Simulating Smart Connected Containers

Eindverslag bachelorproject IN3405

Rob Post (1358383)

Oskar van Rest (1361074)

Tim Roorda (1358405)

TECHNISCHE UNIVERSITEIT DELFT,
FACULTEIT ELEKTROTECHNIEK, WISKUNDE EN INFORMATICA

Examencommissie

Ir. C. Pronk

Dr. M.M. de Weerdt

TECHNISCHE UNIVERSITEIT DELFT

Ir. L. Blom

Dr. A.F. van Lier MCM CMC

CENTRIC IT SOLUTIONS

23 juni 2010
Versie: 1.0

Voorwoord

Het document dat voor u ligt is het eindverslag ter afsluiting van ons bachelorproject. Met dit project ronden wij de bachelorfase af van de opleiding Technische Informatica aan de Technische Universiteit Delft (TU Delft). Ons bachelorproject is uitgevoerd in de vorm van een fulltime stage bij Centric IT Solutions voor een duur van 420 uur (15 ECTS).

Het bachelorproject is opgedeeld in ruwweg drie fasen: de oriëntatiefase, de realisatiefase en de afrondingsfase. De oriëntatiefase bestaat uit het verdiepen in het probleem. Deze fase hebben wij afgesloten met het schrijven van een Bijlage A. Opdrachtomschrijving, een Bijlage B. Plan van Aanpak, en eenBijlage C. Onderzoeksverslag. Deze documenten zijn te vinden als bijlage. Tijdens de realisatiefase hebben we een ontwerp gemaakt en dat vervolgens geïmplementeerd en getest. Het resultaat van deze fase is een Bijlage D. Requirements Analysis Document en een geteste applicatie met documentatie en Bijlage F. Handleiding. Dit eindverslag is geschreven tijdens de afrondingsfase en zal nog gevolgd worden door een eindpresentatie.

Onze dank gaat uit naar alle personen die ons geholpen en gesteund hebben tijdens ons bachelorproject en in het bijzonder naar:

Dr. M.M. de Weerdt (Stagebegeleider TU Delft)

Ir. B.R. Sodoyer (Stagecoördinator TU Delft)

Ir. C. Pronk (Vervanger voor Ir. B.R. Sodoyer)

Ir. L. Blom (Stagebegeleider Centric)

Dr. A.F. van Lier MCM CMC (Directeur Overheid Centric)

A. Mahieu (Manager Microsoft Surface tafel)

A. Knijf (Applicatie ontwikkelaar Microsoft Surface tafel)

J.P. Bregonje (Applicatie ontwikkelaar Microsoft Surface tafel)

Wij kijken terug op een prettige samenwerking met Centric IT Solutions en hopen dat zij in de toekomst door meer TU Delft studenten benaderd zullen worden.

Veel plezier gewenst aan alle lezers van dit document.

Gouda,
23 juni 2010

Rob Post
Oskar van Rest
Tim Roorda

Inhoudsopgave

Lijst van figuren

Samenvatting

Voor ons bachelorproject hebben wij stage gelopen bij het bedrijf Centric IT Solutions. Wij hebben de opdracht gekregen een simulator te maken voor het *Mobile Hubs and Smart Connected Containers* concept. Dit concept is bedacht door Centric en biedt autoriteiten, containereigenaren en de supply-chain mogelijkheden om informatie over containers bijna realtime ter beschikking te hebben wanneer dat nodig is. Volgens het concept wordt met behulp van *clustering*, *respecteren van autonomie* en *informatie bij de bron* de complexiteit van het probleem tegemoet gekomen en worden de belangen van de verschillende spelers behartigd. Informatie over de containers worden in de transportentiteit opgeslagen, in een zogenaamde mobiele hub (MH). Deze MH's kunnen communiceren met permanente hubs (PH's), die een vaste locatie langs de kade of boven de weg hebben. De PH's hebben een verbinding met het internet zodat ze altijd bereikbaar zijn.

Uit de onderzoeksfase is gebleken dat het concept bij een enkele situatie niet goed werkt. Bij een juiste plaatsing van de PH's zou dit probleem voorkomen worden. De door ons gemaakte simulator moest in ieder geval duidelijk maken welke informatie er wordt uitgewisseld, waar de informatie opgeslagen is en hoe de informatie teruggewonnen kan worden. Daarnaast moest het *Target a Container* concept in de simulator verwerkt worden. Dit concept beschrijft de procedure die uitgevoerd moet worden nadat er een melding van een calamiteit plaatsvindt. De simulator moest geschreven worden in .NET en moest op de Microsoft Surface tafel getoond kunnen worden.

De simulator is ontworpen volgens het Model-View-Controller (MVC) ontwerppatroon. Dit betekent dat functionaliteit van de Graphical User Interface (GUI) dus zoveel mogelijk gescheiden wordt gehouden van de rest. Aan het begin van het project ging dit nog wel eens mis doordat GUI functionaliteit gemengd was met functionaliteit voor het achterliggende systeem, maar dit is later in de implementatiefase verbeterd.

Tijdens de implementatiefase moest rekening gehouden worden met de Surface tafel. Dit betekende dat wij gebruik moesten maken van C# en WPF. Daarnaast wordt niet gebruik gemaakt van een muis, maar van aanrakingen met de vingers. Ook moet men op een willekeurige plek rondom de Surface tafel kunnen staan en nog steeds alle informatie kunnen lezen.

Halverwege de implementatiefase is een grote refactor gedaan waarbij veel dubbele code weggewerkt is, nettere code geschreven is, en de simulator efficiënter gemaakt is. Aan het einde van de implementatiefase is de simulator verbeterd door functionaliteit te implementeren voor gestureherkenning en door de procedure bij een calamiteit weer te geven op een tweede monitor. In de code staat bij elke klasse, methode en attribuut commentaar en om het commentaar beter leesbaar te maken is met behulp van Doxygen een documentatierapport gemaakt.

Voor het testen van modelklassen is gebruik gemaakt van NUnit. Door de tests automatisch elk uur te laten uitvoeren zijn een aantal fouten ontdekt. Helaas kwamen wij er te laat achter dat Visual Studio, de programmeeromgeving waarin wij gewerkt hebben, ook goede testmogelijkheden biedt. Naast de

unittests hebben we gebruik gemaakt van de Surface Simulator. Dit programma simuleert de Surface tafel op een gewone computer. Ook werd er regelmatig op de Surface tafel zelf getest en werden hierbij ook informele gebruikerstesten gedaan. Hierbij werden veel kleine bugs en fouten in de opmaak van de GUI gevonden. Ook kwamen hierbij veel suggesties voor verbeteringen van de simulator naar voren.

Bij het ontwerpen van de simulator is rekening gehouden met eventuele uitbreidingen. Zo zal een andere student de code gaan gebruiken om een *Tracking and Tracing* mogelijkheid in te bouwen. Daarom zijn modelklassen zo abstract mogelijk gehouden en is gebruik gemaakt van het MVC ontwerppatroon om de GUI gescheiden te houden van de overige functionaliteit. Er zijn vele uitbreidingen te bedenken die de simulator nog mooier maken of extra functionaliteit toevoegen.

Tijdens de implementatie hebben wij ons aan het spiraalmodel en het V-model gehouden. Wij hadden met de stagebegeleider eens in de twee weken een afspraak gepland. Dit hebben wij als aanknopingspunt genomen om bij elke afspraak een werkende simulator met nieuwe functionaliteit te tonen. We doorlopen dan telkens een iteratie uit het spiraalmodel. Met behulp van het V-model werden achtereenvolgens unittests, integratietests, systeemtests en acceptatietests uitgevoerd.

Tijdens het gehele project is het ons gelukt om ons aan de planning te houden. Dankzij een goede samenwerking en het snel beschikbaar hebben van een versie met beperkte functionaliteit zijn we tijdens de implementatiefase maar twee dagen achterop schema komen te liggen. Gelukkig was er genoeg tijd voor uitloop ingepland om dit op te vangen.

Het resultaat van het project is een goed werkende simulator die aan alle requirements voldoet. We kunnen dit project daarom als een succes beschouwen en hebben de samenwerking met Centric als zeer prettig ervaren.

Een aanbeveling is om het concept en bijbehorende simulator uit te breiden met andere bestaande end-to-end oplossingen zodat getoond kan worden dat het *Mobile Hubs and Smart Connected Containers* breder toepasbaar is. Gedacht kan worden aan *Tracking and Tracing*, monitoren van containers en personeel en het dynamisch toewijzen van resources. Om de simulator aantrekkelijker te maken kan gebruik gemaakt worden van vragenstellende actoren.

1 Inleiding

In de gehele wereld circuleren ongeveer 440 miljoen containers ten behoeve van de supply chain. Deze containers zijn allemaal gestandaardiseerd en rederijen en containereigenaren hebben allerlei systemen (niet gestandaardiseerd) in gebruik voor het verkrijgen en opslaan van hun gegevens. Tegenwoordig blijkt het lastig om de juiste informatie op het juiste tijdstip beschikbaar te hebben wanneer dat nodig is. Doet zich bijvoorbeeld een calamiteit voor, zoals een botsing of brand op een containerschip, dan kan het soms wel tot 24 uur duren voordat hulpdiensten weten wat er in de containers zit. Dit komt doordat de informatie over de containers zoals inhoud, bestemming en locatie vaak alleen bij de containereigenaren zelf beschikbaar is en hun systemen niet door anderen gebruikt kunnen worden. Een oplossing zou zijn om alles te standaardiseren en bijvoorbeeld alle gegevens in een grote database te stoppen, maar het probleem is dat de vele concurrerende partijen die hierbij betrokken zijn dit niet zullen waarderen. Daarnaast moet het systeem op wereldschaal kunnen werken en gezien er in verschillende landen verschillende wetgeving is omtrent controlerende instanties, is het niet aannemelijk dat een allesomvattend systeem ooit werkelijkheid zal worden.

Om deze redenen heeft Centric het concept *Mobile Hubs and Smart Connected Containers* bedacht. Hierbij wordt met behulp van *clustering*, *respecteren van autonomie* en *informatie bij de bron*, de complexiteit van het probleem tegemoet gekomen en de belangen van de verschillende spelers behartigd. De opdracht die wij van Centric hebben gekregen is het visualiseren van het concept. Ons is gevraagd een simulator te maken waarin de verschillende aspecten van het concept naar voren komen. De simulator zal in november 2010 getoond worden op een transportbeurs om te onderzoeken of er interesse in het concept is. De simulator zal ook gebruikt worden om verschillende mensen die werkzaam zijn bij Centric het concept op een intuïtieve manier bij te brengen.

In dit rapport zal allereerst het concept uitgelegd worden en zullen de beperkingen hiervan aan bod komen. Het hoofdstuk daarna beschrijft de probleemstelling en analyse van dit project. Daarna wordt ingegaan op de ontwerpfase en de verschillende ontwerpbeslissingen die daarbij genomen zijn. In hoofdstuk 5 wordt de implementatiefase beschreven. Hierin wordt besproken hoe deze fase verlopen is, hoe de documentatie tot stand gekomen is en hoe de applicatie getest werd.
Het daaropvolgende hoofdstuk bespreekt mogelijke uitbreidingen van de simulator, gevolgd door een hoofdstuk over de organisatie van het project. Hierin komen de gevolgde ontwerpmethodieken, de planning, de samenwerking en taakverdeling en de gebruikte software aan bod. Hoofdstuk 8 sluit af met de conclusie en aanbevelingen.

2 Mobile Hubs and Smart Connected Containers

Hieronder wordt het *Mobile Hubs and Smart Connected Containers* concept uitgelegd. Daarna wordt gekeken naar beperkingen van het concept. Bij deze beperkingen wordt ook een mogelijke oplossing gegeven.

2.1 Uitleg concept

Het *Mobile Hubs and Smart Connected Containers* is gebaseerd op bevindingen uit het onderzoek van Van Lier naar interoperabiliteit [12]. In het kort komt het er op neer dat *samenwerkende systemen* in *tijdelijke coalities* informatie kunnen *uitwisselen en delen* als hun relaties *voldoende zijn vormgegeven* [12]. De heer Blom heeft het concept verder uitgewerkt en is vooral bezig geweest met de technische kant van het verhaal. Het concept is inmiddels al zover uitgedacht dat het in de werkelijkheid gebruikt zou kunnen worden. Er moet nog wel nagedacht worden over de verschillende technieken voor het overbrengen van de informatie. Hierbij kan gebruik gemaakt worden van al bestaande technieken zoals walradarsystemen en wegportalen. Daarnaast dragen de vele betrokkenen en de grote schaal waarop het systeem gebruikt moet worden bij aan de complexiteit van het probleem.

In de huidige situatie is het zo dat informatie over de inhoud van containers meestal alleen bij de eigenaren bekend is. Voor bijvoorbeeld hulpverleners en autoriteiten is het nu dus erg lastig om deze informatie te verkrijgen indien dit nodig is. Het concept biedt daarom de volgende mogelijkheden:

- Autoriteiten in staat stellen om de inhoud en de status van inkomende containers op te vragen (*Trust a Container*).
- Hulpverleners in staat stellen om bij een calamiteit te bepalen welke containers betrokken zijn en wat de inhoud daarvan is (*Target a Container*).
- Voor de supply chain een mogelijkheid bieden om realtime over de meest belangrijke informatie te beschikken, zoals locatie, inhoud en andere gegevens over de status van de goederen (*Network your Container*).

De kracht van het concept is dat de gegevens bij de bron worden opgeslagen. Elke container moet over een apparaat beschikken waarin lading van de container staat opgeslagen. Naast de container zijn er twee andere entiteiten aanwezig voor het uitwisselen en bewaren van informatie: de mobiele hub (MH) en de permanente hub (PH). MH's zullen in transportentiteiten als containerschepen en vrachtwagens geplaatst worden en kunnen bijvoorbeeld aan de boardcomputer (denk aan de Roady) gekoppeld worden. PH's zullen een vaste positie langs vaarroutes of wegen hebben. Ook hier kan weer gebruik gemaakt worden van reeds aanwezige infrastructuur zoals walradarsystemen en wegportalen.

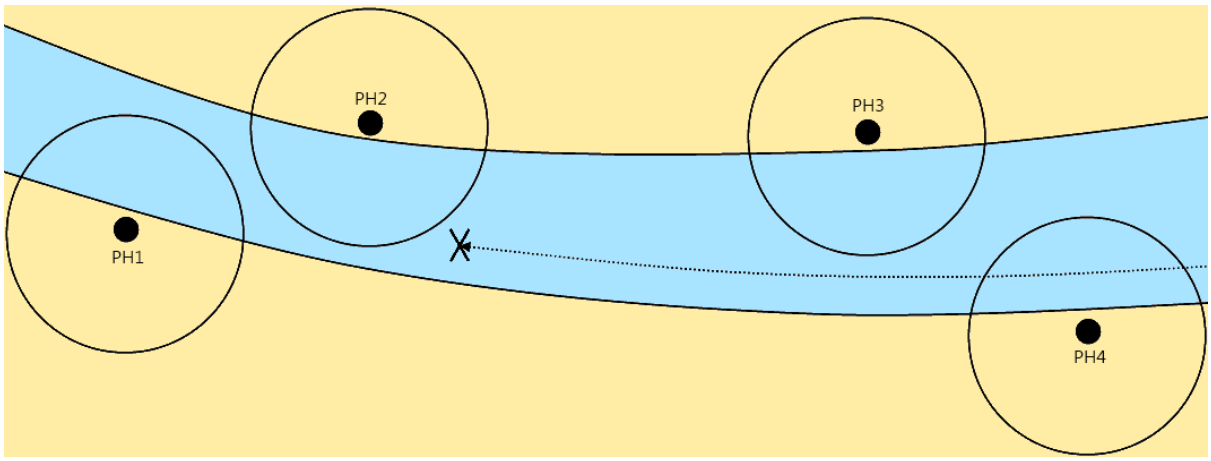
De container moet zijn gegevens door kunnen geven aan een MH. Zo beschikt een containerschip over een MH waarin informatie over de containers die het schip vervoert opgeslagen wordt. MH's kunnen op hun beurt bereikt worden door PH's, die op tactische plaatsen langs de vaargeul of de weg staan opgesteld. Hierin worden gegevens over MH's en de gegevens over de containers tijdelijk opgeslagen. Deze PH's zullen via internet bereikbaar zijn, zodat verschillende toepassingen er gebruik van kunnen maken.

Het concept beschrijft onder andere wanneer gegevens uitgewisseld worden, wanneer PH's informatie opslaan en verwijderen en hoe informatie teruggevonden kan worden. Zo zullen gegevens over goederen telkens maar in een enkele PH opgeslagen staan. De PH's zorgen er onderling voor dat de juiste gegevens overgezonden worden wanneer een MH binnen het bereik van een nieuwe PH komt.

2.2 Beperkingen concept

Een beperking van het concept waar we tijdens de onderzoeksfase achter gekomen zijn, betreft het plaatsen van de PH's aan de kant van het water. Het blijkt dat je deze niet zomaar willekeurig kan plaatsen.

Beschouw de situatie zoals in Figuur 1. Een schip vaart van rechts naar links door de vaargeul. Onderweg communiceert hij met PH4. Vlak voordat hij het bereik van PH2 binnenkomt, gebeurt er iets met het schip en het schip kan niet verder varen. Er wordt een melding van deze calamiteit gemaakt en de calamiteitenprocedure wordt gestart. Eén van de stappen van deze procedure is het vragen om informatie over de schepen die in het bereik zijn van de drie PH's die het dichtst bij de plek van de melding zijn. In dit geval zijn dit PH1, PH2 en PH3. Deze drie PH's weten echter niets over het schip dat problemen heeft en de informatie over het schip met problemen en de lading van dit schip kan niet worden bepaald.



Figuur 1: Situatie waarbij het concept niet werkt.

Om dit probleem te vermijden is het nodig om de PH's zo te plaatsen en het bereik zo in te stellen dat ze de hele breedte van de vaargeul overspannen. Zo komen schepen altijd langs minimaal één van de drie dichtstbijzijnde PH's en kan het hierboven beschreven probleem niet optreden. Een andere mogelijke oplossing is om naar meer PH's in de omgeving te kijken als het schip met problemen niet gevonden wordt bij de drie dichtstbijzijnde PH's.

Een ander probleem dat kan optreden is dat het bereik van twee PH's overlapt. Een schip kan zich zo bij twee PH's aanmelden, wat niet de bedoeling is. Een mogelijke oplossing is dat het schip zich aanmeldt bij de PH met het sterkste signaal. In onze simulator kan het niet voorkomen dat het bereik van twee PH's overlapt, dus hoeven MH's hier geen rekening mee te houden.

3 De probleemstelling en analyse

Centric wil het *Mobile Hubs and Smart Connected Containers* concept presenteren aan potentiële klanten en anderen geïnteresseerden. Daarom leek het hen een goed idee een simulator te ontwerpen waarbij de verschillende aspecten van het concept duidelijk naar voren komen. Op deze manier kan men op een intuïtieve manier de werking van het concept inzien zonder hiervoor allerlei documenten te hoeven bestuderen.

Centric heeft ons gevraagd deze simulator te realiseren. Hierbij moet in ieder geval duidelijk blijken welke informatie er wordt uitgewisseld, waar de informatie opgeslagen is en hoe de juiste informatie toegankelijk gemaakt kan worden op het moment dat dit nodig is. Er moeten dus enkele PH's en MH's weergegeven worden, er moet weergegeven worden wanneer zij informatie uitwisselen en op ieder moment moet de opgeslagen informatie opgevraagd kunnen worden.

Om het enigszins beperkt te houden worden alleen schepen als transportentiteiten gebruikt en wordt een statische achtergrond gebruikt waarop de PH's en MH's en hun informatiestromen in 2D weergegeven worden. Daarnaast wordt de mogelijkheid *Target a Container* erin verwerkt, dat hulpverleners in staat stelt om bij een calamiteit te bepalen welke containers zijn betrokken en wat de inhoud daarvan is.

De simulator moet uiteindelijk op de Microsoft Surface tafel getoond worden [8]. Dit is een tafel met daarop een groot multi-touch screen waarop je met meerdere personen tegelijkertijd interactief bezig kunt zijn. Het voordeel van het tonen van de simulator op een dergelijk scherm, is dat het een hele andere beleving is waarbij het net lijkt alsof je in een meldkamer zit. De Surface tafel biedt de mogelijkheid om eventueel met meerdere mensen tegelijkertijd informatie op te vragen over MH's en PH's en de calamiteitenprocedure te doorlopen. Dit nodigt uit om de mogelijkheden van het concept uit te proberen om zo te zien wat het resultaat is. Uiteindelijk wordt hierdoor bereikt dat men geïnteresseerd raakt in de precieze werking van het concept.

Een eis aan de simulator is dat het in .NET geschreven wordt, omdat dit het framework is waarmee de Surface tafel overweg kan. Zie het Bijlage D. Requirements Analysis Document voor een compleet overzicht van alle requirements waaraan de simulator moet voldoen.

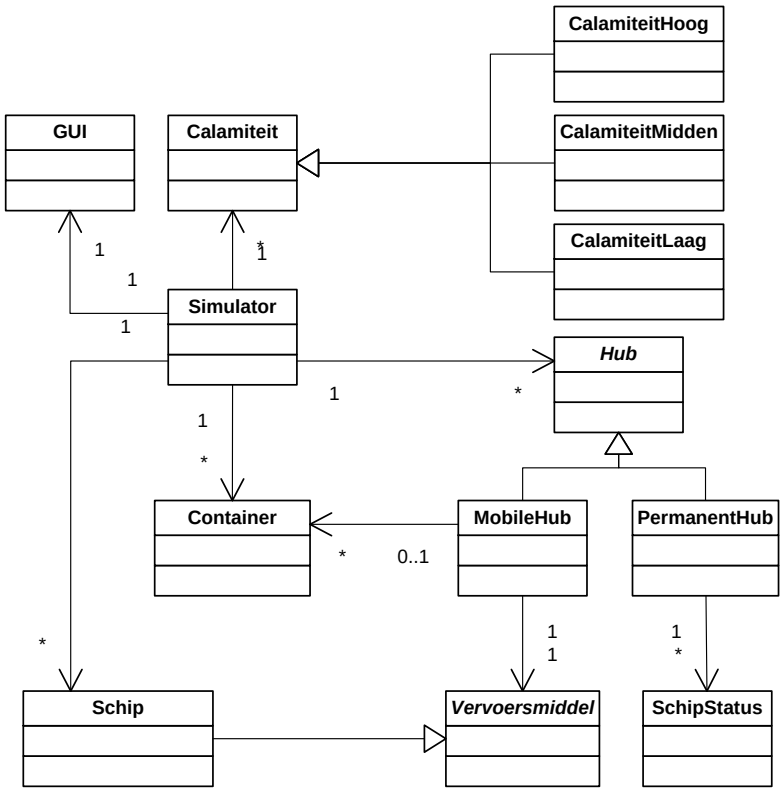
4 Ontwerpfase

In dit hoofdstuk worden de veranderingen beschreven die we tijdens het implementeren aan het ontwerp hebben gedaan. Allereerst wordt de situatie aan het begin van de implementatiefase weergegeven aan de hand van het toen gemaakte klassendiagram. Vervolgens worden de belangrijkste ontwerpbeslissingen beschreven en ten slotte volgt het uiteindelijke klassendiagram zoals het nu is.

4.1 Initiële ontwerp

Tijdens de onderzoeksfase kwam naar voren dat veel simulators worden gemaakt aan de hand van het Model-View-Controller (MVC) ontwerppatroon [5]. Dit houdt in dat het systeem bijna helemaal los staat van de grafische weergave van het systeem. Ook de regelunit, de controller, moet zo los mogelijk komen te staan van de andere componenten. Hierdoor is het makkelijk om bijvoorbeeld de grafische weergave heel anders te maken zonder iets aan het onderliggende systeem te veranderen.

Tijdens het ontwerpen hebben wij zoveel mogelijk geprobeerd deze ontwerpmethodiek te volgen. Dit blijkt ook uit ons eerste klassendiagram dat te vinden is in Figuur 2.



Figuur 2: Klassendiagram tijdens de ontwerpfase.

Hierin is duidelijk het MVC ontwerppatroon te zien. In het klassendiagram stelt de Simulatorklasse de controller voor. De view wordt voorgesteld door de GUI. De overige klassen vormen het model. Tijdens

het implementeren en testen wordt er steeds meer ervaring opgedaan en worden er punten bedacht die beter kunnen. Tijdens de implementatiefase zijn er dus ook een aantal ontwerpbeslissingen genomen. Deze staan hieronder beschreven.

4.2 Ontwerpbeslissingen

Deze sectie bevat de belangrijkste ontwerpbeslissingen die gemaakt zijn tijdens de implementatiefase. Hierbij proberen we uit te leggen waarom sommige aspecten zo geworden zijn zoals ze zijn.

4.2.1 Singleton pattern

We hebben ervoor gekozen het singleton pattern te implementeren voor de simulatorklasse [5]. Het singleton principe zorgt ervoor dat er maar één instantie van de simulatorklasse kan bestaan. Hierdoor communiceren alle andere klassen dus altijd met dezelfde simulator. Ook zijn alle gegevens die nodig zijn in veel klassen van de simulator altijd bereikbaar. De lijst met MH's en de lijst met PH's worden bijvoorbeeld in veel klassen geraadpleegd.

4.2.2 Kiezen van juiste informatie uit PH's

Bij een calamiteit moet bepaald worden welke MH's opgeslagen staan in de drie dichtstbijzijnde PH's. Dit moet gebeuren op een moment vlak voor de calamiteit alsmede op een moment vlak na de melding. In de werkelijkheid is het tijdstip van de calamiteit niet precies te bepalen. Vanaf het tijdstip van de melding moet worden teruggerekend om het tijdstip van de calamiteit te bepalen. In de praktijk zal dit gebeuren met speciale algoritmes of op basis van ervaring. Wij hebben er voor gekozen om het tijdstip waarop de calamiteit plaatsvindt op te slaan. Aan de hand van dit tijdstip is het makkelijk om de situatie vlak voor de calamiteit te vinden. Om de situatie na de melding te bepalen, wordt er tijdens het doorlopen van de calamiteitprocedure een extra scan gedaan door de drie dichtstbijzijnde PH's. Zo is het altijd zeker dat er een situatieschets gemaakt is na het tijdstip van de calamiteit.

Ook bij het weergeven van de situaties en het weergeven van de informatie die een PH bevat, is een keuze gemaakt over welke informatie van MH's moet worden weergegeven. Als iemand op een PH drukt, wordt alleen de meest recente informatie per MH weergegeven. Een PH bevat veel meer gegevens, maar als dit allemaal op het scherm moet worden weergegeven, wordt de gebruiker overspoeld met informatie.

4.2.3 Calamiteitenprocedure

Er is voor gekozen dat wanneer een calamiteit wordt gestart het schip dat problemen heeft steeds langzamer te laten varen. Vervolgens wordt er na een aantal seconden een melding gedaan vanaf een locatie in de buurt van het schip met problemen. Daarna wordt de simulatie gepauzeerd en wordt het calamiteitenscherm weergeven met alle stappen die moeten worden doorlopen. Er is voor gekozen om na elke stap te wachten op interactie van de gebruiker. Dit geeft de gebruiker de mogelijkheid om bij elke stap uitleg te geven en eventuele vragen te beantwoorden.

Daarnaast is er voor gekozen om het calamiteitenscherm op een tweede monitor te laten zien als die aanwezig is. Hierop wordt weergegeven welke stappen er gedaan worden na de melding van een calamiteit. Dit zijn juist de stappen die in werkelijkheid in een meldkamer uitgevoerd worden. Door een

apart scherm te gebruiken kan men zich beter inleven in de situatie. Ondertussen wordt op de Surface tafel ook het één en ander gevisualiseerd. Zo verschijnen er tijdens een calamiteit bijvoorbeeld vraagtekens op de mogelijke schepen die in de problemen zijn. Dankzij het tweede scherm blijft alles heel overzichtelijk en hoeven er geen vensters boven elkaar geplaatst te worden. Als er geen tweede scherm aanwezig is, komt het calamiteitenscherm wel gewoon over de simulatie heen.

4.2.4 Graphical User Interface

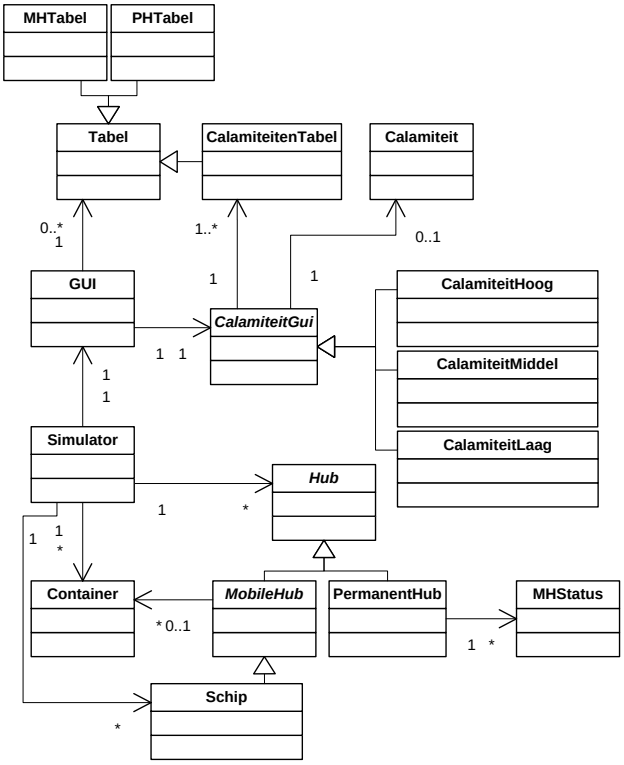
Het grootste gedeelte van de Graphical User Interface (GUI) is geïmplementeerd in de GUI klasse. Er zijn echter ook een paar GUI elementen die we in een aparte klasse hebben geïmplementeerd. Het gaat hier om de tabellen die worden weergegeven als er op een MH of PH wordt gedrukt. Ook de tabellen die in het calamiteitenscherm worden weergegeven, maken gebruik van dezelfde code. We hebben daarom een superklasse Tabel gemaakt die alle gemeenschappelijke methoden en attributen bevat. Voor iedere soort tabel is er een subklasse gemaakt die de specifieke eigenschappen van dat soort tabel implementeert. Een voorbeeld hiervan is dat een tabel van een MH de containers moet laten zien en een tabel van een PH de statussen van de MH's moet weergeven.

