MASTER EN SCIENCES INFORMATIQUES

Modélisations de systèmes coopératifs mobiles à temps réels: évaluation du processus de développement ROPES: analyse de cas pour des systèmes ATC (Air Traffic Control) (Air Traffic Control)

Honet, Jean-François

Award date:
2004

Modélisations de systèmes
coopératifs mobiles à temps réels
Analyse de cas pour des systèmes ATC
(Air Traffic Control)
Evaluation du processus de développement ROPES

Résumé

L'objectif de ce travail est d'éprouver la méthodologie de développement de logiciel ROPES en l'appliquant à un cas d'analyse précis : la modélisation d'un système coopératif embarqué temps réel de gestion de trafic au sol dans un aéroport. En appliquant les principes préconisés par Bruce Powel Douglass dans sa méthodologie, nous avons voulu repérer les points forts et les faiblesses d'un tel processus utilisant exclusivement UML comme langage de spécification et d'analyse. Pour les manques que nous avons repérés, nous proposons des pistes de réflexions qui pourraient amener de futures équipes de développement à utiliser cette méthodologie légèrement adaptée. Ce travail s'attache essentiellement à utiliser celle-ci pour la phase d'analyse des besoins et de l'étude fonctionnelle du système à développer.

Mots clés

UML, ROPES, A-SMGCS, Air Traffic Control, Analyse des Besoins, Temps réel, Systèmes Embarqués, Systèmes coopératifs, Aéroport

Abstract

Improving the ROPES methodology, this is the principle objective of this paper. We will use this software development process, suitable for real-time embedded systems, for a case study : the conception of a real-time & cooperative embedded Airport ground traffic management system. By applying the ROPES' principles, we would like to pick out the strong points and the weaknesses of such a process using exclusively UML as a specification and analysing language. For the lacks which we located, we will propose tracks of thoughts which could lead future development teams to use this slightly adapted ROPES methodology. This work primarily sticks to use this methodology for the early requirements analysis phase and the functional study of the system to be developed.

Key words

UML, ROPES, A-SMGCS, Air Traffic Control, Requirements analysis, Real Time, Embedded systems, Cooperative system, Airport

Remerciements

Anne De Vos est sans doute la personne qui doit être remerciée le plus chaleureusement. Sans elle, jamais, nous n'aurions pu réussir à aboutir au terme de ce mémoire et de ces longues années d'étude. Elle m'a encouragé, soutenu, poussé pour que finalement je puisse clore cette formation par ce travail.

Nous tenons aussi à remercier Vincent Englebert sans les conseils duquel nous nous serions souvent égarés sur des pistes obscures.

Merci à Laurent Honet qui nous a permis d'explorer une problématique inconnue et qui nous a encouragé par son expérience professionnelle et sa patience.

Merci aussi à tous nos compagnons de LIHD avec qui nous avons formé une bande où solidarité et soutien ont été plus que de simples mots. Merci donc à Jean-Marc, Laurent, Jean-Luc, Christine, Marie-Paule, Jacques, Thierry et tous les autres étudiants, professeurs, secrétaires ...

Nous tenons à remercier également toutes les autres personnes qui nous ont aidé en relisant ce travail, en proposant des solutions, en nous soutenant le moral à des moments de découragements, et particulièrement Claire De Vos d'avoir bien voulu se plonger dans un travail dont le domaine ne lui est pas particulièrement familier et de le corriger.

Enfin, nous ne pouvons pas finir ce mémoire sans remercier Léo qui est venu au monde au moment où naissait aussi ce travail de fin d'étude.

Table des matières

1 Introduction générale

Concevoir un système coopératif temps réel d'aide à la gestion du trafic au sol dans un aéroport, tel pourrait être l'objectif ambitieux de ce mémoire. De la conception de ce système, nous nous contenterons d'éprouver un processus de développement conçu pour de tels projets : ROPES. En effet, nous avons voulu montrer les forces et les faiblesses d'une méthodologie qui utilise exclusivement UML pour spécifier les besoins, les contraintes et les exigences des systèmes temps réels embarqués. Langage de modélisation particulièrement à la mode pour les développements de systèmes d'information de gestion, UML s'est largement répandu grâce à l'engouement de la programmation orientée objet et spécialement du langage Java. ROPES est une méthodologie de développement proposée par un des auteurs actuellement les plus en vue dans le domaine de la conception de systèmes embarqués temps réels, Bruce Powel Douglass.

Le secteur de l'aéronautique civile, que nous allons étudier, est aujourd'hui en plein essor. La tendance lourde actuelle est à l'explosion du transport aérien. Aussi, les problématiques liées à la sécurité, au contrôle du trafic aérien, et à une optimalisation de ce trafic deviennent cruciales tant pour les compagnies privées que pour les organismes régulateurs de ce trafic. Dans cette large problématique, nous étudierons celle de la gestion du trafic des véhicules terrestres au sein des aéroports et de l'échange de données entre tous les acteurs du domaine. Ce champs relativement large servira de prétexte à une mise en perspective de la méthodologie ROPES. Notre approche est donc davantage d'ordre méthodologique que technique. Au terme de ce mémoire, nous ne proposerons pas un système clé sur porte permettant de contrôler les mouvements de véhicules dans un aéroport, mais nous espérons pouvoir aider les futurs intervenants d'un tel projet à adopter une méthodologie de développement la mieux adaptée.

