Architectuur document

Project:	WARGAMES
Project team:	Distributed Simulations Ltd.
Teamleden:	Nico Kuijpers Samuil Angelov André Postma Frank Peeters

Opdrachtgever:	Nationaal Coördinator Terrorismebestrijding
Versie:	1.2
Versiedatum:	18 april 2014
Status:	Definitief

Documenthistorie

Versie	Wijzigingen	Auteur	Datum	Verzendlijst	Verzoek
0.1	Opzet document	Nico Kuijpers	24 okt 2012	Samuil Angelov Frank Peeters	C C
0.2	Aanpassing/uitbreiding hoofdstukken 2 en 3	Nico Kuijpers	1 nov 2012	Samuil Angelov Frank Peeters	C C
0.3	Hoofdstukindeling aangepast. Hoofdstuk 1 t/m 7 aangepast.	Nico Kuijpers	7 nov 2012	Samuil Angelov Frank Peeters	C C
1.0	Hoofdstuk 8 ingevuld. Kleine aanpassingen in alle hoofdstukken.	Nico Kuijpers	9 nov 2012	Samuil Angelov Frank Peeters	G G
1.1	Reviewcommentaar verwerkt	Nico Kuijpers	11 nov 2012	Samuil Angelov Michael Franssen Jurjen van Geenen Frank Peeters	K K K G
1.2	Figuur 2 en figuur 3 aangepast. Tekstuele aanpassingen	Nico Kuijpers	18 april 2014	André Postma Marcel Meesters Frank Peeters	K K G

Inhoud

H1 Inleiding

In dit architectuur document worden aanwijzingen gegeven in italic (schuinschrift). Een voorbeelduitwerking van het architectuur document is gegeven in roman (rechte letters).

Beschrijf in de inleiding:

- *De context van het project*
- *De applicatie*
- *De niet-functionele eisen met betrekking tot betrouwbaarheid, performance, beveiliging en schaalbaarheid*
- *Het doel van dit document*

Context
De Nationale Coördinator Terrorismebestrijding (NCT) is verantwoordelijk voor het opsporen van mogelijke terroristische activiteiten in Nederland. Hiertoe wordt nauw samengewerkt met politie, justitie en defensie, waarbij de NCT voornamelijk een coördinerende functie heeft. Een van de taken in het kader van terrorismebestrijding is het onderscheppen van (civiele) vliegtuigen die het Nederlands luchtruim binnenvliegen zonder dat radiocontact is gemaakt. Deze taak wordt uitgevoerd door de Koninklijke Luchtmacht (KLu) middels snelle inzet van twee F16 jachtvliegtuigen. Hiertoe staan te allen tijde ten minste twee toestellen paraat op vliegbasis Volkel.

De Nederlandse regering is voornemens de F16 te vervangen door een ander toestel, mogelijk de Joint Strike Fighter (JSF). In nauwe samenwerking met de KLu en Schiphol wil de NCT een gedistribueerd trainingscentrum oprichten om het onderscheppen van vliegtuigen te oefenen. In deze training ligt de nadruk op het oefenen van de communicatie en afstemming tussen de betrokken actoren. De actoren zijn 2 JSF piloten, 1 piloot van een civiel toestel, 1 civiele verkeersleider (verkeerstoren Schiphol), 1 militair verkeersleider (luchtmachtbasis Volkel), de NCT of diens plaatsvervanger en de trainingsleider. De KLu heeft de beschikking over een tweetal trainingssimulatoren voor de JSF en stelt deze beschikbaar voor het project. Schiphol stelt een trainingssimulator voor de Boeing 747 beschikbaar.

Applicatie
Het gedistribueerd trainingscentrum zal bestaan uit 7 onderdelen, namelijk 2 JSF simulatoren (locatie: Volkel), 1 Boeing 747 simulator (locatie: Schiphol), 1 gesimuleerde radarpost voor de civiele verkeersleider (locatie: Schiphol), 1 gesimuleerde radarpost voor de militaire verkeersleider (locatie: Volkel), een werkstation voor de NCT (locatie: Den Haag) en een werkstation voor de trainingsleider (locatie: Den Haag). De 7 onderdelen worden via internet met elkaar in verbinding gebracht. Naast de drie bemande (man-in-the-loop) vliegtuigsimulatoren kan de trainingsleider vliegbewegingen van andere toestellen simuleren (computergestuurd). Alle vliegbewegingen van de gesimuleerde vliegtuigen kunnen real-time zichtbaar gemaakt worden op de beeldschermen van de simulatoren, op de gesimuleerde radarschermen en op de werkstations. Voor audiocommunicatie tussen de actoren zal gebruik worden gemaakt van commercial off-the-shelf (COTS) producten. Audioberichten worden tijdens de trainingssessie opgeslagen. Na afloop van een training kunnen alle vliegbewegingen opnieuw worden afgespeeld synchroon met de opgenomen audioberichten (after action review, AAR).

Niet-functionele eisen
Onderstaande niet-functionele eisen uit het user requirements document (URS v1.0) worden besproken in dit document:

Betrouwbaarheid:
Gesimuleerde vliegbewegingen worden door werkstation, gesimuleerd radarbeeld en bemande simulatoren met een vertraging van maximaal 1 seconde gerepresenteerd (URS Q.1). In bemande simulatoren worden positie en oriëntatie van vliegtuigen met een afstand tot 1000 m met een vertraging van maximaal 0.1 seconde gerepresenteerd (URS Q.2).

Performance:
Bemande simulatoren vereisen een refresh-rate van 20 ms. De vereiste performance wordt behaald onder de randvoorwaarde dat de benodigde bandbreedte voor internetverkeer maximaal 10 Mb/s bedraagt (URS Q.3). Audioberichten worden zonder vertraging verzonden (URS Q.4).

