Rapport 1 du projet de genlog

Grp 4

24 octobre 2008

Table des matières

Introduction

Chapitre 1

Analyse

Chapitre 2

Résolution d'obstacles

2.1 Achieve[AccurateAmbulancePositionRecorded When AccurateAmbulancePositionSent]

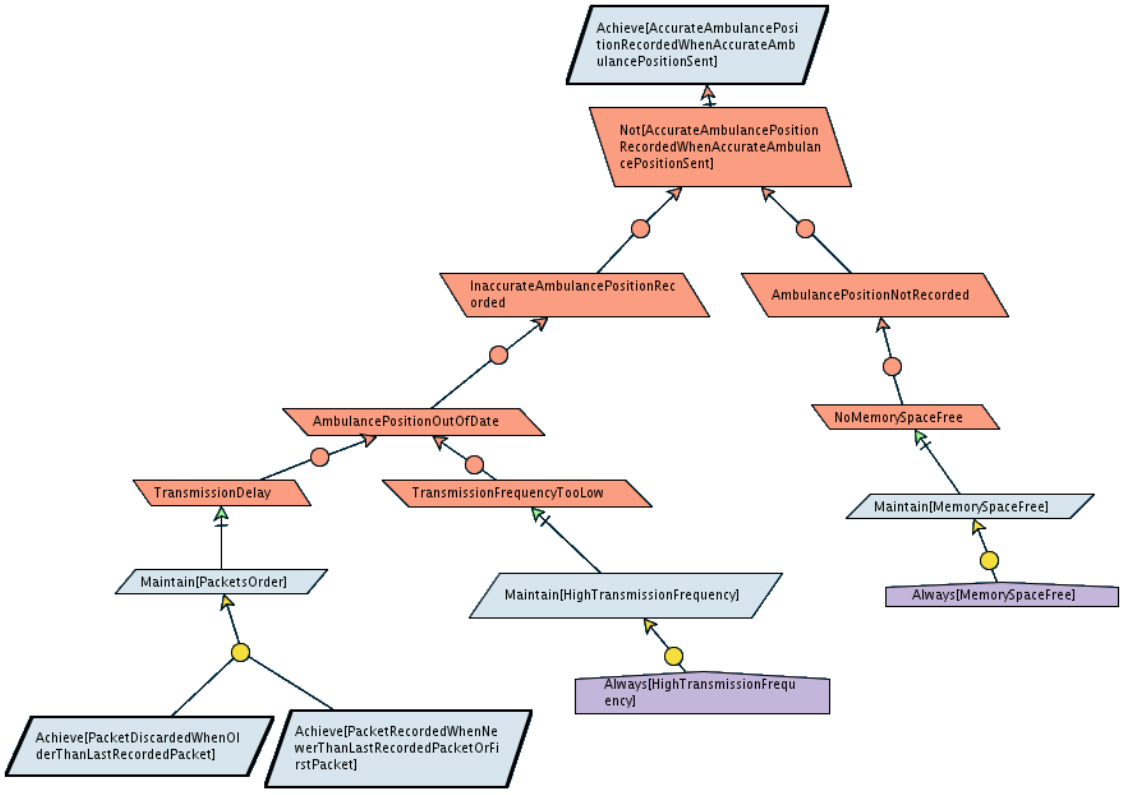

Fig. 2.1 – Diagramme de résolution d'obstacle

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]