Ce mémoire s'ouvrira sur une présentation succincte du milieu de l'aéronautique civile. Sans être exhaustif, cet aperçu permettra au lecteur d'avoir une vision globale de la problématique, des acteurs et des contraintes propres au domaine de l'aviation civile. Nous présenterons ensuite un historique de la surveillance au sol et des technologies successivement utilisées dans les aéroports. Nous expliquerons enfin les notions du A-SMGCS pour Advanced Surface Movement Guidance and Control System.

Notons que chaque chapitre de ce travail sera construit selon une même philosophie. Ils s'ouvriront tous par une introduction qui présentera les grandes lignes de ce qui sera traité dans le chapitre. Nous traiterons ensuite des problématiques proprement dites pour finir par une conclusion qui nous permettra de réunir les idées fortes et de mettre en perspective les concepts utilisés.

La seconde partie de ce travail aura pour objectif d'exposer notre approche méthodologique qui sera la trame de l'étude. Dans un premier temps, nous présenterons les concepts fondamentaux des systèmes temps réels. Nous proposerons ensuite différents moyens que nous avons à disposition pour modéliser et développer de tels systèmes. Nous nous limiterons à expliquer les deux grandes méthodes utilisant UML comme langage de modélisation :à savoir ROOM et ROPES qui sera celui que nous analyserons plus en détail. Enfin, dans une troisième partie, nous évoquerons notre approche personnelle quant au processus utilisé pour concevoir notre système répondant aux normes du A-SMGCS.

Viendront ensuite les parties centrées sur l'analyse et la conception de notre système embarqué. Nous débuterons par l'analyse des besoins de ce système. Nous définirons les concepts de besoins, d'exigences et de contraintes. Après avoir défini les acteurs interagissant avec le système, nous tenterons de saisir leurs buts quant à l'utilisation de ce système. Les fonctionnalités attendues seront exprimées sous forme textuelle. A partir de ces besoins, nous présenterons quelques Use Cases qui permettront de modéliser les interactions entre certains

acteurs et le système à mettre en place. Enfin, pour valider cette approche, nous présenterons un prototype global sous forme d'une mise en scène de différents Use Cases. Ce prototype représentera un incident pouvant survenir dans l'enceinte d'un aéroport, à savoir l'incendie d'un véhicule sur une voie. Nous terminerons cette partie par une critique, à charge et à décharge, de cette démarche.

La capture des besoins achevée, nous affinerons notre démarche par la modélisation des composants-métiers que notre système devra contenir. Nous dévierons quelque peu de la marche à suivre proposée par ROPES en représentant statiquement le domaine étudié. Ce modèle statique, nous permettra de glisser plus naturellement vers une représentation dynamique de chaque composant. Pour chacun de ces composants nous proposerons un diagramme d'état qui montrera les états possibles dans lesquels un module peut se trouver à chaque instant. Cette analyse devrait nous permettre de décomposer les systèmes en modules interdépendants. D'un système global, nous dériverons vers des sous-systèmes particuliers.

Les composants définis, nous reviendrons rapidement sur le modèle statique proposé par ROPES. Cette approche ayant déjà été abordée préalablement, il s'agira essentiellement d'affiner la représentation du système en y ajoutant des contraintes et des exigences que nous avons pu relever grâce à l'étape préalable. Ces raffinements successifs nous aiguilleront parfois vers des solutions architecturales particulières. Si tel est le cas, nous expliciterons les raisons pour lesquelles nous retenons ces options.

Au terme de notre étude, nous reprendrons une critique générale de notre approche. Nous synthétiserons les points positifs et négatifs que nous avons pu relever tout au long de notre démarche et le cas échéant, nous proposerons des voies alternatives qui pourraient nous permettre de résoudre certains problèmes. Ainsi, si nous jugeons que UML n'est pas suffisant pour modéliser certains types de contraintes temporelles, nous pourrions proposer d'autres techniques alternatives offrant un formalisme plus adapté.

Ce mémoire se terminera par une conclusion générale dans laquelle nous rappellerons les grandes étapes de ce travail, les points forts de la méthode suivie ainsi que d'éventuelles pistes de réflexions à poursuivre pour améliorer le processus de développement suivi.

2 Description du domaine

2.1 Introduction

La croissance accélérée du trafic aérien de ces dernières années a un impact très important sur la gestion de ce trafic. Tant la sécurité que la capacité d'accueil des aéroports est sans cesse remise en question. Par ailleurs, il s'avère de plus en plus indispensable de pouvoir faire voler plus d'avions dans un espace parfois identique, mais souvent réduit suite à des décisions politiques justifiées par d'autres impératifs que la seule gestion rationnelle du trafic aérien. L'afflux des vols implique un afflux des déplacements d'avions au sol. Ces carrousels impliquant des avions venant d'atterrir, des avions se préparant au décollage, d'autres se rendant au garage pour un entretien, des camions de ravitaillement, des bus transportant des passagers, d'autres véhicules prioritaires de sécurités,... embouteillent les aéroports et les risques d'accrochages plus ou moins sérieux sont légions. Or, on constate que la plupart des immobilisations d'avions impliquant des retards sont souvent dues à des accidents au sol. Ainsi, une ligne importante de Virgin a été bloquée durant une journée en 2001 suite à l'immobilisation de deux de ses avions impliqués dans des accrochages au sol !

