Naval Strategy Planner

Documentsoort:	Behoeftespecificatie
Versie:	1.0
Datum:	20 Feb 2014
Auteur:	Quinten Soetens
Status:	Opgeleverd

Samenvatting

Dit document bevat de specificaties voor de basis van een informaticasysteem dat toelaat verschillende oorlogs-strategieën te evalueren. Het is geschreven in het kader van het vak "Inleiding software Engineering" (1ste bachelor informatica - Universiteit Antwerpen).

Context

Op haar laatste algemene vergadering besliste de *NAVO* dat ze nood had aan een *Naval Strategy Planner*. Dergelijke planner heeft twee strategische doelen: het opstellen van een aanval-/verdedigings- strategie enerzijds en het visueel voorstellen van de strategie tijdens een pers-conferentie.
Ontwikkelaars die wensten het contract voor de ontwikkeling van dit software binnen te slepen, werden gevraagd een prototype in te sturen.

Om de complexiteit van dergelijk systeem onder controle te houden heeft het team van de UAntwerpen beslist om de ontwikkeling op te delen in twee onafhankelijke producten: een visuele- en een sturende-component. De visuele component: engine, 3d-animaties van aanvallen, ... zal ontwikkeld worden tijdens het vak "Computer Graphics". De component die de posities, bewegingen, ... berekent en bedenkt zal op zijn beurt ontwikkeld worden tijdens het vak "Project Software Engineering." Op het einde moeten beiden componenten echter wel te integreren zijn met elkaar.

Belangrijke criteria voor het prototype zijn een goede aanpasbaarheid (wijzigingen in de wensen van de NAVO moeten eenvoudig op te vangen zijn) en correctheid (berekende posities en simulaties moeten juist zijn).

Legende

Een typische use-case bevat de volgende onderdelen.
Refertenummer & titel:

Wordt gebruikt om naar een bepaalde use-case te verwijzen.

Prioriteit:

De specificatie van een systeem vraagt meer dan wat binnen de voorziene tijd op te leveren is. Vandaar dat we per use-case aangeven in hoeverre die functionaliteit belangrijk is. In volgorde van belangrijkheid kan hier staan: VERPLICHT (deze use-case moet opgeleverd worden), BELANGRIJK (niet essentieel maar bij voorkeur toch opleveren), NUTTIG (interessant maar kan weggelaten worden).

Doel:

Summiere beschrijving van het waarom van de use-case, t.t.z. wat de use-case bijdraagt tot de gehele functionaliteit.

Preconditie:

Summiere beschrijving van de uitgangspunten bij aanvang van de use-case.

Succesvol einde:

Summiere beschrijving van wat opgeleverd zal worden als er niks fout is gegaan.

Stappen:

Een sequentiële beschrijving van hoe de use-case precies zal verlopen als alles goed gaat (het zogenaamde "happy day scenario"). De stappen zijn genummerd en kunnen controle instructies (WHILE, IF, ...) bevatten.

Uitzonderingen:

Een lijst van mogelijke probleemgevallen en hoe die behandeld zullen worden. Een probleem geval (a) verwijst naar het nummer van de stap waar het probleem kan optreden, (b) bevat een conditie die aangeeft wanneer het probleemgeval optreedt, (c) omschrijft heel kort (een lijn) hoe het probleem behandeld zal worden.

Voorbeeld:

Een voorbeeld van wat in- of uitgevoerd kan worden.

Soms is een use-case een uitbreiding van een andere use-case, en dan zijn volgende onderdelen relevant.

Uitbreiding:

Een referte naar de use-case waarvan deze een uitbreiding is.

Stappen:

Een lijst van extra en/of aangepaste stappen t.o.v de use-case waarvan deze een uitbreiding is.