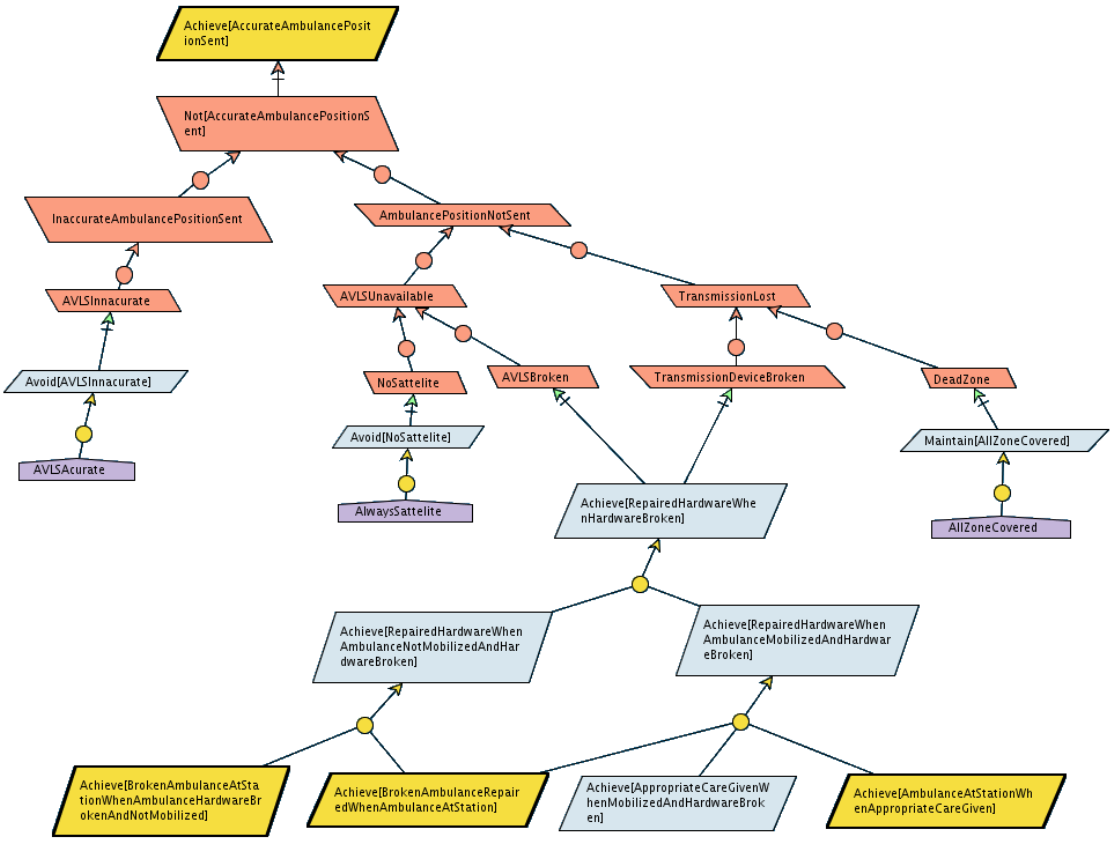

FIG. 2.2 – Diagramme de résolution d'obstacle

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 l'ambulance 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 gérer les objectifs suivants :

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

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

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 transmission 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[AmbulanceOnScene When AmbulanceMobilized]

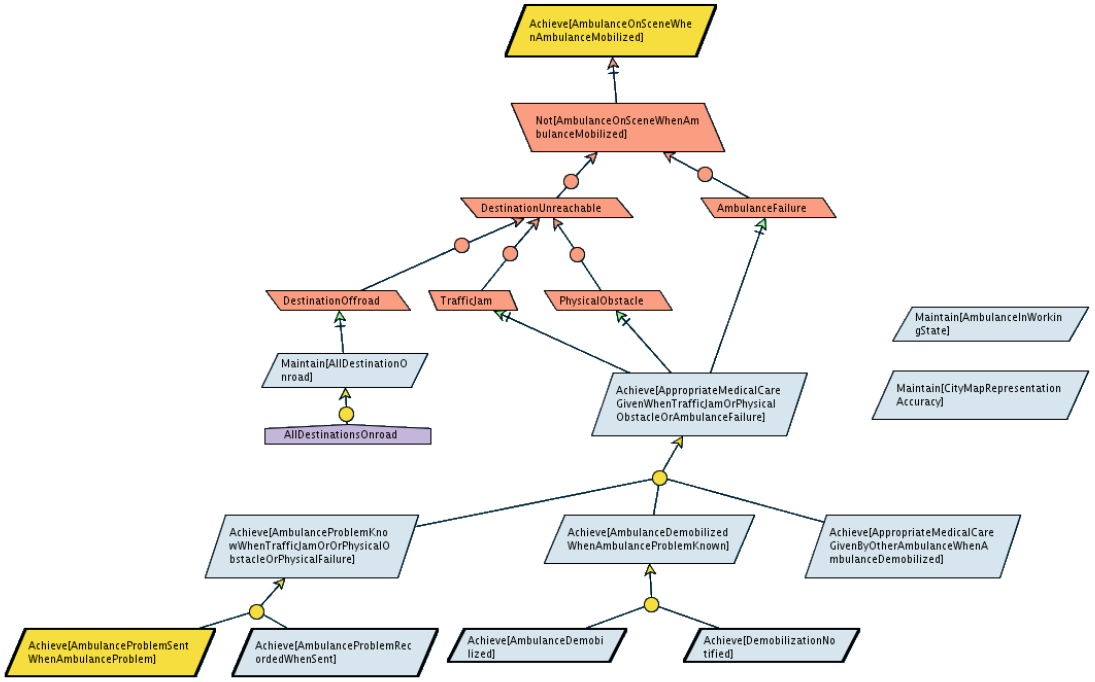

FIG. 2.3 – Diagramme de résolution d'obstacle

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[MedicalCareGivenByOtherAmbulance When AmbulanceDemobilized]* n'est manifestement pas complet ni suffisemment raffiné.

En fait, deux alternatives s'offrent à nous. La première et la moins bonne consiste à répliquer l'entièreté de l'arbre des buts *Achieve[MedicalCareGiven When AmbulanceMobilized* 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[AmbulanceMobilized When IncidentInformationKnown]* par un *Maintain[WorkingAnbulanceMobilized When IncidentInformationKnownAndNotResolved]* Ce but s'assurera donc qu'il y a toujours une ambulance en état de marche mobilisée pour chaque incident.

Cependant cela nécéssite de définir deux états supplémentaires : *Working(a :Ambulance)* et *Resolved(i :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 modêle.

Enfin, Cette résolution n'est pas non plus complète. Il faut en effet modifier la représentation de la carte du systême 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 révèle une fois de plus la nécessité d'une seconde passe d'analyse.