Aujourd'hui, les outils de surveillance du trafic aérien génèrent une quantité importante de données brutes qui sont traitées par des humains ou des outils « intelligents ». Ces données émises par des radars et/ou trackers sont en perpétuel changement et leur traitement nécessite une prise en compte en « temps réel » des informations saisies. En effet, peu de passager accepterait de voir le guidage de leur avion retardé de quelques minutes au moment de l'atterrissage. Ce travail a donc pour but de saisir certains services qui pourraient s'avérer utiles pour la gestion du trafic aérien/ terrestre, de les modéliser et de proposer des solutions pour arriver à combler ces besoins. De nos jours une série de besoins nouveaux apparaisse dans le traitement de ces données.

Aussi, dans cette première partie, nous allons présenter le domaine d'application dans le cadre duquel s'inscrit notre travail. Dans un premier temps, nous introduirons sommairement les concepts clés des aéroports civils, les différents métiers qui s'y exercent et les acteurs qui y évoluent. Ensuite nous présenterons un système de guidage au sol de véhicules qui tend à se répandre dans les aéroports européen, le A-SMGCS (**A**dvanced **S**urface **M**ovement **G**uidance and **C**ontrol **S**ystem). Enfin nous terminerons cette présentation en évoquant les objectifs raisonnables que nous pouvons espérer atteindre dans un travail tel que celui-ci.

2.2 Champ d'application et contexte de l'étude

2.2.1 Les aéroports civils

L'étude que nous voulons mener à pour objectif de modéliser un système de guidage au sol de mobiles au sein d'un aéroport. Par mobiles nous entendons tout véhicule qui se déplace au sein de l'enceinte d'un aéroport. Ce peut être un avion qui vient d'atterrir, un camion de pompier ou un tracteur à bagages. Depuis une dizaine d'années, le trafic aérien est en très large expansion. Cet accroissement a un impact considérable sur l'évolution des aéroports tant en Europe que dans d'autres régions du monde.

En Belgique, ces dernières années ont vu la croissance de Zaventem, par une augmentation des mouvements. Cette augmentation, si elle subit des fluctuations conjoncturelles liées aux situations économiques et politiques, semble inéluctable à long terme. Autres preuves de cet

accroissement est le développement des aéroports régionaux à partir desquels de nombreuses compagnies décident de rayonner.

Aussi, tant les questions de sécurité que d'accroissement de capacité hantent les responsables de ces aéroports. En effet, la multiplication des mouvements impose une meilleure régulation des véhicules tant en vol qu'au sol et ce tout en améliorant les conditions de sécurité.

Un aéroport civil peut être vu comme un microcosme dans lequel évoluent des acteurs aux fonctions bien définies. Si la fonction première d'un aéroport est d'accueillir des avions et d'offrir des infrastructures permettant d'accueillir leur « cargaison », tant matérielle qu'humaine, de nombreuses activités périphériques trouvent également leur place dans cette enceinte fermée.

Nous n'évoquerons pas ici tous les « métiers », toutes les fonctions qui doivent être rendues dans un aéroport tant celles-ci sont multiples. De plus, elles n'ont pas toutes un rapport avec le projet que nous menons. Nous nous bornerons donc à évoquer les métiers liés aux déplacements dans les aéroports.

2.2.1.1 Accueil des passagers –marchandises

Toute une infrastructure est mise en place pour gérer les passagers (arrivants comme partants). Chaque avion doit pouvoir se « parquer » pour que ceux-ci puissent débarquer ou embarquer.

Des passagers doivent également pouvoir se déplacer dans l'enceinte de l'aéroport afin de rejoindre l'avion dans lequel ils doivent embarquer ou atteindre la sortie de l'aéroport. Ainsi, les aéroports, en fonction de critères qui leur sont propres, adoptent des stratégies différentes. Dans certains, les passagers sont transportés dans des bus qui les conduisent jusqu'à leur destination. Ces bus doivent circuler dans l'enceinte du lieu et « affronter » d'autres mobiles qui circulent eux aussi. D'autres aéroports, tel Zaventem, favorisent l'acheminement des passagers jusqu'à leur destination sans déplacement sur les « routes » de l'aéroport.

Certains aéroports régionaux autorisent même aux passagers de circuler à pied sur le tarmac de l'aéroport, tel Charleroi.

Pour ce qui est du (dé)chargement des marchandises, elles sont souvent chargées dans des avions garés à partir de véhicules qui circulent sur les voies de communication de l'aéroport.

2.2.1.2 Contrôle aérien - terrestre

Afin de conduire les avions à bon port, l'une des fonctions abritées dans un aéroport est celle du contrôle aérien. Plusieurs contrôleurs, aidés par des outils, communiquent avec les pilotes pour leur transmettre des informations et des injonctions qui permettent de réguler au mieux les mouvements aériens. Ces contrôleurs peuvent ainsi planifier les décollages et atterrissages. Ils se doivent donc d'être particulièrement méticuleux tant la sécurité des aéroports civils est fondamentale.

Parallèlement à cette fonction de contrôle aérien, la vérification du trafic terrestre entre tous les nombreux mobiles qui circulent dans l'enceinte de l'aéroport est également devenue un point vital pour la bonne gestion de ce microcosme. Très souvent, les accidents dans le milieu de l'aéronautique sont liés à des incidents au sol, impliquant ces mobiles. C'est pourquoi les grands aéroports européens mettent actuellement sur pied des systèmes de contrôle des mouvements au sol.