Beveiliging:
Er gelden geen specifieke eisen met betrekking tot beveiliging.

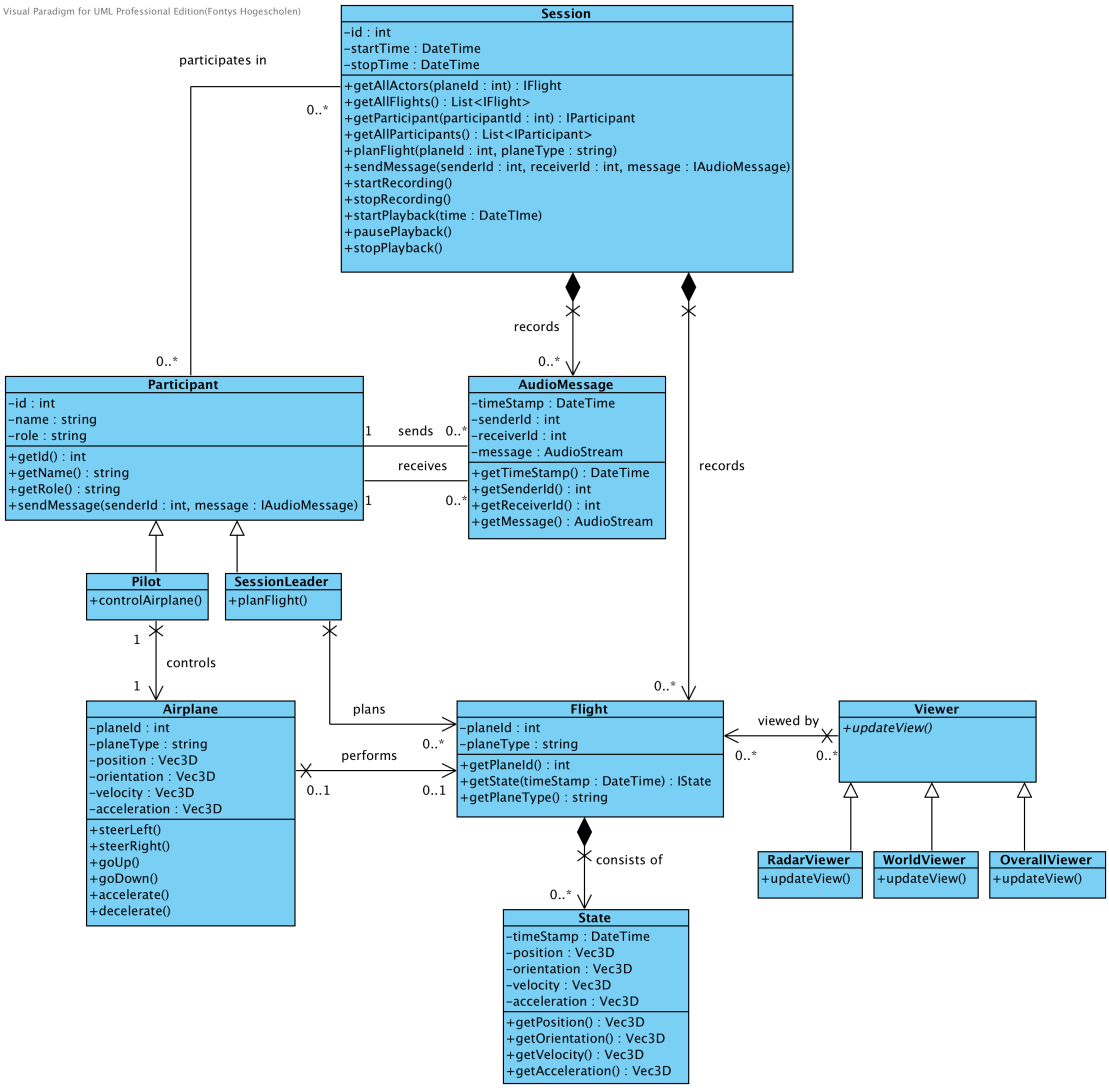
Schaalbaarheid:
Het aantal deelnemende simulatoren en werkstations kan worden uitgebreid tot maximaal 20 (URS Q.5).

Doel van dit document
Dit document verantwoordt de architectuur (software en afbeelding van software modules op hardware) voor de gesimuleerde radarstations en de werkstations. Daarnaast worden de interfaces voor de vliegtuigsimulatoren beschreven met betrekking tot interoperabiliteit. Er worden zeven aspecten van de architectuur beschreven: domeinmodel, persistentie, communicatie, realisatie van niet-functionele eisen, componenten, deployment en specificatie van interfaces.

H2 Domeinmodel

In dit hoofdstuk worden een of meer klassendiagrammen van het domein getoond. Daarnaast wordt een toelichting gegeven en worden gemaakte keuzes verantwoord. Tot slot wordt de afbakening van het project besproken. Tip: gebruik sequentiediagrammen om samenwerking tussen objecten toe te lichten.

Klassendiagram



Figuur 1: Klassendiagram van het domein.

In Figuur 1 is het klassendiagram van het domein afgebeeld. Hierin worden de volgende klassen onderscheiden:

- Session: Representeert een trainingssessie inclusief voorbereiding en After Action Review. Een Session heeft een identificatienummer, een starttijd en een stoptijd. Voorafgaand aan een trainingssessie kunnen vluchten van computergestuurde vliegtuigen worden gepland. Tijdens

een trainingssessie worden alle audioberichten opgenomen en alle vluchten van bemande simulatoren. Audioberichten en alle vluchten kunnen synchroon worden afgespeeld tijdens de After Action Review.

