Containing

Gemaakt door:
Groep 5
Joshua Bergsma
Remco de Bruin
Melinda de Roo
Arjen Pander
Jeffrey Harders
Yme van der Graaf

Inhoudsopgave

1. Inleiding

In het 2de jaar van de informatica opleiding aan de NHL, krijgen wij als studenten een grote opdracht genaamd containing. De opdracht is aanzienlijk groter dan wat we gewend zijn, dus bijna alles moet daarom anders geregeld en gepland worden. Naast taakverdeling, grotere groepen en meer programmeer werk, is er ook een grotere kans op fouten, omdat veel code los van elkaar wordt geschreven, en dan later samen veel problemen op kan leveren. Om dit tegen te gaan is ons aangeraden veel meer met testen te werken. Dit doen wij onder andere met hulp van ATOS, en doormiddel van JUnit.

2. IEEE

Functies:

- Uitladen XML.
- Controller/server toot de gegevens van de XML real time.
- Controller/server stuurt XML door naar Simulator/client.
- Simulator/client ontvangt van de Controller/server.
- Simulator/client moet de gegevens van de Controller/server simuleren.
- Kranen moeten containers van schepen aftillen.
- AGV's moeten deze containers verplaatsen.
- Containers moeten van AGV's af worden gehaald en op voertuigen worden geplaatst.
- Containers moeten van AGV's af worden gehaald en op het opslag terrein worden geplaatst.
- AGV's moeten met container langzamer rijden dan zonder container.
- Er moeten 100 AGV's tegelijk kunnen werken.
- Er moet op een mobiel apparaat een snapshot van de data kunnen worden bekeken.
- De AGV's moeten de kortste/snelste route kiezen.

Wat te testen:

- Uitladen XML.
- Controller/server toot de gegevens van de XML real time.
- Controller/server stuurt XML door naar Simulator/client.
- Simulator/client ontvangt van de Controller/server.
- Kranen moeten containers van schepen aftillen.
- AGV's moeten deze containers verplaatsen.
- Containers moeten van AGV's af worden gehaald en op voertuigen worden geplaatst.
- Containers moeten van AGV's af worden gehaald en op het opslag terrein worden geplaatst.
- Er moeten 100 AGV's tegelijk kunnen werken.
- Er moet op een mobiel apparaat een snapshot van de data kunnen worden bekeken.

Wat niet te testen:

- *Simulator/client moet de gegevens van de Controller/server simuleren:*
  Als dit niet werkt, zou er niks gebeuren, als het wel werkt, zou alles moeten doen wat het hoort te doen. Dit is niet echt een vorm van testen, aangezien je in 1 oog opslag ziet of het werkt of niet.

- *AGV's moeten met container langzamer rijden dan zonder container:*
  Ook hier kun je niet goed testen of het werkt, je moet het gewoon zien aangezien we niet een snelheidsmeter willen inbouwen.

- *De AGV's moeten de kortste/snelste route kiezen:*
  Wij schrijven deze code op zo'n manier dat hij alleen de beste route kan kiezen, wat zou beteken dat hij niks doet als hij die route niet kiest, wat ook weer betekent dat je het niet echt test.

Aanpak:

Zijn er specifieke tools of software die gebruikt moeten worden?

- Java.
- JUnit.

Hebben deze programma's extra training nodig?

- Nee.

Welke variabelen worden verzameld?

- Snelheid van de voertuigen.
- Het aantal containers per platform.

Waar worden deze variabelen verzameld?

- In de Controller/server, sommigen doorgestuurd naar de Simulator/client en/of de Management Interface.

Hoeveel verschillende configuraties worden er getest?

- 7, namelijk door invoering van de 7 XML bestanden.

Hardware:

- De software moet werken op de computer van ieder groepslid van de projectgroep.

Software:

- Moet in Java worden geschreven en getest worden met JUnit.

Hoe worden negatieve of onlogische resultaten verwerkt?

- Er wordt voor gezorgd dat deze met exceptions worden opgevangen en aangegeven zodat wij die kunnen repareren.

Wanneer is het goed:

In onze ogen is alles voldoende, als wij de 7 aangeleverde XML bestanden probleemloos kunnen inladen en simuleren. Dit betekent ook dat alle programma's goed samenwerken, een acceptabele snelheid (geen lag) hebben en de Management Interface de statistieken toont.

3. Dekkingsgraad (Coverage)

Coverage is een manier om te kijken welke instructie het programma door heen gaat. Daarmee wordt bedoeld dat er wordt gekeken dat er bijvoorbeeld een if-statement is waar die nooit inkomt en dus nutteloos is. Daarbij wordt er gebruik gemaakt van een Eclipse plugin genaamd EclEmma. Door EclEmma te laten runnen kan er gekeken worden welke instructie wel wordt uitgevoerd en welke niet. Het resultaat(afbeelding 1) wordt namelijk berekend in percentages. Dus hoe hoger het percentage, hoe meer instructies er zijn gebruikt.