Zoals te zien is in Figuur 2 hadden wij in eerste instantie drie subklasse van Calamiteit gemaakt: CalamiteitHoog, CalamiteitMiddel en CalamiteitLaag. Dit hebben wij op deze manier gedaan omdat we dachten dat de verschillende niveaus verschillende algoritmes zouden hebben voor het vinden van de juiste informatie. Halverwege de implementatiefase bleek dat eigenlijk alleen de weergave van de verschillende niveaus anders was en dat er dus alleen verschil in de GUI zit. Om deze reden hebben we de subklassen van Calamiteit weggehaald en hebben een klasse CalamiteitGUI gemaakt met subklassen CalamiteitHoogGUI, CalamiteitMiddelGUI en CalamiteitLaagGUI.

De logica die nodig is om de calamiteitenprocedure te doorlopen is te vinden in de klasse Calamiteit. In deze klasse staat bijvoorbeeld een methode die uit een lijst PH's de drie dichtstbijzijnde zoekt. Dit zijn dus methoden die niets met de weergave van calamiteiten te maken hebben, maar wel met de achterliggende berekeningen voor de calamiteitenprocedure.

4.3 Uiteindelijke ontwerp

Figuur 3 geeft het uiteindelijke klassendiagram weer. Hier zijn alle bovenstaand beschreven beslissingen in opgenomen. In dit klassendiagram hebben we de klasse Vervoersmiddel weggelaten. Ondanks dat een transportentiteit geen MH is maar een MH aan boord heeft, hebben we besloten om de transportentiteiten toch een subklasse van MH te maken. Dit is gedaan omdat veel transportentiteiten veel dezelfde attributen hebben zoals snelheid. Nu een transportentiteit een subklasse is van MH, beschikken ze over de functionaliteit van een MH en kunnen ze gemakkelijk in de simulator worden geplaatst. Ook de klasse SchipStatus is hernoemd naar MHStatus. Dit zorgt voor een hoger abstractieniveau en maakt de simulator beter uitbreidbaar.



Figuur 3: Uiteindelijke klassendiagram.

5 Implementatiefase

Na de onderzoeksfase zijn we direct begonnen met het implementeren. In dit hoofdstuk wordt beschreven wat we tijdens de implementatiefase hebben gedaan. Allereerst worden de aspecten van de implementatie toegelicht. Daarna wordt er vermeld hoe de documentatie bij de code tot stand is gekomen. Tenslotte wordt er beschreven hoe de geschreven code is getest en wat daarbij aan het licht kwam.

5.1 Implementatie

Om een applicatie te maken voor de Surface tafel moet er worden geprogrammeerd in .NET. Wij hebben gebruik gemaakt van de talen C# en WPF. Een van ons had wat ervaring in C#, maar voor de anderen was deze taal, net als WPF, compleet nieuw. Gelukkig hebben we programmeerervaring in Java, wat veel op C# lijkt. De overschakeling was daarom niet al te groot. Wat we wel nog moesten leren was WPF. Gelukkig kregen we dit ook snel onder de knie.

In het begin van de implementatiefase lag onze focus op het implementeren van het model. Na een ruime week van implementeren was het model af. Ook werden er tegelijkertijd tests geschreven om te controleren of het model ook werkte. Meer over testen is te lezen in hoofdstuk 5.3.

Tijdens de laatste paar dagen van het programmeren aan het model werd er ook al begonnen aan de GUI. Dit maakte het programmeren voor ons ook een stuk leuker. Je kunt gelijk zien wat je hebt gedaan en je kunt gelijk controleren of het ook werkt zoals jezelf in gedachten had. Toen het model eenmaal af was, zijn de taken wat meer verdeeld. Eén persoon richtte zich vooral op de implementatie van de calamiteiten. De andere twee hielden zich vooral bezig met de grafische weergave van het systeem.

Het programmeren voor de Surface tafel is anders dan het programmeren voor een normaal computerscherm. Er moet rekening mee worden gehouden dat het scherm van alle kanten zichtbaar is. Hierdoor is het van belang om zo min mogelijk tekst op het scherm te tonen. Dit kan immers maar vanaf één kant gelezen worden. Moet er toch tekst op het scherm komen, dan is het van belang dat deze tekst draaibaar is. Zo kunnen ook mensen aan de andere kant van de tafel de tekst lezen. Ook moet er rekening mee worden gehouden dat het programma aangestuurd wordt met vingers in plaats van een muis. Hierdoor is het belangrijk dat de interface er intuïtief uit komt te zien.

Halverwege de implementatiefase hebben we een grote refactor gedaan. Voorheen werden alle tabellen die zichtbaar waren, aangemaakt op het moment dat ze nodig waren. De code voor het aanmaken van een tabel stond in één klasse, maar veel tabellen bevatten net wat andere informatie. Hierdoor kwam veel dubbele code voor. Ook was het lastig om de tabellen die bij PH's horen te updaten als er informatie werd geüpdatet. De oplossing die wij hebben toegepast is om voor elke soort tabel een eigen klasse te maken. Ook worden alle tabellen nu aangemaakt als de simulator wordt opgestart. De tabellen hoeven dan alleen maar zichtbaar of onzichtbaar gemaakt te worden als dat nodig is. Bij deze refactor hebben we gelijk de hele code doorlopen en overal de code proberen netter te maken. Ook zijn een aantal methoden herschreven zodat ze veel efficiënter zijn geworden.

Tijdens een van de laatste evaluatiesessies met de begeleiders en met iemand die al meer ervaring heeft met het ontwikkelen voor de Surface tafel, kwamen nog een aantal interessante punten naar boven. Tot nu toe kon er bij onze simulatie gepauzeerd worden of door een calamiteitprocedure worden gelopen door met een vinger twee keer achter elkaar op het scherm te drukken. Dit werkte bij ons niet altijd even goed en het idee werd geopperd om gestures te gaan gebruiken. Gestures zijn figuren die je met je vinger op het scherm kan tekenen. Een speciale bibliotheek kan deze gestures herkennen. In onze simulator wordt nu gebruik gemaakt van deze gestures. Het implementeren van deze functionaliteit kostte ons hooguit één dag. Na het testen op de Surface tafel bleken de gestures veel beter te werken dan het dubbel op het scherm drukken met een vinger.

Naast het gebruik van gestures werd ook het idee geopperd om gebruik te maken van een tweede monitor. Tot nu toe werd het calamiteitenscherm over de simulatie heen gezet. Dit zorgde voor een vrij vol scherm. Het calamiteitenscherm laat eigenlijk zien wat de meldkamer te zien krijgt. Door het gebruik van een twee monitor wordt dus eigenlijk een scheiding gemaakt tussen de “werkelijkheid” en de procedure die in de meldkamer plaatsvindt. Om in de simulatie toch te kunnen zien wat er precies gebeurt, hebben we een aantal icoontjes in de simulator gezet. Mocht er geen tweede scherm zijn aangesloten dan wordt de calamiteitenprocedure over de simulatie weergegeven, zoals in de oude situatie het geval was. De implementatie voor het gebruik van een tweede scherm kostte ons wel iets meer moeite dan het implementeren van de gestures. Toch zorgde dit er niet voor dat we ver achter kwamen te lopen op onze planning.

5.2 Documentatie

Om de code toegankelijk te maken voor ontwikkelaars die met onze code verder moeten werken hebben we geprobeerd om overal zo duidelijk mogelijk commentaar toe te voegen. Visual Studio 2008 biedt een handige functie om commentaar in XML formaat bij elke methode en bij elk attribuut te plaatsen. Ook wordt er een waarschuwing gegeven als je ergens vergeten bent om commentaar te plaatsen.

Het commentaar in XML formaat staat tussen de code en is dus verspreid door het hele project. Dit maakt het niet erg leesbaar, zeker niet als er even snel iets opgezocht moet worden. We hebben er daarom voor gekozen om al het commentaar te exporteren naar HTML bestanden. Dit is gedaan met het programma Doxygen [4]. Dit programma maakt van al het commentaar een handig documentatierapport. Wij hebben voor Doxygen gekozen, omdat dit programma een mooie output geeft en omdat het door veel projecten wordt gebruikt, waaronder OpenOffice [10].

In het documentatierapport is per klasse precies te zien welke attributen en methoden er in die klasse te vinden zijn. Daarnaast wordt gelijk per attribuut en methode het commentaar erbij gezet. Ook is er de mogelijkheid om de hiërarchie tussen de klassen te bekijken. Daarnaast is er functionaliteit aanwezig om op alfabetische volgorde door de klassen of door alle attributen en methoden te lopen.

5.3 Testen

Om de kwaliteit van de code te waarborgen zijn er tegelijkertijd met de implementatie van de modelklassen unittests geschreven. Per modelklasse hebben we een testklasse aangemaakt met dezelfde naam aangevuld met “Test” zodat duidelijk is waar de tests van elke klasse zich bevinden. Hierin staan testmethode die de methoden uit de modelklassen testen, met uitzondering van get en set methoden.

De testklassen zijn subklassen van de klasse TestOmgeving waarin een testomgeving wordt gecreëerd met enkele gegevens zoals MH’s, PH’s en containers. Vóór het uitvoeren van de testmethoden wordt elke keer deze testomgeving gereset.

Als testframework hebben we gebruik gemaakt van NUnit 2.5.5, omdat de functionaliteit en het gebruik hiervan grotendeels overeenkomt met het voor ons bekende JUnit voor het unittesten van Java klassen [9]. Daarnaast is NUnit een veelgebruikt framework voor unittests in C#, is het makkelijk in gebruik en is het goed gedocumenteerd. Er kan zowel van een GUI als een console gebruik worden gemaakt. We hebben dankbaar gebruik gemaakt van onze unittests, want toch werden soms stopcondities niet juist geformuleerd en werden typefouten gemaakt. Bovendien kwamen tijdens unittesten na het refactoren ook een aantal kleine fouten naar voren. Naast het uitvoeren van de unittests na het implementeren van nieuwe functionaliteit werden ook periodiek elk uur alle unittests uitgevoerd, zodat foute code snel opgemerkt kon worden.

Tijdens het project bleek echter dat NUnit ook een groot nadeel heeft: private methoden kunnen niet makkelijk worden getest. Pas toen ontdekten we de testmogelijkheden van Visual Studio 2008. Dit gebeurde nadat alle modelklassen geïmplementeerd en getest waren. Voor de viewklassen hebben we geen unittests geschreven, deze hebben we met printstatements en met behulp van het menselijk oog getest. Vanwege deze redenen en omdat we al tegen het einde van het project aanliepen, besloten we geen tijd te investeren in het overstappen van NUnit naar de Visual Studio testmogelijkheden. Voor een volgend project is het zeker interessant om eens goed te kijken naar de testmogelijkheden van Visual Studio.

Naast automatische unittests hebben we gebruik gemaakt van de Surface Simulator. Dit programma simuleert de Surface tafel, waarbij de muis een vinger simuleert. Ook kunnen er meerdere muizen worden aangesloten, zodat er meerdere vingers gesimuleerd kunnen worden. Zo konden we precies zien hoe ons programma er uitziet op de Surface tafel als we geen beschikking hadden over een Surface tafel. Op deze manier konden we goed de GUI testen. Daarnaast bestaat er ook een stresstest voor de Surface Simulator die random een heleboel vingerbewegingen simuleert. Zo konden we kijken hoe ons programma met meerdere gebruikers en in belastende situaties functioneert. Hierbij zijn geen problemen gevonden.

Ook tegen het einde van het project hebben onze laptops geen minuut stilgestaan. Gedurende anderhalf uur heeft ons programma in de Surface Simulator gedraaid om te testen of ons programma ook voor een langere tijd soepel kan draaien. Hierbij zijn ook geen problemen aan het licht gekomen.

Gedurende het hele project is er ook geregeld op de Surface tafel zelf getest. Dit omdat de Surface Simulator er toch anders uit ziet en anders bediend wordt dan de echte Surface tafel. Hierbij konden we gebruiksvriendelijkheid van ons programma testen alsmede de reactiesnelheid van de Surface tafel. Als de heer Blom beschikbaar was, ging hij ook vaak mee naar de Surface tafel en op die manier vond op informele wijze een gebruikerstest plaats. Kleine bugs en fouten in de opmaak van de GUI werden daarna meteen opgelost.

Bij deze tests zijn de meeste bugs aan het licht gekomen. Denk bijvoorbeeld aan wat er gebeurt als er op twee knoppen tegelijkertijd gedrukt wordt. Dit ging eerst fout bij het tegelijkertijd drukken op twee calamiteitenknoppen, omdat niet duidelijk was welke calamiteit er nou gestart moest worden. Dit probleem hebben wij verholpen door bij het klikken op een calamiteitenknop direct alle calamiteitenknoppen uit te schakelen en te verbergen.

Een meer zichtbare bug was dat de timers niet verwijderd werden en dat de simulatie na pauzeren dubbel zo snel verliep. Er werd namelijk bij het hervatten wel nieuwe timers aangemaakt, terwijl de oude ook hervat werden.

Ook werd duidelijk dat openstaande PH en MH tabellen achter de weergave met de calamiteitenprocedure bleven staan en daardoor niet gesloten konden worden. Bovendien werden sommige plaatjes en tabellen wel toegevoegd aan de GUI, maar nooit meer verwijderd.

Op de Surface tafel konden we ook goed kijken welke gestures goed bruikbaar waren. Zo werd een rondje bijvoorbeeld te snel herkend en moest bij een driehoek de basis evenwijdig aan de fysieke onderkant van de Surface tafel worden getekend.

Een bug die in eerste instantie niet erg opviel en bij de Surface Simulator zelfs helemaal niet opviel was dat als direct onder een lijst met items geklikt werd, er een null reference exceptie optrad. Dit kwam omdat het programma het bijbehorende item probeerde op te halen, terwijl die er niet was. Dit is verholpen door eerst te checken of het bijbehorende item niet null is.

De heren Van Lier en De Weerdt en ook andere geïnteresseerden hebben zelf ook onze simulatie op de Surface tafel zien draaien. Dankzij deze gebruikerstests zijn niet alleen belangrijke bugs aan het licht gekomen, maar zijn ook suggesties voor verbeteringen naar voren gekomen. Hier hebben wij dankbaar gebruik van gemaakt, zoals het implementeren van een sluitenknop in tabellen in plaats van sluiten door ergens op de tabel te klikken. Dit is niet alleen duidelijker, maar ook robuuster omdat de tabel niet direct gesloten wordt als iemand per ongeluk de tabel aanraakt.

Tegen het einde van het project werd opnieuw een gebruikerstest uitgevoerd waarbij twee andere studenten aanwezig waren. Zij onderzoeken of er markt is voor het *Mobile Hubs and Smart Connected Containers* concept en hebben dus wel kennis over het concept, maar niet over onze simulatie. Hierbij werd een geheugenlek ontdekt: als de simulator een aantal keer herstart werd (door een bepaalde vingerbeweging te maken) nam het geheugengebruik enorm toe. Dit was te zien in Windows Taakbeheer. Na ongeveer twintig keer herstarten werd de simulator traag en voeren de schepen

schokkerig. Blijkbaar werden bepaalde resources niet vrijgemaakt door de garbage collector. Het kostte enige tijd voordat we de oorzaak van het probleem gevonden hadden. Met behulp van .NET Memory Profiler kwamen we er al snel achter dat wanneer een scherm gesloten werd, deze nog steeds in het geheugen bleef staan [11]. Blijkbaar waren er ergens in de applicatie nog bepaalde referenties naar het scherm. Uiteindelijk hebben we het probleem opgelost door voor een van de schermen alle methoden één voor één te verwijderen. Na elke methode te hebben verwijderd, lieten we het scherm automatisch honderd keer openen en sluiten om te zien of het geheugen toenam. Zo zijn we er uiteindelijk achtergekomen dat de methode die bij het sluiten van het scherm alle event handlers verwijdert, één event handler niet verwijderde. Na het toevoegen van een enkele regel was het probleem verholpen.

6 Uitbreidingen van de simulator

Onze applicatie moest zo ontworpen worden dat later eventueel eenvoudig nieuwe functionaliteit toegevoegd zou kunnen worden. Zo zal er waarschijnlijk een andere stagiair onze code gaan gebruiken om een applicatie te bouwen die containereigenaren in staat stelt hun containers te volgen (*Tracking and Tracing*). Om deze reden heeft Centric ons gevraagd in hoeverre onze code gebruikt kan worden voor eventuele uitbreidingen. Naast *Tracking and Tracing* zijn er natuurlijk nog veel meer mogelijke uitbreidingen te bedenken die bepaalde functies van het concept verduidelijken of die mogelijk andere toepassingen van het concept illustreren.

Ten eerste het toevoegen van andere transportentiteiten. Het zou bijvoorbeeld interessant zijn om te zien hoe het concept met vrachtwagens en wegportalen overweg gaat. Een andere uitbreiding waarvan we tijdens het implementeren achter zijn gekomen dat dit erg handig zou zijn is het toevoegen van een schuifbalk om de simulator te versnellen of te vertragen. Ook noemen we nog een aantal andere mogelijke uitbreidingen die al in het onderzoeksverslag naar voren zijn gekomen. Nu de simulator zo goed als af is, kunnen we meer zeggen over de haalbaarheid en toegevoegde waarde van deze uitbreidingen. Ook wordt er nog iets verteld over de mogelijkheid om met behulp van het Automatic Identification System (AIS), realtime informatie van (echte) schepen in de simulator te verwerken.

6.1 Tracking and tracing

Voor *Tracking and Tracing* van containers door de containereigenaren kan in ieder geval gebruik gemaakt worden van de modelklassen. We hebben ons tijdens het ontwikkelen van de applicatie zoveel mogelijk aan het MVC ontwerppatroon gehouden en het model dus zoveel mogelijk gescheiden gehouden van de User Interface en de controllerklassen.

Indien het om *Tracking and Tracing* van containers op containerschepen gaat, zou natuurlijk ook een groot deel van de GUI gebruikt kunnen worden. Het deel van de GUI met functionaliteit voor calamiteiten staat namelijk grotendeels los van de rest van de GUI. Het beginscherm, het gedeelte waarbij MH's voortbewegen, PH's aan het scannen zijn en op MH's en PH's gedrukt kan worden, zou prima hergebruikt kunnen worden. Vooral de implementatie van de tabellen die de inhoud van MH's en PH's weergeven, zijn goed te hergebruiken. De implementatie hiervan heeft veel tijd in beslag genomen, omdat ze erg complex zijn en veel gegevens bevatten. Wil men soortgelijke tabellen maken, dan raden wij aan gebruik te maken van onze code.

Alle modelklassen staan in de hoofdmap. In deze hoofdmap is een map "GUI" te vinden waar alle klassen van de GUI in zitten. De controller genaamd "Simulator.cs" staat ook in de hoofdmap.

6.2 Verschillende transportentiteiten

We hebben geprobeerd om de klassen van het klassendiagram en de implementatie hiervan zo abstract mogelijk te houden zodat er naast schepen ook vrachtwagens, treinen of andere transportentiteiten toegevoegd zouden kunnen worden. Om deze reden hebben we een klasse MobileHub gemaakt met

een subklasse Schip. De klasse MobileHub bevat functionaliteit die in principe door iedere andere transportentiteit gebruikt zou kunnen worden.

Gezien een MH en PH ook overeenkomstige functionaliteit hebben, hebben we ook hier weer een klasse Hub voor gemaakt met algemene functionaliteit en twee subklassen MobileHub en PermanentHub die deze functionaliteit overerven.

Zonder al te veel te hoeven veranderen aan de code zou het dus mogelijk zijn om bijvoorbeeld een simulatie voor vrachtwagens met wegportalen te maken. Wel moet er dan het een en ander veranderen bij de PH's, gezien de gesimuleerde walradarsystemen anders werken dan wegportalen. Interessant is ook om het proces van overslag, opslag en transport met verschillende transportentiteiten te laten zien, vooral wanneer men er *Tracking and Tracing* bij wil betrekken.

6.3 Schuifbalk om de simulator te versnellen/vertragen

Een andere uitbreiding die naar ons idee in weinig tijd veel toegevoegde waarde kan hebben, is het toevoegen van een schuifbalk waarmee de simulator te versnellen en te vertragen is. Een van de voordelen is dat je na het opstarten snel waarden in de PH's kunt krijgen door simpelweg de simulator te versnellen. Zo hoeft er niet lang gewacht te worden voordat er voldoende informatie in de PH's is opgeslagen om een calamiteit te starten. Een ander voordeel is dat je de schepen nu op realistische snelheid kunt laten gaan indien gewenst (momenteel varen de schepen sneller dan in de werkelijkheid). Naar schatting kost het implementeren van deze functie niet veel tijd.

6.4 Vragenstellende actoren

Vragenstellende actoren, zoals uitgelegd in het onderzoeksverslag, beschrijft een mogelijke aanvulling op de simulator. Hierbij is het de bedoeling dat verschillende actoren de voor hen relevante vragen kunnen stellen en de simulator laat zien hoe de juiste informatie gevonden wordt om de vragen te kunnen beantwoorden.

Deze vragen kunnen duidelijk maken waar *Trust a container*, *Target a Container* en *Network your Container* toe dienen en hoe deze concepten in hun werk gaan. Gezien *Target a Container* door ons geïmplementeerd is en *Network your Container* al aan bod gaat komen bij de stagiaire die volgend jaar *Tracking and Tracing* gaat inbouwen, wordt het concept vragenstellende actoren al deels gerealiseerd. Het zou een interessante toevoeging zijn wanneer alle drie de concepten gecombineerd worden in een enkele simulator zodat verschillende actoren de voor hen relevante onderdelen kunnen simuleren door bijbehorende vraag te selecteren.

6.5 Schip kunnen besturen

Een uitbreiding die voor het tonen van het concept niet veel toegevoegde waarde heeft, maar de simulator wel een stuk interactiever maakt, is een functie inbouwen waarmee men een of meerdere schepen zelf kan besturen. Dit staat in meer detail beschreven in het onderzoeksverslag. Nu we precies weten wat de functies van de Surface tafel zijn en hoe de simulator in elkaar steekt, kunnen we meer zeggen over hoe dit eruit zou moeten zien. Naar ons idee kan dit het beste geïmplementeerd worden

door met je vingers een pad te trekken en vervolgens het schip op eigen snelheid dit pad af te laten lopen. Het herkennen van een pad zit momenteel al in de simulator, namelijk bij het herkennen van gestures. Het trekken van een pad kan daarom in korte tijd gerealiseerd worden. Wel kan het lastig worden om de schepen vloeiend te laten bewegen en niet van het ene op het andere moment tien graden te zien draaien. Wil men dat het er realistisch uitziet en de schepen laten bewegen zoals ze dat in werkelijkheid ook doen, dan moet er wel het één en ander aangepast worden aan de simulator.

Daarnaast gaat men, wanneer deze functie geïmplementeerd is, er waarschijnlijk vanuit dat er ook botsingen kunnen optreden wanneer bijvoorbeeld twee schepen tegen elkaar varen. Wil men dit implementeren, dan zal dus ook een vorm van collision detection geïmplementeerd moeten worden. Dit is naar ons idee minder eenvoudig te realiseren. Gezien de functie weinig toegevoegde waarde heeft bij het uitleggen van het concept, raden wij af om hiermee verder te gaan.

6.6 Opslaan en afspelen van scenario's

Zoals besproken in het onderzoeksverslag zou het opslaan en afspelen van scenario's de mogelijkheid bieden om vooraf gedefinieerde handelingen automatisch uit te laten voeren. Op de manier kunnen belangrijke situaties getoond worden en weet de spreker wat er komen gaat. Bij nader inzien heeft deze uitbreiding niet heel veel toegevoegde waarde. Op dit moment kan met de simulator binnen enkele seconden een gewenste situatie gecreëerd worden. Meestal hoeft alleen even gewacht te worden totdat de schepen in de juiste positie staan en met een beetje inzicht kan men wel bepalen op welk moment bijvoorbeeld een calamiteit gestart moet worden opdat de gewenste situatie optreedt. De implementatie van deze uitbreiding neemt naar schatting veel tijd in beslag en wij raden het dan ook af.

6.7 Automatic Identification System

AIS biedt de mogelijkheid om realtime informatie te verkrijgen van schepen op zee of in binnenwater. Informatie als positie, snelheid en op de reis betrekking hebbende scheepsgegevens worden met regelmatige tussenpozen geüpdatet. Aan de hand van deze informatie zouden wij van een bepaald gebied, bijvoorbeeld in Rotterdamse haven, kunnen laten zien welke schepen er op dat moment varen. Zo kunnen we PH's op de kaart plaatsen en het concept demonstreren met deze echte schepen.

Hoewel deze uitbreiding behoorlijk wat tijd zal kosten om te implementeren, denken wij dat deze uitbreiding de mogelijkheid biedt om de simulator ook voor andere doeleinden te gebruiken. Dit is daarom een erg interessante uitbreiding. In eerste instantie moet gezegd worden dat het weinig toegevoegde waarde zal hebben voor het kunnen uitleggen van het concept. Dit komt doordat de schepen veel te langzaam zullen varen, er te weinig schepen of helemaal geen schepen aanwezig kunnen zijn en er dus totaal geen controle meer is. Voor het kunnen tonen van het concept is deze uitbreiding misschien dus niet gewenst. Waar een simulator met deze uitbreiding wel erg goed voor gebruikt kan worden, is wanneer gesimuleerd moet worden waar PH's in de werkelijkheid geplaatst moeten worden opdat het concept naar behoren werkt. Door met behulp van AIS de schepen uit de werkelijkheid te simuleren en daarbij fictieve PH's op de kaart te plaatsen, kan worden nagegaan of daadwerkelijk de gewenste informatie in de PH's opgeslagen wordt. Als *Mobile Hubs and Smart Connected Containers* werkelijkheid wordt, lijkt ons een simulator als deze van groot belang.

7 Organisatie

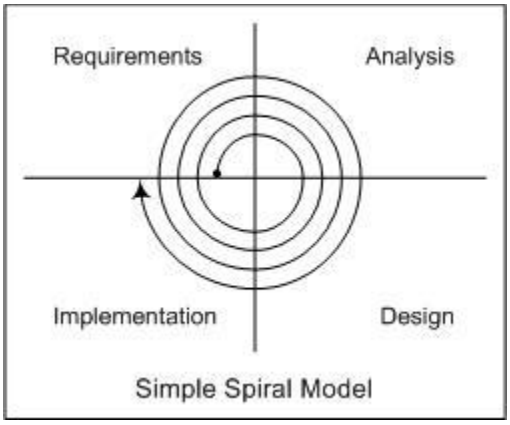
In dit hoofdstuk worden alle organisatorische aspecten van het project besproken. Allereerst wordt behandeld hoe we de gekozen ontwerpmethodiek hebben gevolgd. Hierna is de planning beschreven en daarna volgt een beschrijving over onze samenwerking en taakverdeling. Het hoofdstuk wordt afgesloten met het noemen van de software die wij gebruikt hebben.

7.1 Gevolgde ontwerpmethodieken

Aan het begin van het project hebben wij een ontwerpstrategie gekozen die ons het meest aansprak en die voor ons de beste resultaten zou bieden. In dit hoofdstuk wordt beschreven hoe het spiraalmodel voor ons heeft uitgepakt. Ook wordt toegelicht hoe het V-model terug te zien is in onze aanpak van dit project.

7.1.1 Spiraalmodel

Wij hebben zoveel mogelijk geprobeerd het spiraalmodel (zie Figuur 4) aan te houden tijdens de implementatie. Aan het begin van onze stage hadden we met de stagebegeleider afgesproken om elke twee weken overleg te plegen. Dit was een mooi aanknopingspunt om het spiraalmodel in de praktijk te brengen. Op deze manier moesten we om de week een werkende versie van onze simulator kunnen laten zien aan de stagebegeleiders.



Figuur 4: Visuele weergave van het spiraalmodel [3].

De eerste meeting vond plaats aan het einde van de onderzoeksfase. Om toch iets te kunnen laten zien, hadden we snel een simpel voorbeeld van de simulator gemaakt zonder functionaliteit. Bij dit voorbeeld werd gebruikt gemaakt van een aantal standaardfiguren, zoals rechthoeken en cirkels. Dit voorbeeld werd door de stagebegeleider als eenvoudig bestempeld. Dit klopte ook, omdat we nog niks hadden geïmplementeerd. We wisten dus dat onze volgende versie van de simulator er grafisch beter uit moest zien.

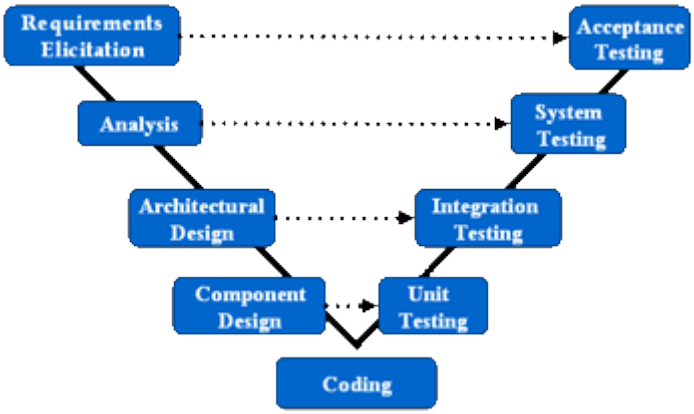
Toen we eenmaal aan het implementeren waren, konden we meer inzicht geven over wat we tijdens de komende meetings wilden laten zien. Voor de tweede meeting hadden we onszelf het doel gesteld om het uitlezen van MH's en PH's en het oversturen van informatie te hebben gevisualiseerd. Dit is allemaal

gelukt en de begeleiders waren blij dat er in ieder geval al iets beschikbaar was om het concept aan andere mensen te laten zien.