- Participant: Representeert een deelnemer aan een trainingssessie. Een actor heeft een identificatienummer en een naam. Een Actor kan de rol van trainingsleider, NCT, piloot of verkeersleider aannemen.
- Pilot: Representeert een piloot. Een piloot kan een vliegtuig besturen.
- SessionLeader: Representeert de trainingsleider. Een trainingsleider kan vluchten van computergestuurde vliegtuigen plannen.
- AudioMessage: Representeert een audiobericht dat verzonden wordt van de ene actor naar de andere actor. Een audiobericht heeft een zender, een ontvanger, een tijdstip en het bericht zelf (audiostream).
- Airplane: Representeert een bemande vliegtuigsimulator en wordt bestuurd door een piloot. Een Airplane heeft een positie, orientatie, snelheid en versnelling. Een Airplane voert een vlucht uit.
- Flight: Representeert een vlucht. Een vlucht heeft het identificatienummer en type van het bijbehorende vliegtuig. Een vlucht bestaat uit opeenvolgende statussen.
- State: Representeert positie, orientatie, snelheid en versnelling van het vliegtuig op een bepaald tijdstip tijdens de vlucht.
- Viewer: Algemene representatie (abstracte klasse) voor de weergave van vluchtbewegingen tijdens een trainingssessie of After Action Review.
- RadarViewer: Representatie van radarbeeld.
- WorldViewer: Representatie van het buitenbeeld getoond in een bemande vliegtuigsimulator.
- OverallViewer: Representatie van het totaaloverzicht zoals getoond op de werkstations voor de trainingsleider en hoofd NCT.

Afbakening

Het volledig ontwerp van de bestaande man-in-the-loop vliegsimulatoren (simulator Boeing 747 op Schiphol en de twee JSF simulatoren) wordt niet besproken in dit document. Alleen de interfaces ten behoeve van interoperabiliteit worden besproken.

H3 Persistentie

In dit hoofdstuk wordt de persistentie van de objecten beschreven. Beschrijf welke eigenschappen van welke objecten moeten worden bewaard. Beschrijf hoe opslag wordt gerealiseerd (database, serialiseren, XML, etc.). Dit kan verschillen per eigenschap. Voeg het ERD met beschrijving toe indien gegevens worden opgeslagen in een database. Beschrijf tot slot wanneer opslag plaatsvindt. Ook dit kan verschillen per eigenschap.

Eigenschappen van objecten m.b.t. persistentie

Om een onderbroken trainingssessie op een later tijdstip te kunnen vervolgen en ten behoeve van After Action Review dienen de volgende eigenschappen van de objecten opgeslagen te worden (URS F.8):

- Session: id, startTime, stopTime, alle AudioMessages en alle Flights.
- Participant: id, name, role.
- AudioMessage: id zendende actor, id ontvangende actor, timeStamp, message.
- Airplane: planeId, planeType.
- Flight: planeId, planeType en bijbehorende States.
- State: timeStamp, position, orientation, velocity, acceleration.

Data-opslag voor, tijdens en na een trainingssessie

Gegevens van actoren (id, name, role) worden opgeslagen in een database. Voor deze oplossing is gekozen zodat deze gegevens gecombineerd kunnen worden met andere gegevens zoals adres en telefoonnummer van actoren en die buiten de applicatie om kunnen worden gebruikt en aangepast.

Ten behoeve van de planning van een trainingssessie worden vliegbewegingen van computergestuurde vliegtuigen voorafgaand ingevoerd door de trainingsleider. Voor ieder computergestuurd vliegtuig wordt het id en type opgeslagen en daarnaast de status voor iedere seconde gesimuleerde tijd. Op tussenliggende momenten wordt de status alleen opgeslagen indien er wordt gestuurd, versneld of afgeremd. Door middel van serialisatie van Session, Flight en State kan een volledige sessie worden opgeslagen in een bestand. Er is voor serialisatie gekozen omdat dit een efficiënte manier is voor data-opslag en omdat deze gegevens niet met andere applicaties hoeven te worden uitgewisseld. Ten behoeve van de After Action Review wordt de volledige sessie ingelezen, zodat deze opnieuw afgespeeld kan worden (URS F.8).

H4 Communicatie

In dit hoofdstuk wordt de communicatie van en tussen objecten beschreven. Beschrijf welke eigenschappen van welke objecten moeten worden gecommuniceerd. Beschrijf hoe communicatie wordt gerealiseerd (bijvoorbeeld m.b.v Remote Method Invocation, RMI). Dit kan verschillen per eigenschap. Beschrijf tot slot wanneer communicatie plaatsvindt (push-pull, welk object is leidend?, etc.) Ook dit kan verschillen per eigenschap.