Coverage	Covered Instructio...	Missed Instructions	Total Instructions
12,1 %	46	333	379
76,4 %	1.045	323	1.368
80,7 %	1.045	250	1.295
80,7 %	1.045	250	1.295
0,0 %	0	3	3
0,0 %	0	3	3
100,0 %	3	0	3
100,0 %	3	0	3
100,0 %	170	0	170
98,2 %	650	12	662
0,0 %	0	42	42
40,3 %	125	185	310
94,9 %	94	5	99

Afbeelding 1.

4. Optimalisatie (Optimalisation)

De combinatie van een server en een client geven veel kansen op fouten en minpunten. Om dat zo veel mogelijk te voorkomen proberen wij door middel van optimalisatie ons product zo goed, vlot en stabiel mogelijk te maken.

Om de FPS in de client hoog te houden, gebruiken we breakpoints. Dit doen we om tussen de methodes, assets en animaties steeds datgene te vinden wat veel ruimte/data inneemt. De models worden waar mogelijk zodanig aangepast dat dezelfde visuele kwaliteit wordt behouden, terwijl er minder lijnen en triangels hoeven te worden ingeladen. Ook proberen we de client niks te laten renderen wat niet te zien is, zoals bv. de achterkant van een container of de onderkant van het platform.

Er kan veel snelheid verloren gaan in de communicatie tussen de server en de client. Om dit (grotendeels) te voorkomen hebben we de connectie/netwerk-code door één persoon laten schrijven. Dit om zo een zo perfect mogelijke compatibiliteit en stabiliteit te garanderen.

We verdelen ook de “taken” tussen de server en de client; zo laten we de ene zware helft (het rekenwerk) door de server doen en de andere zware helft (het simuleren) door de client. Ook dit zou voor meer stabiliteit en snelheid in het programma moeten zorgen.

Als laatste is er voor gekozen om te switchen van DOM xml parsen naar sax. Ook deze keus is gemaakt omdat dit naar onze verwachting betere resultaten op zal leveren. Dit denken wij omdat het een veel steviger en sneller systeem is.

5. Test Technieken (Test Techniques)

Tijdens dit project moet er veel getest worden. Gelukkig zijn er technieken bedacht om dit een stuk makkelijker te maken.

Grenswaarde-analyse:
Er wordt namelijk gebruik gemaakt van een Grenswaarde-analyse, daarmee wordt bedoeld dat je een bepaalde statement hebt, bv. "groter als 20", waarbij je gaat kijken wat er gebeurd als er een getal kleiner, gelijk of groter is als de grenswaarde.

Compatibiliteit test:
Ook wordt er gebruik gemaakt van de Compatibiliteit test, deze test is vooral belangrijk voor de Management Interface. Dit komt omdat er verschillende browsers in de omloop zijn en deze ook allemaal gebruikt worden. Conclusie: je weet niet wie welke browser gebruikt en er moet rekening gehouden worden met elke mogelijke browser.

Unit testen:
Een andere vorm van testen die gebruikt wordt is unit testen. Het doel hiervan is om units onafhankelijk van elkaar te testen om te kijken of het goed werkt. Hierbij wordt vooral gekeken of de software goed blijft werken bij een goede/foute invoer van bepaalde gegevens, zoals bv. een verkeerde positie van een container.

Regressie testen:
Naast deze vormen wordt tijdens dit project ook veel gebruikt gemaakt van regressie testen, dit als er bv. iets wordt aangepast in de server. Met regressie testen kun je kijken of de client dan nog (goed) werkt. Regressie testen wordt ook gebruikt als er bv. een locatie wordt veranderd in die client, ook dan moet gekeken worden of de server nog steeds hetzelfde doet.

6. Test Resultaat (Test Deliverables)

Hieronder een voorbeeld van wat wij gaan opleveren als Test Deliverables. Zo'n soort tabel wordt er gemaakt voor de volgende onderdelen:

- Controller (Server)
- Protocol (Server – Client)
- Simulator (Client)
- Management Interface (Mobiel)

Testverslag 1			Datum : 3-12-2013	
Controller (Server):				
Punten	**Status**	**Omschrijving**	**Datum**	**Prioriteit**
	(Open/Close)	(Bij Open probleem, bij Close eventueel opmerking)	(Datum van Close of wanneer dit punt in de planning staat)	(Hoog, middel, laag)
Uitladen XML	Closed	Het werkt!	2-12-2013	Hoog
Controller toont de gegevens van de XML real time	Open	Wordt al wel gedisplayed, maar nog niet real time.	10-12-2013	Laag
Communicatie tussen de Controller en de Client d.m.v. een protocol	Closed	Er kunnen 'berichten' worden verstuurd!	3-12-2013	Hoog
Communicatie tussen de Controller en de Management Interface	Closed	Het werkt!	3-12-2013	Middel