Er waren weinig op- of aanmerkingen over het product tot nu toe. Wij hadden onszelf daarom als doel gesteld om bij de volgende meeting bijna alle functionaliteit te hebben geïmplementeerd. Dit betekende dat het mogelijk moest zijn om MH's qua lading zelf in te kunnen stellen en dat het mogelijk moest zijn om een calamiteit te veroorzaken zodat de calamiteitenprocedure weergegeven kon worden. Ook deze versie werd goed ontvangen door de begeleiders. De enige opmerkingen die er waren, hadden betrekking op specifieke onderdelen voor de Surface tafel. Deze hebben we in de laatste versie van de simulator geïmplementeerd.

7.1.2 V-model

Tijdens het project hebben we het V-model (zie Figuur 5) gevolgd. Eerst zijn we achtereenvolgens alle ontwerpfasen doorgelopen (linker tak van de V) en daarna begonnen met implementeren. Tijdens elke iteratie van het spiraalmodel hebben we achtereenvolgens unittests, integratietest, systeemtest en als de heer Blom aanwezig was ook een acceptatietest uitgevoerd. Soms werden de integratie- en systeemtest gecombineerd en meestal werden deze tests uitgevoerd op de Surface Simulator. De acceptatietests vonden plaats op de Surface tafel zelf en werden meestal door de heer Blom en soms door de heer Van Lier uitgevoerd.



Figuur 5: Visuele weergave van het V-model [1].

7.2 Planning

De planning zoals we die aan het begin van het project hebben opgesteld is te zien in Figuur 6.

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11
Taken	19/4 - 23/4	26/4 - 30/4	3/5 - 7/5	10/5 - 14/5	17/5 - 21/5	24/5 - 28/5	31/5 - 4/6	7/6 - 11/6	14/6 - 18/6	21/6 - 25/6	28/6 - 2/7
Tijdsplanning											
Probleemstelling											
Onderzoek											
Analyse											
Ontwerp											
Implementatie											
Testen											
Evaluatie											
Verslag											
Uitloop											

Figuur 6: Planning aan het begin van het project.

Wij hebben geprobeerd om ons zoveel mogelijk aan deze planning te houden. Dit is goed gelukt. Onze eerste versies van de ontwerpdocumenten waren op zeven mei beschikbaar. Deze moesten echter nog wat aangepast worden en dit gebeurde in het begin van week vier. Hierdoor begonnen we één dag later met de implementatie dan gepland. Ook aan het einde van de geplande implementatiefase zijn we iets langer doorgegaan. Dit kwam omdat er bij de laatste evaluatiesessie een aantal handige tips werden gegeven die we nog wel wilden implementeren. Ook hadden we in de laatste week van de implementatiefase nog een test gedaan op de Surface tafel. Hierbij werden nog een aantal kleine foutjes gevonden die ook nog opgelost moesten worden. Zodoende begonnen we ook een dag of twee later met het schrijven van het verslag. Dit heeft echter geen effect op de afronding van het project. De eindpresentatie vindt plaats in week elf, dus het verslag moet in week tien af zijn. Over het algemeen bleek dat ons werktempo goed aan onze planning, die we aan het begin van het project hadden opgesteld, voldeed.

7.3 Samenwerking en taakverdeling

Wij kennen elkaar inmiddels drie jaar en hebben samen al meerdere projecten met succes afgerond. Voor ons was dit project op het gebied van onderlinge samenwerking dan ook geen grote uitdaging. Wat wel nieuw voor ons was, was het periodieke overleg met de werkgever. Hieruit bleek wel dat periodiek overleg heel verstandig is. Ook is het van belang om met mensen van verschillende vakgebieden te praten. Deze kunnen handige tips geven over zaken waar je zelf niet aan denkt.

De taakverdeling tussen ons drieën kwam eigenlijk vanzelf tot stand. Als iemand klaar was met zijn opdracht koos hij een volgende opdracht uit de lijst met taken die nog gedaan moesten worden. Op de takenlijst werd ook rekening gehouden met hoe belangrijk elke taak was. Zo werden de belangrijkste taken eerst gedaan.

Als we met het project bezig waren, zaten we altijd met zijn drieën bij elkaar. Dit is vooral handig als één van ons iets niet helemaal begreep. Bij vragen konden we gelijk met zijn drieën praten over de beste oplossing. Ook bij vragen over het werk wat een ander gedaan had, kon gelijk antwoord gegeven worden. Zodoende hoefde niemand lang te wachten op een antwoord op zijn vraag.

7.4 Gebruikte software

Het klassendiagram en de use cases zijn gemaakt in Microsoft Office Visio 2007 en de sequencediagrammen in JUDE [2]. Onze simulator is geschreven in C# in combinatie met WPF. Dit hebben wij geprogrammeerd gebruikmakend van Visual Studio 2008. Gezien er Surface functies gebruikt worden moet ook de Surface SDK geïnstalleerd zijn om het programma te kunnen compileren [8]. De Surface SDK is ontwikkeld voor een 32-bit versie van Microsoft Vista en wordt niet ondersteund door andere besturingssystemen. Wel is het met behulp van een hack werkend te krijgen op Windows 7 en de 64-bit versie van Windows Vista.

8 Conclusies en aanbevelingen

Met dit hoofdstuk wordt het verslag afgerond. Hierbij kijken we terug op hoe het project verlopen is en geven we aanbevelingen voor mogelijke uitbreidingen. Hierbij zullen we ook beschrijven hoe onze simulator gebruikt kan worden door andere applicatie ontwikkelaars.

8.1 Conclusies

Het resultaat van dit project is een goed werkende simulatie die de kracht van het *Mobile Hubs and Smart Connected Containers* concept weer kan geven. Samen met een goede presentatie kan het concept nu gemakkelijk bij andere mensen worden geïntroduceerd. Deze simulator is voor Centric ook de eerste mogelijkheid om het concept te visualiseren.

Tijdens het uitvoeren van dit project hebben wij veel geleerd. We hadden alle drie nog geen ervaring met het ontwikkelen in .NET. Dankzij dit project hebben we dat nu wel en dat kan ons goed van pas komen bij volgende projecten. Daarnaast hebben we gemerkt hoe belangrijk het is om het software ontwikkelproces goed te doorlopen en overal goed over na te denken. De onderzoeksfase die door de TU Delft verplicht werd, heeft ons goed gedaan. Hierdoor denk je dieper na over wat je precies wilt gaan maken en daar wordt het eindproduct alleen maar beter van. Ook was Centric blij met de ideeën waarmee we gekomen waren. Wel blijft het moeilijk om aan het begin van het project al een goed klassendiagram op te stellen. Hier is tijdens het project wel het een en ander aan veranderd.

Wat wij aan Centric hebben overhandigd is de simulator, de broncode van de simulator, het documentatierapport bij de broncode en een handleiding bij de simulator. Met deze middelen kan Centric alle kanten op om de simulator uit te breiden of om de simulator te gebruiken bij een presentatie. Daarnaast hebben wij zelf nog een filmpje opgenomen waarin wij het concept en de simulator demonstreren.

Alle requirements die aan het begin van het project waren opgesteld door Centric, zijn door ons nageleefd. Deze requirements zijn samen met door ons opgestelde requirements ondergebracht in een zogenaamd MoSCoW-overzicht. Alle requirements uit de categorieën “must haves” en “should haves” zijn geïmplementeerd alsmede enkele requirements uit de andere twee categorieën. Wij kunnen dit project als een succes beschouwen, omdat alle requirements zijn geïmplementeerd en alles op tijd is afgerond.

Wij hebben de samenwerking met Centric als heel prettig ervaren. We waren heel vrij in onze werktijden en waar we werkten. Daarnaast werden we voorzien in alles wat we nodig hadden, van koffie tot laptops. Ook werden we bij het bedrijf betrokken als er iets voor het personeel werd georganiseerd. Wij willen daarom Centric bedanken dat ze ons de mogelijkheid hebben gegeven om ons bacheloreindproject bij hen te mogen doen. Wij hopen dat Centric veel profijt van onze simulator zal hebben en dat het *Mobile Hubs and Smart Connected Containers* concept een succes wordt.

8.2 Aanbevelingen

In de toekomst kan de simulator uitgebreid worden met andere transportentiteiten om te laten zien dat het concept ook toegepast kan worden buiten de binnenvaartsector. Bij vrachtauto's kan gebruik worden gemaakt van reeds bestaande wegportalen en bij treinen kunnen PH's bevestigd worden aan bestaande constructies voor bovenleidingen. Hoewel het concept hetzelfde blijft bij andere transportentiteiten, kan het wel zo zijn dat PH's op een andere manier MH's scannen. Vrachtwagens kunnen bijvoorbeeld ingehaald worden en van baan verwisselen. Hier moeten de PH's mee om kunnen gaan. In hoeverre aanpassingen nodig zijn, zal onderzocht moeten worden.

Naast het uitbreiden met andere transportentiteiten verdient het ook aanbeveling om het concept en de simulator daarvan uit te breiden met andere bestaande end-to-end oplossingen. Hierbij kunnen bestaande systemen gebruikt en gecombineerd worden zodat niet voor elke end-to-end oplossing nieuwe hardware aangeschaft hoeft te worden. Denk bijvoorbeeld aan *Tracking and Tracing*, monitoren van containers (temperatuur, luchtdruk, luchtvochtigheid, etc.) en monitoren van personeel (werktijden, productiviteit, etc.). Ook kan gedacht worden aan het dynamisch toewijzen van resources. Blijkt er ergens file te staan, dan kan de vervoerder misschien beter gebruik maken van scheepvaart in plaats van vrachtwagens. Door het toewijzen van resources dynamisch te laten gebeuren kunnen tijd en kosten bespaard worden.

Het vragenstellende actoren idee lijkt ons een aantrekkelijke manier om verschillende actoren bij de simulator te betrekken. Zo kan voor elke actor relevante aspecten van het concept worden gesimuleerd.

Voor mensen die met de code van onze simulator aan de slag willen gaan is het raadzaam om het klassendiagram te bekijken, de handleiding te lezen en eventueel de documentatie te bekijken alvorens aan de code te gaan sleutelen. Hieruit zal blijken hoe onze simulator geprogrammeerd is en wat de functionaliteiten van de simulator zijn. Voor het testen van de simulator is het aan te raden om eerst te testen met de Surface Simulator en daarna pas op de Surface tafel. Op die manier wordt een hoop tijd bespaard die nodig is om de Surface tafel op te starten. Bovendien is het een aanrader om te zoeken naar een tool die automatisch de GUI kan testen zodat de GUI niet elke keer handmatig getest hoeft te worden.

Begrippen en afkortingen

.NET Framework	Het Microsoft .NET Framework is een software framework ten behoeve van de samenwerking van applicaties geschreven in verschillende programmeertalen.
AIS	Automatic Identification System (AIS) is een aanvulling op het bestaande verkeersmanagement van verkeersposten door Rijkswaterstaat en de bestaande schip-schip communicatie.
C#	C# is een objectgeoriënteerde programmeertaal ontwikkeld door Microsoft als deel van het .NET Framework.
Garbage collector	De garbage collector probeert geheugen vrij te geven dat in gebruik is door objecten die niet meer door de applicatie aangeroepen zullen worden.
Gesture	Een gesture is een vorm die met behulp van vingers of handen op de Surface tafel getekend is. Gestures vervangen de functionaliteit van het toetsenbord en de muis.
GUI	Graphical User Interface (GUI) is de grafische weergave van het systeem.
HTML	HyperText Markup Language (HTML) is een opmaaktaal voor de specificatie van documenten, voornamelijk bedoeld voor het Internet.
Mobiele hub (MH)	Een klein apparaatje in een vervoersmiddel met een communicatiemogelijkheid voor korte afstand naar containers om ladinggegevens uit te wisselen en een communicatiemogelijkheid voor de middellange afstand waarmee uitgelezen kan worden welke lading in het vervoersmiddel aanwezig is.
Model-View-Controller (MVC)	Ontwerppatroon waarbij klassen worden ingedeeld aan de hand van verschillende verantwoordelijkheden: datamodel (model), datapresentatie (view) en applicatielogica (controller).
Permanente hub (PH)	Een apparaat met een vaste positie en met een communicatiemogelijkheid voor de middellange afstand waarmee ladinggegevens en informatie over een vervoersmiddel opgevraagd en tijdelijk opgeslagen kunnen worden.

Smart container	Een container met een chip of zeer klein apparaatje dat dient als gegevensdrager waarin de inhoud van de container wordt bewaard die wordt ingevoerd bij laden en lossen. Beschikt over een communicatiemogelijkheid voor de korte afstand naar een mobiele hub waarmee gemeld kan worden welke lading de container bevat.
Mobile Hubs and Smart Connected Containers	Door Centric bedacht concept waarbij voor de binnenvaart sector met behulp van smart containers, mobiele hubs en permanente hubs ten altijd snel informatie over de lading van een schip opgevraagd kan worden.
Spiraalmodel	Ontwerpmethodiek waarbij vier fasen worden onderscheiden: eisen opstellen, analyse, ontwerpen en implementeren. Nieuwe functionaliteit wordt incrementeel ingevoerd door de spiraal te doorlopen.
Surface Simulator	Een door Microsoft ontwikkelde simulator voor de Surface tafel die het mogelijk maakt om applicaties voor de Surface tafel te testen zonder gebruik te maken van de Surface tafel.
Surface tafel	De Surface tafel is een multi-user-multi-touch-product van Microsoft dat ontwikkeld is als een software en hardware combinatie die één of meerdere gebruiker(s) toestaat om content te bewerken door middel van menselijke bewegingen of objecten.
V-model	Ontwerpmethodiek met gelijke aandacht voor ontwikkeling en verificatie waarbij softwareontwerp en verificatie zijn opgedeeld in een aantal fasen en elke fase de basis vormt voor een nieuwe fase.
WPF	Windows Presentation Foundation (WPF), het grafische subsysteem dat een onderdeel is van het Microsoft .NET Framework (sinds versie 3.0).

Literatuurlijst

[1] Cetic (z.d.). *The Acceptance Tests Generator.* Geraadpleegd op 23 juni 2010, http://www.cetic.be/article221.html

[2] Change Vision, Inc. (2010). *JUDE : UML, ER, CRUD, DFD, Flowchart and Mind Map: Design & Modeling Tool.* Geraadpleegd op 23 juni 2010, http://jude.change-vision.com/jude-web/index.html

[3] Compass Software, Inc. (2007). *Iterative Development.* Geraadpleegd op 23 juni 2010, http://www.compass-se.com/iterativeDev.html

[4] Dimitri van Heesch (2010). *Generate documentation from source code.* Geraadpleegd op 23 juni 2010, http://www.stack.nl/~dimitri/doxygen/

[5] Gamma, E, Helm, R, Johnson, R en Vlissides, J, 1995, *Design patterns: elements of reusable object-oriented software*, Addison-Wesley, Reading, MA

[6] Microsoft Corporation (2010). *Microsoft Visual Studio.* Geraadpleegd op 23 juni 2010, http://msdn.microsoft.com/nl-nl/vstudio/default(en-us).aspx

[7] Microsoft Corporation (2010). *Visual C# Developer Center*. Geraadpleegd op 23 juni 2010, http://msdn.microsoft.com/nl-nl/vcsharp/default(en-us).aspx

[8] Microsoft Surface (2010). *The Microsoft Surface platform.* Geraadpleegd op 23 juni 2010, http://www.microsoft.com/surface/en/us/Pages/Product/Platform.aspx

[9] NUnit.org (2010). *What Is NUnit?* Geraadpleegd op 23 juni 2010, http://www.nunit.org/

[10] Open Office (2010). *OpenOffice Intro.* Geraadpleegd op 23 juni 2010, http://nl.openoffice.org/

[11] SciTech Software (2010). *.NET Memory Profiler.* Geraadpleegd op 23 juni 2010, http://memprofiler.com/

[12] Van Lier, B, (2009), *Luhmann ontmoet 'the Matrix',* Eburon Academic Publishers, Delft, 206p

[13] W3.org (2010). *HTML Working Group.* Geraadpleegd op 23 juni 2010, http://www.w3.org/html/wg/

Bijlagen

Bijlage A. Opdrachtomschrijving

Opdrachtomschrijving

Bachelorproject Rob Post, Oskar van Rest en Tim Roorda

20 april 2010

1 Introductie

Als onderdeel van onze Bachelorfase van de opleiding Technische Informatica aan de TU Delft zullen wij een stage lopen. Hiervoor hebben wij contact gezocht met het bedrijf Centric en uit enkele gesprekken is een opdracht naar voren gekomen die ons aanspreekt. In dit document zullen wij deze opdracht formuleren en omschrijven in welke omgeving deze opdracht gemaakt zal worden. Voordat we dat doen zullen we eerst iets over het bedrijf Centric vertellen.

In 1978 begon Gerard Sanderink als zelfstandig ondernemer en in 1992 startte hij de Sanderink Groep. Sinds 2000 heet het bedrijf Centric: een dienstverlener met een breed pakket aan diensten en producten. Centric is inmiddels uitgegroeid tot een van de grootste ICT-organisaties van Nederlandse oorsprong. Met rond 10.000 werknemers heeft Centric vestigingen in onder andere Nederland, België, Luxemburg en andere Europese landen en is daarmee de grootste niet-beursgenoteerde onderneming van de Benelux. Centric is actief in diverse branches waaronder overheid, financiële dienstverlening, woningcorporaties, groothandel, industrie, retail en zorginstellingen. Centric is een innovatief bedrijf en hecht grote waarde aan duurzaamheid en maatschappelijk verantwoord ondernemen. Voor meer informatie over Centric zie http://www.centric.eu.

2 Projectopdracht

In dit hoofdstuk wordt de projectopdracht beschreven. Allereerst wordt iets verteld over de projectomgeving: in welke omgeving speelt het door Centric bedachte *Mobile Hubs and Smart Connected Containers* concept zich af. Daarna komt de doelstelling van ons project naar voren en zal onze opdracht geformuleerd worden. Vervolgens worden de in te leveren producten opgenoemd en dit hoofdstuk sluit af met minimale eisen aan het eindproduct.

2.1 Projectomgeving

De haven van Rotterdam behoort tot de grootste en belangrijkste havens ter wereld en is daarom van groot belang voor de supply chain. In de gehele wereld circuleren zo'n 440 miljoen containers en hoewel deze containers gestandaardiseerd zijn, hebben we te maken met verschillende transportmodaliteiten (bijvoorbeeld zeeschip/vrachtwagen), concurrentie van verschillende commerciële bedrijven en verschillende wetgeving van landen.

De informatie over de locatie van de containers, de inhoud, de eigenaar, de bestemming, etc., is vaak alleen maar beschikbaar bij een klein aantal partijen die er direct iets mee te maken hebben. Mocht er in Rotterdam bijvoorbeeld een ongeluk gebeuren met een vrachtschip (denk aan een botsing of brand), dan weet eigenlijk niemand zo snel van wie de betreffende containers zijn en wat er precies in zit. Dit maakt het hulpdiensten bijvoorbeeld erg lastig (denk aan een brandweer die niet weet of een container explosieve/giftige stoffen bevat). Deze informatie moet mogelijkerwijs beschikbaar komen, zodat de juiste instanties op het juiste moment over de juiste informatie beschikt.

Om dit te bewerkstelligen is het concept *Mobile Hubs and Smart Connected Containers* bedacht, waarbij met behulp van *clustering*, *respecteren van autonomie* en *informatie bij de bron*, de complexiteit van het probleem tegemoet gekomen wordt en de belangen van de verschillende spelers behartigd worden.

2.2 Doelstelling project

Het doel van het project is om het concept *Mobile Hubs and Smart Connected Containers* visueel te maken, zodat eventuele klanten van Centric geïnteresseerd raken en het als een mogelijke oplossing gaan zien. Ons is gevraagd een demo te maken die dit bewerkstelligd en onder andere in november dit jaar op een beurs getoond worden om klanten te trekken. Op die manier wil Centric onderzoeken of er een markt is voor de realisatie van hun model. Bij de presentatie op de beurs wordt gebruik gemaakt van Surface tafels. Op deze manier kunnen verschillende aspecten van het model eenvoudig gevisualiseerd en uitgelegd worden.

2.3 Opdrachtformulering

De projectopdracht is het maken van een demo die een mogelijk scenario simuleert, zodat de werking van het concept *Mobile Hubs and Smart Connected Containers* gevisualiseerd wordt. De demo moet duidelijk laten zien welke informatie er uitgewisseld wordt door de verschillende systemen, welke informatie waar wordt opgeslagen en hoe deze informatie toegankelijk gemaakt kan worden. Het systeem moet voor eventuele klanten transparant, betrouwbaar en realistisch blijken zodat ze overtuigd raken van de werking van het concept.

De presentatie van de demo gebeurt op een zogenaamde Surface tafel. Het is dus belangrijk dat de demo voor dit systeem ontwikkeld wordt. Daarnaast is het wenselijk dat de demo ook op normale pc's werkt. Andere zaken waaraan de projectgroep moet voldoen zijn te vinden in het hoofdstuk eisen en beperkingen.

2.4 In te leveren producten

Het hoofdproduct dat geleverd moet worden is de demo die de werking van het systeem laat zien. Daarnaast moet er natuurlijk documentatie bij deze demo komen, zodat dat duidelijk is hoe de demo werkt en hoe de demo in de toekomst uitgebreid kan worden.

2.5 Eisen en beperkingen

Er wordt gebruik gemaakt van een vooraf willekeurig gekozen landkaart.
De schaal van de kaart moet realistisch zijn voor het aantal hubs.
Op de kaart moet een rechte vaarroute te zien zijn.
Op de kaart moeten 5 tot 7 MH's te zien zijn, met de volgende eisen:

- Varen van links naar rechts of visa versa
- Varen ze aan de ene kant van de kaart af komen ze aan de andere kant er weer op
- 3 van de MH's zijn wat betreft schip en lading instelbaar
- De overige MH's zijn dummy's en dus niet instelbaar

Op de kaart zijn 3 tot 5 PH's zichtbaar die de aanwezigheid en vracht van MH's registreren. De PH's zijn altijd uitleesbaar.

Er zijn drie soorten calamiteiten, waarvan de locatie bekend is:

- (Hoog) er is geen response van de MH en de bemanning; daarnaast is lading verloren.
- (Middel) Er is response van de MH, maar niet van de bemanning; daarnaast is (een deel van de) lading verloren.
- (Laag) Response van MH en bemanning; schip en/of lading is beschadigd.

De demo wordt ontwikkeld in C# in combinatie met Windows Presentation Foundation (WPF) die beide onderdeel zijn van het .NET Framework. Er moet goede documentatie bij zitten zodat eventuele uitbreiding of aanpassing van de software, nadat wij dit project hebben afgerond, mogelijk is. Het eindproduct moet eind juni beschikbaar komen.

Bijlage B. Plan van Aanpak

Plan van Aanpak

Bachelorproject Rob Post, Oskar van Rest en Tim Roorda

22 april 2010

Voorwoord

Het document dat voor u ligt is een beschrijving van het project en de manier van hoe wij het willen aanpakken. Daarbij wordt de opdracht omschreven, het probleem afgebakend, een globale tijdsplanning gegeven en de inrichting van het project beschreven. Ook wordt de aanpak en kwaliteitswaarborging kort toegelicht. In dit verslag wordt vastgelegd waaraan de opdrachtgever (Centric) en wij moeten voldoen. Bovendien worden overige afspraken met betrekking tot het verloop van het project vastgelegd.

1 Introductie

Wij moeten als onderdeel van onze bachelor een bachelorproject doen en dat doen wij in de vorm van een stage. Hiertoe hebben wij het bedrijf Centric benaderd met de vraag of zij een opdracht voor ons hadden. Centric heeft een concept voor smart containers bedacht waarbij op een eenvoudige manier bijna realtime de lading van een container kan worden opgevraagd zonder daarbij van een centrale database gebruik te maken. Centric wil een simulatie van dit concept hebben om hiermee te laten zien hoe het concept werkt en daarmee te bekijken of er markt is voor een realisatie van dit concept. Ons is gevraagd die simulatie te leveren, zodat deze gebruikt kan worden bij een beurs in november om te laten zien dat en hoe het concept werkt. Deze simulatie zou voor een Surface tafel ontwikkeld moeten worden.

In het volgende hoofdstuk zal de projectopdracht preciezer worden omschreven. Ook wordt de opdracht afgebakend en zal duidelijk worden waaraan het product zal moeten voldoen. In hoofdstuk drie wordt de aanpak omschreven en wordt een globale tijdsplanning gegeven. De projectinrichting zal aan bod komen in hoofdstuk vier. Het laatste hoofdstuk zal gaan over de kwaliteitsborging waarin wordt beschreven hoe de kwaliteit van het project hoog gehouden wordt en risico's worden voorkomen.

2 Projectopdracht

In dit hoofdstuk wordt de projectopdracht beschreven. Bij deze beschrijving wordt het probleem vastgesteld en afgebakend. Daarnaast wordt duidelijk waaraan het product moet voldoen.

2.1 Projectomgeving

De haven van Rotterdam behoort tot een van de grootste en belangrijkste havens ter wereld. De Rotterdamse haven is daarom van groot belang voor de supply chain in een groot deel van Europa. In de gehele wereld circuleren zo'n 440 miljoen zeecontainers en hoewel deze containers gestandaardiseerd zijn, hebben we te maken met verschillende transportmodaliteiten zoals vrachtwagens, schepen en treinen, concurrentie van verschillende commerciële bedrijven en verschillende wetgeving van landen.

De informatie over de locatie van de containers, de inhoud, de eigenaar, de bestemming, enzovoorts, is vaak alleen maar beschikbaar bij een klein aantal partijen die er direct iets met deze container te maken hebben. Mocht er in ergens een ongeluk gebeuren waarbij een zeecontainer betrokken is, dan weet niemand zo snel van wie de betreffende container is en wat er precies in zit. Dit maakt het werk voor hulpdiensten bijzonder lastig. Deze informatie moet mogelijkerwijs beschikbaar komen, zodat de juiste instanties op het juiste moment over de juiste informatie beschikken.

Om dit te bewerkstelligen is het concept *Mobile Hubs and Smart Connected Containers* bedacht, waarbij met behulp van *clustering*, *respecteren van autonomie* en *informatie bij de bron*, de complexiteit van het probleem tegemoet gekomen wordt en de belangen van de verschillende spelers behartigd worden.

2.2 Doelstelling project

Het doel van het project is om het concept *Mobile Hubs and Smart Connected Containers* visueel te maken, zodat eventuele klanten van Centric geïnteresseerd raken en het als een mogelijke oplossing gaan zien van het hierboven beschreven probleem. Ons is gevraagd een demo te maken die dit bewerkstelligt en onder andere in november dit jaar op een transportbeurs getoond kan worden om klanten te trekken. Op die manier wil Centric onderzoeken of er een markt is voor de realisatie van hun model. Bij de presentatie op de transportbeurs wordt gebruik gemaakt van een Surface tafel. Met deze manier van presenteren kunnen verschillende aspecten van het concept duidelijk gevisualiseerd en uitgelegd worden.

2.3 Opdrachtformulering

De projectopdracht is het maken van een demo die de werking van het concept *Mobile Hubs and Smart Connected Containers* visualiseert. Dit moet worden gedaan door middel van het simuleren van een aantal schepen die containers vervoeren. De demo moet duidelijk laten zien welke informatie er uitgewisseld wordt tussen de verschillende componenten van het systeem, welke informatie waar wordt opgeslagen en hoe deze informatie toegankelijk gemaakt kan worden. De demo moet ook een aantal rampscenario's bevatten zodat de kracht van het concept getoond kan worden. Het systeem moet voor eventuele klanten transparant, betrouwbaar en realistisch blijken zodat ze overtuigd raken van de werking van het concept.

De presentatie van de demo gebeurt op een zogenaamde Surface tafel. Het is dus belangrijk dat de demo voor dit systeem ontwikkeld wordt. Daarnaast is het wenselijk dat de demo ook op normale pc's werkt, zodat de demo ook daarop gepresenteerd kan worden.

2.4 In te leveren producten

Het hoofdproduct dat geleverd moet worden is de demo die de werking van het systeem laat zien. Daarbij is het minder belangrijk dat het product er grafisch goed uitziet, de prioriteit ligt bij het laten zien van de werking van het concept. De documenten die ontwikkeld worden tijdens het ontwerp van de demo worden ook meegeleverd met het eindproduct.

2.5 Eisen en beperkingen

Door de opdrachtgever worden een aantal eisen aan de demo gesteld. De volgende punten moeten in de demo verwerkt zijn:

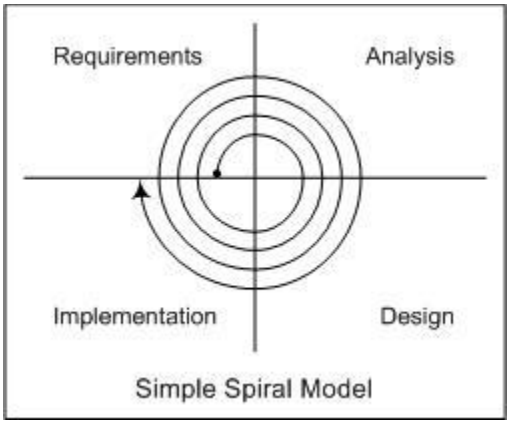
- Er wordt gebruik gemaakt van een vooraf gekozen landkaart.
- De schaal van de kaart moet realistisch zijn voor het aantal hubs.
- Op de kaart moet een rechte vaarroute te zien zijn.
- Op de kaart moeten 5 tot 7 MH's (schepen) te zien zijn, met de volgende eisen:
  - Varen van links naar rechts of visa versa.
  - Varen ze aan de ene kant van de kaart af komen ze aan de andere kant er weer op.
  - 3 van de MH's zijn wat betreft schip en lading instelbaar.
  - De overige MH's zijn dummy's en dus niet instelbaar.
- Op de kaart zijn 3 tot 5 PH's zichtbaar die de aanwezigheid en vracht van MH's registreren. De PH's zijn altijd uitleesbaar.
- Er zijn drie soorten calamiteiten, waarvan de locatie bekend is:
  - (Hoog) Er is geen response van de MH en de bemanning; daarnaast is lading verloren.
  - (Middel) Er is response van de MH, maar niet van de bemanning; daarnaast is (een deel van de) lading verloren.
  - (Laag) Er is response van MH en bemanning; schip en/of lading is beschadigd.
- De demo wordt ontwikkeld in C# in combinatie met Windows Presentation Foundation (WPF) die beide onderdeel zijn van het .NET Framework.