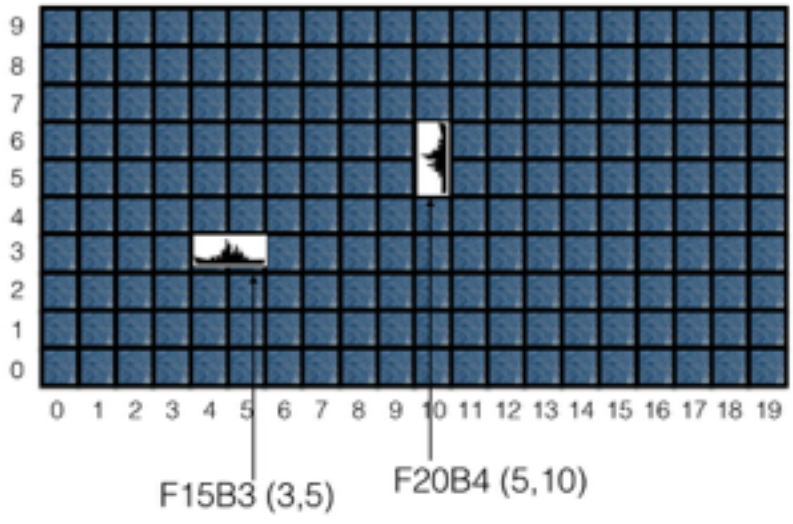
Een uitbreinding (a) verwijst naar het nummer van de stap die uitgebreid wordt, (b) zegt of de uitbreiding voor, na of tijdens de normale stap zal gebeuren, (c) omschrijft wat precies in de uitbreiding zal gebeuren.

Behoeftes

Hieronder volgt een opsomming van alle use-cases inclusief hun prioriteit.

Use-case	**Prioriteit**
1: Invoer	
1.1. Slagveld inlezen	VERPLICHT
1.2. Bewegingen inlezen	VERPLICHT
2: Uitvoer	
2.1. Huidig slagveld wegschrijven	VERPLICHT
2.2. Resterende bewegingen wegschrijven	VERPLICHT
3: Simulatie	
3.1. Bewegen over het slagveld	VERPLICHT
3.2. Automatisch uitvoeren van bewegingen	VERPLICHT

1. Invoer

1.1. Slagveld inlezen	
Prioriteit	VERPLICHT
Doel	De begin-situatie van het systeem bestaat uit een zee waarop 2 schepen gepositioneerd zijn. Informatie over de vorm van de zee en de posities van de schepen is beschikbaar in een XML-gestructureerde file. Doel van deze usecase is het inlezen van de XML-file en controleren of de beginsituatie fysisch haalbaar is.
Preconditie	Een XML bestand met daarop een beschrijving van de zee en de twee schepen. (Zie Appendix 1 voor meer informatie over het XML formaat)
Succesvol einde	Het systeem bevat de beginsituatie van het slagveld.
Stappen	1. Open het invoerbestand 2. Parse het bestand met TinyXML) 3. WHILE Er zijn nog elementen in het geparste document 3.1. Herken het soort element (één van ZEE of SCHIP) 3.2. Lees de verder informatie voor dit element 3.3. IF Verifieer de geldigheid van het element 3.3.1. THEN voeg element toe aan de virtuele zee 4. Verifieer de consistentie van de virtuele zee.
Uitzonderingen	3.1. [Onherkenbaar element] Foutboodschap + positioneer op volgende element in het bestand => verdergaan vanaf stap 3 3.3. [Ongeldige informatie] Foutboodschap + positioneer op volgende element in het bestand => verdergaan vanaf stap 3 4. [Inconsistent virtuele stad] Foutboodschap
Voorbeeld	Een virtuele zee met 2 schepen (zoals op onderstaande prent) staat beschreven in het bestand Slagveld1.0.xml. 

1.2. Bewegingen inlezen	
Prioriteit	VERPLICHT
Doel	Lees de verschillende bewegingen in voor de schepen.
Preconditie	Een XML bestand met de beschrijvingen van de bewegingen.
Succesvol einde	Het systeem bevat de bewegingen die sequentieel in de volgorde van voorkomen in het bestand uitgevoerd kunnen worden.
Stappen	1. Open het invoerbestand met de bewegingen 2. Parse het bestand (met TinyXML) 3. WHILE Er zijn nog elementen in het geparste document 3.1. Herken het soort element (BEWEGING) 3.2. Lees de verder informatie voor dit element 3.3. IF Verifieer de geldigheid van het element 3.3.1. THEN voeg beweging toe aan de overige bewegingen
Uitzonderingen	3.1. [Onherkenbaar element] Foutboodschap + positioneer op volgende element in het bestand => verdergaan vanaf stap 3 3.3. [Ongeldige informatie] Foutboodschap + positioneer op volgende element in het bestand => verdergaan vanaf stap 3
Voorbeeld	Zie het bestand Bewegingen1.0.xml