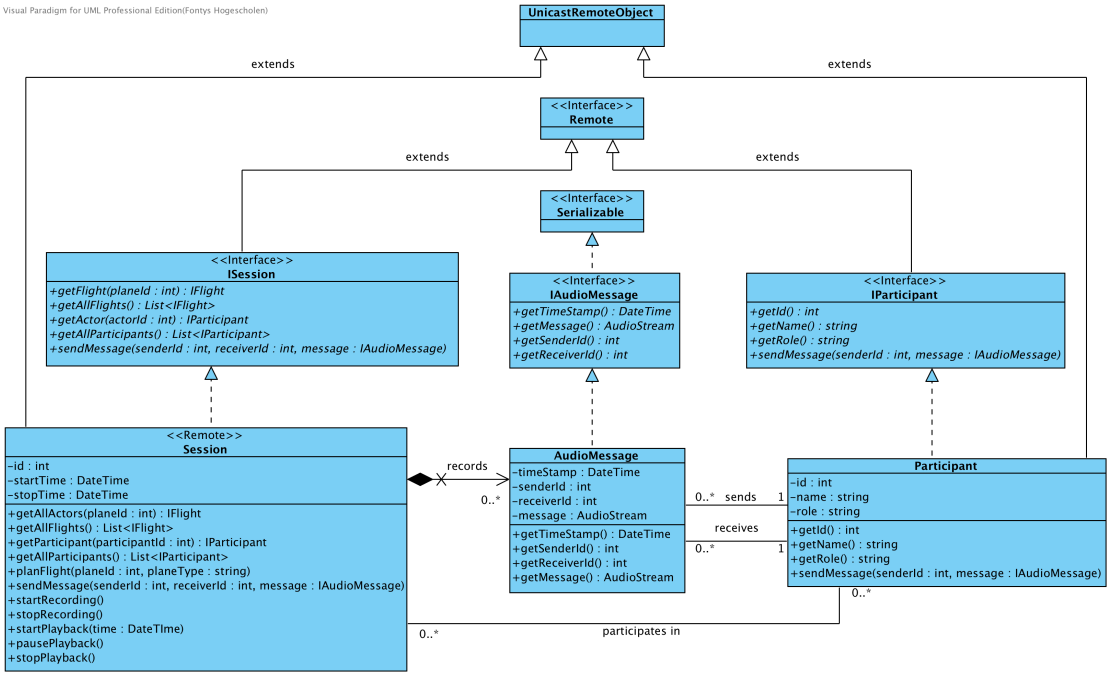
Eigenschappen van objecten m.b.t. communicatie

Tijdens een trainingssessie en tijdens de After Action Review worden de volgende eigenschappen van objecten gecommuniceerd:

- AudioMessage: id zendende actor, id ontvangende actor, timeStamp, message.
- State: timeStamp, position, orientation, velocity, acceleration.

Dataverkeer tijdens een trainingssessie en AAR

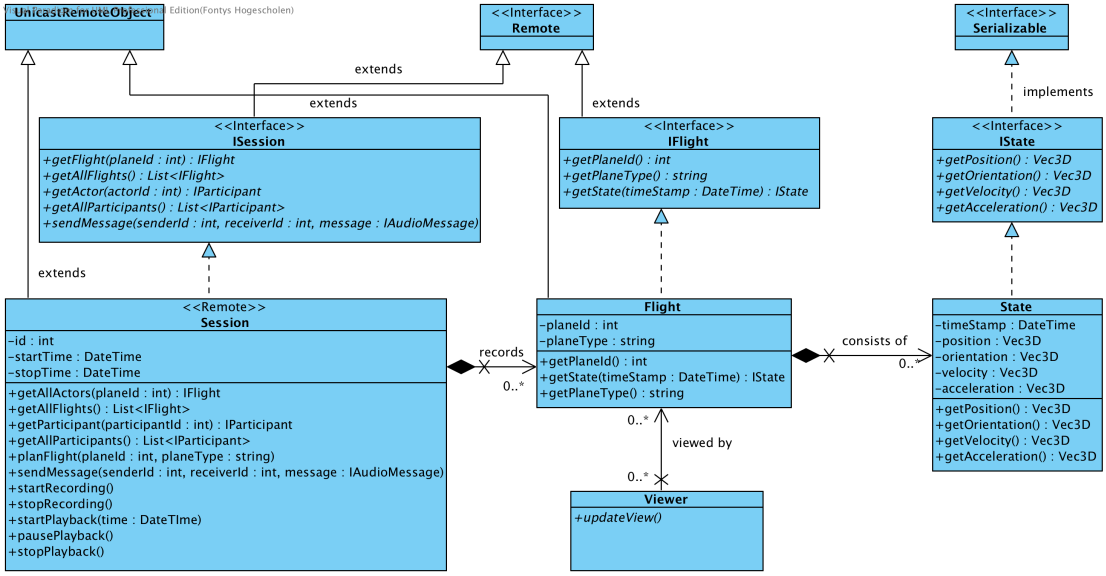
Tijdens een trainingssessie en tijdens de After Action Review worden audioverkeer en state-updates gecommuniceerd. Omdat communicatie real-time moet geschieden is gekozen voor Remote Method Invocation (RMI). Voor audioverkeer wordt het push-principe gehanteerd en voor state-updates het pull-principe. Voor RMI wordt de standaardport 1099 gebruikt.



Figuur 2: Klassendiagram voor audiocommuncatie m.b.v. RMI.

Het klassendiagram in Figuur 2 representeert de relaties tussen betrokken klassen en interfaces t.b.v. audiocommunicatie volgens het push-principe. De zendende deelnemer (instantie van klasse Participant) is hierin leidend. Na remote aanroep van de methode sendMessage van klasse Participant wordt het

audiobericht geserialiseerd en verstuurd naar de betreffende deelnemer. Na aanroep van methode sendMessage() van klasse Session wordt het audiobericht geserialiseerd en verstuurd naar Session (remote object). Session slaat het bericht vervolgens op zoals beschreven in Hoofdstuk 3.



Figuur 3: Klassendiagram voor communicatie van state-updates m.b.v. RMI.

