen in grote lijnen overeen (eigenlijk is dit bij alle methodes zo). Het grote voordeel van DSDM is dat er gebruik gemaakt wordt van het MoSCoW principe en timeboxing. Dit is voor dit project erg handig, omdat de functionaliteiten wel redelijk vaststaan, maar het traject een onderzoekend karakter heeft. Hierdoor zou het kunnen gebeuren dat functionaliteiten tijdens de ontwikkeling veranderen. Met behulp van het MoSCoW principe en timeboxing zijn deze veranderende functionaliteiten goed in de hand te houden. Daarom zal er bij dit project gebruik gemaakt worden van de ontwikkelmethode DSDM.

Wel moet er vermeld worden dat hoogstwaarschijnlijk niet alle elementen van de ontwikkelmethode precies uitgevoerd gaan worden. Dit omdat bepaalde delen misschien niet uitgevoerd kunnen worden of omdat deze voor dit project niet van belang zijn. De ontwikkelmethode die tijdens dit project gebruikt gaat worden, bevat dus elementen van DSDM.

## Bronnenlijst

**Personen:**

Sander Bosma  
Chris van Aart  
Thomas Hood  
Ard Peek  
Gerard Wagenaar

**Rol:**

(Algemeen) Bedrijfsbegeleider / Unitmanager Rotterdam  
Bedrijfsbegeleider / Innovatie & Inspiratie Rotterdam  
Algemeen projectleider D-CIS Lab  
Technisch begeleider Webattach B.V.  
Docentbegeleider

**Websites:**

[www.google.nl](http://www.google.nl)  
[www.wikipedia.nl](http://www.wikipedia.nl)  
[www.nl.wikipedia.org/wiki/PRINCE2](http://www.nl.wikipedia.org/wiki/PRINCE2)  
[www.nl.wikipedia.org/wiki/Scrum\\_%28softwareontwikkelmethode%29](http://www.nl.wikipedia.org/wiki/Scrum_%28softwareontwikkelmethode%29)  
[www.nl.wikipedia.org/wiki/Extreme\\_Programming](http://www.nl.wikipedia.org/wiki/Extreme_Programming)  
[www.nl.wikipedia.org/wiki/Rational\\_Unified\\_Process](http://www.nl.wikipedia.org/wiki/Rational_Unified_Process)  
[www.nl.wikipedia.org/wiki/Watervalmethode](http://www.nl.wikipedia.org/wiki/Watervalmethode)  
[www.nl.wikipedia.org/wiki/System\\_Development\\_Methodology](http://www.nl.wikipedia.org/wiki/System_Development_Methodology)  
[www.nl.wikipedia.org/wiki/Dynamic\\_Systems\\_Development\\_Method](http://www.nl.wikipedia.org/wiki/Dynamic_Systems_Development_Method)  
[www.u-vision.be](http://www.u-vision.be)  
[www.u-vision.be/pdf/ProjectManagement.pdf](http://www.u-vision.be/pdf/ProjectManagement.pdf)