Figure 2.2.1.2 : croisement d'avions sur la piste – décollage et atterrissage [VALLEE,2001]

2.2.1.3 Sécurité dans un aérodrome

Vu le nombre important de mouvements de passagers comme de marchandises, il est fondamental que la sécurité des personnes comme des biens soit assurée. Aussi, toute une série d'acteurs intervient pour la garantir. Les pompiers, les secouristes, les antennes médicales, les forces de l'ordre, les douaniers, .. sont autant de personnes qui circulent continuellement sur les « routes » de l'aéroport.

Ces acteurs jouent des rôles différents. Des tâches précises leur sont attribuées et ils peuvent également intervenir en cas d'urgence. Si un accident survient dans l'aéroport, ils seront immédiatement avertis.

2.2.1.4 Entretien

Un tel monde ne peut vivre sans que des spécialistes n'entretiennent quotidiennement tant les infrastructures de l'aéroport (pistes, signaux, ..)que les mobiles qui circulent régulièrement sur ces pistes. Ainsi, les mécaniciens aéronautiques, les transporteurs de carburant, les réparateurs de pistes, ... sont autant de spécialistes qui doivent pouvoir se déplacer dans les aéroports afin d'en assurer le bon fonctionnement.

2.2.1.5 Un microcosme sous haute surveillance

Une véritable ruche ! C'est ainsi que l'on pourrait qualifier un aéroport. Une foule d'acteurs aux fonctions différentes mais complémentaires se croisent, se gênent, se soutiennent, tout cela pour assurer le rôle premier d'un aéroport : l'accueil de transporteurs aériens, de leurs passagers et/ou de marchandises qui doivent être acheminés de façon sécurisée et rapide d'un point de départ vers une destination.

A l'instar de la société dans son ensemble, il est à noter que tous ces acteurs ne dépendent pas d'une autorité unique. Les policiers doivent rendre compte à leur autorité de tutelle et peuvent agir assez librement dans certains cas. Il en va de même pour les pompiers, les douaniers, les médecins, ... Cette multiplication des responsables a pour effet de complexifier la gestion des aéroports.

Dans cet ensemble, nous nous limiterons à étudier la régulation de la circulation terrestre des mobiles dans l'enceinte des aéroports. Après avoir rappelé les grandes étapes qui ont amené les autorités à se pencher sur cette problématique, nous allons présenter les grandes lignes du système A-SMGCS.

2.2.2 Guidage au sol – une tentative d'harmonisation des procédures

2.2.2.1 Organismes régulateurs

Dans le monde de l'aviation civile, nombre d'organisations nationales et internationales ont été mises sur pied pour tenter d'uniformiser les règles et pratiques dans différents domaines.

Les plus connus sont[1] :

- OACI : Organisation de l'Aviation Civile Internationale
- EUROCONTROL
- JAA : Joint Aviation Authorities
- EUROCAE : Organisation Européenne pour l'Equipement de l'Aviation Civile
- CEAC : Conférence Européenne de l'Aviation Civile
- BELGOCONTROL
- …

Depuis plusieurs années, différents organismes internationaux édictent des règles permettant d'uniformiser les modes de contrôle aérien. En Europe, EUROCONTROL est un organisme qui tente d'harmoniser ces règlements. Aussi, tout pays Européen souhaitant faire partie d'un espace aérien unique, doit se conformer aux indications émises par cet organisme. Ces règles recouvrent tous les domaines du contrôle aérien.

En Belgique, l'organisme responsable de l'application des directives proposées par EUROCONTROL est BELGOCONTROL. Cette organisation s'occupe de la mise en œuvre concrète du contrôle aérien pour tous les avions civils qui traversent l'espace aérien de notre pays et du Luxembourg.

Il serait vain de passer en revue toutes les missions de tous ces organismes, mais il est toujours intéressant de se rendre compte que la complexité du domaine a forcé les états à se regrouper afin d'unifier leurs réglementations en terme de gestion du trafic aérien civil.

---

[1] Pour les organismes cités cf. :

OACI- IACO : http://www.icao.int

Eurocontrol : http://www.eurocontrol.int

JAA : http://www.jaa.nl/

EUROCAE : http://www.eurocae.org/

CEAC : http://www.ecac-ceac.org/

BELGOCONTROL : http://www.belgocontrol.be/

Parmi les nombreuses problématiques, la sécurisation et l'optimisation des mouvements au sol des mobiles au sein d'un aéroport ont longtemps intéressé ces organismes, pour aboutir à deux normes : le SMGCS et sa version améliorée, le A-SMGCS.

2.2.2.2 Principes actuels : Vu et être vu - SGMCS[2]

Vu l'absence de réglementation uniforme en terme de contrôle du trafic terrestre dans les aéroports à travers le monde, l'OACI (Organisation de l'Aviation Civile Internationale) a proposé en 1986 qu'un système soit mis en place pour lutter contre les accidents au sol dans les aéroports civils. Ainsi est né le **Surface Movement Guidance and Control System** (SMGCS)

Sur cette base, la sécurité des mouvements au sol pouvait se résumer par « Vu et être vu ». Encore aujourd'hui, les procédures opérationnelles dépendent essentiellement de la vision directe des conducteurs de mobiles terrestres et du(des) contrôleur(s). Le contrôleur, de sa tour, doit avoir une vision aussi globale que possible d'un aéroport. A l'inverse, les pilotes et autres conducteurs de mobiles voient la situation devant eux. La circulation dans l'enceinte d'un aéroport se fait donc majoritairement par « ajustements mutuels » entre les différents chauffeurs, également guidés par un ou plusieurs contrôleur(s).