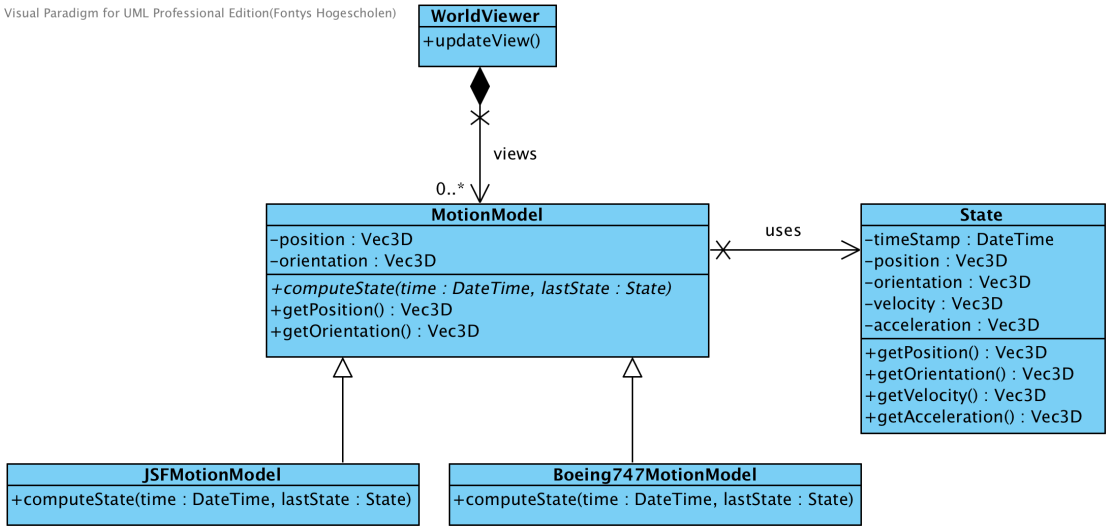
Het klassendiagram in Figuur 3 representeert de relaties tussen betrokken klassen en interfaces voor communicatie van state-updates volgens het pull-principe. Hierin is de Viewer leidend. Steeds als een Viewer het scherm refresht, wordt de state van alle relevante vliegtuigen opgevraagd door remote aanroep van de methode getState() van klasse Flight. Het huidige (simulatie)tijdstip wordt meegegeven als parameter. Vervolgens wordt de state geserialiseerd en verzonden naar de betreffende Viewer.

H5 Realisatie niet-functionele eisen

In dit hoofdstuk wordt de realisatie van (overige) niet-functionele eisen besproken. De volgende onderwerpen dienen tenminste te worden besproken: betrouwbaarheid, performance, beveiliging, schaalbaarheid. Eventueel kunnen nog extra niet-functionele eisen worden besproken.

Betrouwbaarheid

Tijdens een trainingssessie en tijdens de After Action Review dienen positie en oriëntatie van alle deelnemende toestellen zo nauwkeurig mogelijk te worden afgebeeld door alle deelnemende applicaties (werkstations, gesimuleerde radarposten en man-in-the-loop simulatoren, URS Q.1). Met name voor man-in-the-loop simulatoren is hoge nauwkeurigheid vereist (klasse WorldViewer, URS Q.2). Door gebruik te maken van zogenaamde bewegingsmodellen kunnen positie en oriëntatie nauwkeurig worden berekend op basis van de laatst bekende status (onder de aanname dat de versnelling onveranderd is gebleven). Indien de laatste status-update ouder is dan 5 seconden zal het betreffende vliegtuig niet worden afgebeeld, omdat positie en oriëntatie dan niet meer betrouwbaar kunnen worden weergegeven.



Figuur 4: Klassendiagram voor bewegingsmodel.

Het klassendiagram in Figuur 4 representeert de relatie tussen de klassen WorldViewer, MotionModel en State. Op basis van de laatst bekende state kan de huidige state nauwkeurig worden bepaald aan de hand van het betreffende bewegingsmodel. Dit geschiedt onder de aanname dat versnelling onveranderd is gebleven gedurende het tijdstip van de laatst bekende state en het huidige tijdstip. Daarnaast is van belang dat de (simulatie)tijd van alle deelnemers gesynchroniseerd is.

Aangezien bewegingsmodellen voor een straaljager en een verkeersvliegtuig verschillend zijn, worden er specifieke bewegingsmodellen voor de JSF en voor de Boeing 747 gerealiseerd. De bewegingsmodellen worden gemaakt op basis van gepubliceerde modellen in vaktijdschriften [under construction].

Performance
Status update van deelnemende vliegtuigen gebeurt volgens het RMI protocol (pull principe). Hiervoor worden objecten (instanties van klasse Status) geserialiseerd en via het internet verstuurd. De performance is sterk gerelateerd aan de zendtijd, die ook weer afhankelijk is van het onderliggende netwerk en overig dataverkeer, waaronder de audio-communicatie.

De WorldViewer heeft een refresh-rate van 20 ms. Positie en oriëntatie van vliegtuigen die nabij zijn dienen met een hogere nauwkeurigheid te worden afgebeeld dan de positie en oriëntatie van vliegtuigen die verder weg zijn. Toepassing van bewegingsmodellen zoals hierboven beschreven maakt het mogelijk om het aantal RMIs t.b.v. state-update te beperken. Voor vliegtuigen met een afstand tot 1000 m wordt 10 keer per seconde een state-update uitgevoerd. Voor de overige vliegtuigen gebeurt dit 1 keer per seconde. Ditzelfde principe wordt ook toegepast bij de RadarViewer (representeert radarscherm) en de OverallViewer (representeert totaaloverzicht). Toepassing van dit principe reduceert de benodigde bandbreedte. De daadwerkelijk benodigde bandbreedte wordt nog onderzocht [under construction].

Beveiliging

Beveiliging van internetverkeer voorafgaand aan, tijdens en na afloop van een trainingssessie is van ondergeschikt belang. Er worden geen speciale maatregelen genomen om het onderscheppen van berichten door derden te voorkomen.

Schaalbaarheid

Het gedistribueerd trainingscentrum zal in eerste instantie bestaan uit 7 onderdelen (3 bemande vliegtuigsimulatoren, 2 gesimuleerde radarposten en 2 werkstations) die onderling (point-to-point) met elkaar verbonden zijn via het internet. Door gebruik te maken van RMI en bewegingsmodellen zoals hierboven beschreven is het systeem uit te breiden met nieuwe (vliegtuig)simulatoren en andere werkstations tot een maximum van 20 deelnemers (URS Q.5).

