Tentamen Software Engineering

5 juli 2002

In dit tentamen staat de case "You Got To Move (YGTM)" centraal. Vragen over modelleertechnieken hebben betrekking op deze case.

> In de jaarlijkse ledenvergadering van beroepsverhuizers kwam aan de orde dat kleine verhuizers het steeds moeilijker krijgen. Het grootste probleem waar zij mee te maken hebben is het efficiënt inzetten van resources, d.w.z. verhuizers en materieel. Het is hollen of stilstaan. Ook zijn de investeringen in modern materieel, zoals grote hijskranen en liften, voor een kleine verhuizer economisch niet rendabel. Een duidelijke conclusie van de vergadering was dat kleine verhuizers zouden moeten samenwerken, zonder daarbij hun identiteit en zelfstandigheid te verliezen.
>
> Zoals te doen gebruikelijk op vergaderingen wordt voor het oplossen van een lastig probleem een werkgroep opgericht; in dit geval de werkgroep *You Got To Move* (YGTM). YGTM ging voortvarend van start en onderkende dat het probleem voornamelijk een resource sharing probleem is dat met moderne ICT middelen kan worden opgelost. Het idee is dat verhuisbedrijven resources hebben die kunnen worden verhuurd: aan klanten die hun spullen van A naar B laten verhuizen en aan aangesloten verhuisbedrijven die tijdelijk te kampen hebben met ondercapaciteit. Voor het gebruik van deze resources moet uiteraard worden betaald. Een verhuisbedrijf mag zelf bepalen welke tarieven het aan haar klanten wil doorberekenen, terwijl voor de aangesloten verhuisbedrijven eenduidige (goedkopere) prijsafspraken gelden. Het benodigde dure materieel wordt door de aangesloten verhuisbedrijven gezamenlijk ingekocht. Dit materieel wordt beheerd en vervolgens verhuurd door een hiervoor speciaal in het leven geroepen bedrijf.
>
> De expertise van de leden van de werkgroep YGTM is verhuizen. Omdat ze van automatiseren weinig verstand hebben, geven ze de firma Van Vliet & De Bruin opdracht de mogelijkheden van een modern - op Internet gebaseerd systeem - te onderzoeken. Naast de functionele eisen zoals hierboven beschreven, moet het systeem ook nog voldoen aan een aantal randvoorwaarden en niet-functionele eisen. Een belangrijke randvoorwaarde is dat een verhuizer zijn zelfstandigheid behoudt. Dit betekent dat een gecentraliseerde oplossing uit den boze is. Iedere verhuizer krijgt een systeem dat in principe stand-alone is te gebruiken. Als resources moeten worden ingehuurd, moet uiteraard contact worden gezocht met systemen van andere verhuizers. Verder zijn hoge betrouwbaarheid, performance en veiligheid van essentieel belang.

Let op

Merk op dat deze probleemstelling een aantal onduidelijkheden en ambiguïteiten bevat. Bij het beantwoorden van de vragen ben je vrij om onvolledigheden zelf in/aan te vullen. Vergeet niet gemaakte aannames kort toe te lichten.

Vraag 1: Requirements Engineering

a) Identificeer de stakeholders in de bovenstaande YGTM-case en geef een profiel (de karakteristieke eigenschappen van de stakeholders die relevant zijn in de context van de YGTM-case).
b) Wat is het verschil tussen functionele en niet-functionele eisen?
c) Een overduidelijke functionele eis voor het YGTM-systeem is dat een klant een offerte kan aanvragen voor het laten verhuizen van goederen. Werk deze "Must Have" uit d.m.v. een use case in de vorm van een opeenvolging van genummerde stappen en alternatieven voor bepaalde stappen.
d) Stel niet-functionele eisen op m.b.t. de aspecten hoge betrouwbaarheid, performance en veiligheid. Van ieder aspect wordt tenminste één (gekwantificeerde) eis gevraagd.

Vraag 2: Analyse

"Van Vliet & De Bruin" is nog niet zeker van zijn zaak. De software engineers weten niet of de YGTM-case nu het beste met OOA/D kan worden geanalyseerd en ontworpen of beter met SA/SD. Er wordt besloten om van beide gebruik te maken.

a) Analyseer de YGTM-case m.b.v. een class diagram (met gebruikmaking van de specialisatie/generalisatie (is-a), aggregatie (part-of) en/of associatie (is-related-with) relatie). Het gaat in deze vraag om een conceptueel model met de nadruk op concepten en hun onderlinge relaties. Class attributen en methoden hoeven niet in het diagram te worden opgenomen. Geef korte omschrijvingen van de classes en hun onderlinge relaties. Verklaar de gebruikte notatie.
b) Formaliseer de use case uit vraag 1.c m.b.v. één van de dynamische OO technieken (bijvoorbeeld, interaction (scenario) diagram of activity diagram). Verklaar de gebruikte notatie.
c) Analyseer dezelfde YGTM-case met een data flow diagram (DFD). Er wordt hier een uitwerking verwacht die bestaat uit tenminste twee niveaus: een level 1 context diagram en een level 2+ DFD met een aantal data stores en processen. Verklaar de gebruikte notatie.
d) Stel jij bent een software engineer in dienst van "Van Vliet & De Bruin" en je krijgt de opdracht om een afweging te maken tussen OOA/D en SA/SD voor dit specifieke geval. Beschrijf kort maar krachtig je overwegingen in ongeveer 50 tot 100 woorden.

Vraag 3: Software Architectuur

a) Wat wordt verstaan onder een architectural/design pattern? Waar dienen ze voor?
b) Stel een (globale) software architectuur op voor het YGTM-systeem m.b.v. architectural/design patterns. Maak aan de hand van deze architectuur aannemelijk dat het te bouwen YGTM-systeem aan de niet-functionele eisen voldoet die zijn opgesteld in vraag 1.d. Verklaar de gebruikte notatie.

Vraag 4: Testen

a) Stel een test plan op voor het YGTM-systeem. Het test plan moet ingaan op de doelstellingen (test objectives), aanpak (test approach) en de middelen (test resources).

Vraag 5: Software Maintenance

a) Leg uit waarom in de praktijk software altijd zal evolueren.
b) Wat voor soorten maintenance kunnen we onderscheiden? Geef van iedere soort een korte omschrijving.

Puntentelling

Met dit tentamen zijn 90 punten te verdienen. Het eindcijfer wordt als volgt bepaald:

Eindcijfer = (#punten + 10) / 10	zonder huiswerkopdracht bonus
Eindcijfer = min((#punten + 15) / 10 , 10)	met huiswerkopdracht bonus

Het gewicht van de vragen is als volgt verdeeld:

1a: 5	b: 2	c: 10	d: 3
2a: 10	b: 5	c: 10	d: 5
3a: 5	b: 15		
4a: 10			
5a: 2	b: 8		