2. Uitvoer

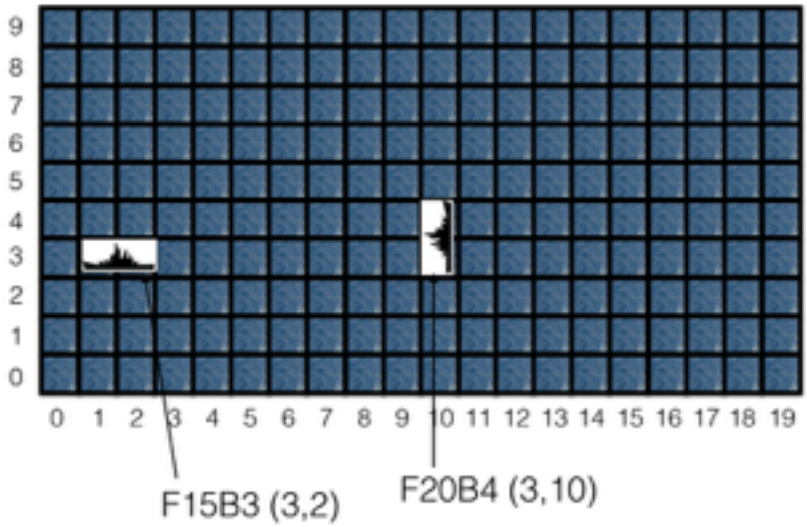
2.1. Huidig slagveld wegschrijven	
Prioriteit	VERPLICHT
Doel	Volledige slagveld zoals hij op moment van uitvoeren is, wegschrijven naar een bestand. Dit bestand beschrijft dus a) hoe de zee eruit ziet, b) hoe de schepen gepositioneerd zijn.
Preconditie	Het systeem is correct geinitialiseerd Een slagveld werd ingelezen
Succesvol einde	Het systeem heeft een tekstbestand (ASCII) uitgevoerd, waarin de informatie over de virtuele zee netjes is uitgeschreven.
Stappen	1. Creëer uitvoerbestand 2. Open uitvoerbestand 3. Schrijf gegevens uit voor de zee. 4. FORALL Schepen 4.1. Schrijf gegevens uit voor het schip 5. Sluit uitvoerbestand
Uitzonderingen	1. [Creatie mislukt] Foutboodschap + schrijf naar console ipv bestand

Voorbeeld	gegeven de input uit het voorbeeld van 1.1, waarbij we 1 zee hebben ingelezen met twee schepen:
	`Bestand: HuidigSlagveld.txt`
	
	`Het huidige slagveld bevindt zich in de Kaspische zee.`
	`Eigenschappen van deze zee:`
	`-Breedte 20`
	`-Lengte 10`
	
	`Schip F20B4 bevindt zich in dit slagveld.`
	`Eigenschappen van dit schip:`
	`-X,Y (5,10)`
	`-orientatie N`
	`-lengte 2`
	
	`Schip F15B3 bevindt zich in dit slagveld.`
	`Eigenschappen van dit schip:`
	`-X,Y (3,5)`
	`-orientatie W`
	`-lengte 2`

2.2. Resterende bewegingen wegschrijven	
Prioriteit	VERPLICHT
Doel	Alle nog niet gemaakte bewegingen wegschrijven naar een bestand.
Preconditie	Het systeem is correct geinitialiseerd Een slagveld werd ingelezen
Succesvol einde	Het systeem heeft een tekstbestand (ASCII) uitgevoerd, waarin de informatie over de resterende bewegingen netjes is uitgeschreven.
Stappen	1. Creëer uitvoerbestand 2. Open uitvoerbestand 3. WHILE nog te maken bewegingen aanwezig 3.1. Schrijf gegevens uit voor de beweging 4. Sluit uitvoerbestand
Uitzonderingen	1. [Creatie mislukt] Foutboodschap + schrijf naar console ipv bestand 3. [Geen te maken bewegingen aanwezig] Als er geen te maken bewegingen zij, is de output “geen bewegingen”