H6 Componenten

In dit hoofdstuk wordt de opdeling in software componenten besproken aan de hand van een Componentendiagram met toelichting. Daarnaast wordt koppeling en synchronisatie tussen de componenten besproken en de opdeling in software packages.

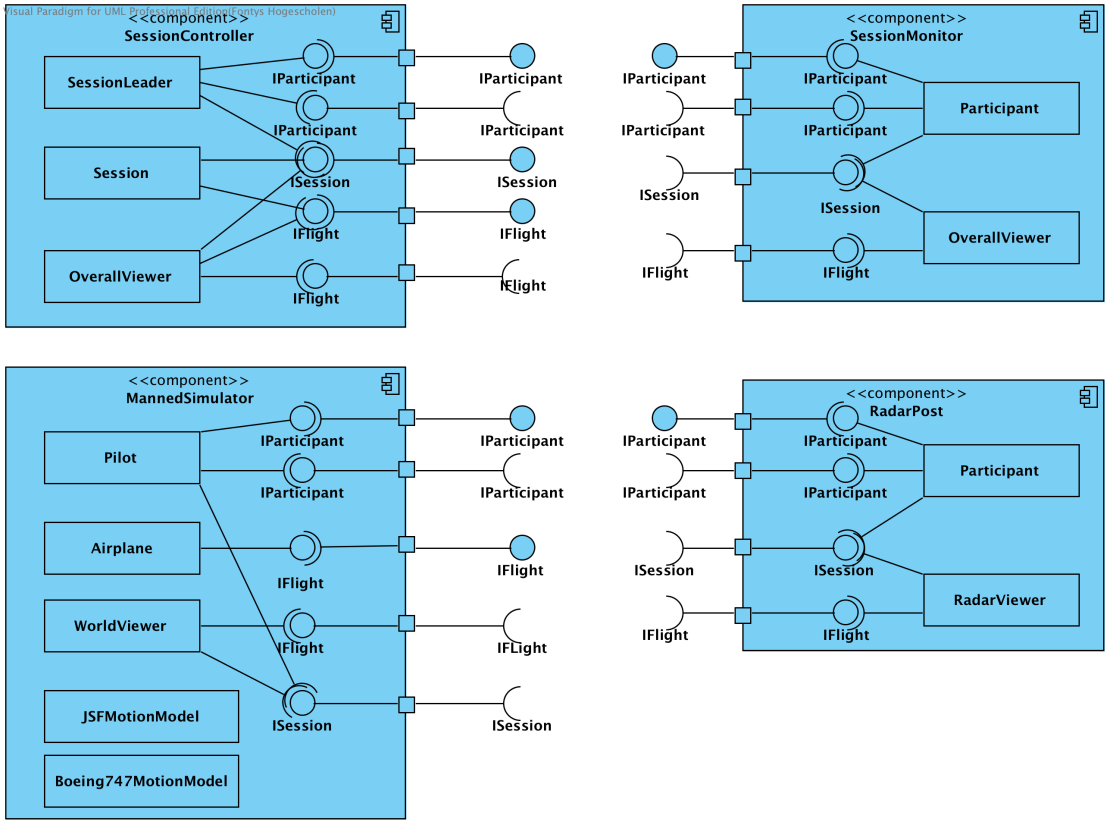
Een component is een modulair deel van het systeem dat beschreven wordt in termen van aangeboden en gevraagde interfaces. In principe kan een component vervangen worden door een andere component, mits die andere component aan dezelfde interfaces voldoet. Er wordt onderscheid gemaakt tussen componenten en subsystemen: een component is altijd een executeerbare eenheid, maar een subsysteem is geen executeerbare eenheid. Tip: kies voor iedere component een naam die duidelijk verschilt van de namen van de klassen die in die component zitten.

De volgende onderwerpen worden in dit hoofdstuk besproken:

- *Componentendiagram (opdeling componenten)*
- *Soort koppeling (RMI, HTTP, SQL, …)*
- *Synchronisatie (zie ook persistentie/communicatie)*
- *Service(s) per component (voor welke component interessant; aanduiding van behoeften (naam methode) per interface)*
- *Allocatie objecten (binnen welke component)*
- *Remote objecten (welke objecten worden op afstand aangesproken)*
- *Packagestructuur*

Er zijn verschillende manieren om componentdiagrammen te tekenen, zelfs binnen UML. Hier is gekozen voor componenten opgebouwd uit klassen en interfaces. Klassen kunnen interfaces realiseren of hebben behoefte aan interfaces. Deze interfaces kunnen binnen een component met elkaar verbonden worden en/of via een zogenaamde poort beschikbaar worden gesteld aan andere componenten. Een poort (engels: port) wordt weergegeven door middel van een vierkantje op de rand van de component. Een poort kan een interface aanbieden (aangeboden poort) of behoefte hebben aan een interface (benodigde poort). Een poort die zowel een interface aanbiedt als een interface nodig heeft noemen we een complexe poort. Meestal wordt een poort gerealiseerd door middel van een object dat de aanvraag van services doorgeeft naar een ander object binnen de component. Tijdens uitvoering van het programma zal een benodigde poort een verbinding leggen met een aangeboden poort van een andere component. Voor meer informatie over componentendiagrammen, zie Hoofdstuk 14 van Praktisch UML van Jos Warmer en Anneke Kleppe (vijfde editie, uitgever Pearson).

Componentendiagram



Figuur 5: Componentendiagram.

Het componentendiagram in Figuur 5 definieert 4 componenten met klassen en interfaces. De volgende componenten worden onderscheiden:

- SessionController. Deze component representeert de software applicatie voor de trainingsleider met klassen SessionLeader, Session en OverallViewer.
- SessionMonitor. Deze component representeert de software applicatie voor de NCT met klassen Participant (er is geen aparte subklasse voor de NCT) en OverallViewer.
- MannedSimulator. Deze component representeert de software applicatie voor een piloot (zowel JSF als Boeing 747) met klassen Pilot, Airplane, WorldViewer, JSFMotionModel en Boeing747MotionModel.
- RadarPost. Deze component representeert de software applicatie voor een verkeersleider met klassen Participant (er is geen aparte subklasse voor verkeersleider) en RadarViewer.