3 Aanpak

In dit hoofdstuk wordt onze manier van werken uitgelegd. Allereerst geven we aan volgens welke methodiek we te werk zullen gaan en welke technieken wij zullen gebruiken. Vervolgens geven we een tijdsindeling.

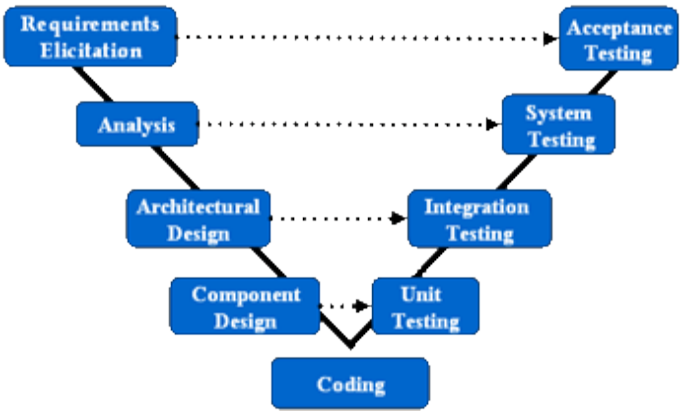
3.1 Methodiek

We willen zoveel mogelijk te werk gaan volgens het spiraal model (zie Figuur 1) en al zo vroeg mogelijk een werkend geheel beschikbaar hebben. Vervolgens kunnen we incrementeel functionaliteit toevoegen door nieuwe functionaliteit telkens weer de verschillende fasen te laten ondergaan. Op deze manier kan al in een vroeg stadia nagegaan worden of het product aan de verwachtingen van de opdrachtgever voldoet en ook blijft voldoen. We zijn van plan deze cirkel 4 keer te doorlopen (zie paragraaf 3.3) en het resultaat telkens aan de opdrachtgever laten zien.



Figuur 1: Spiraalmodel (http://www.compass-se.com/iterativeDev.html).

Nadat het systeem ontworpen is en in verschillende componenten verdeeld is beginnen we gelijktijdig met zowel het implementeren als het testen van het systeem. Bij het testen willen we te werk gaan volgens het V-model (zie Figuur 2). Elke fase van het project wordt hierbij op een toepasselijke manier getest: nieuw toegevoegde code wordt eerst apart getest (Unit Testing), vervolgens worden samenhangende componenten getest (Integration Testing), daarna wordt het gehele systeem getest aan de hand van use cases (System Testing) en ten slotte wordt getest of het aan de verwachtingen van de opdrachtgever voldoet door een demonstratie te geven van de nieuwe functionaliteit (Acceptance Testing). Natuurlijk kunnen we voor elke iteratie (zoals beschreven hierboven) het V-model opnieuw gebruiken. We voegen bij een iteratie functionaliteit toe, testen de nieuwe functies door middel van Unit Testing, Integration Testing en System Testing en laten de nieuwe functionaliteit aan de opdrachtgever zien en starten weer een nieuwe iteratie.



Figuur 2: V-model (http://www.cetic.be/article221.html).

3.2 Technieken

Eén van de eisen is dat wij het C# in combinatie van WPF gebruiken, zie ook 2.4. De code willen we mogelijkerwijs testen met behulp van NUnit testing.

Voor het delen van documenten tussen de groepsleden willen we gebruik maken van Office Groove, eventueel in combinatie met Office Sharepoint.

Er moet nog onderzoek gedaan worden welke techniek we willen gebruiken voor het opslaan van data en naar al beschikbare simulatiesoftware die we eventueel kunnen gebruiken.

3.3 Planning

In Figuur 3 is een globale planning te zien die we zo strak mogelijk aan proberen te houden. We willen in principe in week 8 klaar zijn met het implementeren, zodat we in week 9 en 10 tijd hebben om aan documentatie en het verslag te werken en ook om de presentatie op de TU Delft te geven. Mocht dit niet lukken, dan kunnen we week 11 nog voor uitloop gebruiken. We werken 5 dagen in de week waarbij 3 dagen bij Centric en 2 dagen aan de TU Delft. Eén keer in de twee weken evalueren we onze voortgang met de begeleiders van Centric.

Zoals te zien is wordt onze voortgang om de week geëvalueerd door de op opdrachtgever. Deze evaluaties vinden plaats aan het begin van de week (maandag of dinsdag). Wij proberen bij iedere evaluatie een ronde uit het spiraal model doorlopen te hebben, zodat we nieuw toegevoegde functionaliteit telkens aan de opdrachtgever kunnen laten zien (Acceptance Testing). Dit zal de eerste keer in week 4 plaatsvinden, waarbij wij hopen een mockup-versie te kunnen laten zien. In week 6 en week 8 is de bedoeling dat wij een werkend systeem kunnen laten zien waarbij een deel van de functionaliteit geïmplementeerd is, en in week 10 zou alle functionaliteit geïmplementeerd moeten zijn en zou het systeem aan alle eisen moeten voldoen.

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11
Taken	19/4 - 23/4	26/4 - 30/4	3/5 - 7/5	10/5 - 14/5	17/5 - 21/5	24/5 - 28/5	31/5 - 4/6	7/6 - 11/6	14/6 - 18/6	21/6 - 25/6	28/6 - 2/7
Tijdsplanning											
Probleemstelling											
Onderzoek											
Analyse											
Ontwerp											
Implementatie											
Testen											
Evaluatie											
Verslag											
Uitloop											

Figuur 3: Tijdsplanning aan het begin van het project.

4 Projectinrichting

In dit hoofdstuk wordt de inrichting van het project toegelicht. Hoe wordt het project in goede banen geleidt en wat is hiervoor nodig? Ook wordt uitgelegd op welke manier gecommuniceerd zal worden naar de opdrachtgever. Als laatste worden de werkplekken omschreven en wat de benodigde middelen zijn bij het uitvoeren van dit project.

Dit project telt drie projectleden (Oskar, Rob en Tim) die alle drie dezelfde verantwoordelijkheden dragen. Door elkaars werk te controleren blijven we constant op de hoogte wie wat doet en of het goed gebeurt. De projectleden worden begeleidt door een stagebegeleider van de TU Delft, Mathijs de Weerdt, en een begeleider van Centric, Leen Blom. Daarnaast houdt ook Ben van Lier onze voortgang in de gaten. Leen en Ben hebben samen het smart container concept bedacht en bij hen kunnen we terecht met inhoudelijke vragen. Verder hebben we contact gelegd met Jean-Paul en Alex die met de Surface-tafel werken.

Om onze aanwezigheid bij Centric te bekijken kunnen Leen en Ben onze Outlook agenda's bekijken en op die manier worden ook afspraken gemaakt. Verder houden wij tijdens de onderzoeksfase een TU Delft weblog bij (http://rot.weblog.tudelft.nl) die toegankelijk is of wordt voor Mathijs, Leen, Ben en onze stagecoördinator vanuit de TU Delft, Bernard Sodoyer. Als laatste gebruiken we Groove, een versiebeheer programma, zodat alle betrokkenen binnen Centric ons werk op elk moment kunnen inzien. Ook spreken we tenminste elke twee weken af met Leen en zullen we geregeld afspreken met Mathijs om onze voortgang te bespreken.

Communiceren met Leen en Ben kan via de email, maar we kunnen als ze aanwezig zijn ook gemakkelijk even langslopen. Dit geldt ook voor Jean-Paul en Alex als we iets over de Surface tafel moeten weten. Verder hebben we elke twee weken een afspraak om de voortgang in persoon te bespreken met Leen en Ben. Ook zullen er ontwerpdocumenten als het Bijlage D. Requirements Analysis Document worden geschreven zodat op papier terug te vinden is hoe we te werk gaan en wat we doen.

Wij zullen drie dagen in de week werkplekken van Centric gebruiken en twee dagen in de week op de TU Delft. Centric voorziet ons van elk een laptop zodat we overal terecht kunnen. Hierop staat de benodigde software geïnstalleerd bestaande uit Visual Studio 2008, Microsoft Office 2007 en Groove. Verder hebben we op afspraak toegang tot een Surface tafel.

5 Kwaliteitsborging

Om de kwaliteit van de demo hoog te houden is er tussen de opdrachtgever en de opdrachtnemers afgesproken om elke twee weken een evaluatiegesprek te houden. Hierin wordt medegedeeld wat er in de afgelopen twee weken bereikt is en waar er tegen problemen is aangelopen. De opdrachtgever kan zo tijdig ingrijpen als het project de verkeerde kant op dreigt te gaan. Ook wordt er periodiek met de stagebegeleider van de TU Delft afgesproken om het proces door te spreken.

Tijdens de eerste drie weken wordt door ons ook een weblog bijgehouden waarop onze werkzaamheden in die drie weken te komen te staan. Hierop kunnen de stagebegeleiders en opdrachtgever kijken waarmee we bezig zijn en feedback daarop geven.

Bijlage C. Onderzoeksverslag

Onderzoeksverslag

Bachelorproject Rob Post, Oskar van Rest en Tim Roorda

6 mei 2010

Inhoudsopgave

1 Inleiding

Het document dat voor u ligt is het resultaat van de onderzoeksfase met betrekking tot het visualiseren van het smart container concept dat door Centric is bedacht. Allereerst wordt in dit document de probleemstelling omschreven die wij tijdens dit project op gaan lossen. In hoofdstuk drie laten we zien wat een simulator nu precies is en welk ontwerppatroon vaak gebruikt wordt bij het implementeren van simulatiesoftware. In het daarop volgende hoofdstuk dragen wij alternatieve visualisaties aan, dus andere mogelijkheden om de probleemstelling op te lossen. In hoofdstuk vijf geven we mogelijke verbeteringen en aanvullingen op het simulatie concept zoals dat door Centric voorgedragen is. In het hoofdstuk Programmeeromgeving doen wij onderzoek naar de verschillende technieken en technologieën die wij zullen gebruiken bij het realiseren van de simulatie. Ten slotte sluiten we af met een discussie waarbij we de gemaakte keuzes discussiëren en waarbij we reflecteren op de opdracht en het proces.

2 Probleemstelling

Centric heeft het smart container concept bedacht en wil dit presenteren aan potentiële klanten. Hiertoe is door Centric al een mogelijke uitwerking van een simulatie bedacht [2][3]. Bij het presenteren van het smart container concept is interactie wenselijk en moeten zoveel mogelijk actoren geïnteresseerd raken. De presentatie van het smart container concept richt zich vooral op de volgende actoren:

Direct betrokkenen:

- Douane
- Havenautoriteit
- Rijkswaterstaat
- Hulpdiensten:
    - Politie
    - Brandweer
    - Ambulance

Stakeholders:

- Havenbedrijven
- Vervoerder
- Verlader

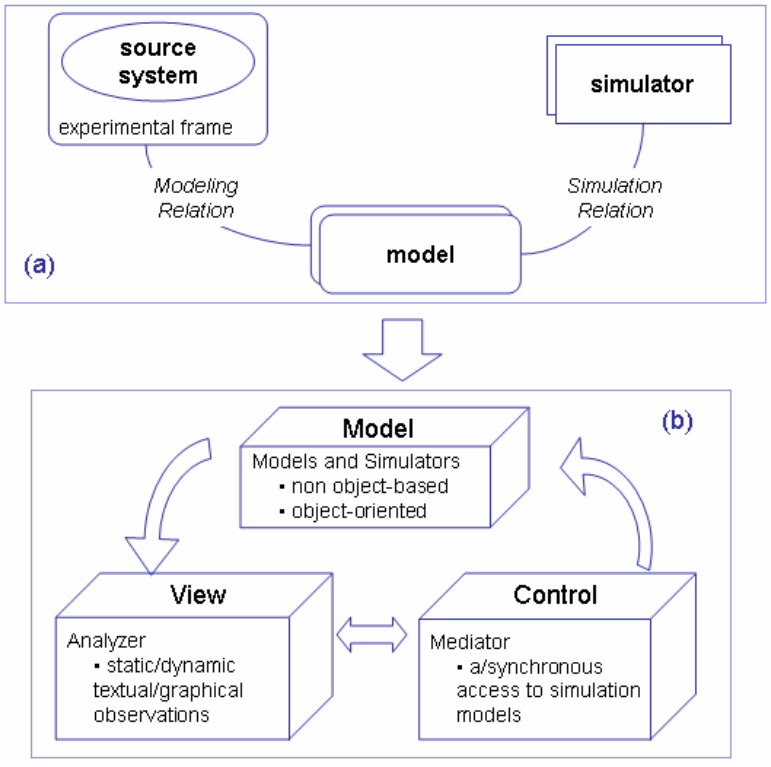
De verschillende actoren kunnen door de netcentrische aanpak alleen bij de gegevens die voor hen relevant zijn. Voor elke actor zijn verschillende vraagstukken interessant. Een vervoerder wil bijvoorbeeld weten waar zijn voertuigen zijn en een havenbedrijf wil weten waar een bepaalde container opgeslagen staat. Al deze aspecten moeten gepresenteerd kunnen worden aan potentiële klanten. In dit onderzoeksverslag zullen wij eerst uitwerkingen van een aantal alternatieve oplossingen

van het probleem behandelen. Daarna verfijnen we het door Centric voorgestelde concept dat we tijdens ons project zullen realiseren.

3 Het ontwerpen van een simulator

Een simulator is een nabootsing van de werkelijkheid, waarbij de meestal continue veranderingen in de tijd gesimuleerd worden door discrete stappen in de tijd. Dit wordt ook wel Discrete-Event System Simulation genoemd [1]. Omdat we in de werkelijkheid te maken hebben met vele variabelen en een grote hoeveelheid data, worden zulke simulaties vaak met behulp van een computer gedaan.

Sarjoughian en Singh suggereren dat het verstandig is het concept uit Figuur 1 aan te houden bij het ontwerpen van zo'n simulator [10]. In dit concept worden de verschillende onderdelen van een simulator geïmplementeerd door middel van het Model-View-Controller (MVC) ontwerppatroon. Dit vereenvoudigt het implementatieproces omdat het systeem een hoger abstractieniveau krijgt. Ook is het systeem hierdoor beter te valideren en dit leidt dus tot hogere kwaliteit software. Naast het bieden van een grafische user interface bevat het view-gedeelte nu ook logica voor het experimenteren met het simulatiemodel. De controller biedt de logica om de aanvragen die de gebruiker via het user interface doet uit te voeren en gebruikt daarbij het model. Daarnaast kan de controller logica bevatten voor het simuleren van delen uit de werkelijkheid waarbij geen gebruikersinteractie nodig is.



Figuur 1: Het afbeelden van een simulator op het MVC ontwerppatroon [10].

4 Alternatieve visualisaties

Dit hoofdstuk beschrijft een aantal alternatieve oplossingen om het door Centric bedachte concept te visualiseren.

4.1 Voor elke actor één simulatie

Een mogelijkheid om het concept aan verschillende actoren te verkopen is om voor elke actor een simulatie te maken die vanuit het perspectief van die actor naar het vervoeren van containers aankijkt. Zo zou voor een containereigenaar een simulatie gebouwd kunnen worden die één specifieke container van die eigenaar blijft volgen. Dit volgen moet mogelijk zijn vanaf het laden bij het vertrekpunt tot en met het lossen op de bestemming. Op die manier zouden alle informatiestromen die betrekking hebben op deze ene container gevisualiseerd kunnen worden. Eenzelfde idee gaat op voor een rederij die één van haar schepen wil volgen. Zo is er voor elke actor wel een specifieke simulatie te maken.

Het grote voordeel van het bouwen van deze verschillende simulaties is dat elke actor voor haar interessante aspecten van het concept te zien krijgt. De actor krijgt geen overbodige informatie te zien waarvan zij misschien niks begrijpt. Ook kan de actor zich dan beter indenken hoe de situatie voor haar verandert en worden de voordelen misschien beter zichtbaar. Bovendien zal op deze manier elke simulatie minder complex worden.

Een nadeel is echter dat er verschillende applicaties ontwikkeld moeten worden. Dit kost meer tijd en resources en is voor ons niet haalbaar binnen de gestelde termijn van tien weken. Ook is het niet praktisch om verschillende applicaties tegelijkertijd op bijvoorbeeld een beurs te presenteren als er maar één spreker of onvoldoende hardware beschikbaar is.

4.2 Spelvorm

Serious games zijn games die gemaakt zijn zonder het hoofddoel om mensen te entertainen. Serious games worden onder andere gemaakt voor training, simulatie, educatie en reclame maken. Games kunnen goed worden gebruikt bij het verbeteren van vaardigheden en inzicht, oog-handcoördinatie en het selecteren van belangrijke elementen in het gezichtsveld [11]. Een groot voordeel van simulaties is dat het voor de gebruikers mogelijk is om in een omgeving te komen die niet bestaat in de echte wereld. Dit kan gebruikt worden om een serious game te maken van het concept dat Centric heeft bedacht. Uit onderzoek blijkt ook dat simulatiegames goed helpen bij het begrijpen van de werking van systemen [9]. Ook wordt door veel mensen beweerd dat games helpen bij het begrijpen van complexe situaties [5]. Een game zou dus de potentiële klanten goed kunnen overtuigen van de kracht van dit concept. De vraag is natuurlijk hoe dit concept kan worden omgevormd tot een game.

Enkele ideeën voor een spelvorm van het concept zijn de volgende:

- De gebruiker krijgt een schip te besturen en moet een vaarroute bevaren. Tijdens deze route moeten een aantal containers geladen en gelost worden. Ook vaart de gebruiker langs een aantal PH's (PH's). Tijdens deze run worden de informatiestromen van het concept duidelijk. Ook zouden er calamiteiten in de game kunnen gebeuren. Er kan bijvoorbeeld een ander schip

tegen het schip van de gebruiker aanvaren. Zodoende krijgt de gebruiker ook te zien welke informatiestromen er plaatsvinden bij een calamiteit en hoe het speuren naar de inhoud van een container in het nieuwe concept te werk gaat.

- De gebruiker krijgt de controle over alle schepen in een havengebied en moet er voor zorgen dat alle schepen naar hun los- en laadplaatsen worden geloodst. Tijdens dit proces worden de informatiestromen van het concept duidelijk. Een calamiteit kan optreden als een schip opeens niet luistert naar een order van de gebruiker en tegen een ander schip aan vaart. Hierna wordt de calamiteitenprocedure gestart en kan de gebruiker zien hoe de inhoud van de verloren containers wordt bepaald.
- De gebruiker ziet de simulatie een paar minuten draaien voordat er een calamiteit gebeurd. Hierdoor ziet de gebruiker de informatiestromen tussen de verschillende hubs. De gebruiker moet, nadat er een calamiteit heeft plaatsgevonden, de stappen uitvoeren die normaal bij de meldkamer worden gedaan. Hierdoor ziet de gebruiker hoe het proces in zijn werk gaat om de inhoud van een verloren container te achterhalen.

Zo zijn er een aantal spelvormen te bedenken die de kracht van het concept weergeven. Het spel zou kunnen dienen ter ondersteuning van een presentatie over het concept. Echter moet het spel niet speelbaar zijn tijdens de presentatie, anders kunnen gebruikers afgeleid worden van de presentatie. Wel is het goed om de gebruikers het spel mee te geven zodat die thuis het spel kan spelen om zo het concept nog eens in werking te zien. Een spelvorm kan dus bijdragen aan het beter begrijpen van het concept.

Om een spelvorm te implementeren moeten naast het concept ook enkele spelelementen worden geïmplementeerd en een handleiding voor het spel worden geschreven. Vanwege het korte tijdsbestek waarin wij iets moeten opleveren lijkt ons een spelvorm niet haalbaar binnen de beschikbare tijd.

4.3 Huidige situatie / nieuwe situatie

In de huidige situatie wordt het opvragen van bijvoorbeeld de inhoud van de containers van een gezonken schip bewerkstelligd door de eigenaar van de containers vast te stellen en deze eigenaren te benaderen om hen te vragen wat er in hun containers zit. In de nieuwe situatie wordt dit bewerkstelligd door de informatie uit de juiste PH's op te vragen. Een mogelijkheid is om beide situaties op te nemen in de simulatie.

Dit zou bijvoorbeeld geïmplementeerd kunnen worden door het computerscherm in tweeën te splitsen: een helft voor de huidige situatie en een helft voor de nieuwe situatie. Beide helften moeten dan eenzelfde scenario gelijktijdig simuleren. Op beide helften kan dan getoond worden waar de informatie wordt opgeslagen en hoe de informatiestromen lopen. Zo zou je op de ene helft kunnen zien dat informatie in MH's en PH's wordt opgeslagen, terwijl op de andere helft te zien is dat verschillende partijen over verschillende informatie beschikken. Een alternatieve implementatie is door de twee runs achtereenvolgens uit te voeren en het scherm niet in tweeën te splitsen.

Voordelen van het weergeven van beide situaties zijn dat er gemakkelijker een vergelijking tussen de twee situaties gemaakt kan worden zodat hierdoor de voordelen van het nieuwe systeem beter begrepen worden.

Nadelen zijn dat de implementatie veel meer tijd in beslag neemt en het daardoor niet haalbaar is binnen ons tijdsbestek. Daarnaast is het lastig om de huidige situatie te simuleren gezien die erg complex is en er gebruik wordt gemaakt van vele verschillende communicatiemiddelen. Bovendien maakt deze oplossing het mogelijkerwijs lastig om het idee over te brengen, gezien er veel meer informatie op het scherm komt waardoor het voor gebruikers meer tijd kost om het te begrijpen.

4.4 Film of powerpoint

Om een nieuw systeem te presenteren aan potentiële klanten kan er natuurlijk gebruik gemaakt worden van traditionele presentatietechnieken zoals een film of een powerpointpresentatie. Op zich zijn dit goede technieken om een product te promoten. Het smart container concept is echter een hele andere denkwijze dan hedendaagse toegepaste technieken. Een presentatie is te statisch om potentiële klanten van dit concept te overtuigen. Ook is het niet erg interessant voor potentiële klanten om op een beurs naar een presentatie te luisteren. Een film zou hier dus beter op zijn plaats zijn. Echter zou hierin telkens hetzelfde scenario te zien zijn, terwijl de omgeving waarin het concept van toepassing is heel dynamisch is en er elke keer andere situaties plaatsvinden. Om dit toonbaar te maken is het beter om een interactieve demo te ontwikkelen. Bij deze demo kan door de presentator uitleg worden gegeven over wat er precies gebeurt. Een demo is uitermate geschikt om het dynamische karakter van de omgeving waarin het concept van toepassing is weer te geven.

5 Verbeteringen en toevoegingen simulatie concept

Dit hoofdstuk beschrijft de door de opdrachtgever voorgestelde oplossing omtrent het presenteren van calamiteiten en de mogelijke verbeteringen hiervan en mogelijke toevoegingen hieraan.

5.1 Target a container

Het hoofddoel van de voorgestelde simulatie is het kunnen nabootsen van calamiteiten om zo de werking van het concept *Target a Container* te kunnen demonstreren. Allereerst moet de gebruiker een soort calamiteit (hoog, middel of laag) kunnen kiezen. Vervolgens moet er een procedure op gang worden gebracht die laat zien hoe de nabijgelegen schepen worden geselecteerd met behulp van de drie PH's die het dichts bij de calamiteit zijn. Vervolgens moet getoond worden hoe het schip met de calamiteit gevonden wordt en welke goederen deze vervoert.

5.1.1 Starten van de procedure

Het triggeren van een calamiteit kan op een tweetal manieren gebeuren. Een eerste optie is om het triggeren te programmeren. Hierbij wordt de simulatie zó geprogrammeerd dat er om de zoveel tijd een calamiteit plaatsvindt. Een voordeel is dat dit eenvoudig is te programmeren en dat bovendien de gebruiker of de spreker niet nodig zijn om een calamiteit weer te geven. Een nadeel is dat er niet direct

een calamiteit kan worden weergegeven als de gebruiker of spreker dat wenst. Ook zal er elke keer dezelfde calamiteit op hetzelfde tijdstip plaatsvinden, waardoor niet duidelijk wordt dat het concept voor elke calamiteit op elk tijdstip werkt.

Er kan ook gekozen worden om het triggeren te laten gebeuren wanneer er op een knop gedrukt wordt. Bij deze variant staat van te voren niet vast waar en op welk tijdstip een calamiteit plaatsvindt. Bovendien kan de gebruiker of spreker op elk gewenst moment een calamiteit triggeren. Een nadeel is echter dat interactie van de gebruiker of spreker noodzakelijk is om een calamiteit te weergeven.

Als het starten van de procedure handmatig gebeurt (via een knop), dan zijn er weer twee mogelijkheden voor het aanwijzen van het schip dat bij de calamiteit betrokken is. Er kan gekozen worden het schip handmatig te kiezen of willekeurig door de computer te laten bepalen. Het willekeurig kiezen van een schip is eenvoudiger, maar degene die het concept presenteert heeft misschien voorkeur voor een bepaald schip. Zo zou men bijvoorbeeld meer geïnteresseerd kunnen zijn in een schip dat buiten het bereik van een PH valt, dan een schip dat binnen het bereik van een PH valt. Het systeem zal meer dynamisch zijn als het schip handmatig gekozen kan worden.

Voorgesteld is de simulatie te stoppen op het moment dat zich een calamiteit voordoet. Het voordeel hiervan is dat men niet afgeleid wordt door bewegende elementen op het scherm en dat de gegevens die opgeslagen zijn in de PH's niet meer veranderen. Het nadeel is dat dit niet helemaal realistisch is, gezien in de werkelijkheid ook gewoon alles door gaat.

5.1.2 Visualiseren calamiteit

Nadat het schip met de calamiteit bepaald is, moet dit gevisualiseerd worden. Het schip zou tot stilstand kunnen komen en bij elke soort calamiteit een andere animatie kunnen tonen (denk aan een rookwolkje, vuur of het zinken van het schip). Het op de kant varen of het botsen van schepen zou meer realistisch zijn, maar het probleem is dat dit niet gevisualiseerd kan worden wanneer de procedure op een door de gebruiker bepaald moment gestart kan worden. Wel zou er een vertraging kunnen zitten tussen het opstarten van de procedure en het tonen van de calamiteit, zodat er tijd is om een botsing te simuleren.

Vervolgens moet een melding van de calamiteit gemaakt worden. Gedacht kan worden aan het tonen van een informatiestroom tussen degene die de melding maakt (het schip zelf, een ander schip, iemand langs de kant) en de meldkamer. Om het realistisch te maken zou een melding door een ander schip alleen mogelijk moeten zijn als dit schip zich dicht genoeg bij het schip met de calamiteit bevindt. Zo zou ook het melden door iemand langs de kant realistischer zijn als dit alleen gebeurt wanneer de calamiteit in werkelijkheid zichtbaar zou zijn (denk aan het zinken van een schip of lading; soort calamiteit: hoog/middel).

5.1.3 Bepalen van het schip

Nadat een melding van een calamiteit gemaakt is, moet op het scherm getoond kunnen worden hoe precies bepaald wordt welk schip de calamiteit ondergaat. Hiervoor moet het volgende gevisualiseerd worden (dit hoeft alleen als de bemanning van het schip zelf geen melding gemaakt heeft):

- Het bepalen van de drie dichtstbijzijnde PH's.
- Het weergeven van de opgeslagen gegevens in deze drie PH's op een bepaald moment vóór de calamiteit en op een moment na de melding.
- Het weergeven welke schepen betrokken kunnen zijn en hoe contact met de bemanning opgenomen wordt.
- Het bepalen van het schip met de calamiteit (het schip dat niet op de oproep van de meldkamer reageert).
- Het tonen van de inhoud van de containers van dit schip.

Gedacht kan worden om dit te visualiseren door informatiestromen weer te geven met behulp van lijntjes, een tabel weer te geven met de gegevens van de PH's en bepaalde rijen te kleuren, schepen die betrokken zijn te kleuren, enzovoorts.

5.1.4 Eind van de procedure

Als de procedure afgelopen is en de lading van het schip getoond is, kunnen er twee dingen gebeuren: de simulatie kan opnieuw gestart worden of de simulatie kan voortgezet worden. In het eerste geval wordt het beginscherm met instellingen voor de simulatie getoond en wordt een nieuwe run gestart. Bij het tweede geval varen alle schepen gewoon weer verder en wordt de simulatierun voortgezet. Dit laatste is meer realistisch en is misschien meer gebruikersvriendelijk, gezien niet elke keer weer door het instellingenmenu gegaan hoeft te worden. Ook hoeft er niet gewacht te worden totdat er voldoende informatie in PH's is opgeslagen om de procedure van een calamiteit te starten. Aan de andere kant kan het juist wenselijk zijn dat begonnen wordt met een schone lei waarbij de PH's nog geen informatie opgeslagen hebben. Op deze manier wordt de gebruiker niet overspoeld met informatie die reeds in de PH's opgeslagen is.

5.2 Vragenstellende actoren

Een mogelijke aanvulling op het huidige concept voor de simulatie is de actoren meer betrekken bij de simulatie door hen een belangrijke rol te geven in de simulatie. Elke actor heeft bepaalde belangen en wil daarom een aantal dingen weten. Ons idee is om in de simulatie de verschillende actoren te laten zien en dat die actoren aangeklikt kunnen worden. Na het aanklikken van een actor zullen dan enkele vragen worden weergegeven die voor deze actor interessant zijn. Deze vragen kunnen aangeklikt worden en zo nodig kunnen meerdere vragen tegelijkertijd aangeklikt worden. De objecten en informatiestromen die betrekking hebben op de aangeklikte vragen zullen duidelijk zichtbaar worden gemaakt. Dit kan bijvoorbeeld door de objecten en informatiestromen die bij een aangeklikte vraag horen allemaal dezelfde kleur te geven en dan per vraag een andere kleur te gebruiken.