Aussi, lorsque les conditions de visibilité ne sont plus optimales, il est de plus en plus difficile d'appliquer ce principe de guidage au sol. La détection des anomalies est parfois impossible ce qui oblige souvent les autorités aéroportuaires à restreindre la circulation. L'absence de radar de surface implique souvent une réglementation stricte de la circulation en surface, voire son interdiction. La circulation des avions peut, elle-même être fortement réduite et strictement réglementée.

Ces incertitudes embarrassent tant les responsables des aéroports que les chauffeurs eux-mêmes. Les risques d'accidents sont nombreux ainsi que les retards entraînés par les annulations de circulation.

Aussi, différents acteurs de la normalisation internationale dans le domaine aérien se sont penchés sur ces problématiques pendant plusieurs années pour aboutir à un nouveau standard qui devrait permettre une meilleure gestion du trafic terrestre. Guidés par le double facteur du maintien de la sécurité au sol et de l'augmentation de la capacité des aéroports, ces « normalisateurs » ont conçu le **Advanced-Surface Movement Guidance and Control System** (A-SMGCS).

2.2.2.3 A-SMGCS

- **Présentation**

D'« Automatic », le « A » a très vite été remplacé par « Advanced ». Les systèmes respectant ces directives, devront donc aider à assurer la sécurité et la capacité voulue tant dans des circonstances « normales » que dans des circonstances « exceptionnelles ». Ainsi, l'aide à la circulation terrestre devrait être facilitée tant en cas d'engorgement que lors de mauvaises situations météorologiques. Les quatre fonctions de base définies par l'OACI sont :

- Surveillance
- Routage

[2] Cette présentation du SMGCS et surtout du A-SMGCS est largement inspirée des articles de [VALLEE, 2001 ; STOICA, 2002]

- Guidage
- Contrôle (alertes, autonomisation du système)

L'application de tels besoins nécessite bien entendu le développement d'outils aidant à la mobilité. C'est ainsi que la multiplication des radars au sol et des systèmes permettant de localiser plus précisément les mouvements de mobiles, ont favorisé l'adoption de ces principes par plusieurs organismes régionaux telle la branche européenne de l'OACI, EUROCAE et EUROCONTROL,...)

Suite aux réflexions sur l'utilisabilité de tels systèmes par EUROCONTROL, la Communauté Européenne a financé plusieurs études sur le sujet tandis que les gros aéroports européens s'équipaient en matériels susceptibles de favoriser l'utilisation future du A-SMGCS[3].

Par ailleurs, EUROCONTROL a également étudié la normalisation des formats d'échange de données utilisés dans le A-SMGCS (ASTERIX 10, ASTERIX 11).

- **Objectifs du A-SMGCS**

Les objectifs principaux de ces systèmes sont d'offrir un très haut niveau de sécurité en toute condition [VALLEE, 2001], plus précisément :

- D'offrir à tous les acteurs (pilotes, contrôleurs, conducteurs de véhicule) un même niveau de service
- De préciser très clairement les responsabilités de chacun
- D'élaborer des moyens améliorés à l'intention de ces acteurs pour qu'ils puissent avoir une meilleure prise en compte de la situation
- De réduire les délais et d'augmenter le nombre de mouvements sans augmenter le temps de roulage
- D'améliorer les indications au sol et les procédures
- De réduire la charge de travail du contrôleur et du pilote par l'automatisation de certaines fonctions et l'amélioration de l'ergonomie
- D'offrir des solutions modulaires adaptées à chaque type d'aérodrome
- D'assurer la détection, l'analyse et la résolution des conflits
- De garantir un environnement plus sûr et efficace par l'automatisation en pouvant y inclure des éléments de contrôle, de guidage et d'assignation de routes

- **Fonctions de base du A-SMGCS**

a. Surveillance

A la manière d'un contrôleur humain par beau temps et dans des circonstances normales, la fonction de surveillance doit pouvoir donner au système la connaissance de la position et de l'identification de tous les mobiles dans toutes les conditions météorologiques et toutes les configurations d'aérodrome.

Par ailleurs, cette connaissance de la situation, à chaque instant, devra pouvoir être donnée aux acteurs concernés (contrôleurs, pilotes, conducteurs de mobiles, ...), mais elle sera également utilisée pour activer les autres fonctions du A-SMGCS (guidage, contrôle).

[3] Citer quelques études.

Idéalement, cette fonction devrait couvrir l'entièreté d'un aérodrome à une hauteur suffisante que pour traquer des hélicoptères évoluant à basse altitude.

Dans la plupart de la littérature [EUROPEAN ORGANISATION FOR THE SAFETY OF AIR NAVIGATION, 2003; VALLEE, 2001] présentant A-SMGCS, la fonction de surveillance A-SMGCS est assurée par les senseurs et un module de fusion des données. Nous avons opté pour la solution différenciant la capture des données à celle de leur traitement. Ainsi les senseurs (radars, GPS, transpondeurs, ...) seront considérés comme fournisseurs de données au système A-SMGCS et non comme partie intégrante.

b. Routage

Cette fonction est celle qui désignera la route à suivre pour chaque mobile.