Koppeling tussen componenten

Voor onderlinge real-time communicatie wordt Remote Method Invocation (RMI) gebruikt (zie Hoofdstuk 4). De koppeling tussen componenten wordt gerealiseerd middels drie interfaces, te weten ISession, IParticipant en IFlight. Omdat alle componenten informatie nodig hebben van alle andere componenten zijn alle aangeboden interfaces verbonden met alle bijbehorende benodigde

interfaces. Om de figuur overzichtelijk te houden zijn deze verbindingen niet opgenomen in Figuur 5.

Synchronisatie tussen componenten

Ten behoeve van nauwkeurige weergave door de WorldViewer met behulp van bewegingsmodellen (zie Hoofdstuk 5) is tijdsynchronisatie tussen de componenten van groot belang (URS Q.1 en URS Q.2). Daarom wordt aan het begin van een trainingssessie of AAR een protocol uitgevoerd om de systeemklokken gelijk te zetten [under construction].

Services per component

Services worden gerealiseerd middels drie interfaces, te weten ISession, IParticipant en IFlight:

ISession:
- Vraag referentie naar een specifieke deelnemer.
- Vraag lijst referenties naar alle deelnemers.
- Vraag referentie naar de vlucht van een specifiek vliegtuig.
- Vraag referenties naar alle vluchten.
- Verstuur een audiobericht t.b.v. opslag.

IParticipant
- Vraag id van de betreffende deelnemer.
- Vraag naam van de betreffende deelnemer.
- Vraag rol van de betreffende deelnemer.
- Verstuur audiobericht naar de betreffende deelnemer.

IFlight
- Vraag id van het betreffende vliegtuig.
- Vraag type van het betreffende vliegtuig.
- Vraag status van het betreffende vliegtuig op bepaald tijdstip.

De volgende services worden per component aangeboden:
- SessionController: ISession, IParticipant (trainingsleider), IFlight (computergestuurde vliegtuigen).
- SessionMonitor: IParticipant (hoofd NCT).
- MannedSimulator: IParticipant (piloot), IFlight (bemand vliegtuig).
- RadarPost: IParticipant (verkeersleider).

De volgende services worden per component gevraagd:
- SessionController: IParticipant (overige deelnemers), IFlight (bemande vliegtuigen).
- SessionMonitor: ISession (sessiegegevens), IParticipant (overige deelnemers), IFlight (alle vliegtuigen).
- MannedSimulator: ISession (sessiegegevens), IParticipant (overige deelnemers), IFlight (overige vliegtuigen).

- RadarPost: ISession (sessiegegevens), IParticipant (overige deelnemers), IFlight (alle vliegtuigen).

Voor de specificatie van de interfaces wordt verwezen naar Hoofdstuk 8.

Allocatie van objecten

Allocatie van objecten per component:
- SessionController: instanties van SessionLeader, Session en OverallViewer.
- SessionMonitor: instanties van Participant (met rol NCT) en OverallViewer.
- MannedSimulator: instanties van Pilot, Airplane, WorldViewer, JSFMotionModel en Boeing747MotionModel.
- RadarPost: instanties van Participant (met rol verkeersleider) en RadarViewer.

Remote objecten

Remote objecten per component:
- SessionController: remote instanties van Participant en Flight.
- SessionMonitor: remote instanties van Session, Participant en Flight.
- MannedSimulator: remote instanties van Session, Participant en Flight.
- RadarPost: remote instanties van Session, Participant en Flight.

Packagestructuur

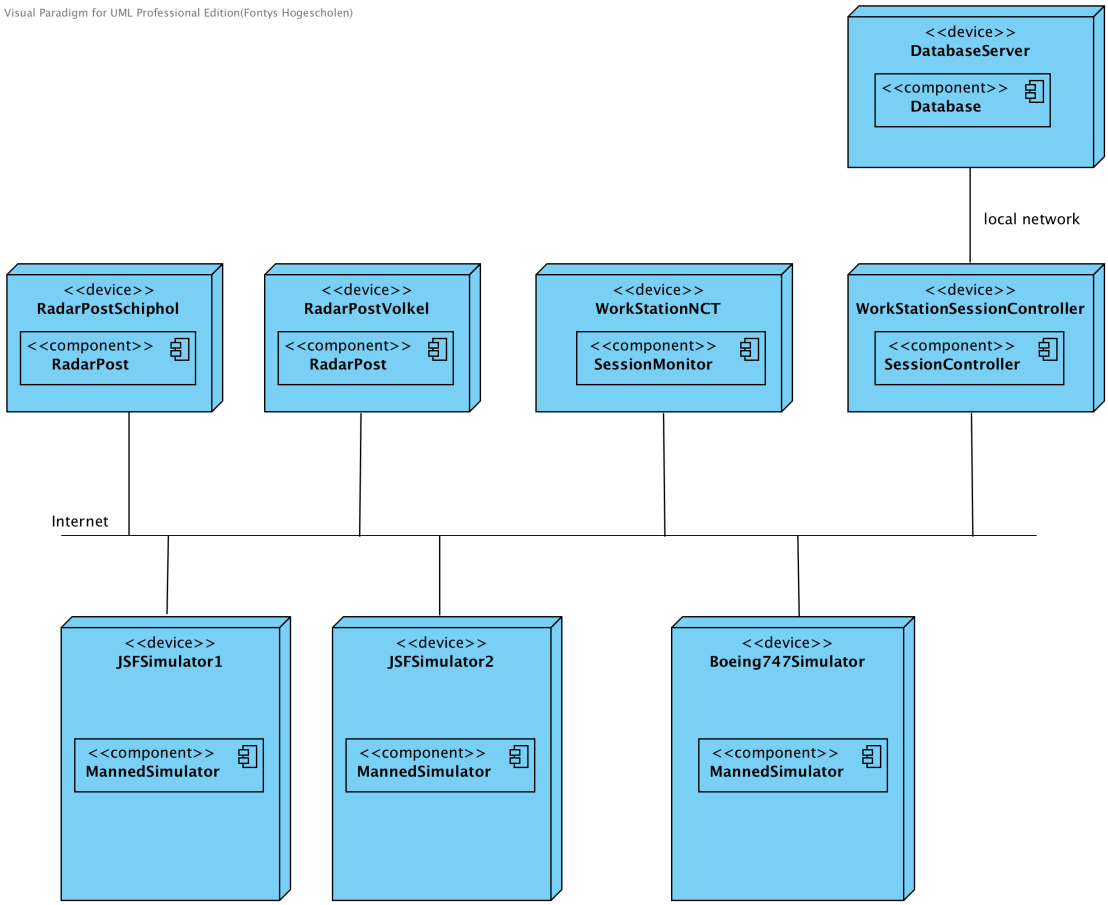
Er is een package Generic met algemene klassen voor alle componenten en een package Interfaces met interfaces voor alle componenten. Omdat de componenten SessionController en SessionMonitor beide gebruikmaken van de klasse OverallViewer is er een apart package SessionViewer. Voor de componenten SessionController, MannedSimulator en RadarPost is er een package met specifieke klassen voor de betreffende componenten.