Met dit idee kunnen verschillende actoren tegelijkertijd met de simulatie aan de gang gaan. Op die manier wordt de simulatie erg interactief en blijven de gebruikers bezig, ook terwijl de spreker tegen andere personen praat. Ook kunnen zij voor hen relevante vragen stellen en de simulatie zal duidelijk visualiseren hoe de informatiestromen verlopen. Zo kan het concept duidelijk worden gemaakt. Elke actor kan dan ook de vragen van andere actoren zien zodat ook meteen duidelijk is wat andere actoren willen weten en hoe dit in het concept naar voren komt.

Een nadeel van dit idee is dat deze actoren ruimte op het scherm in beslag nemen. Een ander nadeel is dat de vragen in tekstvorm op het scherm komen en daardoor dus niet door mensen aan de andere kant van de tafel tegelijkertijd gelezen kunnen worden. Ook zal er slechts een van te voren vastgestelde verzameling van vragen te zien zijn bij de simulatie en kunnen andere vragen van de actoren niet op eenzelfde manier worden getoond.

5.3 Mogelijkheid tot pauzeren

De demo wordt onder andere gepresenteerd op een beurs. Daar is het vanzelfsprekend dat mensen vragen hebben over wat er allemaal gebeurt in de demo. Voordat de demo verder gaat met het laten zien van de kracht van het concept kan het van belang zijn dat de vragen van de geïnteresseerden eerst worden beantwoord. Ook kan het van belang zijn om op bepaalde momenten te bekijken welke informatie nu waar is opgeslagen. Wij denken daarom dat het van belang is dat er een pauzemodus in de demo gemaakt wordt. Dit kan bijvoorbeeld op de volgende manieren worden ingevoerd:

- Er bevindt zich ergens op het scherm een pauzeknop. Deze knop zorgt er voor dat de demo op elk willekeurig moment gepauzeerd kan worden. Deze knop kan ook gebruikt worden om de demo weer verder te laten gaan.
- Tijdens het bekijken van de inhoud van een hub wordt de demo gepauzeerd. Dit zorgt ervoor dat de inhoud goed kan worden bekeken zonder dat er allerlei informatie opeens verdwijnt of bij komt. Het sluiten van het scherm met de informatie zorgt ervoor dat de demo weer verder gaat.

5.4 Collision detection

Om de simulatie toch wat meer op de werkelijkheid te laten lijken, lijkt het ons wel van belang om enigszins collision detection in de demo te verwerken. Mensen zullen toch wat raar op kijken als ze opeens twee schepen door elkaar heen zien varen. Het door elkaar laten varen van schepen kan ook de geloofwaardigheid van het systeem nadelig beïnvloeden. Daarnaast kan het ook verwarring veroorzaken als er een calamiteit gebeurt en er opeens wel twee schepen op elkaar kunnen varen. De collision detection in onze simulatie zal wel erg basic blijven, omdat we niet veel tijd beschikbaar hebben voor een uitgebreide collision detection.

5.5 Geluid toevoegen

Tijdens de simulatie kan op willekeurige momenten ineens informatie uitgewisseld worden, bijvoorbeeld wanneer een MH in het bereik van een PH komt. Een dergelijk event kan mogelijkerwijs niet opgemerkt worden door de gebruiker. Hierdoor lijkt de simulatie niet transparant en is het voor de gebruiker lastig om het concept te begrijpen. Wanneer er een geluidje afgespeeld wordt wanneer een event optreedt, dan heeft de gebruiker sneller in de gaten dat er iets gebeurt en kan dan aan de gevisualiseerde informatiestromen zien wat er precies gebeurd is. Zo gaat een event niet aan de gebruiker voorbij.

Gezien het uiteindelijke doel is deze simulatie op een beurs te tonen is dit misschien toch niet zo'n goed idee. Op een beurs wordt veel gepraat en is het rumoerig. Hierdoor kunnen geïnteresseerden zich alleen

maar gaan ergeren aan het systeem omdat ze het geluid nauwelijks horen. Het toevoegen van geluid is dus misschien niet helemaal nodig.

5.6 Koppeling tussen permanente en mobiele hubs visualiseren

Elke MH (een schip in onze situatie), is verbonden met één of geen PH. De visualisatie van het concept kan verbeterd worden door deze koppeling expliciet te visualiseren. Dit kan bijvoorbeeld gedaan worden door een lijntje zichtbaar te maken tussen een MH en zijn PH. Dit kan echter een beetje misleidend zijn aangezien MH en PH geen constante verbinding hebben. Het zou beter zijn om de verbinding weer te geven wanneer daarom gevraagd wordt door een gebruiker. Zo zou een MH en bijbehorende PH op kunnen lichten op het moment dat de MH aangeklikt wordt.

5.7 Vinden van mobiele hubs binnen bereik visualiseren

PH's sturen na een bepaald tijdsinterval telkens een signaal uit om alle MH's binnen bereik te kunnen vinden. In onze demo zou dit gevisualiseerd kunnen worden door een rondje om de PH weer te geven die vanaf het moment dat de hub een signaal uitzendt. Dit rondje kan dan groter worden en vervagen en uiteindelijk verdwijnen wanneer de rand op een bepaalde afstand van de PH komt te liggen. Dit simuleert dus dat het signaal op grote afstand te zwak wordt om nog door een MH ontvangen te kunnen worden. Wanneer een MH binnen bereik is, zou deze opgelicht kunnen worden op het moment dat de lijn over de MH gaat (bijvoorbeeld door middel van een glowing effect).

5.8 Schip kunnen besturen

Om de demo nog interactiever te maken kan gedacht worden aan de gebruiker in staat te stellen een schip, of meerdere schepen te kunnen besturen. Een gebruiker zou bijvoorbeeld met zijn vinger een schip over het scherm kunnen slepen, waarbij het schip dus met de vinger meebeweegt. Een andere manier is om een willekeurig punt op de kaart aan te klikken zodat het schip daar vervolgens met zijn eigen snelheid heen vaart. Een derde mogelijkheid is om de gebruiker in staat te stellen een pad te trekken met zijn vinger. Het schip loopt dan op eigen snelheid dit pad af. Het besturen van een schip is een would have en zal zeer waarschijnlijk niet geïmplementeerd worden gezien dit niet direct bijdraagt aan het doel van het project. Ook is dit waarschijnlijk niet haalbaar is binnen ons tijdsbestek.

5.9 Opslaan en afspelen van scenario's

In de simulatie kan het wenselijk zijn om bepaalde handelingen automatisch te laten uitvoeren of om bijzondere situaties te herhalen. Dit kan mogelijk worden gemaakt door het gebruik maken van scenariobestanden. In deze bestanden kunnen bepaalde handelingen worden opgeslagen. Deze bestanden kunnen vervolgens door de simulatie worden geopend zodat de handelingen automatisch worden uitgevoerd.

Een groot voordeel hiervan is dat je altijd dezelfde simulatie kunt laten zien en zo een goede presentatie over het concept kan geven bij deze simulatie. Daarnaast hoeft de presentator niet zelf steeds de simulatie te besturen.

De nadelen hiervan liggen vooral bij de ontwikkelaars. Zij moeten komen met een systeem dat de scenariobestanden kan aanmaken en vervolgens weer kan inlezen en uitvoeren. Daarnaast moet dit systeem robuust zijn. Iemand zou iets kunnen aanpassen in een scenariobestand waardoor de simulatie niet meer zou kunnen werken. Vanwege de beperkte tijd die wij tot onze beschikking hebben, hebben we besloten om niet te gaan werken met scenariobestanden.

6 Programmeeromgeving

De programmeeromgeving die bij dit project gebruikt gaat worden is heel anders dan dat wij gewend zijn. Daarom hebben wij informatie opgezocht over de te gebruiken technieken en hieronder een overzicht daarvan gegeven.

6.1 Microsoft .NET Framework

Bij het schrijven van deze sectie is gebruik gemaakt van bron [7]. Het Microsoft .NET Framework is een software framework dat geïnstalleerd kan worden op computers met een Microsoft Windows besturingssysteem. Op 12 april dit jaar is .NET 4.0 uitgegeven. Voor ons project gebruiken wij .NET 3.5 in combinatie met Visual Studio 2008. Dit doen wij omdat deze versie in de praktijk betrouwbaar is gebleken. Daarnaast heeft de nieuwe versie nauwelijks toevoegingen die we bij ons project kunnen gebruiken. Bovendien is de nieuwe versie zo nieuw dat het nog maar op weinig systemen gebruikt wordt. Hieronder worden enkele belangrijke eigenschappen van het .NET Framework toegelicht.

- Interoperabiliteit: middelen om functionaliteit van programma's buiten de .NET omgeving te benaderen.
- Common Language Runtime (CLR): virtuele machine zodat alle .NET programma's draaien onder de supervisie van de CLR. Op die manier worden bepaalde eigenschappen gegarandeerd en wordt geheugen beheerd, veiligheid gewaarborgd en vindt foutafhandeling plaats.
- Base Class Library (BCL): library van functionaliteit die beschikbaar is voor alle talen binnen het .NET Framework. De functies in de BCL bestaan uit onder meer bestanden lezen en schrijven, grafische weergave, database interactie, XML bestandsmanipulatie, etcetera.
- Veiligheid: het ontwerp is bedoeld om een aantal kwetsbaarheden op te vangen zoals buffer overflows. Bovendien voorziet het .NET in een veiligheidsmodel voor alle applicaties.
- Portabiliteit: het ontwerp van het .NET Framework is theoretisch platformonafhankelijk. Dat betekent dat elk programma geschreven om het framework te gebruiken altijd zou moeten draaien op elk systeem waarvoor het framework geïmplementeerd is.

Het .NET Framework kent enkele verschillen en overeenkomsten ten opzichte van Sun Java. Ze zijn beide gebaseerd op een virtuele machine dat de details van de hardware verbergt. Beide gebruiken ze hun eigen tussencode. De bytecode van .NET wordt altijd gecompileerd vóór executie, terwijl de bytecode van Java van te voren gecompileerd kan worden of in runtime geïnterpreteerd kan worden. Het .NET Framework kan alleen worden geïnstalleerd op computers met een Microsoft Windows besturingssysteem, terwijl Java op een scala aan besturingssystemen kan draaien. Een ander groot

verschil is dat het .NET Framework onder de Microsoft Reference License valt en Java open source is onder de GNU GPL license.

6.2 De programmeertaal C# vergeleken met Java

De programmeertaal C# (spreek uit als C Sharp) is geschreven voor het .NET Framework. Het kent vele overeenkomsten met Java, maar ook enkele verschillen. Omdat wij door onze studie bekend zijn met Java zullen we hieronder de belangrijkste verschillen van C# met Java uitlichten. Bij het schrijven van deze sectie is gebruik gemaakt van bron [12].

- Java is ontworpen om uitgevoerd te worden op een Java platform via de Java Runtime Environment (JRE).
- C# is ontworpen om uitgevoerd te worden op the Common Language Runtime (CLR) dat een core component is van het .NET Framework. JRE is beschikbaar voor Solaris, Windows en Linux, terwijl CLR alleen beschikbaar is voor Windows.
- C# staat het gebruik van pointers toe, maar Java heeft geen pointer data types.
- C# implementeert Object Oriënted (OO) method pointers in de vorm van delegates. Java heeft geen soortgelijke contructie op taalniveau en gebruikt een OO ontwerp, de Observer.
- In C# zijn methoden standaard niet-virtueel, maar kunnen wel als zodanig expliciet worden gedeclareerd. In Java alle niet-statische niet-private methoden virtueel. Virtualiteit garandeert dat de meest recente overschrijving van de methode altijd aangeroepen wordt, maar dit gaat ten koste van de runtime.
- Java bestaat al langere tijd en wordt al jaren door veel mensen gebruikt. C# daarentegen is een relatief jonge programmeertaal en is bij het ontwerpen hiervan is ook gekeken naar andere programmeertalen. De ontwikkeling van Java wordt bekritiseerd als langzaam, terwijl C# zich erg snel heeft ontwikkeld.

Er zijn nog vele andere kleine verschillen en verschillen in semantiek op te noemen, maar dat valt buiten het bereik van dit onderzoeksverslag.

6.3 Windows Presentation Foundation

Windows Presentation Foundation (WPF) is het grafische systeem om de interface te tekenen van het .NET Framework. Het werd voor het eerst meegeleverd met .NET Framework 3.0. WPF maakt gebruik van DirectX zodat hardware-acceleratie wordt toegepast en er kan gebruik worden gemaakt van moderne features zoals transparantie in de interface. WPF maakt gebruik van de programmeertaal Extensible Application Markup Language (XAML). Deze op XML gebaseerde taal volgt het succes van andere opmaaktalen. Het wordt gebruikt om grafische interfaces te beschrijven, vergelijkbaar met ActionScript voor Flash. WPF biedt naast 2D en 3D graphics ook de mogelijkheid voor animaties, rotaties en een aantal grafische effecten [5][9].

Er bestaan heel wat programma's om interfaces in WPF mee te maken. Deze bieden vaak de mogelijkheid om de interface naast de code erachter te zien, zodat beide aangepast kunnen worden en het resultaat gelijk te zien is. Een voorbeeld hiervan is Microsoft Expression Blend.

6.4 Microsoft Expression Blend

Microsoft Expression Blend is een programma waarmee user interfaces in WPF kunnen worden gemaakt. Het is programma is een zogenaamde 'what you see is what you get' editor. Hierdoor kan je doormiddel van het toevoegen van controls en deze op hun plek te slepen een user interface maken. Code achter de controls moeten daarentegen natuurlijk wel zelf geprogrammeerd worden. Daarnaast biedt Expression Blend uitgebreide tools om je design volledig te naar je hand te zetten door middel van het toevoegen van kleuren en plaatjes. Ook biedt Expression Blend de mogelijkheid tot het creëren van zogenaamde storyboards. Een storyboard is een tijdlijn waarin acties gedefinieerd kunnen worden. Acties kunnen van alles zijn, bijvoorbeeld het laten veranderen van kleur van een knop of het laten bewegen van een invoerveld. Deze storyboards kunnen dan door een event, bijvoorbeeld het klikken van een knop, geactiveerd worden.

Expression Blend creëert de XAML code voor de user interface die net gemaakt is. Deze code kan worden geïmporteerd in Visual Studio om zo functionaliteit achter de user interface te maken.

6.5 Programmeren voor de Microsoft Surface Tafel

De demo die ontwikkeld wordt, moet geprogrammeerd worden voor de Microsoft Surface tafel. Deze tafel zal de demo draaien ter ondersteuning van de presentatie. Een Surface tafel biedt een hoop mogelijkheden voor het geven van een presentatie. Een belangrijk aspect om rekening mee te houden is dat het scherm van alle kanten leesbaar moet zijn. Potentiële klanten kunnen immers rondom de tafel gaan staan en voor hun moet het scherm goed te zien zijn. Tekst of een plaatje moet dus niet één bepaalde richting hebben zodat het maar vanaf één kant te zien is. Deze '360 degrees' denkwijze is iets heel nieuws voor traditionele software ontwikkelaars [4]. Met deze denkwijze moet daarom rekening worden gehouden tijdens de ontwerpfase.

De interface van de Surface Tafel wordt gezien als een Natural User Interface (NUI). Dit houdt in dat er na een leerperiode alles van zelf gaat door middel van natuurlijk bewegingen. Bij conventionele user interfaces blijft alles toch kunstmatig. Een NUI wordt ook wel omschreven als een interface die te bedienen is zonder toetsenbord en muis.

Voor het programmeren van de Surface Tafel kan gewoon WPF in combinatie met C# worden gebruikt. Voor user interfaces kan gebruik gemaakt worden van speciale Surface controls, zoals knoppen, scrolbalken en invoervelden die zich in de Surface SDK bevinden. De controls reageren automatisch op aanrakingen op het scherm. Daarnaast bevatten ze leuke effecten die het programma er mooi uit laten zien. Een speciale control, de ScatterView, is speciaal voor multi touch ontwikkeld. De ScatterView kan door middel van de besturing door twee vingers gedraaid, vergoot en verkleind worden.

6.6 Microsoft SQL Server

Microsoft SQL server is een relationeel databasebeheersysteem dat een dialect van SQL (de meest gebruikte databasetaal) genaamd Transact-SQL (T-SQL) ondersteunt. T-SQL is een uitbreiding van SQL92 die voornamelijk uit extra mogelijkheden bestaat voor het gebruik van opgeslagen procedures.

Microsoft SQL Server gebruikt het “Tabular Data Stream”-protocol (TDS) om op applicatieniveau te communiceren over netwerken.

Voor onze simulatie is beveiliging, betrouwbaarheid en snelheid nog niet zo van belang gezien het in eerste instantie alleen door onszelf en de opdrachtgever gebruikt gaat worden en er maar een beperkte hoeveelheid data opgeslagen gaat worden. Wil men het systeem in de toekomst uitbreiden en misschien het algoritme erachter werkelijk gaan gebruiken dan zijn beveiliging, betrouwbaarheid en snelheid van uiterst belang. Gelukkig biedt Microsoft SQL Server voldoende mogelijkheden om dit te garanderen [8]:

- Het biedt sterke authenticatie en toegangscontrole en krachtige encryptie om de informatie zo goed mogelijk te beveiligen.
- Het beschikt over hulpmiddelen en functies die nodig zijn voor de performance en schaalbaarheid van grote databases.
- Met behulp van Always On technologieën wordt de downtime geminimaliseerd.
- Het biedt de mogelijkheid geografische gegevens op te slaan (denk aan de locaties van MH’s en PH’s).

7 Discussie

In dit hoofdstuk komt een overzicht van de gemaakte keuzes aan bod. Daarna wordt er gereflecteerd op de opdracht en het proces wat betreft de onderzoeksfase.

7.1 Gemaakte keuzes

Het maken van een simulatie lijkt ons beste optie om binnen de beschikbare tijd op een zo goed mogelijke manier het smart container concept te visualiseren. Waarom we precies tot die keuze gekomen zijn, is te lezen in hoofdstuk 3 waar de voor- en nadelen van de verschillende alternatieven besproken zijn. Als we tijd genoeg hebben kunnen we altijd nog met het idee 4.3 Huidige situatie / nieuwe situatie.

In hoofdstuk 5 zijn een aantal verbeteringen en aanvullingen op het simulatie concept van Centric besproken en hier wordt vermeld welke van deze punten wij zullen meenemen in onze simulatie [2]. Target a container is de basis van onze simulatie en die zullen wij zeker in de simulatie opnemen. Vragenstellende actoren is een erg leuke toevoeging, maar waarschijnlijk niet haalbaar binnen de beschikbare tijd. Wij zullen dit dus niet aan onze simulatie toevoegen. Mogelijkheid tot pauzeren vinden wij zeer wenselijk en zal bovendien niet veel extra complexiteit met zich meebrengen. Daarom zullen wij deze optie wel aan onze simulatie toevoegen. Ook zullen wij een simpele methode toevoegen om te voorkomen dat schepen door elkaar heen kunnen varen. Een zeer uitgebreide collision detection is echter niet haalbaar binnen de beschikbare tijd. Het toevoegen van geluid bij de simulatie lijkt ons geen grote toevoeging, omdat het op de beurs waar onze simulatie getoond zal worden rumoerig kan zijn. Bovendien draagt geluid niet veel bij aan het visualiseren van het smart container concept. Geluid zullen wij dus ook niet opnemen in onze simulatie. De koppeling tussen PH’s en MH’s en het vinden van MH’s

binnen bereik van een PH visualiseren draagt wel veel bij aan het uitleggen van het smart container concept en zal niet heel veel tijd kosten. Deze twee opties zullen wij wel verwerken in de simulatie. Het besturen van een schip met behulp van een vinger verhoogt de interactie met de gebruiker, maar draagt niet bij aan het bereiken van het doel van de simulatie. Bovendien kost deze optie flink wat tijd en zal waarschijnlijk niet haalbaar zijn binnen de beschikbare tijd. Deze optie zullen we dus niet in onze simulatie verwerken.

7.2 Reflectie op de opdracht en het proces

Wij zijn erg tevreden over onze plek bij Centric. Zij hebben ons vriendelijk ontvangen en we voelen ons erg thuis bij Centric. Ook over de opdracht zijn wij enthousiast omdat we met iets nieuws bezig zijn dat later echt gebruikt kan worden. Daarnaast vinden wij het een leuke manier om met de Surface tafel kennis te maken en op die manier een hele nieuwe manier van user interface ontwerpen te leren kennen.

Tijdens de onderzoeksfase was ons grootste probleem dat voor ons niet duidelijk was wat er precies van ons verwacht werd. Daarnaast hadden we wat problemen met het TU Weblogsysteem, maar we hadden al snel een goed alternatief gevonden. Wat we erg leuk vonden was het onderzoek naar de programmeeromgeving en dan met name naar WPF en de Surface tafel omdat dit geheel nieuw voor ons was. Het onderzoeken van alternatieven voor het simulatie concept kwam wat moeizaam opgang, omdat al vrij duidelijk was wat Centric van ons wil. Toen echter een aantal wilde brainstormsessies opgang kwamen, werd het toch wel interessant. Vooral bij het nadenken over de aanvulling ‘vragenstellende actoren’ hebben we veel lol gehad.

Wat we beter hadden kunnen doen is bijvoorbeeld het meerekenen van feestdagen in de planning. Dat we op Koninginnedag vrij waren was uiteraard erg aangenaam, maar we hadden hiermee geen rekening gehouden tijdens het opstellen van de planning. Een manier om dat op te vangen is om thuis ook het een en ander te doen en dit zullen we met Bevrijdingsdag ook doen. Verder hadden we misschien de tijd wat beter kunnen verdelen tussen brainstormen en literatuuronderzoek. Hoewel brainstormen een goede methode is om op nieuwe ideeën te komen kost het wel veel tijd. Voor de onderzoeksfase hadden we slechts drie weken waarin ook ontwerpdocumenten gemaakt moesten worden. Wat we naar ons idee goed hebben gedaan is de communicatie tussen TU Delft, Centric en ons. Dit hebben we bereikt door snel op e-mails te reageren, weblog bij te houden, duidelijke documenten te maken en door vaste dagen in Delft en Gouda aanwezig te zijn zodat we altijd snel terecht kunnen met vragen.

8 Referenties

[1] Banks, J, Carson II, JS, Nelson, BL en Nicol, DM, 2009, *Discrete-event system simulation, fifth edition*, Prentice Hall, New Jersey.

[2] Centric IT Solutions 2010, *Functioneel Ontwerp TaCS*.

[3] Centric IT Solutions 2010, *Mobile Hubs and Smart Containers visie*.

[4] dotNed, geraadpleegd op 26 april 2010, <http://community.dotned.nl/blogs/dennis_blog/archive/2009/06/30/997.aspx>

[5] Douma, AM, Hillegersberg, J van en Schuur, PC (2008). Using a Management Game to Exemplify a Multi-Agent Approach for the Barge Rotation and Quay Scheduling Problem in the Port of Rotterdam. In Proceedings of the HICL 2008 (pp. 227-244). Erich Schmidt Verlag.

[6] Extensible Application Markup Language, geraadpleegd op 26 april 2010, <http://en.wikipedia.org/wiki/Extensible_Application_Markup_Language>

[7] Microsoft .NET, geraadpleegd op 26 april 2010, <http://www.microsoft.com/net/>

[8] Microsoft SQL Server, geraadpleegd op 27 april 2010, <http://www.microsoft.com/sqlserver>

[9] Ryan, T, The role of simulation gaming in policy-making. Systems research and behavorial science, 2000. pp. 359-364.

[10] Sarjoughian, HS en Singh, RK, 2004, *Building Simulation Modeling Environments Using System Theory and Software Architecture Principles*, April, Wash. DC, SCS.

[11] T. Susi, M. Johannesson, and P. Backlund. Serious games - an overview. Technical Report HS-IKI-TR-07-001, School of Humanities and Informatics, University of Skovde, Sweden, February 2007.

[12] The Java Language Environment, geraadpleegd op 3 mei 2010, <http://java.sun.com/docs/white/langenv/Simple.doc2.html>

[13] Windows Presentation Foundation, geraadpleegd op 26 april 2010, <http://en.wikipedia.org/wiki/Windows_Presentation_Foundation>

Bijlage D. Requirements Analysis Document

Requirements Analysis Document

Bachelorproject Rob Post, Oskar van Rest en Tim Roorda

11 mei 2010

Inhoudsopgave

1 Introductie

Informatie over de inhoud en de locatie van zeecontainer wordt momenteel opgeslagen door de eigenaar van de zeecontainer. Bij calamiteiten moest dus eerst aan deze eigenaar worden gevraagd wat de inhoud van de betrokken zeecontainers is. Het hele proces om daar achter te komen kan wel vierentwintig uur in beslag nemen. Dit kan veel te lang zijn als er giftige stoffen bij betrokken zijn en de hulpdiensten hiervan op de hoogte gesteld moeten worden.

Centric heeft een systeem bedacht waarbij de inhoud en de locatie van zeecontainers die betrokken zijn bij een ongeluk binnen enkele minuten kan worden opgevraagd. Bij dit systeem wordt de informatie over de inhoud en de locatie van zeecontainers decentraal opgeslagen. Voor meer informatie over dit concept zie het document 'MH's and Smart Containers visie'. Om belanghebbenden te overtuigen van de werking van het concept, moet het concept gevisualiseerd worden. De simulatie om de werking van het concept aan te tonen moet nog geïmplementeerd worden. Dit document beschrijft waaraan de simulatie moet voldoen, zodat de kracht van het concept naar voren komt. Daarnaast geeft dit document een abstract ontwerp van het te bouwen systeem.

2 Huidige situatie

In de huidige situatie wordt de positie van containerschepen bijgehouden door de eigenaar van het schip. Dit wordt vaak gedaan via een satellietverbinding of via het Automatic Identification System (AIS). Ook is vaak de lading van een schip alleen bekend bij de eigenaar van dat schip. Als er iets met dit schip of de lading gebeurt moet alle communicatie via de eigenaar van het schip verlopen. Deze vertelt dan wie de eigenaren zijn van de containers die aan boord zijn. Via de containereigenaren moet dan weer de inhoud van de containers achterhaald worden. Dit proces kan soms wel vierentwintig uur duren. Als er gevaarlijke stoffen bij het ongeluk betrokken zijn, kan dit veel te lang zijn. Rederijen en containereigenaren gebruiken ook vaak hun eigen systeem om deze gegevens op te slaan. De combinatie van verschillende communicatiesystemen, verschillende opslagsystemen en verschillende betrokkenen zorgen ervoor dat het hele proces van het vervoeren van containers niet transparant is.

3 Het te bouwen system

Dit hoofdstuk begint met een korte introductie over het te bouwen system en waarvoor we dit systeem maken. Daarna komen de functionele requirements aan bod die aan het systeem gesteld worden. Vervolgens worden de niet-functionele requirements besproken. In de daarop volgende sectie worden de randvoorwaarden aan het project toegelicht. Dit wordt gevolgd door de indeling van requirements in categorieën volgens de MoSCoW indeling. Dit hoofdstuk sluit af met verschillende modellen van het te bouwen systeem. Hierin komen onder andere scenario's, use cases en use case diagrammen, sequencediagrammen en het ontwerp van de user-interface aan bod.

3.1 Introductie

Er is gevraagd een simulatie te ontwikkelen om de werking van het smart container concept te demonstreren. Daarbij moet nadruk worden gelegd op hoe het concept kan bijdragen aan een vlotte en veilige afhandeling van calamiteiten. In november zal onze simulatie gebruikt worden bij een transportbeurs, zodat men de werking van het concept kan zien. Daar moeten bezoekers van de beurs met de simulatie aan de gang kunnen zonder dat zij verdere uitleg nodig hebben. Er zal wel een spreker bij aanwezig zijn die het concept nog verder toe kan lichten en zelf de simulatie kan bedienen. Bij het bouwen van het systeem heeft het laten zien van de werking van het concept de hoogste prioriteit en is de grafische kant van minder belang. Om de werking van het concept te laten zien zullen er drie soorten calamiteiten optreden. De simulatie zal laten zien wat er bij deze calamiteiten gebeurt en hoe de informatie over vervoersmiddelen en containers uitgewisseld wordt.

3.2 Functionele requirements

Dit hoofdstuk beschrijft de functionele requirements waaraan de simulatie moet voldoen. Dit hoofdstuk komt grotendeels overeen met het document ‘Functioneel Ontwerp v0.0.5’.

3.2.1 De kaart

- Er wordt gebruik gemaakt van een vooraf willekeurig gekozen landkaart waarop een vaarroute zichtbaar is.
- De schaal van de kaart moet realistisch zijn voor het aantal hubs.
- Op de kaart zullen vijf tot zeven MH’s (MH’s) getoond worden.
- Op de kaart zullen drie tot vijf PH’s (PH’s) getoond worden. Deze staan langs de vaarroute.
- Alvorens de kaart weergegeven wordt, is er de mogelijkheid drie MH’s in te stellen (wat betreft schip en lading). De overige MH’s zijn dummy’s (wat betreft schip en lading).

3.2.2 Mobiele Hubs

- Alle MH’s zijn schepen.
- Per MH worden de volgende gegevens bijgehouden:
    - ID
    - Begin positie in graden
    - Naam
    - Vlag
    - Scheepstype
    - Status
    - Begin snelheid in knopen
    - Begin koers in graden
    - Afmetingen in meter
    - Diepgang in meter
    - Laadvermogen in TEU
    - Bestemming
    - Roepnaam

- Een MH beschikt over nul, één of meerdere containers.
- De totale omvang van de containers mag het laadvermogen van de MH niet overschrijden.
- De MH's varen van links naar rechts of van rechts naar links (horizontaal over het scherm).
- De MH's die aan de ene zijde de kaart verlaten komen er op de andere zijde weer bij.
- Door op een MH te klikken worden de gegevens van deze MH zichtbaar en moet het mogelijk zijn de inhoud van de containers te bekijken.

