Adres: | Studierichting: | Tentamen: SE
Postcode en | | Datum: 14-04-2005
Woonplaats: | Jaar van eerste inschrijving: | Naam docent:

1 a - requirements specificatie
- development
0,5 - validatie
- evolutie
- verificatie

b Bij gebruik van het ~~Het~~ watervalmodel, ~~is~~ is het lastig om gedurende het proces de requirements te
0,2 veranderen. Bij automotive systemen staan de eisen aan het systeem vrijwel vast. Ze kunnen ~~voor~~ voor het ontwikkelproces duidelijk geformuleerd worden en zullen waarschijnlijk weinig of niet veranderen. Daarom is de watervalmethode hier een geschikte methode. ✓ formele methoden + [illegible]

c Het systeem moet opgedeeld kunnen worden in
0,2 verschillende componenten. Delen van het systeem moeten ontwikkeld en getest kunnen worden.

d In elke iteratie vaststellen welk deel van
0,1 het systeem ontwikkeld zal worden, welke requirements, gedetailleerd ontwerp van component, implementatie en integratietest doen.
Als voorbereiding requirements elicitatie en analyse. Architectuur maken, high-level design. ✓ roadmap

e - kans op falen van het proces is kleiner
- je kunt vrijwel altijd een werkend product
0,2 leveren (vanaf na een bepaalde ~~eerste~~ iteratie)
- fouten in het systeem worden in een vroeg stadium ontdekt omdat er bijna ~~bijna~~ gedurende het hele proces getest wordt

2 a - het volume van het alarm ~~kan pas~~ gaat na 2 minuten omhoog ~~uitge~~ — functioneel
- het opblazen van de airbags moet binnen een halve seconde gebeuren (performance)?
- de airbag moet paars zijn
- het alarm mag niet hoger dan 200 decibel

b de tweede van onderdeel a, want daar is speciale, dure apparatuur voor nodig.
De eerste en laatste van a kunnen gemakkelijk softwarematig geregeld worden.
De kleur van de airbag zal ook niet moeilijk te realiseren zijn.

c Een toestandsdiagram lijkt me handig, want het gedrag van het systeem is erg belangrijk. Het systeem moet in elke situatie op een bepaalde manier reageren.

d Requirements moeten goed gedocumenteerd worden. Bepaalde weergaven, zoals dependency graphs, zijn erg nuttig. Wanneer er een change request komt, kan makkelijk gezien worden wat de impact is. Dat maakt het makkelijker om te beslissen of aanpassingen doorgevoerd moeten worden. Traceability is erg belangrijk.

e Bepaalde fases moeten vaker doorlopen worden. Je weet van te voren niet hoe vaak, dus dat kun je moeilijk plannen. Er zou gebruik gemaakt kunnen worden van het spiraalmodel, dat continu is.

Afdeling Wiskunde en Informatica R.U.G.

Naam:	Studentnummer:	Bladnr.: 2
Adres:	Studierichting:	Tentamen:
Postcode en		Datum:
Woonplaats:	Jaar van eerste inschrijving:	Naam docent:

3 a In een groot team moeten taken zo verdeeld worden dat niet te veel mensen (te erg) afhankelijk van elkaar zijn. Want het systeem kan zo gemakkelijk instorten! Een goede taakverdeling is nodig, en daarvoor een expliciet architectuurontwerp (Bovendien moet duidelijk zijn waar iedereen mee bezig is.)

b
- performance
- ~~fie~~ reliability
- maintainability
- robustness
- safety
- security

c - Client-server model

voor:
- datamanagement is straightforward
- makkelijk om servers (en aus services) toe te voegen

tegen:
- veel communicatie → minder performance
- zelfde data op meerdere plekken → minder maintainability

- Repository model

voor:
- alle gebruikers kunnen gemakkelijk bij alle data → weinig security
- data centraal op één plek → maintainability

tegen:
- moeilijk om datatransformaties uit te voeren → lage performance
- als het repositorysysteem faalt, kan niemand meer bij de data → minder reliability

+0,3

- het proces verloopt sneller omdat je niet (meerdere malen) achteraf hoeft te testen

0/2 mogelijk nadeel:
Wanneer requirements veranderen, moeten nu ook de tests herschreven worden voor je verder kunt met implementeren.

b Unit test werken meestal zo, behalve dat de tests niet per se van te voren geschreven hoeven worden. Integratie tests kunnen pas uitgevoerd worden wanneer de implementatie ver genoeg gevorderd is. ~~Testen doorgaan~~ Het heeft daarvoor niet heel veel zin om deze tests voor het hele implementatieproces te schrijven. Het levert dan misschien juist alleen maar extra werk, want de tests zouden misschien veranderd moeten worden, doordat requirements veranderen, voordat ze gebruikt worden. Het beschreven principe is dus meer toepasbaar op unit tests.

c Vanuit de use-cases kunnen test cases gemaakt worden voor integration tests.

