

# Rapport 1 du projet de genlog

Grp 4

22 octobre 2008

# Table des matières

<b>1</b>	<b>Analyse</b>	<b>5</b>
<b>2</b>	<b>Résolution d'obstacles</b>	<b>6</b>
2.1	Achieve[AccurateAmbulancePositionRecordedWhenAccurateAmbulancePositionSent]	6
2.2	Achieve[AccurateAmbulancePositionSent]	7
2.3	Achieve[AmbulanceOnSceneWhenAmbulanceMobilized]	8
<b>3</b>	<b>Cahier des charges</b>	<b>10</b>
3.1	Modèle de buts	10
3.2	Modèle des objects	10
3.2.1	Diagramme des objets	10
3.2.2	Spécification des concepts	10
3.3	Modèle des agents	10
3.3.1	Diagramme de contexte	10
3.4	Modèle des opérations	10
3.4.1	processIncidentInfo	11
3.4.2	recordAccurateAmbulancePosition	11
3.4.3	choseAmbulance	11
3.4.4	sendMobilizationOrder	11
3.5	Modèle de comportement	11

3.5.1	Scénario . . . . .	11
3.5.2	Machine à état . . . . .	11

# Introduction

# Chapitre 1

## Analyse

# Chapitre 2

## Résolution d'obstacles

### 2.1 Achieve[AccurateAmbulancePositionRecordedWhenAccurateAmbulancePositionSent]

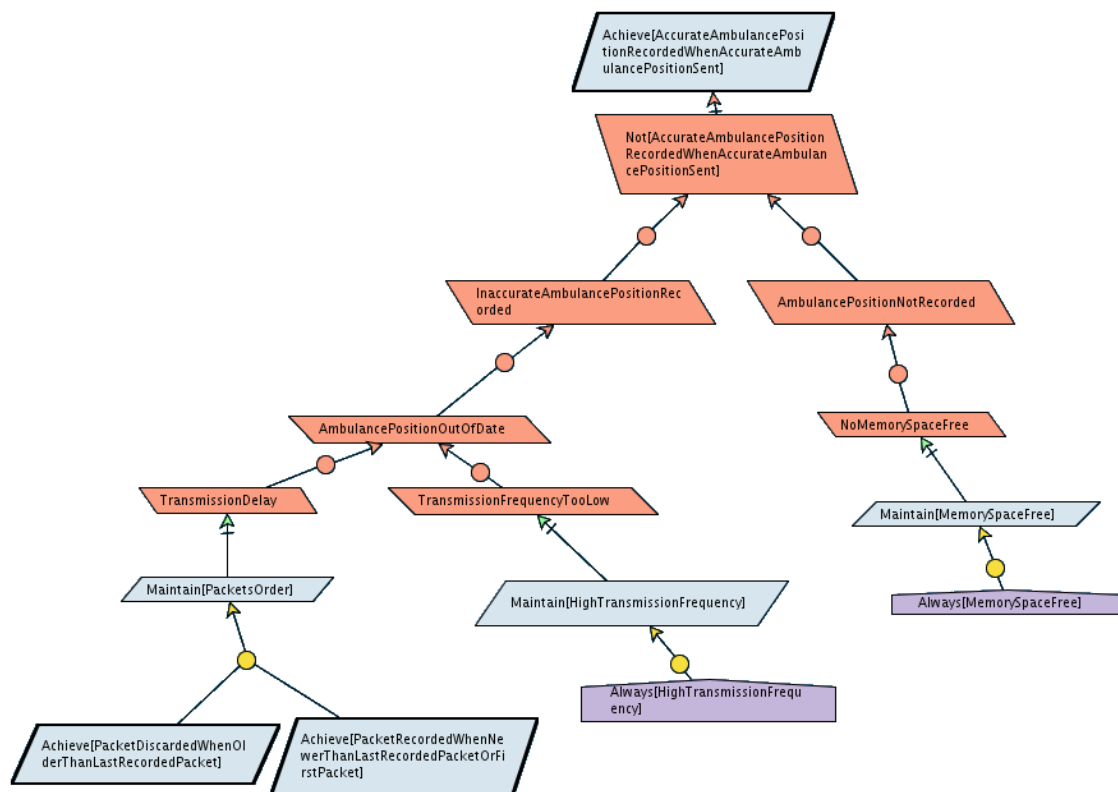


FIG. 2.1 – Diagramme de résolution d'obstacle

Le but qui nous concerne ici est d'assurer que la précision des données envoyées est garantie à la réception et que ces données puissent être enregistrées. Les obstacles

et leur résolution coulent de source.