3.2.3 Instelbare Mobiele Hubs

- De MH kan worden gekozen uit een vooraf gedefinieerde lijst waarin per schip het volgende getoond wordt:
  - Naam van het vervoersmiddel
  - Omvang (lengte x breedte)
  - Maximale lading in TEU
  - Maximale lading in kilogram
  - Beginpositie
  - Beginsnelheid
  - Vaarrichting (90^o of 270^o)
- Er bestaat ook een voorgedefinieerde lijst waarin ladingseenheden (containers) met de volgende gegevens getoond worden:
  - Inhoud
  - Omvang
  - Gewicht
  - Verzender
  - Ontvanger
  - Vervoerder
  - Verlader
- Nadat er een MH is gekozen, kunnen er ladingseenheden gekozen worden. Dit kan zolang de maximale lading niet overschreden wordt.

3.2.4 Permanente Hubs

- Per PH worden de volgende gegevens bijgehouden:
  - ID
  - Positie in graden
- Door op een PH te klikken moeten de opgeslagen gegevens zichtbaar gemaakt worden.
- PH's hebben een bereik van ongeveer vijf kilometer en hun bereik hoeft niet te overlappen met het bereik van andere PH's.
- Elke vijf seconden scannen de PH's op MH's binnen bereik.
  - Dit scannen moet gevisualiseerd worden door te laten zien wanneer er gescand wordt, wat de reikwijdte van het signaal is en welke MH's tijdens het scannen gevonden worden.
  - Als de MH voor het eerst binnen bereik van de PH komt gebeurt het volgende:

        - Er wordt een melding gemaakt aan alle nabijgelegen PH's dat het schip nu met binnen het bereik van de PH is.
        - Alle gegevens over de containers van de MH worden opgeslagen in de PH.
        - Er wordt een SchipStatus (zie hieronder) aangemaakt en in de PH opgeslagen.
    - Als de MH reeds in bereik was wordt alleen een SchipStatus opgeslagen.
    - De informatiestroom van MH naar PH, bij het opslaan van een SchipStatus en de gegevens over de containers, wordt gevisualiseerd.
- Als door PH x een melding wordt ontvangen van een naburige PH y dat MH q in bereik van PH y is, dan worden alle gegevens over MH q uit PH x verwijderd. Daarnaast wordt in PH x opgeslagen dat MH q verbonden is met PH y.

3.2.5 SchipStatus

- Alle gegevens van de schepen die verandering ondergaan in combinatie met een timestamp noemen we een SchipStatus. Per SchipStatus worden de volgende gegevens bijgehouden:
    - ID
    - Positie in graden
    - Koers in graden
    - Snelheid in knopen
    - Timestamp GMT+1

3.2.6 Containers

- Per container worden de volgende gegevens bijgehouden:
    - BIC-code
    - Status
    - Volume in TEU
    - Gevaarsnummer
    - UN-nummer

3.2.7 Calamiteiten

- Er zijn drie soorten calamiteiten
    - Hoog: er is geen response van de MH en de bemanning (semafoon); daarnaast is de lading verloren;
    - Middel: er is wel response van de MH, maar geen response van de bemanning (semafoon) en (een deel van) de lading is verloren;
    - Laag: er is wel response van MH en de bemanning, maar het schip en/of de lading is beschadigd.
- Gedurende de simulatie moet het op ieder moment mogelijk zijn een calamiteit te triggeren.
- Er moet aangegeven kunnen worden welk schip betrokken is bij de calamiteit.

3.2.8 Target a container

- Op ieder moment kan de gebruiker een calamiteit triggeren door op een MH te klikken en voor "Calamiteit" te kiezen.

- De gebruiker kan uit drie soorten calamiteiten kiezen: hoog, middel en laag.
- Het schip wordt na een aantal seconden stilgezet en er wordt per soort calamiteit een ander soort animatie getoond. Het tijdstip waarop dit gebeurt noemen we t_c.
- Na een aantal seconden wordt automatisch een melding gemaakt door:
    - Het schip zelf indien het om de soort calamiteit "laag" gaat.
    - Een ander schip indien dit andere schip in de buurt van het schip met de calamiteit ligt.
    - Iemand langs de kant in het geval de andere twee gevallen niet van toepassing zijn.
- Deze melding wordt gevisualiseerd door een informatiestroom tussen de melder en meldkamer weer te geven. Het tijdstip waarop de melding plaatsvindt heet t_m.
- Indien het om de calamiteit " laag" gaat heeft de meldkamer dus een melding van het schip zelf ontvangen en weet dus om welk schip het gaat. Indien het om de calamiteit "middel" of "hoog" gaat moet eerst gevisualiseerd worden hoe het schip met de calamiteit bepaald wordt.

Schip met calamiteit bepalen

- Om het schip met de calamiteit op te sporen worden de volgende stappen achtereenvolgens uitgevoerd door de gebruiker per stap op het scherm te laten klikken:
    - De drie dichtstbijzijnde PH's worden opgelicht. Een paar seconden later wordt een informatiestroom tussen de meldkamer en de drie PH's zichtbaar.
    - Er worden twee tabellen weergegeven, een voor t_c en een voor t_m.
        - Elk schip dat op tijdstip t_c met één van de drie PH's verbonden was, wordt weergegeven in de tabel voor t_c. Hetzelfde geldt voor t_m.
        - Elk schip dat op tijdstip t_c niet met een van de drie PH's verbonden was maar wel in één of meerdere van de drie PH's opgeslagen is, wordt ook weergegeven in de tabel voor t_c. Hetzelfde geldt voor t_m.
        - Voor elk schip wordt het volgende in de tabellen opgenomen:
            - Schip ID
            - PH waar het schip aangemeld is
            - Coördinaten en koers indien MH verbonden is met een van de drie PH's. Anders "Left PH – joined PH_x".
    - Er wordt een derde tabel, de "conclusietabel" weergegeven.
        - Deze tabel is een samenvoeging van de andere twee tabellen.
        - Alle records voor schepen uit de tabel van t_m zijn hierin opgenomen.
        - Alle records voor schepen uit de tabel van t_c zijn hierin opgenomen indien het schip niet voorkomt in tabel t_m (we gebruiken de meest recente gegevens).
        - Er is een extra kolom in de tabel genaamd "Status"opgenomen, die voor elk schip op "Onbekend" geïnitialiseerd wordt.
    - Er moet nagegaan worden welk van de schepen uit de conclusietabel het schip met de calamiteit is. Allereerst wordt aan alle schepen die zich als laatst bij een van de drie dichtstbijzijnde PH's hebben aangemeld, om een confirmatie van de

bemanning/systeem gevraagd om na te gaan wat het schip met de calamiteit is. Dit gaat als volgt:

- Eén voor één wordt er voor een schip een informatiestroom tussen meldkamer en schip zichtbaar.
- Indien het schip niet bij de calamiteit betrokken is, wordt de status veranderd in “Bevestigd, bemanning en systeem reageren”.
- Indien het wel om het schip met de calamiteit gaat, wordt de status veranderd in:
  - “Onbekend, bemanning reageert, systeem niet”, indien het om de soort calamiteit ‘middel’ gaat.
  - “Onbekend, bemanning en systeem reageren niet”, indien het om de soort calamiteit ‘hoog’ gaat.

- Vervolgens wordt voor alle schepen die zich **niet** als laatst bij een van de drie dichtstbijzijnde PH’s hebben aangemeld, aan de PH waar ze wel mee verbonden zijn gevraagd of ze de status kunnen confirmeren.
  - Indien het schip bij de PH aangemeld is, wordt de status veranderd in “Bevestigd”.
  - Indien het schip niet bij de PH aangemeld is blijft de status van het schip “Onbekend”.
- Het schip waarvan de status nog steeds “Onbekend” is, is het schip met de calamiteit. Deze wordt opgelicht in de conclusietabel.

Inhoud van de containers tonen

- Nadat bekend is welk schip in de problemen is, moet de inhoud van zijn containers getoond worden. Nadat de gebruiker op het scherm klikt, verschijnt er een informatiestroom tussen meldkamer en de PH waar het schip zich het laatst aangemeld heeft.
- Na klikken wordt een tabel met de inhoud van de containers weergegeven op het scherm.

3.2.9 Overig

- Het systeem moet op ieder moment gepauzeerd en herstart kunnen worden.

3.3 Niet-functionele requirements

Dit hoofdstuk bevat een aantal niet-functionele eisen waaraan de simulatie moet voldoen.

3.3.1 User-interface

De eisen aan de user-interface zijn de volgende:

1. De user-interface moet rustig ogen, geen overbodige eye candy bevatten, maar wel aantrekkelijk zijn om mee te werken.
2. De simulatie zal door veel verschillende mensen worden bekeken, mogelijk ook door mensen met minder zicht. De user-interface moet daarom goed leesbaar zijn door middel van een groot lettertype en goed contrast.

3. De user-interface moet consistent zijn, alle mogelijkheden van het programma moeten op een consistente manier worden gepresenteerd aan de gebruiker.
4. Alle aanwezige knoppen moeten een duidelijke functie hebben.
5. De simulatie moet voor iedereen direct te gebruiken zijn, met of zonder minimale uitleg. De user-interface moet intuïtief werken en overeenkomsten tonen met andere programma's.
6. De user-interface moet met de handen te bedienen zijn en mogelijk door meerdere users tegelijkertijd.
7. De user-interface moet zo min mogelijk tekst bevatten.
8. Elementen van de user-interface moeten draaibaar en schaalbaar zijn, zodat ze van alle kanten goed zichtbaar zijn.

3.3.2 Documentatie

De simulatie wordt alleen gebruikt bij het geven van presentaties. Daarom wordt alleen voor degene die de presentatie gaat houden een korte handleiding geschreven. De simulatie hoeft verder geen helpfunctie te bevatten.

3.3.3 Hardware en software eisen

De simulatie moet draaien op een Microsoft Surface tafel of op een willekeurige andere computer met een besturingssysteem van Microsoft en vergelijkbare of betere prestaties als de Surface tafel. Voor een mooie weergave van de simulatie zal de monitor van de computer, net als de Surface tafel, een resolutie van 1024 x 768 moeten hebben.

De simulatie moet vlot werken. De simulatie moet zonder haperingen laten zien wat er gebeurt en moet direct reageren op input van de gebruiker.

3.3.4 Kwaliteit

De simulatie moet robuust en stabiel zijn. De simulatie moet rekening houden met mogelijke fouten die kunnen plaatsvinden. Als een fout plaatsvindt moet de simulatie een melding geven. Dit is netter dan dat de simulatie crasht.

3.3.5 Systeem veranderingen

De simulatie moet zo ontwikkeld en gebouwd worden dat het in de toekomst eenvoudig uitgebreid kan worden met bijvoorbeeld andere transportentiteiten. Ook moet de simulatie schaalbaar zijn: als er meerdere transportentiteiten worden weergegeven moet de simulatie nog steeds soepel werken.

3.4 Randvoorwaarden

De functionaliteit van het te bouwen systeem zal geschreven worden in C#. Voor de grafische kant van het systeem zullen wij Windows Presentation Foundation (WPF) gebruiken. Om gegevens op te slaan maken wij gebruik van SQL Server 2008. Verder mogen wij er vanuit gaan dat er internettoegang beschikbaar is op de plaats waar onze simulatie zal worden getoond, hoewel dit niet nodig zal zijn voor de essentie van de simulatie. Voor de hele ontwikkeling van ons product (van onderzoek en ontwerp tot oplevering) zijn tien weken beschikbaar. De eindpresentatie van ons product zal plaatsvinden op 1 juli 2010 en kort daarvoor zal het product klaar moeten zijn. We ontwerpen en maken onze simulatie

speciaal voor de Microsoft Surface tafel, maar bij onze presentatie gaan we er vanuit dat we niet de beschikking hebben over een Surface tafel. Wel beschikken we over een Surface Simulator die een Surface tafel simuleert. Deze draait, tezamen met de hele Surface SDK, alleen op Windows Vista 32-bit.

3.5 MoSCoW indeling

Hieronder wordt weergegeven welke eisen in het eindresultaat *moeten* komen (must have), welke eisen zeer gewenst zijn (should have), welke eisen alleen aan bod komen indien er genoeg tijd is (could have) en welke eisen niet geïmplementeerd zullen worden, maar eventueel wel in de toekomst interessant zijn (would like to have).

3.5.1 Must have

- Mogelijkheid de simulatie te starten en schepen (MH's) te zien bewegen op een willekeurig gekozen kaart.
- Enkele PH's tonen en deze gegevens over de MH's op te laten slaan.
- Tijdens de simulatie de inhoud van alle PH's en MH's en de containers kunnen uitlezen.
- Mogelijkheid ieder moment een calamiteit te starten en de procedure van melden, traceren van het schip en het verkrijgen van de inhoud van de containers te visualiseren (*Target a Container*).
- Mogelijkheid het systeem te herstarten.

3.5.2 Should have

- Verschillende calamiteitniveaus (hoog, middel, laag) gebruiken.
- Koppeling tussen PH's en MH's visualiseren.
- Vinden van MH's binnen bereik visualiseren.
- Alle tabellen en schermen draaibaar maken zodat de simulatie op de Surface tafel van verschillende kanten bekeken kan worden.
- Schepen kunnen instellen.

3.5.3 Could have

- Het blokkeren van het toevoegen van een extra container aan een schip wanneer dit de maximale lading overschrijdt.
- Coördinaten van de MH's en de schaal van de kaart overeen laten komen met de werkelijkheid.
- Mogelijkheid te pauzeren.
- Collision detection.
- Het bereik van verschillende PH's laten overlappen.

3.5.4 Would have

- Niet alleen schepen, maar ook andere transportentiteiten toevoegen.
- Het proces van overslag en opslag weergeven.
- Vragenstellende actoren.
- Uitgebreide collision detection.
- Schip kunnen besturen.

- Opslaan en afspelen van scenario's.
- Huidige situatie en nieuwe situatie simuleren.
- Voor elke actor één simulatie.
- Schepen niet alleen van links naar rechts of van rechts naar links laten varen, maar ook in andere vaarrichtingen.
- Realistisch water

3.6 Modellen van het systeem

Om een zo goed mogelijk beeld te krijgen van het te vormen systeem zijn de gebruikelijke modellen opgesteld. Eerst worden mogelijke scenario's gegeven en de use case modellen die hieruit volgen. Vervolgens wordt een klassendiagram gegeven en sequencediagrammen die met behulp van dit klassendiagram gevormd zijn. Er wordt afgesloten met een voorstel voor de grafische user-interface.

3.6.1 Scenario's

Hieronder volgen een aantal scenario's die zijn opgesteld voor de te ontwerpen simulatie.

Instellingen opgeven vooraf aan de simulatierun

1. De gebruiker start het programma.
    a) Het instellingenscherm wordt getoond. Hierin staat een horizontale lijst met vooraf gedefinieerde MH's en daaronder enkele eigenschappen van de MH. Hieronder staat een lijst met de huidige lading van het schip en een lijst met vooraf gedefinieerde containers die toegevoegd kunnen worden.
2. De gebruiker kiest MH01 (Smart Barge).
    a) Het programma geeft een lijst van beschikbare containers en een lijst met de actuele lading (geen) van MH01 en bovendien de eigenschappen van MH01 zoals de naam Smart Barge.
3. De gebruiker kiest de container HOYU7510136.
    a) Het programma heeft de container toegevoegd aan MH01 en deze is verschenen in de lijst met huidige lading van MH01.
4. De gebruiker herhaalt stap 3 voor de containers MSKU6230244, AJCU1079606 en POCU2453671.
5. De gebruiker klikt op de starten knop.
    a) Het programma start de simulatierun met de opgegeven instellingen.

3.6.1.1 Simulatie pauzeren, hervatten en beëindigen

1. De gebruiker heeft het programma gestart en MH01, MH02 en MH03 gekozen en ingesteld. De simulatierun is bezig en de informatie van MH02 wordt bekeken (zie scenario *3.6.1.3 Informatie bekijken tijdens de simulatierun*).
2. De gebruiker klikt op de pauze knop.
    a) De simulatierun wordt gepauzeerd. Dit wil zeggen dat de schepen niet langer bewegen. Openstaande tabellen blijven open staan (die van MH02) en MH's en PH's kunnen nog steeds bekeken worden zoals in scenario 'Informatie bekijken tijdens de simulatierun'.

b) De pauze knop wordt vervangen door een doorgaan knop.

3. De gebruiker klikt als hij verder wil gaan op de doorgaan knop.
   a) De simulatierun wordt voortgezet en opnieuw blijven openstaande tabellen open staan.
   b) De doorgaan knop wordt vervangen door de pauze knop.
4. De gebruiker klikt op beëindigen als hij de simulatie wil beëindigen.
   a) De simulatierun wordt beëindigd en het instellingenscherm wordt weer zichtbaar om een nieuwe simulatierun in te stellen.

3.6.1.2 Informatie bekijken tijdens de simulatierun

1. De gebruiker start het programma, stelt de simulatie in en start de simulatie (zie scenario *3.6.1.1 Instellingen opgeven vooraf aan de simulatierun*).
   a) Het programma laat de simulatie zien.
2. De gebruiker klikt op de meest linkse PH in het scherm.
   a) Het programma geeft een tabel met informatie over het verkeer dat recent langs deze PH is gevaren.
3. Als de gebruiker klaar is klikt deze op de tabel.
   a) Het programma laat de tabel verdwijnen en de simulatie komt weer in beeld.

3.6.1.3 Procedure calamiteit laag weergeven

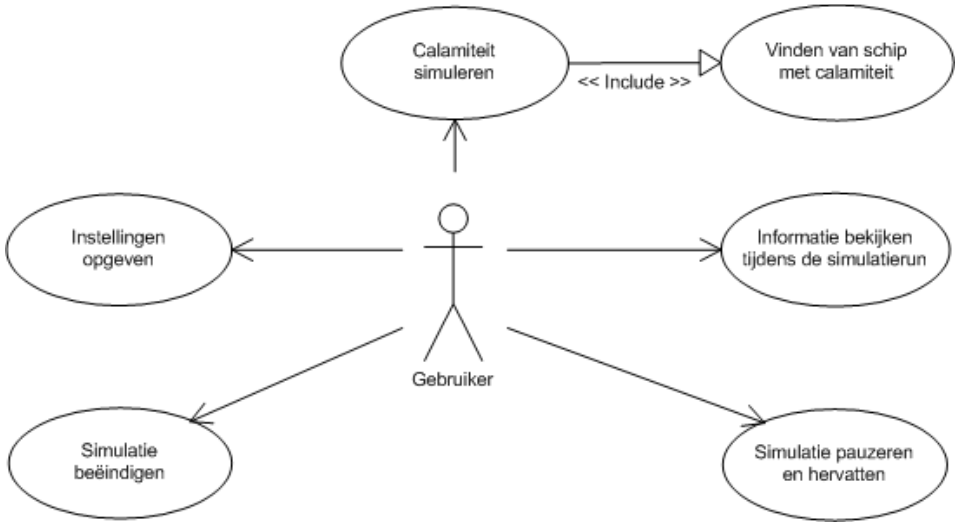
1. De gebruiker start het programma, stelt de simulatie in en start de simulatie (zie scenario *3.6.1.1 Instellingen opgeven vooraf aan de simulatierun*).
   a) Het programma laat de simulatie zien.
2. De gebruiker klikt op het schip met de naam Smart Barge dat aangemeld is bij PH01.
   a) Het programma toont de informatie van het schip Smart Barge en een calamiteitenknop.
3. De gebruiker klikt op de calamiteiten knop.
   a) Het programma laat de drie categorieën voor calamiteiten zien.
4. De gebruiker klikt op laag.
   a) Het programma begint na enkele seconden met het laten zien van de calamiteit en de calamiteitenprocedure.
   b) Het programma wacht een aantal seconden en visualiseert dan dat de eigen bemanning de calamiteit meldt bij de meldkamer.
   c) Het schip licht op.
5. De gebruiker klikt op “Volgende”.
   a) Het programma geeft een informatiestroom weer tussen de meldkamer en PH01 waar het schip het laatst bij aangemeld was.
   b) Het programma wacht enkele seconden en geeft dan een tabel met de inhoud van de containers van het schip weer.
6. De gebruiker klikt als hij klaar is met het bekijken van de tabel op de tabel.
   a) De tabel verdwijnt, de simulatierun wordt beëindigd en het instellingenscherm wordt zichtbaar.

Procedure calamiteit hoog weergeven

1. De gebruiker volgt de eerste drie stappen van het scenario *3.6.1.4 Procedure calamiteit laag weergeven.*
2. De gebruiker klikt op hoog.
    a) Het programma begint na enkele seconden met het laten zien van de calamiteit en de calamiteitenprocedure.
    b) Het programma wacht een aantal seconden en visualiseert dan dat er in de buurt van het schip een melding van een calamiteit wordt gedaan.
    c) De drie dichtstbijzijnde PH's lichten op.
3. De gebruiker klikt op volgende.
    a) Het programma laat de inhoud van de drie PH's zien vlak voor de melding van de calamiteit.
4. De gebruiker klikt op volgende.
    a) Het programma laat de inhoud van de drie PH's zien vlak na de melding.
5. De gebruiker klikt op volgende.
    a) Het programma laat een informatiestroom van de schepen die in het bereik van de drie PH's zijn naar de meldkamer zien.
6. De gebruiker klikt op volgende.
    a) Het programma laat een tabel zien met de schepen in de buurt en met het schip dat in de problemen is.
7. De gebruiker klikt op de knop bekijk van het getroffen schip.
    a) Het programma geeft een tabel met de inhoud van de containers van het schip weer.
8. De gebruiker klikt als hij klaar is met het bekijken van de tabel op de tabel.
    a) De tabel verdwijnt, de simulatierun wordt beëindigd en het instellingenscherm wordt zichtbaar.

3.6.2 Use case diagram

Onderstaand diagram laat de verschillende use cases die de gebruiker kan uitvoeren en de onderlinge samenhang van de use cases zien.



Figuur 1: Use case diagram met de use cases die de gebruiker kan uitvoeren.

3.6.3 Use cases

Hieronder zijn de verschillende use cases uitgewerkt.

3.6.3.1 Instellingen opgeven

Actoren: Iedere gebruiker.
Doel: Het instellen van enkele MH's die in de simulatie aanwezig zijn.
Bijbehorend scenario: *3.6.1.1 Instellingen opgeven vooraf aan de simulatierun*.
Samenvatting: Enkele MH's worden qua lading en schip door de gebruiker ingesteld.
Precondities: Het programma is gestart en het instellingenscherm wordt getoond.
Stappen:

1. De actor klikt op het schip dat hij wil beladen.
   a) Het programma geeft een lijst van beschikbare containers en een lijst met de actuele lading van het geselecteerde schip.
2. De actor sleept een container van de lijst met beschikbare containers naar de lijst met de lading van het schip.
   - Het programma voegt de container toe aan het schip als de container nog op het schip past.
3. De actor kan stap 1 of 2 herhalen.
4. De actor klikt op toevoegen.
   a) Het programma voegt het schip toe aan de simulatie als het aantal toegevoegde schepen kleiner is dan drie en haalt het schip van het instellingenscherm af.
5. De actor klikt op de starten knop.

Postcondities: De simulatie is gestart met onder andere de door de actor ingestelde schepen.

3.6.3.2 Informatie bekijken tijdens de simulatierun

Actoren: Iedere gebruiker.
Doel: Het bekijken van informatie die aanwezig is in een hub.
Bijbehorend scenario: *3.6.1.3 Informatie bekijken tijdens de simulatierun*.
Samenvatting: De informatie in een hub wordt op het scherm weergegeven.
Precondities: De simulatie is gestart of bevindt zich in de pauze modus.
Stappen:

1. De actor klikt op een hub.
   a) Het programma geeft alle informatie die in die hub aanwezig is weer in tabelvorm.
   b) Indien de informatie in de hub verandert, wordt dit gelijk weergegeven in de tabel.
2. De actor klikt op het kruisje van de tabel of op een willekeurige plek in de tabel.
   a) De tabel verdwijnt van het scherm en de simulatie is weer vol in beeld.

Postcondities: Het geopende scherm is weer gesloten en de simulatie gaat verder of blijft in de pauze modus als de simulatie zich daar in bevond.

3.6.3.3 Simulatie pauzeren en hervatten

Actoren: Iedere gebruiker.
Doel: Pauzeren en hervatten van de simulatie.
Bijbehorend scenario: *3.6.1.2 Simulatie pauzeren, hervatten en beëindigen*.
Gerelateerde use case: *3.6.3.2 Informatie bekijken tijdens de simulatierun*.
Samenvatting: De simulatie is in gang gezet en wordt tijdelijk gepauzeerd.
Precondities: De simulatie is gestart en alle instellingen zijn opgegeven.
Stappen:

1. De actor klikt op de pauze knop.
   a) De simulatierun wordt gepauzeerd. Alle MH's en PH's veranderen niet meer van state, maar de door hun opgeslagen data kan wel bekeken worden (zie use case *3.6.3.2 Informatie bekijken tijdens de simulatierun*). Als de calamiteitenprocedure werd uitgevoerd, wordt ook deze gepauzeerd.
2. De actor klikt op de doorgaan knop.
   a) De simulatierun wordt voortgezet.

Postcondities: De simulatierun is voortgezet en alle MH's en PH's kunnen weer van state veranderen.

3.6.3.4 Simulatie beëindigen

Actoren: Iedere gebruiker.
Doel: Beëindigen van de simulatie.
Bijbehorend scenario: *3.6.1.2 Simulatie pauzeren, hervatten en beëindigen*.
Samenvatting: De simulatie is in gang gezet en wordt beëindigd zodat het instellingenscherm weer zichtbaar wordt.
Precondities: De simulatie is gestart en bevindt zich niet in de pauze modus.
Stappen:

1. De actor klikt op de beëindig knop.
   a) De simulatierun wordt beëindigd.
   b) Het instellingenscherm wordt getoond.

Postcondities: De simulatierun is beëindigd en het instellingenscherm wordt weergegeven.

Calamiteit simuleren

Actoren: Iedere gebruiker.
Doel: Het simuleren van de procedure die afgelegd wordt wanneer een calamiteit optreedt om zo het concept *Target a Container* te tonen.
Bijbehorend scenario: *3.6.1.4 Procedure calamiteit laag weergeven* en *3.6.1.5 Procedure calamiteit hoog weergeven*.
Precondities: Alle schepen zijn reeds ingesteld en de simulatie is gestart. Het programma staat niet op pauze.
Samenvatting: Door steeds op "Volgende" te klikken laat het programma de calamiteitenprocedure stap voor stap zien.
Stappen:

1. De actor klikt op een calamiteit.
   a) Het programma vraagt een schip aan te klikken.
2. De actor klikt op een schip.
   a) Het programma wacht een aantal seconden en verandert dan de animatie van het schip in een animatie van een schip met calamiteiten.
   b) Het programma wacht een aantal seconden en visualiseert dan het melden van de calamiteit aan de meldkamer.
   c) Als het schip zelf de melding maakt (calamiteit 'laag'), wordt dit schip opgelicht. Indien dit niet het geval is, wordt met behulp van use case '3.6.3.6 Vinden van schip met calamiteit' het schip bepaald en opgelicht.
3. De actor klikt "Volgende".
   a) Het programma geeft een informatiestroom weer tussen de meldkamer en de PH waar het schip het laatst bij aangemeld was.
   b) Het programma wacht enkele seconden.
   c) Het programma geeft de inhoud van de containers van het schip weer.

Postcondities: De calamiteit is afgehandeld, de simulatie is beëindigd en het instellingenscherm wordt weergegeven.

3.6.3.5 Vinden van schip met calamiteit

Actoren: Iedere gebruiker.

Doel: Visualiseren hoe het schip met de calamiteit gevonden wordt.

Bijbehorend scenario: *3.6.1.4 Procedure calamiteit laag weergeven* en *3.6.1.5 Procedure calamiteit hoog weergeven*.

Samenvatting: Er wordt gevisualiseerd hoe het schip dat bij de calamiteit betrokken is gevonden wordt door handmatig alle schepen in de buurt te contacteren en change-overs te bevestigen.

Precondities: De simulatie is in gang gezet en de calamiteitenprocedure is gestart. De bemanning van het betrokken schip heeft niet zelf de calamiteit gemeld.