5 a Bij een iteratief proces heb je vele milestones. Aan het einde van elke iteratie heb je iets wat opgeleverd kan worden. Het is belangrijk dat bijgehouden wordt welke versies bij elkaar horen en wat bij elke iteratie hoort. Change management is daarom erg belangrijk. Veranderingen in requirements moeten goed gedocumenteerd worden (en in de goede versies). Als dit goed gebeurt, kunnen ook dingen die ervan afhankelijk zijn op de goede manier aangepast worden.

b In een groot softwareproject heb je ook vele documenten en code. Omdat de verschillende bestanden afhankelijk van elkaar zijn, moet alles goed onderhouden worden.

Afdeling Wiskunde en Informatica R.U.G.

Naam:	Studentnummer:	Bladnr.: 3
Adres:	Studierichting:	Tentamen:
Postcode en		Datum:
Woonplaats:	Jaar van eerste inschrijving:	Naam docent:

vervolg 5 b - versioning: versies van documenten en code bijhouden

~~- bijhouden wat er veranderd is~~

4,3 - branches bijhouden

- baselines bijhouden

  L.i. de functie daar van?

0,1 je zo veel mogelijk gebruikmaken van wat je al hebt. Je wilt geen nieuwe architectuur vanaf de grond opbouwen. Daarom is een goede architectuur belangrijk. Een architectuur die gemakkelijk aangepast kan worden en een waar je op voort kunt bouwen.

↳ De architectuur hoeft je maar één keer op te stellen

b Als hiermee rekening gehouden wordt, kunnen
0,2 nieuwe producten makkelijker, ~~en~~ sneller en goedkoper ontwikkeld worden. De extra kosten bij het maken van de architectuur wanneer rekening gehouden wordt met mogelijke variabiliteiten, moeten afgewogen worden tegen de verwachte besparingen die het oplevert en hoeveel hiervan gebruik gemaakt zal worden.

c Verschillende producten in een product familie
0,1 ~~xxxx een gexx gxxx~~ verschillen zeer weinig van elkaar. Het ontwikkelproces voor twee verschillende producten is ~~ex~~ tot op een bepaald punt gelijk. Daarna vindt specialisatie plaats en ontkoppelen ~~volgen~~ de ~~producten een~~ ontwikkelprocessen zich.

↳ domain engineering

d Wanneer een change request binnen komt, moet
X er analyse gedaan worden ~~xxxxxxx~~ (Eerst wordt natuurlijk gekeken of de change consistent is ~~en~~ met bestaande requirements en of het valid is.) Er wordt gekeken waar de change invloed op heeft en wat de impact is. Als het om een productfamilie gaat, kan het doorvoeren van de verandering ~~xxx~~ ervoor zorgen dat het product niet meer binnen de familie past. Er moet dan mogelijk ook met ~~xxxx~~ andere producten binnen de familie rekening gehouden worden bij het beslissen om een verandering wel of niet door te voeren. Dit maakt het lastiger dan bij een "normaal" proces.

↳ wat gebeurt er met een change request op een specifiek product?

e Wanneer bij ~~een product uit~~ een productfamilie
0,2 veranderingen doorgevoerd worden in de architectuur

Afdeling Wiskunde en Informatica R.U.G.

Naam:	Studentnummer:	Bladnr.: 4
Adres:	Studierichting:	Tentamen:
Postcode en		Datum:
Woonplaats:	Jaar van eerste inschrijving:	Naam docent:

vervolg 6 e uit de familie gebeuren. Dit betekent dat voor alle producten documenten en code aangepast moeten worden. Er komen dan heel veel nieuwe versies. Omdat erg veel aangepast moet worden, is het lastiger om allemaal bij te houden. Configuration management is dan moeilijker.

7 a - ~~Quality~~ procedures en standaarden establishen
- procedures en standaarden selecteren voor een project, documenteren
- ervoor zorgen dat de geselecteerde procedures en standaarden ~~negeten~~ gevolgd worden (bijvoorbeeld door reviews)

b Helemaal aan het begin moet een Quality Assurance Plan geschreven worden (eerste twee fases als in a beschreven). Tijdens documentatie en implementatie moet ervoor gezorgd worden dat het Quality Assurance Plan gevolgd wordt (derde fase van a).

c - met een goede documentstandaard, ~~zal~~ zullen geen belangrijke onderdelen in het document ontbreken (waardoor de kwaliteit minder zal zijn)
- een goede, heldere ~~s~~structuur en ordening van een document (waar een document-standaard voor kan zorgen), kan ~~ervoor~~ zorgen voor beter inzicht en dus een betere kwaliteit van het ontwerp

d Een processtandaard kan zijn dat een document een x aantal keren gereviewed moet worden. Dit kan zorgen voor betere documenten. Omdat uiteindelijk alles in het proces één of meerdere documenten als basis heeft, kan de kwaliteit van het eindresultaat hoger zijn ~~door~~ wanneer de proces-standaard gevolgd wordt dan wanneer dat niet gebeurd.