Klassen/interfaces per package:
- Package Generic: Participant, Viewer, AudioMessage, Flight, State.
- Package Interfaces: ISession, IParticipant, IAudioMessage, IFlight, IState.
- Package SessionController: SessionLeader en Session.
- Package SessionViewer: OverallViewer.
- Package MannedSimulator: Pilot, Airplane, WorldViewer, JSFMotionModel en Boeing747MotionModel.
- Package RadarPost: RadarViewer.

H7 Deployment

In dit hoofdstuk wordt de toekenning van software componenten aan hardware besproken aan de hand van een Deploymentdiagram met toelichting. Daarnaast wordt de koppeling (lokaal netwerk, internet, etc.) tussen de nodes besproken.

Deployentdiagram



Figuur 6: Deploymentdiagram.

Deployment van de gedistribueerde trainingsfaciliteit is afgebeeld in Figuur 6 middels een deploymentdiagram. De trainingsfaciliteit bestaat uit de volgende onderdelen:

- RadarPostSchiphol: gesimuleerd radarstation voor de verkeersleider van Schiphol (locatie: Schiphol).
- RadarPostVolkel: gesimuleerd radarstation voor de verkeersleider van Volkel (locatie: Volkel).
- WorkStationNCT: werkstation voor de NCT (locatie: Den Haag).
- WorkStationSessionController: werkstation voor de trainingsleider (locatie: Den Haag).

- DatabaseServer: Database server voor database met gegevens van de deelnemers. Deze is via een lokaal netwerk bereikbaar vanuit WorkStationSessionController.
- JSFSimulator1: eerste bemande JSF simulator (locatie: Volkel).
- JSFSimulator2: tweede bemande JSF simulator (locatie: Volkel).
- Boeing747Simulator: bemande Boeing 747 simulator (locatie: Schiphol).

H8 Specificatie van interfaces

In dit hoofdstuk wordt de specificatie van interfaces besproken. Voor ieder interface wordt per methode gedefinieerd:

- *Naam methode*
- *Naam en type argumenten*
- *Precondities*
- *Type returnwaarde*
- *Beschrijving*
- *Aanleiding voor excepties*

Interface ISession

Methode:	getFlight(planeId : int) : IFlight
Preconditie:	lijst van (remote) instanties van Flight bestaat nrPlanes is lengte van deze lijst 0 <= planeId < nrPlanes
Beschrijving:	geeft referentie naar (remote) instantie van Flight met overeenkomend planeId.
Excepties:	NotInitializedException als lijst niet bestaat. InvalidParameterException als planeId < 0 of planeId >= nrPlanes

Methode:	getAllFlights() : List<IFlight>
Preconditie:	lijst van (remote) instanties van Flight bestaat
Beschrijving:	geeft referentie naar lijst van (remote) instanties van Flight.
Excepties:	NotInitializedException als lijst niet bestaat.

Methode:	getParticipant(participantId : int) : IParticipant
Preconditie:	lijst van (remote) instanties van Participant bestaat nrParticipants is lengte van deze lijst 0 <= participantId < nrParticipants
Beschrijving:	geeft referentie naar (remote) instantie van Flight met Overeenkomend participantId.
Excepties:	NotInitializedException als lijst niet bestaat InvalidParameterException als participantId < 0 of participantId >= nrParticipants

Methode:	getAllParticipants() : List<IParticipant>
Preconditie:	lijst van (remote) instanties van Participant bestaat
Beschrijving:	geeft referentie naar lijst van (remote) instanties van Participant.
Excepties:	NotInitializedException als lijst niet bestaat.

Methode:	sendMessage(senderId : int, receiverId : int, message : IAudioMessage)
Preconditie:	0 <= senderId < nrParticipants 0 <= receiverId < nrParticipants

Beschrijving: stuurt audiostream (geserialiseerd) naar Session object.
Excepties: InvalidParameterException als senderId < 0
of senderId >= nrParticipants
InvalidParameterException als receiverId < 0
of receiverId >= nrParticipants

Interface IParticipant
[under construction]

Interface IAudioMessage
[under construction]

Interface IFlight
[under construction]

Interface IState
[under construction]