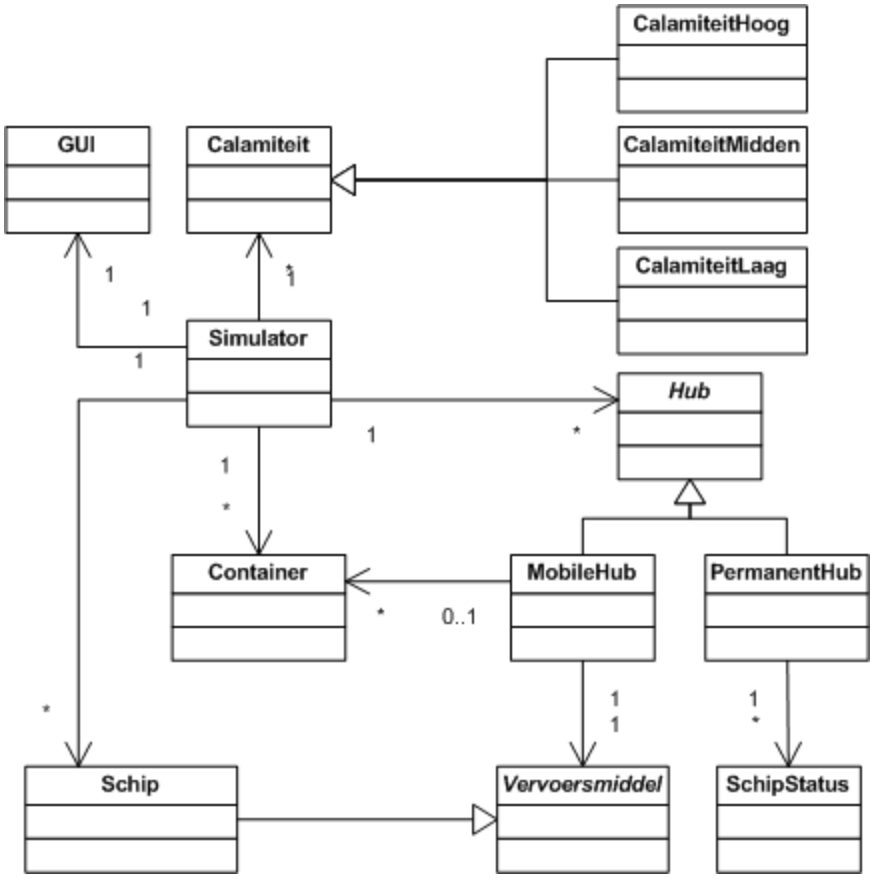
Stappen:

1. De actor klikt “Volgende”.
   a) Het programma laat de drie dichtstbijzijnde PH’s doen oplichten.
   b) Het programma wacht enkele seconden.
   c) Het programma geeft een informatiestroom tussen meldkamer en elk van de drie PH’s weer.
2. De actor klikt “Volgende”.
   a) Het programma geeft de tabellen t_c en t_b weer.
3. De actor klikt “Volgende”.
   a) Het programma geeft de conclusietabel weer.
4. De actor klikt “Volgende”.
   a) Het programma geeft voor alle schepen die wel als laatst bij een van de drie dichtstbijzijnde PH’s verbonden waren, een voor een voor elk schip een informatiestroom weer en verandert de status van het schip als volgt:
      - “Bevestigd” indien het schip niet het schip met de calamiteit is.
      - “Onbekend, bemanning reageert, systeem niet”, indien het om de soort calamiteit ‘middel’ gaat.
      - “Onbekend, bemanning en systeem reageren niet”, indien het om de soort calamiteit ‘hoog’ gaat
5. De actor klikt “Volgende”.
   a) Het programma geeft voor alle schepen die **niet** als laatst bij een van de drie dichtstbijzijnde PH’s verbonden waren, één voor één voor elk schip een informatiestroom weer tussen meldkamer en de PH waar het schip het laatst aangemeld is en verandert de status van het schip als volgt:
      - “Bevestigd”, als het schip bij de PH aangemeld is.
      - “Onbekend”, als het schip niet bij de PH aangemeld is.
6. De actor klikt “Volgende”.
   a) Het programma licht het schip met de calamiteit op.

Postcondities: Er wordt weergegeven welk schip bij de calamiteit betrokken is.

3.6.4 Klassendiagram

Hieronder is het klassendiagram weergegeven. De klasse Simulator is de controller. Deze klasse stuurt dus de hele simulatie aan. Verder verzorgt de klasse GUI de interface met de gebruiker en is dit dus het view gedeelte van de simulator. De overige klassen behoren tot het model gedeelte. De klassen Hub en Vervoersmiddel zijn abstract omdat hier geen instanties van gemaakt kunnen worden. Alleen van de subklassen kunnen instanties worden gemaakt.

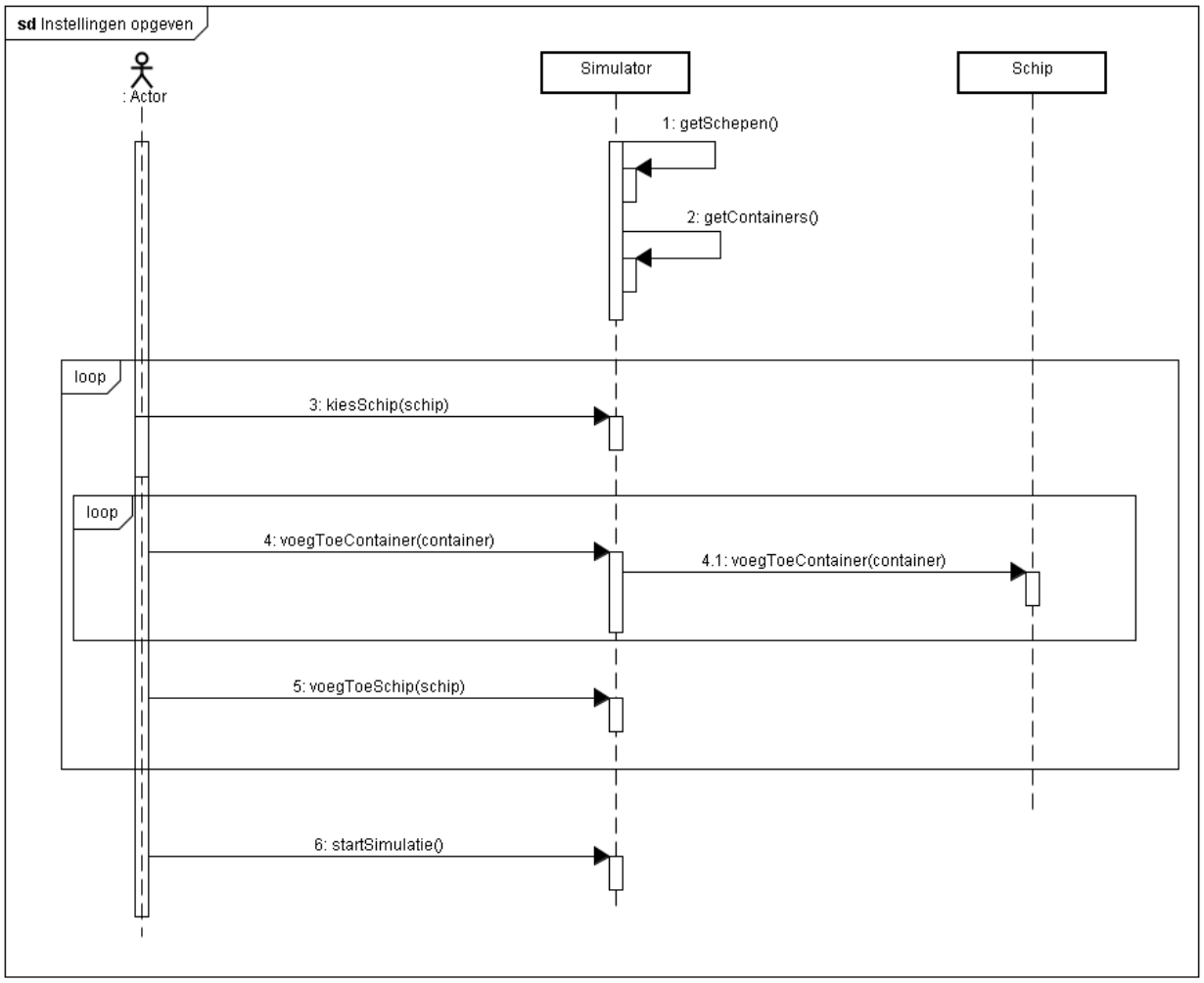


Figuur 2: Klassendiagram met Simulator als hoofdklasse.

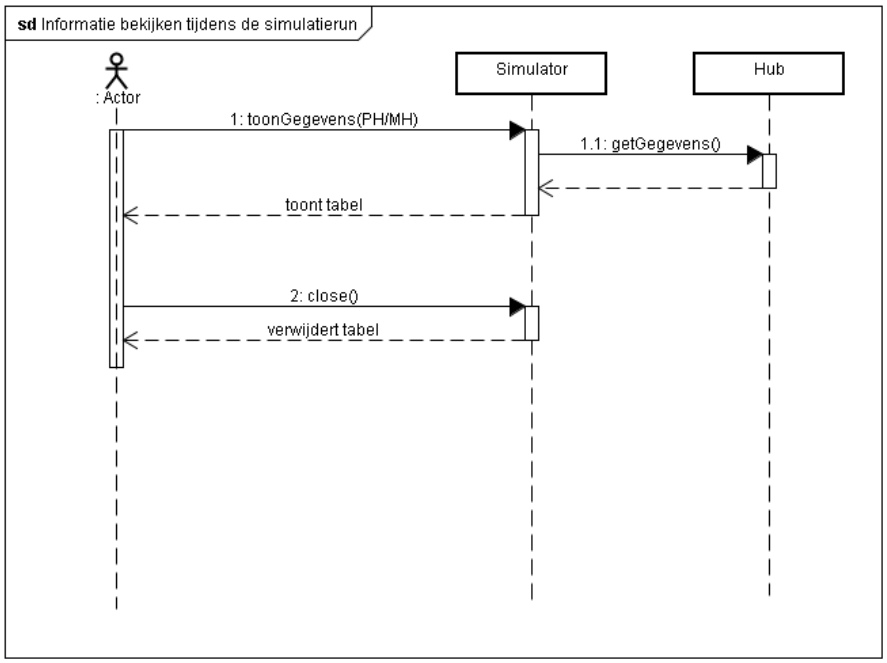
3.6.5 Sequencediagrammen

Hieronder volgen de sequencediagrammen. Deze lopen parallel met de use cases.

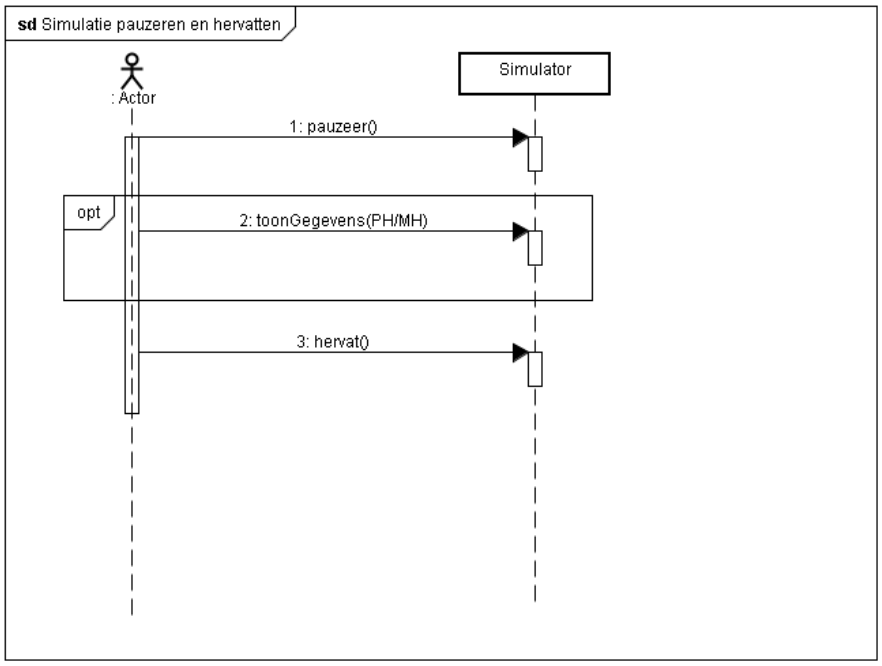
3.6.5.1 Instellingen opgeven



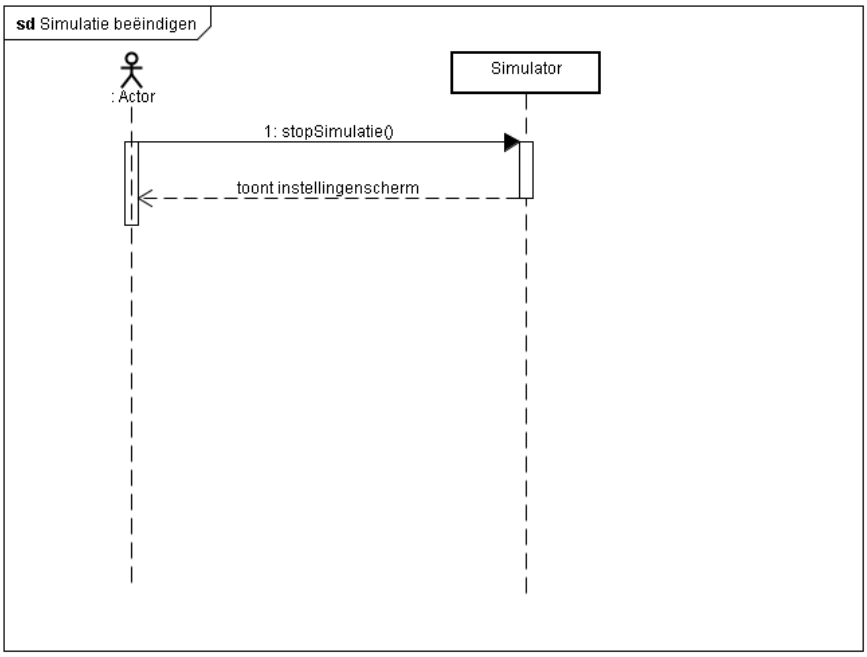
3.6.5.2 Informatie bekijken tijdens de simulatierun



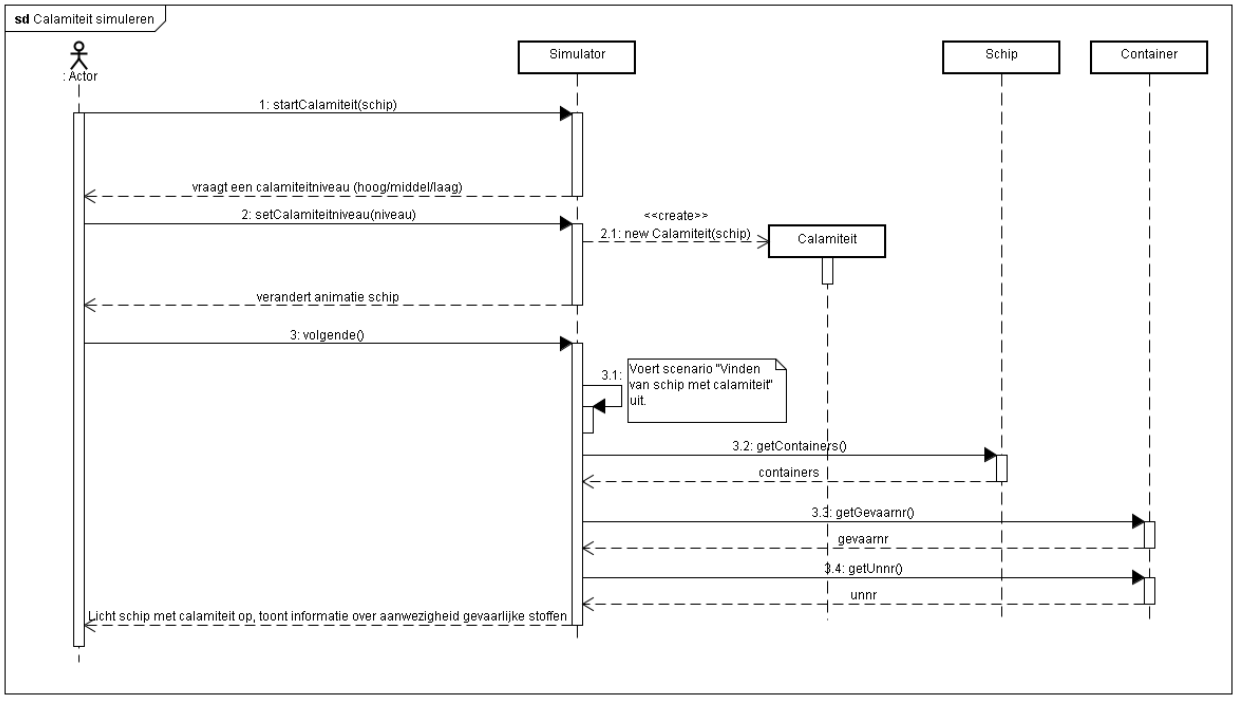
3.6.5.3 Simulatie pauzeren en hervatten



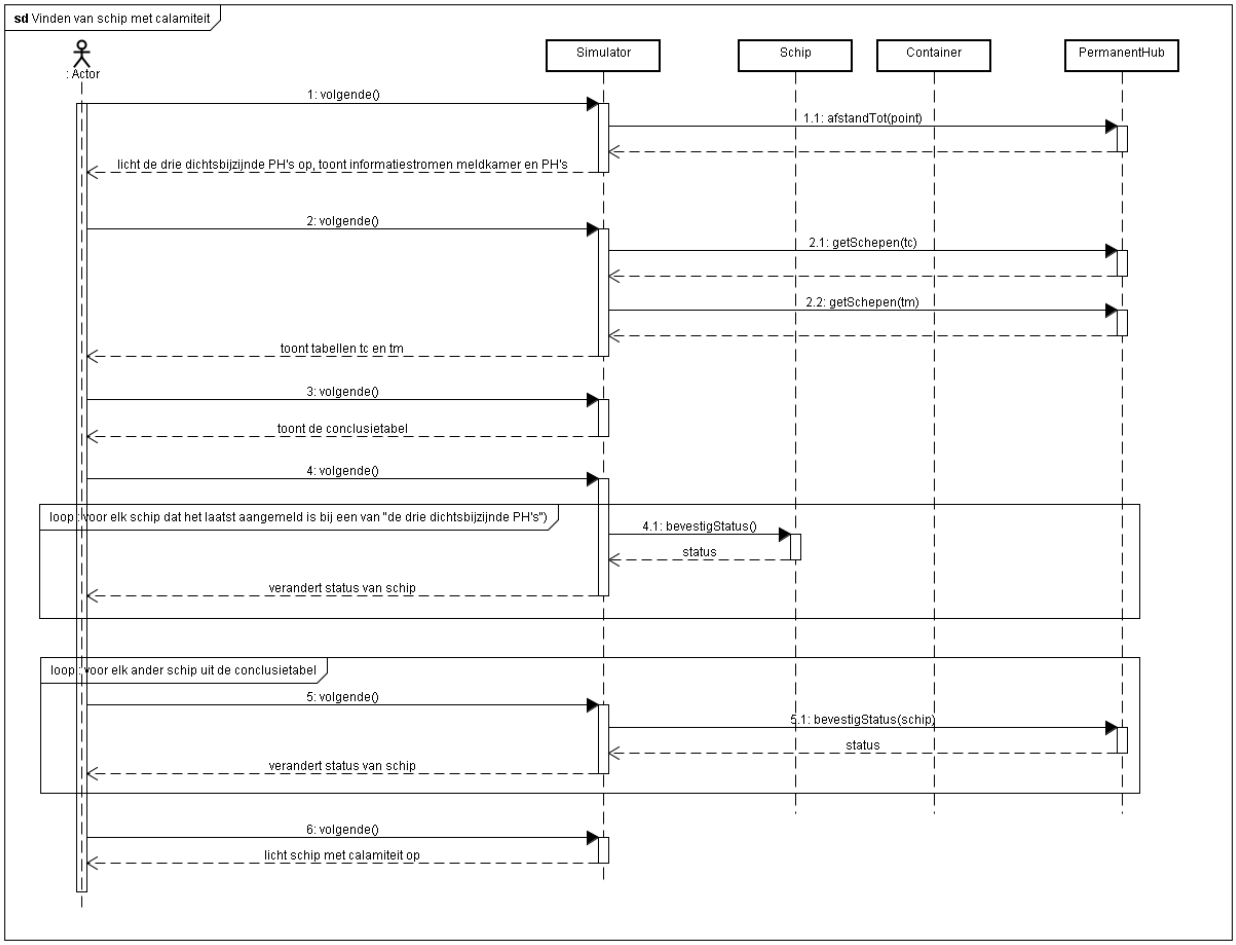
3.6.5.4 Simulatie beëindigen



3.6.5.5 Calamiteit simuleren



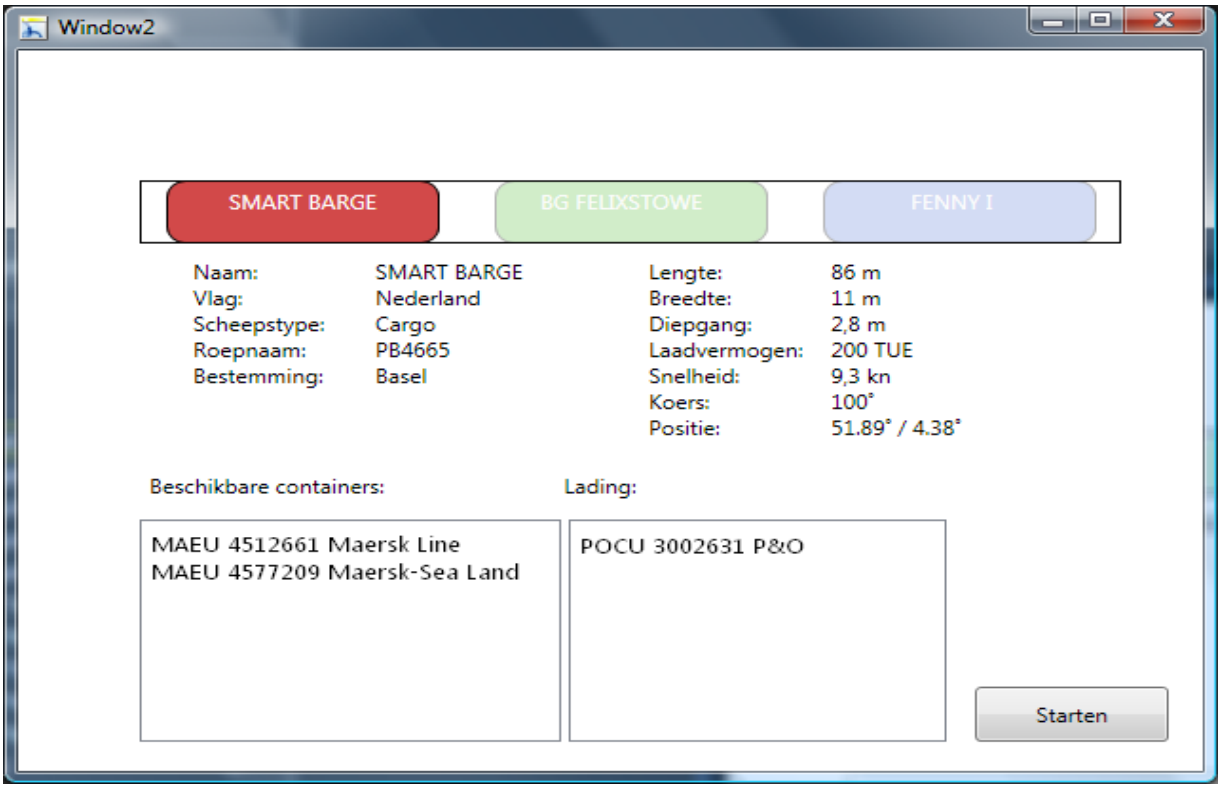
3.6.5.6 Vinden van schip met calamiteit



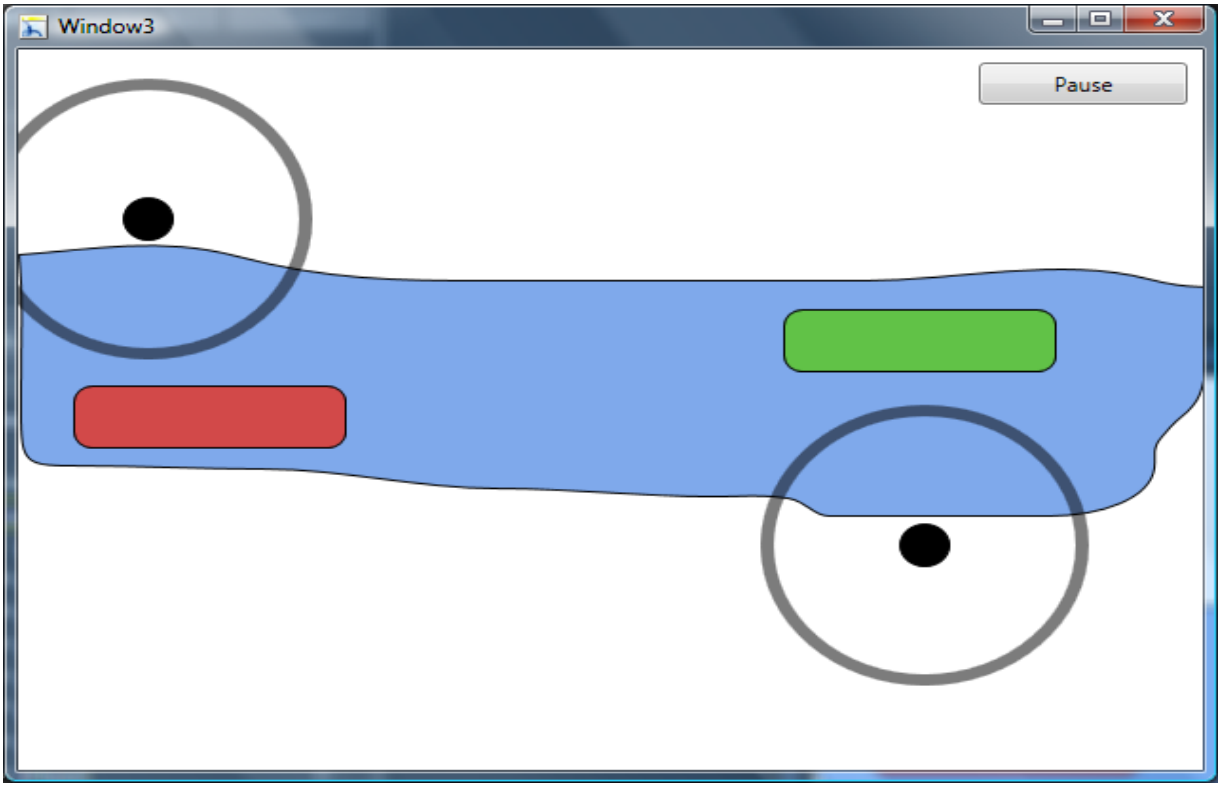
3.6.6 User-interface

Hieronder worden de belangrijkste weergaven van het systeem getoond. Alle tabellen en tekstbevattende gebieden zullen draaibaar zijn, zodat ze van elke positie leesbaar zijn. Onderstaande afbeeldingen dienen alleen om de functionaliteit van het systeem te tonen en de werkelijke weergave zal er anders uitzien.