Voorbeeld	gegeven de input uit het voorbeeld van 1.2,
	
	`Bestand: ResterendeBewegingen.txt`
	
	`Schip F20B4 zal volgende bewegingen nog uitvoeren:`
	`-Beweeg achteruit over een afstand 2.`
	
	`Schip F15B3 zal volgende bewegingen nog uitvoeren:`
	`-Beweeg vooruit over een afstand 3.`

3. Simulatie

3.1: Bewegen over het slagveld	
Prioriteit	VERPLICHT
Doel	Het simuleren van een bewegend schip
Preconditie	Het systeem is correct geinitialiseerd. Een slagveld werd ingelezen. Een beweging werd ingelezen.
Succesvol einde	Het schip waarvoor de beweging werd gedefineerd bevindt zich op de juiste eindpositie.
Stappen	1.1 IF richting beweging is “vooruit” 1.1.1 schuif schip over afstand van de beweging in de richting van de huidige oriëntatie van het schip. 1.2. ELSE 1.2.1 schuif schip over de afstand van de beweging in de tegengestelde richting van de huidige oriëntatie van het schip
Uitzonderingen	1.1.1,1.2.1 [Schip raakt kant] stop beweging & geef foutmelding wanneer nog niet over de volledige afstand verschoven.
Voorbeeld	`Na het uitvoeren van de bewegingen uit voorbeeld 1.2 toegepast op de situatie uit voorbeeld 1.1 krijg je volgende eindsituatie.` 

3.2 Automatisch uitvoeren van bewegingen	
Prioriteit	VERPLICHT
Doel	Een volledig scenario van meerdere bewegingen wordt uitgevoerd.
Preconditie	Het systeem is correct geinitaliseerd. Een slagveld werd ingelezen. Bewegingen werden ingelezen.
Succesvol einde	Er zijn geen onafgewerkte bewegingen. Alle schepen bevinden zich in de juiste eindpositie.
Stappen	1. WHILE nog niet afgewerkte bewegingen 1.1. beweeg schip volgens beweging (zie use-case 3.1)
Uitzonderingen	Geen
Voorbeeld	

Appendix 1
Geldige informatie

We moeten nu nog de tags en hun waarde vastleggen die gelden voor ons probleem-domein. Dit bestaat uit het vastleggen van de mogelijke tags, Attributen en de verwachte inhoud (content).
De mogelijke tag-identifiers zijn:
slagveld, zee, naam, lengte, breedte, schip, oriëntatie, bewegingen, beweging, schipnaam, afstand, richting
Coordinaten in de virtuele zee worden weergegeven met attribuut-identifiers X en Y.
Tag-identifiers en attribuut-identifiers zijn niet hoofdletter gevoelig, inhouden zijn dit wel.

Tags	Subtags	Attributen	Content
Zee	Naam		String
	Lengte		Integer
	Breedte		Integer
Schip		X	Integer
		Y	Integer
	Naam		String
	Lengte		Integer
	Orientatie		String
Beweging	Schipnaam		String

	Richting		String
	Afstand		Integer

(In)consistente virtuele zee

Het bestand met de in te lezen zee wordt met de hand geschreven. Om de ingelezen virtuele zee te kunnen simuleren moet de informatie consistent zijn.
Een virtuele zee is consistent als:

- Een schip binnen de grenzen van de zee ligt.
- Een zee geen negatieve dimensies (lengte, breedte) heeft.
- Een schip geen negatieve coordinaten (x,y) heeft.
- De coordinaten (x,y) de positie van de achtersteven vastleggen.
- Een schip heeft een lengte van minstens 1.
- Een schip een naam heeft.
- Een schip een oriëntatie “N”,”O”,”Z” of “W” heeft

Een beweging is consistent als:
- De afstand niet negatief of nul is.
- De richting “vooruit” of “achteruit” is.