## 2.2 Achieve[AccurateAmbulancePositionSent]

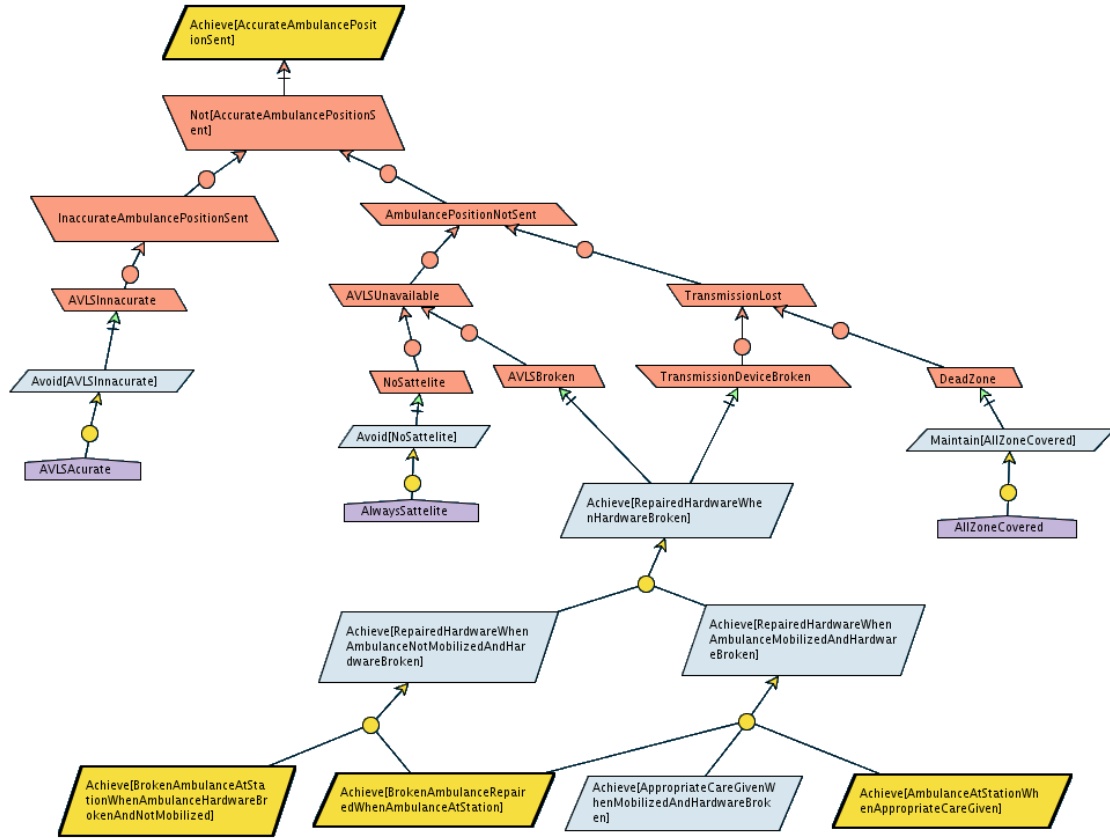


FIG. 2.2 – Diagramme de résolution d'obstacle

Ce but correspond à l'envoi des données de l'AVLS et de s'assurer que celles-ci sont correctes.

L'obstacle dont la résolution est la plus intéressante est le non fonctionnement du matériel responsable de l'exécution de ce but.

La résolution distingue deux cas, celui où elle est libre, et celui où elle est mobilisée. Dans le premier cas, on envoie directement l'ambulance à une station où elle sera réparée. Dans le second cas, l'ambulance effectue son travail et retourne ensuite à la station pour réparation.

Cette résolution est incomplète car notre modèle des buts et objet ne permet pas de considérer les cas suivant :

- Empêcher de choisir ou de mobiliser l'ambulance lorsqu'elle doit être réparée ou en réparation,
- Ne pas mobiliser une ambulance choisie si elle a eu un problème entre temps.