Chapitre 3

Cahier des charges

3.1 Modèle de buts

Dans cette partie, nous vous présentons une partie du modèle de buts. En particulier, nous vous présentons la formalisation que nous avons utilisé pour découler de manière rapide et cohérente les opérations, les machines à états et augmenter la cohérence de notre modèle de buts.

$$\begin{aligned} &AmbulanceFree(a : Ambulance) \\ &= \neg AmbulanceMobilized(a) \\ &\quad \wedge \neg AmbulanceChosen(a) \end{aligned}$$

$$\begin{aligned} &AmbulanceMobilized(a : Ambulance) \\ &= \exists i : Incident, \\ &\quad MobilizationOrderConfirmed(a, i) \end{aligned}$$

$$\begin{aligned} &AmbulanceOnScene(a : Ambulance, I : Incident) \\ &= a.pos = i.pos \end{aligned}$$

$$\begin{aligned} &AmbulanceChosen(i : IncidentInfo) \\ &= \exists a : AmbulanceInfo, a.id = b.id \\ &\quad \wedge a.choice = 1 \\ &\quad \wedge \#i.choice = 1 \\ &\quad \wedge a.choice \rightarrow i \end{aligned}$$

$$\begin{aligned} &MobilizationOrderConfirmed(a : Ambulance, i : Incident) \\ &= \exists m : MoblisationOrder : \\ &\quad m.ambulance = a \wedge m.incident = 1 \\ &\quad \wedge m.confirmed = True \end{aligned}$$

$MobilizationOrderTransmitted(a : Ambulance, i : Incident)$
$= \exists m : MobilizationOrder,$
$m.ambulanceId = a$
$\wedge m.incident = 1$

$MedicalCareGiven(i : Incident)$
$= \exists AmbulanceMobilized(a, i),$
$\wedge AmbulanceOnScene(a, i)$
$\wedge \#a.medicalCare = 1$
$\wedge \#i.victim.medicalCare = 1$
$\wedge a.medicalCare \rightarrow i.victim.medicalCare$

$IncidentInfoKnown(i : IncidentInfo)$
$= IncidentInfoProcessed(i)$

$CallReceived(i : Incident)$
$= \exists c : Call,$
$c.about \rightarrow i$
$\wedge \#c.about = 1$
$\wedge \#i.about = 1$

$AvailabilityKnown(a : Ambulance)$
$= \exists b : AmbulanceInfo,$
$b.id = a.id$
$\wedge \#b.mobilized = \#a.miblized$

$PositionKnown(a : Ambulance)$
$= \exists b : AmbulanceInfo,$
$b.id = a.id$
$\wedge a.pos \cong b.pos$

$IncidentInfoProcessed(j : IncidentInfo)$
$= j.pos \neq null$
$\wedge j.ambulanceKindNeeded \neq null$
$\exists c : Call, \exists i : Incident,$
$(c.about \rightarrow i \wedge c.reporting \rightarrow j)$

$IncidentInfoRecorded(c : Call)$
$= \exists j : IncidentInfo, \exists Incident,$
$c.about \rightarrow i$
$\wedge j.reporting \rightarrow c$
$\wedge j.localisation \neq nul$
$\wedge j.description \neq null$
$\wedge j.victimAge \neq null$
$\wedge j.victimPregrant \neq null$
$\wedge j.id \neq null$

$PositionAccurate(a : Ambulance)$
$= \exists b : AmbulanceInfo,$
$a.id = b.id$
$\wedge b.pos = a.pos$

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 exploitabular par le système Le type d'ambulance nécessaire de i est calculé selon les règles données par le gouvernement et sur base des informations présentent dans i

TAB. 3.1 – processIncidentInfo

Goal	Achieve[AccurateAmbulancePositionRecorded WhenAccurateAmbulancePositionSent]
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

3.4.2 recordAccurateAmbulancePosition

3.4.3 choseAmbulance

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

3.4.4 sendMobilizationOrder

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

3.5 Modèle de comportement

3.5.1 Scénario

3.5.2 Machine à état

Conclusion

Chapitre 4

Annexe : Évaluation d'Objectiver

Objectiver nous a été fort utile durant cette première partie du projet, cependant quelques petites améliorations pourraient rendre son utilisation nettement plus efficace.

- Le format de sauvegarde en .xml met tout le fichier sur une seule ligne, S'il état correctement indenté, des outils tels que svn / git pourraient gérer automatiquement les conflits de version qui arrivent régulièrement lorsqu'on travaille à 8 sur un même fichier.
- Un "Search and Replace" dans le noms des buts et leurs définitions nous aurait fait gagner un temps précieux.
- La manipulation des graphes pourrait être nettement améliorée, par exemple s'il était possible de déplacer un noeud parent et tous ses enfants en même temps, ou de "minimiser" un noeud et ses enfants.
- Il serait pratique de pouvoir disposer de plusieurs instances d'un même but/objet/obstacle dans les diagrammes, cela permettrait de les arranger plus clairement.
- Une vraie version linux ne serait pas de refus.