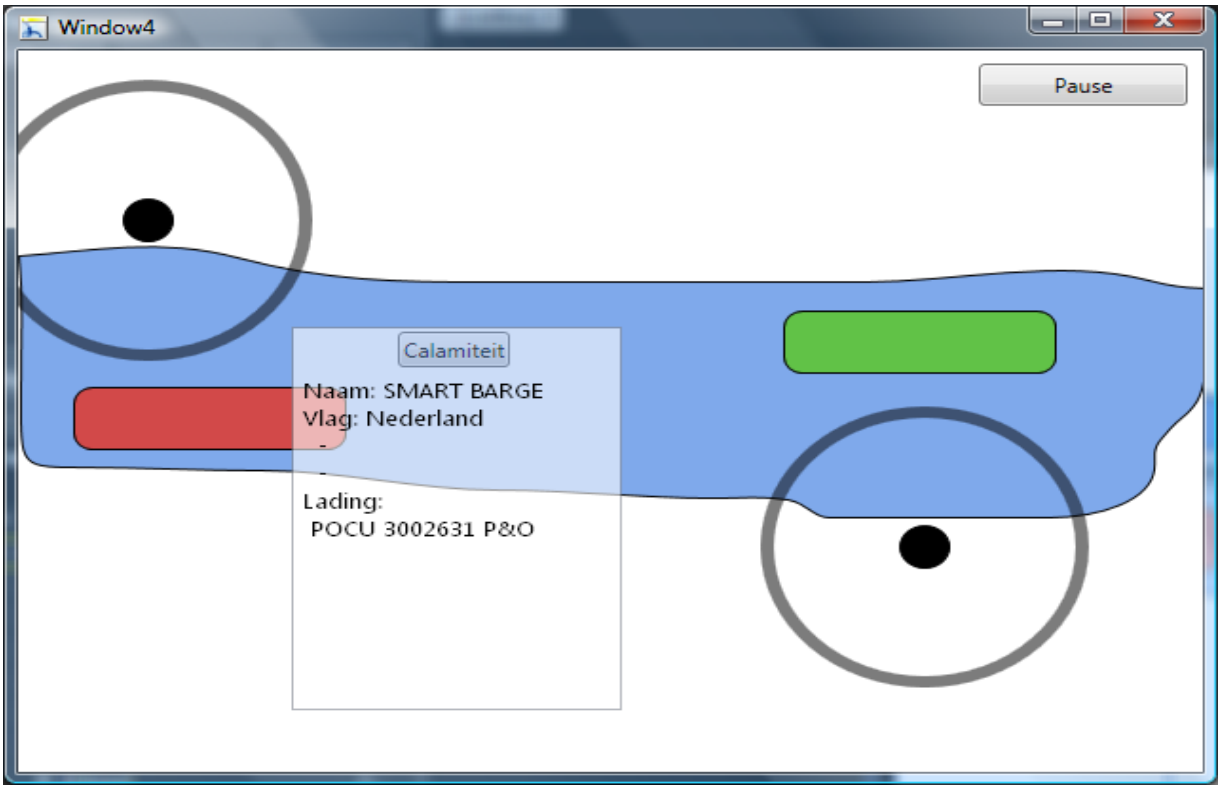
3.6.6.1 Instellingenscherm



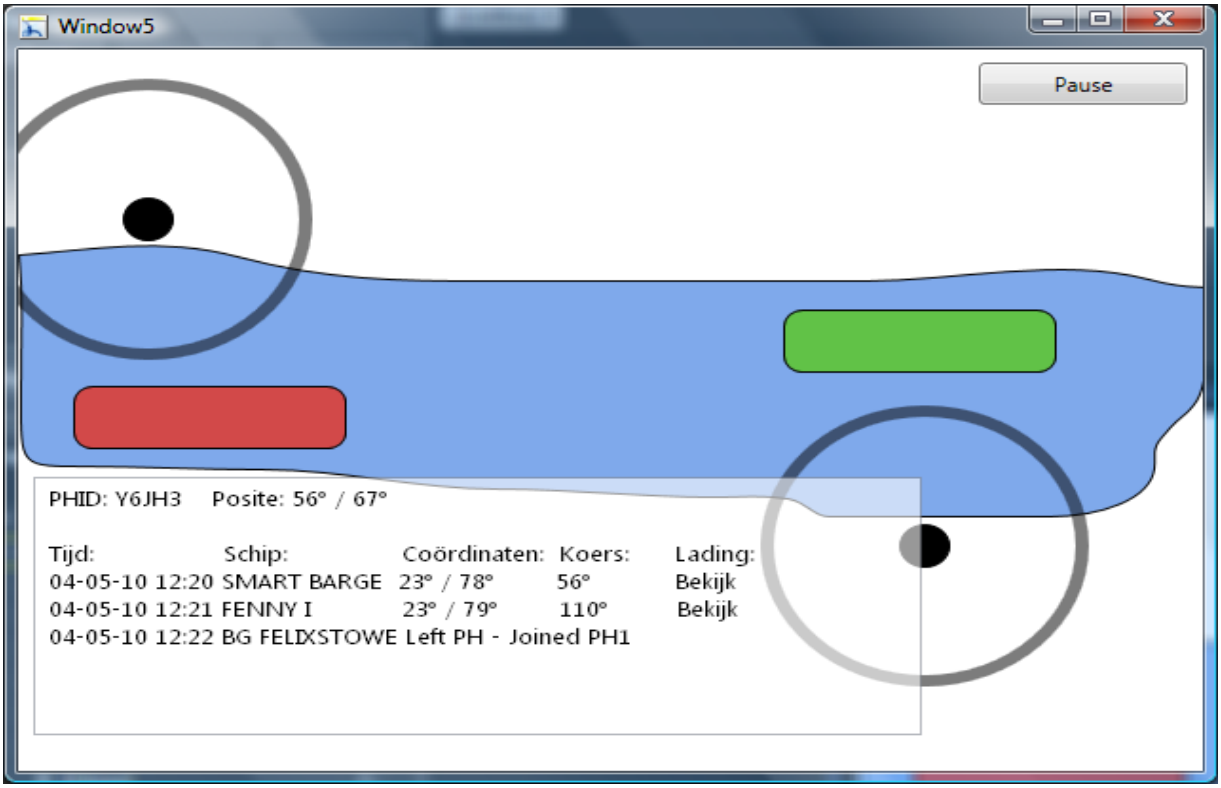
3.6.6.2 Simulatiescherm



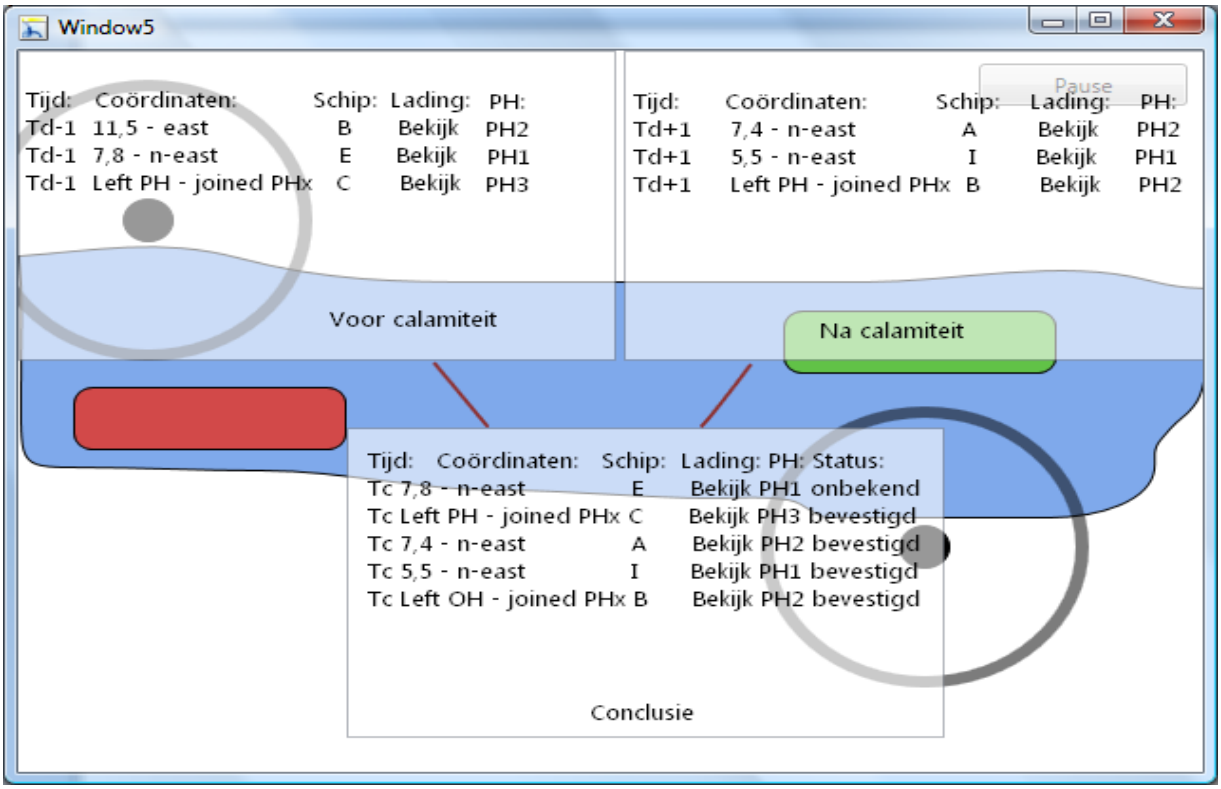
3.6.6.3 Informatie van een MH weergeven



3.6.6.4 Informatie van een permantent hub weergeven



3.6.6.5 Procedure van calamiteit weergeven



4 Begrippenlijst

AIS	Automatic Identification System (AIS) is een aanvulling op het bestaande verkeersmanagement van verkeersposten door Rijkswaterstaat en de bestaande schip-schip communicatie.
BIC-code	Een code voor het identificeren van een container (uitgegeven door het International Container Bureau).
C#	C# is een objectgeoriënteerde programmeertaal ontwikkeld door Microsoft als deel van het .NET Framework.
SchipStatus	Een datatype dat bestaat uit een timestamp en uit alle gegevens van schepen die verandering ondergaan. Bij het scannen door de PH's wordt voor elk gevonden schip een SchipStatus opgeslagen met de meest recente positie, koers en snelheid van het schip.
Smart containers	Een door Centric bedacht concept om op elk moment bijna realtime informatie over vervoersmiddelen en zeecontainers te kunnen opvragen. Dit concept is uitgewerkt in het document 'MH's and Smart Containers visie'.
Surface	De Surface tafel is een multi-touch-product van Microsoft dat ontwikkeld is als een software en hardware combinatie die één of meerdere gebruiker(s) toestaat om content te bewerken door middel van menselijke bewegingen of objecten.
TEU	TEU is de aanduiding voor de afmetingen van containers. De afkorting staat voor Twenty feet Equivalent Unit.
UN-nummer	Nummer om gevaarlijke stoffen aan te duiden.
WPF	Windows Presentation Foundation, het grafische subsysteem dat een onderdeel is van het .NET Framework van Microsoft.

Bijlage E. Database ontwerp

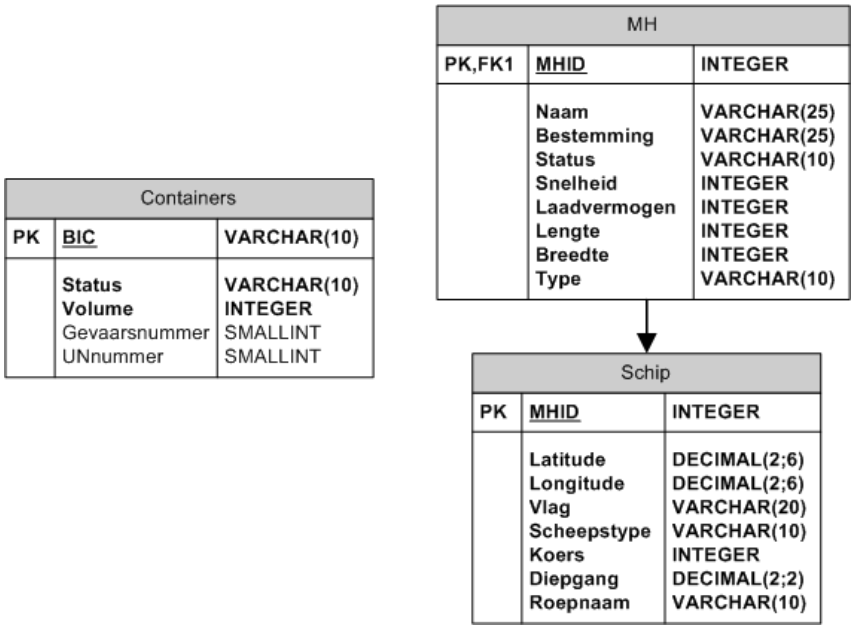
1 Introductie

Tijdens de ontwerpfase hadden we het idee om voor elke hub een eigen database bij te houden. Deze manier van opslaan van informatie nadert de opslag van gegevens in de werkelijkheid. Het gebruik van een database brengt echter ook wat moeilijkheden met zich mee. Zo moet er een databasecontroller worden gemaakt en moet er een databaseserver beschikbaar zijn. Op aanraden van onze stagebegeleider hebben we de database er uit gelaten. Hij vond dat we de simulatie zo simpel mogelijk moesten houden. Ook konden we de informatie gewoon opslaan in objecten, omdat het maar om een kleine hoeveelheid informatie ging. Hieronder volgt een overzicht van de door ons ontworpen databases.

2 Databaseontwerp Mobiele Hubs

Figuur 1 geeft het databaseontwerp van een MH weer. Elke MH krijgt zijn eigen database. Zoals te zien is bestaat dit ontwerp uit een drietal tabellen. Hoewel er in de simulatie alleen schepen als transportentiteiten worden gebruikt, is er voor gekozen om een tabel genaamd 'MH' te maken die gegevens bewaard die voor elke soort transportentiteit van belang zijn. Dit biedt mogelijkheden voor een eventuele uitbreiding van het systeem, zodat ook andere transportentiteiten toegevoegd kunnen worden en er gegeneraliseerd kan worden over al deze entiteiten. Voor een bepaald schip, komt de 'MHID' in de tabel 'MH' overeen met de 'MHID' van de tabel 'Schip'. In de tabel 'MH' is een veld 'Type' opgenomen die de soort transportentiteit beschrijft. Bij de simulatie zal dit veld voor elk record de waarden 'Schip' bevatten.

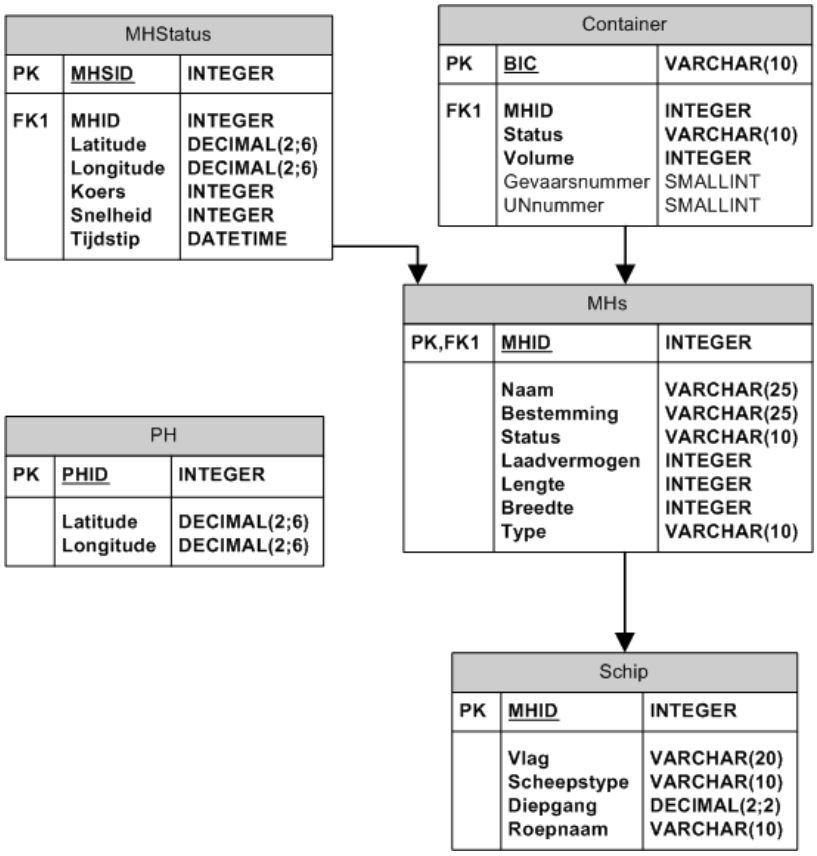
Daarnaast is er een derde tabel genaamd 'Container' in het figuur opgenomen. Deze bevat alle containers die vervoerd worden door de MH. Een koppeling tussen de MH en containers is niet nodig, gezien alle containers in deze tabel tot één en dezelfde MH toebehoren.



Figuur 1: Database ontwerp van een mobiele hub.

3 Databaseontwerp Permanente Hubs

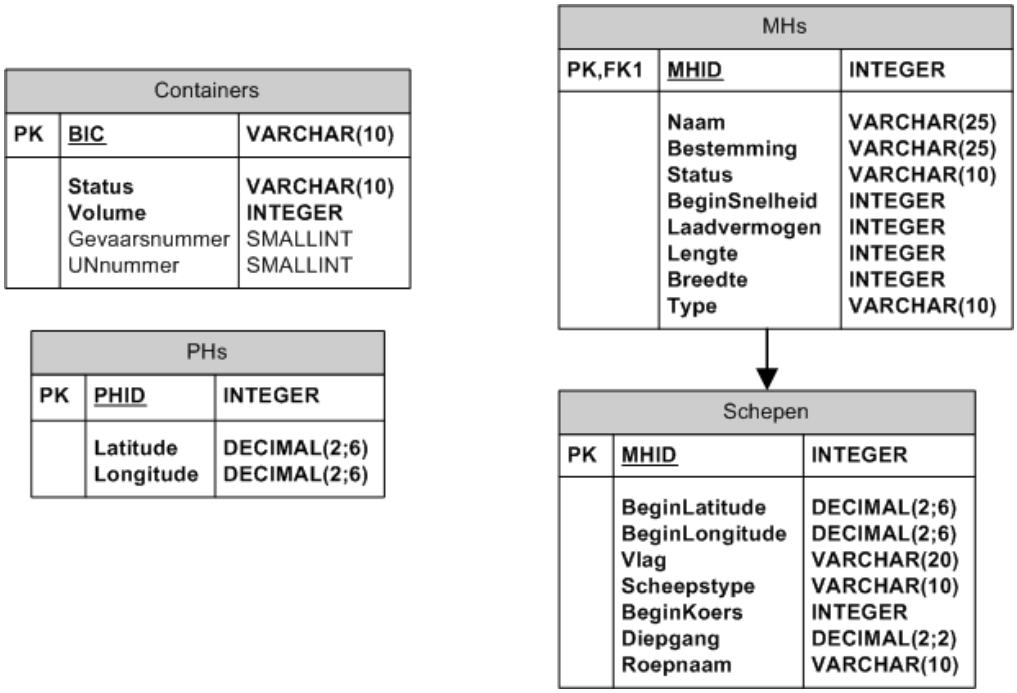
Figuur 2 geeft het databaseontwerp van een PH weer. Elke PH krijgt weer zijn eigen database en beschikt alleen over gegevens en containers van MH's die recentelijk binnen bereik van de PH zijn geweest. Naast de gegevens en containers van MH's slaat de PH ook MHStatussen op. Ook dit gebeurd alleen voor MH's die recentelijk binnen bereik zijn geweest. Ten slotte is een tabel 'PH' opgenomen in het figuur. Deze zal maar één record bevatten, namelijk de gegevens van de PH zelf. Immers hoeft een PH niets over anderen PH's te weten.



Figuur 2: Database ontwerp van een permanente hub.

4 Databaseontwerp simulator

Naast databases voor de MH's en PH's komt er nog een extra database die nodig is voor juiste werking van de simulator, zie Figuur 3. De simulator moet namelijk wél over gegevens van alle containers, MH's en PH's kunnen beschikken. Bij het opstarten moet de gebruiker immers uit een lijst met MH's en containers kunnen kiezen. Ook bij calamiteit moet de simulator weten welke PH's in de buurt liggen. Zo'n centrale database is dus alleen noodzakelijk voor de simulatie en zal in werkelijkheid niet bestaan. Wel zal er in werkelijkheid bijvoorbeeld bij de meldkamer gegevens over alle PH's beschikbaar zijn, maar niet van alle gegevens die deze PH's hebben opgeslagen van MH's en containers.



Figuur 3: Database ontwerp van de simulator.

Bijlage F. Handleiding

Simulator Smart Connected Containers

Handleiding

Rob Post
Oskar van Rest
Tim Roorda

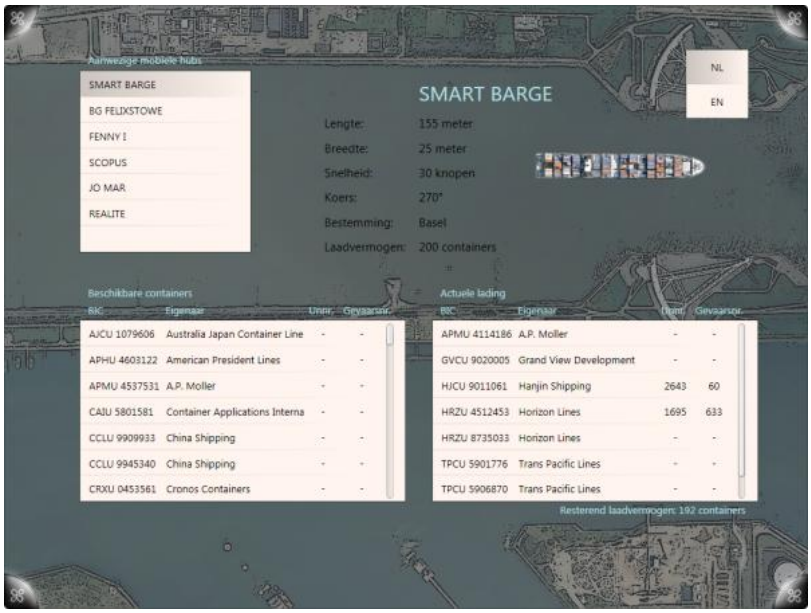
16 juni 2010
Versie 1

1 Introductie

Dit document beschrijft hoe de simulator voor het *Mobile Hubs and Smart Connected Containers* concept bediend kan worden. Hierin komen alle functies van het programma aan bod zodat u het concept duidelijk aan anderen kunt overbrengen. Om te beginnen worden alle mogelijkheden van het instellingenscherm besproken. Dit wordt gevolgd door uitleg over het simulatiescherm. Er wordt afgesloten met een beschrijving van het calamiteitenscherm.

1.1 Instellingenscherm

Zodra u het programma start, ziet u het instellingenscherm. Het instellingenscherm is weergegeven in Figuur 1.



Figuur 1: Het instellingenscherm.

Het instellingenscherm haalt alle gegevens uit het bestand *data.xml*. In dit bestand staats alles gedefinieerd wat in de simulator te zien is. Door het aanpassen van dit bestand kunt u de simulator naar eigen wensen instellen. U kunt bijvoorbeeld MH's toevoegen, verwijderen of aanpassen. Het zelfde geldt voor PH's en containers. De structuur van het bestand is vanzelfsprekend zodat het makkelijk is om aanpassingen te maken. Het *data.xml* bestand is te vinden in de hoofdmap van de simulator.

Dit scherm bevat een aantal lijsten en wat tekst. Rechtsboven op het scherm bevindt zich een lijst met talen die gebruikt kunnen worden in de simulatie. Door op de gewenste taal te drukken kunt u de taal van de simulator veranderen. Denk erom dat bij het drukken op een taal de containertoewijzing van alle schepen verandert.

De lijst linksboven is de lijst met alle MH's die in de simulatie aanwezig zullen zijn. Door met één vinger op een MH te drukken krijgt u rechts alle informatie van deze MH te zien alsmede een plaatje van deze MH.

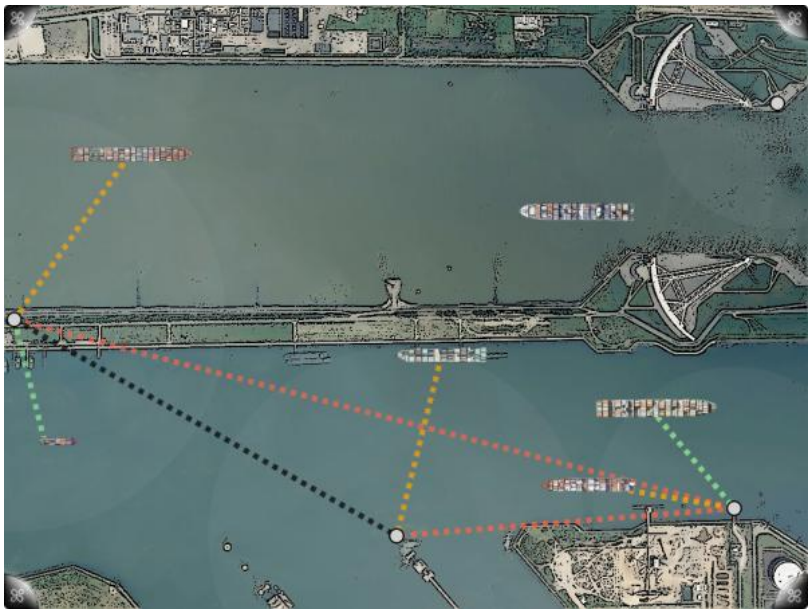
De lijst linksonder geeft alle containers weer die in de simulatie aanwezig zijn. De lijst rechtsonder geeft de actuele lading van de geselecteerde MH weer. Zodra het instellingenscherm gestart wordt, wordt een aantal containers willekeurig op MH's geplaatst. Door met uw vinger op een container te drukken en deze naar de MH te slepen is het mogelijk om het schip met extra containers te beladen. Ook is het op deze manier mogelijk om containers van de MH af te halen. U kunt ook met meerdere vingers tegelijk meerdere containers laden of lossen.

Het instellingenscherm is ook draaibaar zodat u het scherm aan mensen kan laten zien die zich aan de andere kant van de tafel bevinden. Om het instellingenscherm te draaien drukt u met twee vingers op het instellingenscherm en beweegt u deze vingers in een cirkelachtige beweging naar de kant waarnaar u het instellingenscherm wilt draaien. Als u uw twee vingers naar elkaar toe beweegt of van elkaar af beweegt kunt u het instellingenscherm schalen naar de door u gewenste grootte.

Als u klaar bent met het instellen van de simulator kunt u de simulatie starten door met u vinger op het scherm een (korte) lijn van boven naar beneden of van beneden naar boven te trekken.

1.2 Simulatiescherm

Als de simulatie is gestart ziet u het simulatiescherm. Dit scherm is weergegeven in Figuur 2.



Figuur 2: Het simulatiescherm.

In het simulatiescherm is een nabootsing van de werkelijkheid te zien. Op dit scherm ziet u de door uw ingestelde MH's bewegen. Door met uw vinger op het scherm een streep van boven naar beneden,

beneden naar boven, links naar rechts of van rechts naar links te bewegen kunt u de simulatie pauzeren. Door nogmaals deze beweging uit te voeren gaat de simulatie weer verder. Om terug te gaan naar het instellingenscherm kunt u met uw vinger een vierkantje teken op het scherm. Alle eerder ingevoerde instellingen gaan dan verloren.

De MH's op het scherm sturen periodiek hun informatie door naar PH's. Door met uw vinger op een MH's te drukken krijgt u de informatie die in deze MH is opgeslagen te zien. Deze informatie bestaat onder andere uit informatie over de vervoersentiteit en de lading die deze vervoersentiteit bij zich heeft. Dit venster kunt u op dezelfde manier als het instellingenscherm groter maken, kleiner maken en draaien. Het verplaatsen van dit venster gaat met één vinger. Het schalen, draaien en verplaatsen kan alleen als u uw vingers op het bovenste deel van de tabel plaatst en niet op de lijst met containers. Het sluiten van dit venster gebeurd door op de knop met het kruisje te drukken.

Het periodiek oversturen van informatie van MH's naar PH's is te zien aan de gekleurde lijnen die bij een scan van de permanente hubs in de simulatie verschijnen. Er zijn vier verschillende kleuren lijnen te onderscheiden:

- Groen: van MH naar PH; dit betekent dat een MH zich voor het eerst aanmeldt bij een PH. Alle informatie wordt overgestuurd.
- Oranje: van MH naar PH; dit betekent dat een MH reeds is aangemeld bij deze PH en alleen zijn informatie, zoals zijn positie, updatet.
- Rood: van PH naar PH; dit betekent dat een MH zich bij een nieuwe PH heeft aangemeld en de nieuwe PH laat de vorige PH weten dat de MH nu bij hem is aangemeld.
- Zwart: van PH naar PH; dit betekent dat de nieuwe PH tegen de voor vorige PH zegt dat die alle informatie over deze MH kan verwijderen.

Door met uw vinger op een PH te drukken kunt u de meest recente informatie van elke aangemelde of overgestapte MH zien die in de PH wordt opgeslagen. Ook dit venster kunt u verplaatsen, schalen en draaien. Dit gebeurd op dezelfde manier als bij het venster van een MH. Ook hier geldt dat het schalen, draaien en verplaatsen alleen mogelijk is als u uw vingers op het bovenste deel van de tabel plaatst en niet op de lijst met aangemelde en overgestapte MH's. Deze lijst is tevens de belangrijkste informatie die in een PH wordt opgeslagen. Door op een item uit deze lijst te drukken krijgt u ook de informatie en gegevens over de lading van deze MH te zien. Het sluiten van dit venster gebeurt weer door op de knop met het kruisje te drukken.

Zodra de simulatie eenmaal aan de gang is, beginnen de PH's met het verzamelen van informatie. Het is aan te raden om dit proces een tijdje (tenminste vijftien seconden) te laten gaan, zodat de PH's genoeg informatie kunnen verzamelen. Dit zorgt ook voor een meer realistische weergave van het concept. Om de kracht van het concept te tonen is er de mogelijkheid om een MH in de problemen te laten komen. Dit kunt u doen door met uw vinger op een MH te drukken en vervolgens op één van de drie knoppen achter 'simuleer calamiteit' te drukken. De drie knoppen staan voor de volgende calamiteitniveaus:

- Laag: dit wordt voorgesteld door de knop met één ster. Dit zorgt ervoor dat de MH problemen krijgt maar de bemanning en de transponder kunnen wel reageren.

- Middel: dit wordt voorgesteld door de knop met twee sterren. Dit zorgt ervoor dat de MH problemen krijgt en alleen de transponder nog kan reageren.
- Hoog: dit wordt voorgesteld door de knop met drie sterren. Dit zorgt ervoor dat de MH problemen krijgt en de bemanning noch de transponder kunnen reageren.

Na enkele seconden ziet u een ontploffing verschijnen op de door u geselecteerde MH. Vervolgens krijgt u het calamiteitenscherm te zien.

1.3 Calamiteitenscherm

In het calamiteitenscherm zijn alle stappen weergegeven die moeten worden doorlopen om het getroffen schip op te kunnen sporen. Als er een tweede monitor op de Microsoft Surface tafel is aangesloten zal het calamiteitenscherm op de twee monitor worden weergegeven. Is dit niet het geval dan wordt het calamiteitenscherm boven het simulatiescherm weergegeven zoals te zien is in Figuur 3.

Figuur 3: Het calamiteitenscherm.

Om in het calamiteitenscherm naar de volgende stap te gaan gebruikt u hetzelfde gebaar als het gebaar dat u gebruikt om de simulatie te pauzeren. Dit is het van boven naar beneden, beneden naar boven, links naar rechts of van rechts naar links bewegen van uw vinger op het scherm. De calamiteitenprocedure bestaat uit de volgende stappen (in een andere taal hebben deze stappen vanzelfsprekend een andere naam):

- Melding: in deze stap wordt een melding aan de meldkamer gedaan vanaf de kant of vanaf een andere MH als deze in de buurt is. De plaats van de melding wordt aangegeven door een telefoon. De meldkamer wordt weergegeven door een zwaailicht. De melding wordt gevisualiseerd door de blauwe lijn tussen de telefoon en de meldkamer.

- Dichtstbijzijnde PH's: in deze stap worden de drie dichtstbijzijnde PH's bij de locatie van de melding gezocht. Deze lichten rood op.
- Situatie voor calamiteit: in deze stap wordt de informatie die aanwezig is in de drie dichtstbijzijnde PH's van vlak voor het tijdstip van de calamiteit weergegeven.
- Extra scan: er wordt een extra scan gedaan door de drie dichtstbijzijnde PH's. Zo wordt er voor gezorgd dat er ook informatie van na het tijdstip van de calamiteit beschikbaar is.
- Na extra scan: de informatie die de extra scan heeft opgeleverd wordt weergegeven.
- Mogelijk betrokken MH's: de informatie van voor de calamiteit en de informatie van na de calamiteit worden samengenomen en weergegeven. Een van deze MH's is in de problemen.
- Contacteer MH's: de bemanning van de MH's in de samengenomen informatie wordt gecontacteerd om na te gaan of ze een calamiteit hebben. Het contacteren wordt gevisualiseerd door een blauwe lijn tussen de MH's en de meldkamer. De MH die afwijkend reageert is bij de calamiteit betrokken en deze wordt aangegeven met een rood kruis. De andere MH's die niet betrokken zijn bij de calamiteit worden aangegeven met een groen vinkje.
- Betrokken MH: het is duidelijk welke MH bij de calamiteit betrokken is en zijn informatie en ladinggegevens worden weergegeven.

Door nogmaals de pauze beweging met uw vinger te doen verdwijnt het calamiteitenscherm weer en keert u terug naar het simulatiescherm. Vanuit hier kunt u een nieuwe calamiteit starten of terug gaan naar het instellingenscherm.