De plus, d'autres problèmes apparaissent ; L'état "AmbulanceOnScene" ne saura être détecté par le système informatique dans le cas d'une ambulance disfonctionnelle.

Pour résoudre ces problèmes il faut ajouter un attribut broken à l'ambulance et modifier les définitions des buts de choix et de mobilisation de l'ambulance.

De plus l'AVLS et le système de transmissions ne sont sans doute pas les seuls appareils de l'ambulance pouvant tomber en panne, le but de réparation correspond très certainement à un but de général de plus haut niveau consistant à maintenir l'ambulance en état de marche.

## 2.3 Achieve[AmbulanceOnSceneWhenAmbulanceMobilized]

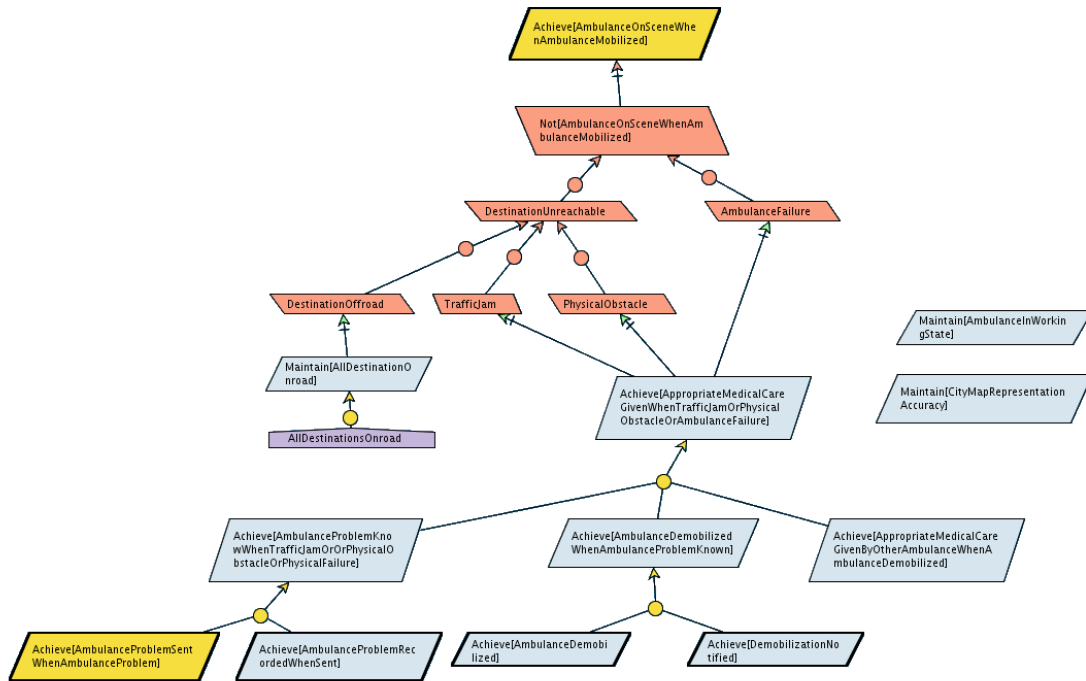


FIG. 2.3 – Diagramme de résolution d'obstacle

Ce but correspond à l'arrivée de l'ambulance sur les lieux indiqués de l'incident après sa mobilisation.

Le raffinement consistant à démobiliser une ambulance accidentée ou coincée peut sembler superflu, mais est justifié par le respect des cardinalités de notre modèle



objet.

Le but *Achieve[MedicalCareGivenByOtherAmbulanceWhenAmbulanceDemobilized]* n'est manifestement pas complet ni suffisamment raffiné.

En fait, deux alternatives s'offrent à nous. La première et la moins bonne consiste à répliquer l'entière de l'arbre des buts *Achieve[MedicalCareGivenWhenAmbulanceMobilized]* comme raffinement du précédent. L'autre alternative consiste à modifier les buts parents afin qu'ils se chargent de mobiliser l'ambulance de secours. Pour cela, il faut transformer *Achieve[AmbulanceMobilizedWhenIncidentInformationKnown]* par un *Maintain[WorkingAnbulanceMobilizedWhenIncidentInformationKnownAndNotResolved]*. Ce but s'assurera donc qu'il y a toujours une ambulance en état de marche mobilisée pour chaque incident.

