# Rapid Application Development (RAD) voor het practicum Software Engineering

Bastiaan Schönhage

practicum SE 1997


## Samenvatting

Dit artikel geeft een inleiding voor het gebruik van de *rapid application development* methode bij het practicum SE 1997. Hiertoe wordt eerst een korte beschrijving van RAD gegeven waarin de belangrijkste kenmerken naar boven komen. In het tweede deel staat hoe we de methodologie willen toepassen voor het practicum SE van 1997.


## 1 Rapid Application Development

Rapid Application Development, voortaan kortweg RAD genoemd, is een software ontwikkelmethode. In vergelijking met een traditionele aanpak zoals de waterval methode, streeft RAD naar een snellere ontwikkeling en hogere kwaliteit van de systemen. Snelle ontwikkeling betekent in deze context niet *quick and dirty* maar veeleer systemen van hoge kwaliteit. Het kernwoord kwaliteit betekent nu dan ook niet:

> *precies voldoen aan de specificaties die vooraf zijn gesteld*

zoals vaak wordt gedacht, maar kwaliteit is

> *een oplossing die goed genoeg is om de bedrijfsdoelstellingen op het moment van oplevering te ondersteunen.*

RAD is gebaseerd op de volgende vier essentiële aspecten, ook wel de pijlers van de rapid application development genoemd: methodologie, mensen, management en tools.

### 1.1 Methodologie

De doelen die de RAD methodologie nastreeft met betrekking tot het creëren van software systemen zijn: *high speed, high quality* en *low cost.* Om dit te bereiken worden bij RAD-ontwikkeling de fasen van requirements analyse tot invoering een aantal malen doorlopen en wordt het systeem in incrementen opgeleverd. Belangrijk hierbij is dat de gebruikers bij alle fasen betrokken worden om het systeem in een vroeg stadium in de juiste richting te sturen. Nadat alle fasen eenmaal doorlopen zijn, wordt een prototype uitgebracht dat door de gebruikers kan worden geëvalueerd. De resultaten van deze evaluatie kunnen als invoer worden gebruikt voor een volgende requirements analyse; de eisen van mensen blijken namelijk vaak te veranderen zodra er enige tijd met een (deel van het) systeem is gewerkt. De prototypes geven vaak vast een goed beeld van het uiteindelijke systeem omdat in veel gevallen de zogenaamde 80/20 regel geldt: 80% van de functionaliteit van een systeem kan worden gerealiseerd met 20% van de code.

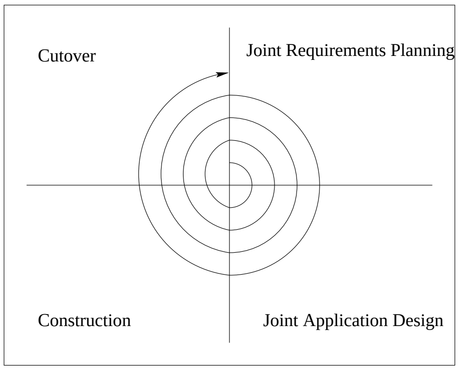
De RAD methodologie bestaat uit het een aantal keer doorlopen van de volgende vier fasen die grofweg overeenkomen met de fasen uit het waterval model.

1. **Joint Requirements Planning (JRP)**
   In deze fase worden de eisen van het nieuwe systeem vastgelegd door de gebruikers die ondersteund worden door IT professionals. Belangrijk hierbij is dat de functionele eisen een prioriteit krijgen zodat later duidelijk is welke eisen als eerste geimplementeerd dienen te worden.

2. **Joint Application Design (JAD)**
   Tijdens deze fase wordt een ontwerp van het systeem gemaakt. Ook bij deze fase is het van belang dat gebruikers participeren in de workshop die het ontwerp moet voortbrengen.

3. **Construction Phase**
   Tijdens de constructie fase wordt een werkend systeem (prototype) gebouwd dat voldoet aan de eisen en het ontwerp van de vorige fasen. In deze fase wordt het gebruik van geavanceerde tools sterk aanbevolen.

4. **Cutover Phase**
   Bij deze fase wordt het systeem ingevoerd. Hierbij dienen gebruikers te worden getraind en moet het systeem grondig worden getest en geëvalueerd.




Figuur 1: De fasen van RAD

## 1.2 Mensen

De mensen die bijdragen in een RAD project zijn gegroepeerd in kleine teams van specialisten. Vier belangrijke teams die samenvallen met de vier belangrijke fasen in RAD zijn:

- Requirements Planning Team (JRP Team)
- User Design Team (JAD Team)
- Construction Team (SWAT Team)
- het Cutover Team.

In analogie met het leger wordt een constructie team vaak een SWAT-team genoemd, SWAT staat voor *Skilled With Advanced Tools*. Belangrijk is dat bij diverse activiteiten zowel de gebruikers als de ontwikkelaars in één team zitten.