- **En mode manuel** : cette route est proposée au contrôleur qui la transmet au mobile concerné.
- **En mode automatique** : cette route est directement transmise au mobile. Le contrôleur quant à lui, en est informé. Il a toujours la possibilité de prendre la main sur le système. La fonction de routage, doit tenir compte d'un ensemble de données et contraintes et doit pouvoir réagir en « temps réel » à des événements imprévus.

c. Guidage

Cette fonction permet de donner aux conducteurs de mobiles (pilotes ou autres) des indications claires et précises. Ces informations doivent leur permettre de suivre leur route. Cette fonction fournit le guidage nécessaire pour tous les mouvements autorisés et est disponible pour toutes les sélections de route possible.

« *....Lorsque les conditions de visibilité permettent un acheminement sûr, ordonné et rapide des mouvements autorisés, la fonction guidage sera essentiellement fondée sur les aides visuelles normalisées. Si les délais d'acheminement sont augmentés en raison d'une mauvaise visibilité, d'autres équipements au sol ou embarqués seront nécessaires pour compléter les aides visuelles afin de maintenir la cadence d'écoulement du trafic et d'appuyer la fonction guidage.....* » [VALLEE, J-CL, 2001]

d. Contrôle

Il est à noter que le contrôleur doit toujours conserver son rôle d'organisateur et de garant de la sécurité. Aussi, la fonction de contrôle est là pour assister ce contrôleur. Elle doit être capable de l'aider à organiser l'ensemble du trafic, à évaluer les priorités entre mobiles, les séparations nécessaires entre eux, à donner les autorisations de circulation, à lancer des alarmes en cas de danger... Cette fonction peut également permettre l'activation de certains dispositifs de sécurité suite au lancement d'une alerte (feux, barrières, alarmes, ...).

Ainsi, lorsque certaines règles prédéfinies ne sont plus respectées, le système peut déclencher des alertes. Ces alertes, peuvent être soit automatiquement envoyées aux mobiles concernés soit être transmises par le contrôleur qui jugera de leur pertinence. Cette exigence d'alerte peut donc être considérée comme une sous-fonctionnalité mais elle est non imposée par les concepteurs du A-SMGCS.

Enfin, A-SMGCS autorise également que le système prenne la « main » sur les contrôleurs au cas où les situations l'imposent (non visibilité, accident grave, conditions météorologiques particulières) . Cette fonction peut soit être intégrée dans cette fonction de contrôle ou être considérée comme une autre fonctionnalité. Nous considérons que cette fonctionnalité en est une supplémentaire et peut donc constituer un « module fonctionnel » du A-SMGCS.

Un système A-SMGCS peut donc être modélisé par un diagramme de Use Cases dans lequel chaque grande fonction du A-SMGCS représente un Use Case [LEMOINE, 2000].

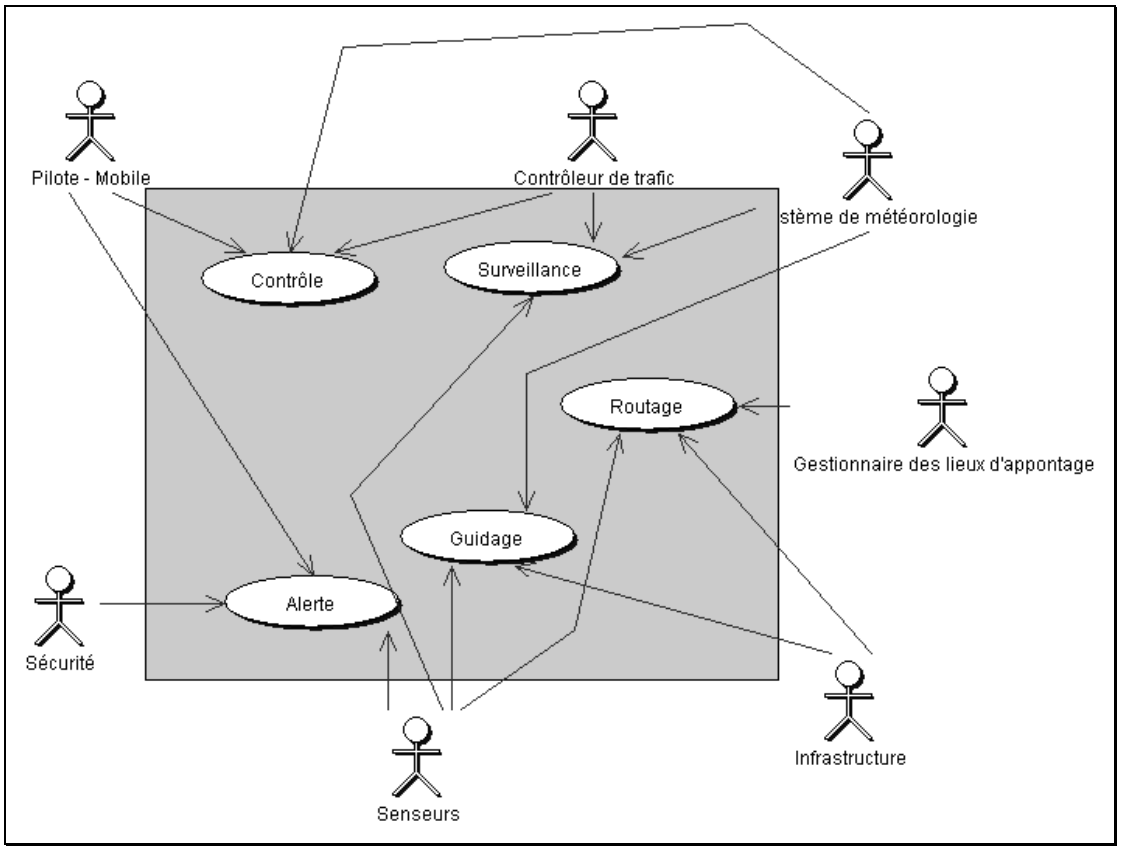

Figure 2.2.2.3a : A-SMGCS (inspiré de [LEMOINE, 2000])

Ce diagramme reprend les principaux acteurs (humains ou systèmes) qui peuvent intervenir dans la mise en place d'un tel système d'information impliquant dans la réalité plusieurs dizaines de types d'acteurs différents.