Cependant cela nécessite de définir deux états supplémentaires : *Working(Ambulance)* et *Resolved(IncidentInformation)*. L'ajout de ces deux états nécessite de modifier notre modèle objet et d'ajouter de nouveaux buts et raffinements, à identifier lors d'une deuxième passe d'analyse et de réflexion sur notre modèle.

Enfin, Cette résolution n'est pas non plus complète. Il faut en effet modifier la représentation de la carte du système de routage et de choix d'ambulance pour que celui ci n'envoie pas une deuxième ambulance dans un embouteillage.

Pour finir, il faut aller rechercher l'ambulance accidentée et la réparer.

Ces deux buts sont à rattacher comme raffinements de buts plus généraux qui ne se trouvent pas dans notre modèle actuel. Cela encore révèle la nécessité d'une seconde passe d'analyse.

# Chapitre 3

## Cahier des charges

### 3.1 Modèle de buts

### 3.2 Modèle des objects

#### 3.2.1 Diagramme des objets

#### 3.2.2 Spécification des concepts

### 3.3 Modèle des agents

#### 3.3.1 Diagramme de contexte

### 3.4 Modèle des opérations

Cette section présente un ensemble d'opération. Nous avons choisi les quatres opérations correspondant à des buts feuilles et assignées à des agents logiciels à développer.

Afin d'illustrer les liens entre les modèles, nous reprenons le nom du but qui est opérationnalisé par l'opération, le nom de l'agent qui effectuera l'opération, l'état à atteindre après cette opération et enfin, l'évènement attaché à cette opération.

### 3.4.1 processIncidentInfo

Goal	Achieve[IncidentInfoProcessedWhenIncidentInfoRecorded]
Agent	InfoProcessor
Goal state	IncidentInfoProcessed
Event	IncidentInfoProcessing
In	$i : IncidentInfo$
Out	$i : IncidentInfo$
Pre	$\exists c : Call, \exists j : Incident(c.about \rightarrow j \wedge c.reporting \rightarrow i)$
Post	$i.pos! = " \wedge i.ambulanceKindNeeded! = "$ La position (pos) contenue dans i correspond à la position (localisation) de i sous forme exploitable par le système Le type d'ambulance nécessaire de i est calculé selon les règles données par le gouvernement et sur base des informations présentées dans i

TAB. 3.1 – processIncidentInfo

### 3.4.2 recordAccurateAmbulancePosition

Goal	Achieve[AccurateAmbulancePositionRecordedWhenAccurateAmbulancePositionSent]
Agent	AmbulanceTracker
Goal state	AmbulancePositionAccurate and AmbulancePositionKnown
Event	AccurateAmbulancePositionRecording
In	$a : Ambulance$
Out	$b : AmbulanceInfo$
Pre	$\exists b : ambulanceInfo : a.id = b.id$
Post	$b.pos = a.pos$

TAB. 3.2 – recordAccurateAmbulancePosition

### 3.4.3 choseAmbulance

### 3.4.4 sendMobilizationOrder

## 3.5 Modèle de comportement

### 3.5.1 Scénario

### 3.5.2 Machine à état

Goal	Achieve[AmbulanceChosenWhen AvailabilityKnownAnd AmbulanceKindKnownAnd AccurateAmbulancePositionKnown]
Agent	InfoProcessor
Goal state	AmbulanceChosen
Event	AmbulanceChoice
In	$i : IncidentInfo$
Out	$a : AmbulanceInfo$
Pre	$\exists a : AmbulanceInfo : \#a.mobilisation = 0 \wedge \#a.choice = 0$
Post	$\#a.choice = 1 \wedge \#i.choice = 1 \wedge i.choice \rightarrow a$

TAB. 3.3 – choseAmbulance

Goal	Achieve[MobilizationOrderSentWhenBestAmbulanceChosen]
Agent	InfoProcessor
Goal state	MobilizationOrderTransmitted
Event	MobilizationOrderTransmittion
In	$a : Ambulance, i : Incident$
Out	$m : MobilizationOrder$
Pre	$\#a.choice = 1 \wedge a.choice \rightarrow i$
Post	$\exists m : MobilizationOrder : m.ambulance = a \wedge m.incident = i$

TAB. 3.4 – sendMobilizationOrder

# Conclusion