## 1.3 Management

Het management van een RAD project moet betrokken zijn bij de ontwikkeling van het systeem. Het dient de kwaliteit en deadlines van het project te bewaken. Belangrijk is dat beslissingen snel genomen kunnen worden.

### 1.4 Tools

Geavanceerde gereedschappen en hulpmiddelen ondersteunen de RAD methode. Zo kunnen code generatoren gebruikt worden voor prototypes en kunnen consistency checks gebruikt worden om de documentatie consistent te houden.

### 1.5 Workshops

Een belangrijke vorm om tot snelle besluiten te kunnen komen is door gebruik te maken van workshops. Vooral in de eerste twee fasen (definitie van eisen en ontwerp) worden workshops veel gebruikt. Workshops zijn nuttig voor het snel consolideren van kennis en het uitdragen van kennis naar de overige leden van de groep. Tevens kan er tijdens een workshop overeenstemming worden bereikt omtrent bepaalde zaken. Met andere woorden, tijdens een workshop kan een draagvlak voor beslissingen gecreeerd worden.

Opdat de workshops geen eindeloze vergaderingen worden, moeten de workshops aan een aantal vormeisen voldoen. Zo wordt gebruik gemaakt van *timeboxing*. Hierbij wordt voor een bepaald onderwerp een bepaalde tijd afgesproken, zodra deze tijd voorbij is moet van het onderwerp worden afgestapt (dit om oeverloze discussies tegen te gaan). Indien het onderwerp naar de mening van de meerderheid nog niet voldoende is uitgediept, kan het in een later stadium alsnog aan bod worden gesteld. Een andere manier om nutteloze discussies tegen te gaan is het *red card* principe. Hierbij kan iemand een rode kaart opsteken indien volgens hem de discussie niet meer relevant is. Als een bepaald percentage de rode kaart heeft opgestoken wordt de discussie over dit onderwerp gestaakt en wordt verder gegaan met het volgende onderwerp op de agenda.

## 2 RAD bij SE

Omdat een RAD-achtige aanpak zeer geschikt lijkt voor het practicum Software Engineering zal met ingang van cursus-jaar 1997 de overgang van een waterval naar een RAD methode gemaakt worden. We volgen hierbij niet de volledige RAD methode, maar zullen de belangrijkste concepten gebruiken binnen een practicum gerichte aanpak. Vooral op het *advanced-tool* gebied zullen we afwijken van de aanbevelingen van RAD omdat voor een eenmalig practicum het aanleren van krachtige maar ingewikkelde tools de inspanning niet loont. Tevens moeten de studenten tijdens het practicum al zeer veel technische dingen leren zoals programmeren in C++ en hush zodat het leren van geavanceerde tools te veel wordt.

### 2.1 Tijdschema

Het practicum SE duurt in totaal 14 weken. De bedoeling is dat tijdens deze 14 weken de groepen de totale cyclus twee keer doorlopen. De eerste drie weken van het practicum worden gebruikt voor het leren van C++ en hush. Vervolgens komt er een eerste cykel die 4 weken duurt. Na deze vier weken moet er een prototype klaar zijn dat een beeld geeft van het programma. Dit prototype bestaat dan ook voornamelijk uit een user-interface en de belangrijkste interacties van de gebruiker met het systeem. Aan onderliggende datastructuren, file I/O en ingewikkelde functionaliteit wordt in deze eerste fase nog geen aandacht besteed. In de tweede fase, wordt het prototype gebruikt als beginpunt. Dit prototype wordt dan uitgebreid met de balangrijkste functionaliteit zoals gedefinieerd staat in de definitie van eisen. De onderstaande tabel geeft een overzicht van hoe de fasen verdeeld worden over de weken.

### 2.2 Begeleiding

De begeleiding van de groepen zal op twee verschillende manieren plaatsvinden. Ten eerste zal er tijdens de gemeenschappelijke workshops iemand van de begeleiding aanwezig zijn om de rol van de klant/gebruiker te spelen. Hiernaast kan hij of zij als begeleider ingegrijpen zodra de workshop dreigt te mislukken. Wanneer er geen workshops zijn kan een groep bij een begeleider langs gaan

Tabel 1: Tijdschema SE najaar 1997

| deadline | #weken | omschrijving |
|---|---|---|
| 19-9 | 3 | Exploratie C++ en hush |
| 26-9 | 1 | JRP |
| 3-10 | 1 | JAD |
| 10-10 | 1 | Construction |
| 17-10 | 1 | Cutover |
| 24-10 | 1 | JRP |
| 7-11 | 2 | JAD |
| 28-11 | 3 | Construction |
| 5-12 | 1 | Cutover |

met vragen en problemen. Ook technische problemen bij de constructie worden op deze wijze behandeld. Om de terugkoppeling met andere groepen en het college te verbeteren worden de groepen verplicht om hun tussenresultaten aan de andere studenten te presenteren als laatste uur van enkele colleges.

## Referenties

[1] J. Martin, *Rapid Application Development*, Macmillan, 1991.