Les recommandations du A-SMGCS permettent donc de le représenter tel que le diagramme ci-dessous le propose [EUROPEAN ORGANISATION FOR THE SAFETY OF AIR NAVIGATION, *A-SMGCS*, 2003, p.30].

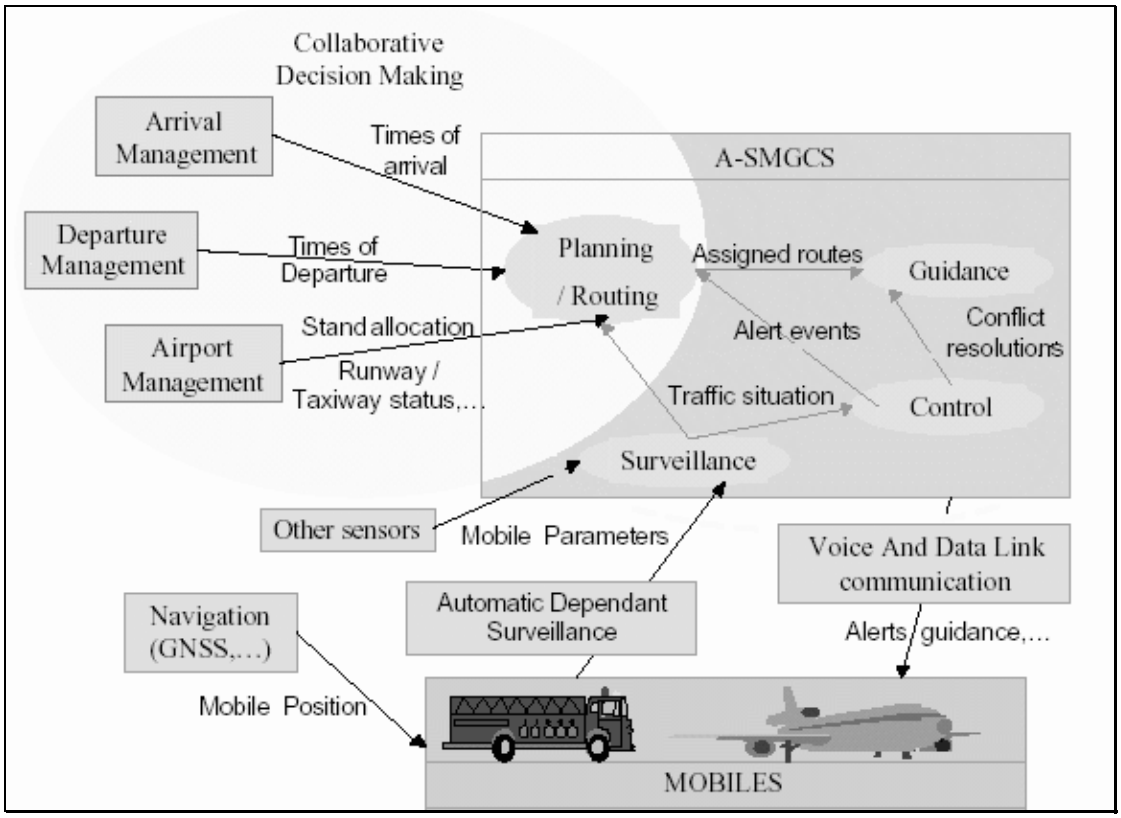

Figure 2.2.2.3b: Modélisation de A-SMGCS (extrait et adapté de [EUROPEAN ORGANISATION FOR THE SAFETY OF AIR NAVIGATION, *A-SMGCS*, 2003, p.30])

Dans notre analyse ultérieure, nous n'étudierons pas tous les acteurs, et toutes les interactions possibles pour un tel système. Un tel travail n'est pas réaliste dans le cadre d'un mémoire de licence.

- **Classifications de A-SMGCS**

L'OACI indique clairement que ces systèmes doivent être modulaires et pouvoir s'adapter à tous les aéroports. Ainsi, il propose une classification des besoins en fonction des aéroports et des conditions d'exploitation, en trois critères :

- Condition de visibilité où l'on définit les conditions de visibilité en fonction des pilotes, contrôleurs et autres mobiles
- Densité de trafic : nombre de décolages – atterrissages et de mouvements au sol
- Dispositions générales de l'aérodrome : nombre de pistes et voies de circulations.

- **Mise en oeuvre du A-SMGCS**

Qui dit surveillance, dit connaissance du milieu à surveiller. Aussi, l'identification des mobiles et leur positionnement exact dans l'espace en temps réel est une des premières phases à mettre en oeuvre pour l'élaboration d'un tel système.

Actuellement, aucun senseur, ni radar ne permet d'obtenir à la fois ce positionnement et l'identification de tous les mobiles en tous lieux et à chaque instant.

Aussi, est-il indispensable aujourd'hui d'associer différents senseurs, radars et autres techniques complémentaires (GPS, transpondeurs, détecteurs au sol, ...) et d'en fusionner les données. Ces données, une fois fusionnées peuvent alors être utilisées par les différents modules du système.

Tous les moyens sont bons pour donner une idée juste de l'état du trafic au sol dans un aéroport. Plusieurs catégories d'outils peuvent être utilisées pour arriver à ce résultat.

Premièrement les senseurs peuvent être coopératifs ou non-coopératifs. Les coopératifs sont ceux qui interagissent avec un élément actif (transpondeur,...). Ils permettent d'obtenir des informations liées au mobile. Le senseur envoie des signaux qui sont récupérés, analysés et conservés. Les non-coopératifs permettent de détecter les mobiles sans intervention de celui-ci (radar). De tels senseurs peuvent donc repérer des mobiles intrus ou des mobiles dont l'élément interactif est en panne. Le senseur repère des mobiles en mouvement dans un espace sans que ce mobile ne se soit déclaré.

De plus, les senseurs peuvent être dépendants (mobile génère lui-même l'information) ou non dépendants.

Ces différents types de senseurs peuvent donc se résumer par le tableau ci-dessous [VALLEE, 2001].

	Non coopératifs	**Coopératifs**
Non dépendant **Couverture totale**	Moyens humains Radar primaire de surface	Multilatération VHF ou UHF Multilatération mode S
Non dépendant **Couverture ponctuelle**	Capteurs hyperfréquences Capteurs optiques Capteurs acoustiques Capteurs infrarouges Boucles magnétiques	Porte mode S Radar secondaire dans l'axe d'approche
Dépendant		ADS ADS/B Transpondeur

a. Senseurs non-coopératifs

Parmi les senseurs coopératifs, citons les suivants :

- **Moyens Humains.** Les pilotes, contrôleurs, chauffeurs, et toute personne ayant la possibilité de donner des informations au système.
- **Radar primaire de surface.** Moyen le plus efficace pour avoir une information sur la position des mobiles situés dans l'enceinte d'un aéroport. Il existe plusieurs techniques.
- **Capteurs ponctuels.** Ils peuvent être utilisés pour protéger certaines zones, pour compléter l'information ou pour commander des équipements particuliers (feux, bornes, ...)

b. Senseurs coopératifs

Ces senseurs impliqueront l'installation de transpondeurs à bord des mobiles, tant dans les mobiles terrestres que dans les mobiles aéroterrestres. Pour cette dernière catégorie, cela est plus complexe car les normes ne sont pas uniformisées, et des avions de toutes origines peuvent évoluer sur un aéroport et évoluent sur différents aéroports.

- **Equipement de véhicules** : balises spécifiques qui émettent et/ou répondent à une fréquence spécifique (UHF – VHF). La localisation de chaque mobile « transpondé » se fait par triangulation.

- **Radar secondaire A/C** : Pour les avions une norme différente est en voie d'uniformisation : le radar mode S. Il permet d'identifier l'avion dès son arrivée jusqu'à son acheminement vers son poste de stationnement.
- **Triangulation ou multilatération Mode S** : Utilisation de trois radars Mode S qui par triangulation localisent les mobiles.

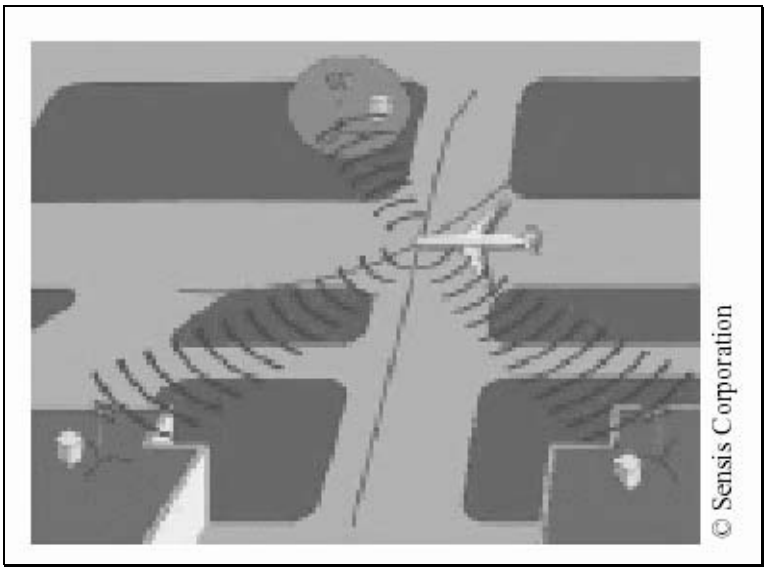

Figure 2.2.2.3c: Triangulation, multilatération Mode S[4], image extraite de [VALLEE, 2001].

- ...

[4] Pour plus d'information sur les radars mode S : http://www.eurocontrol.int/mode_s

c. Senseurs dépendants

Actuellement encore peu utilisés, ces senseurs ont néanmoins un avenir à court terme dans le domaine qui nous intéresse. En effet, dans cette configuration, c'est le mobile qui réalise le plus gros du travail. Il évalue sa position (via GPS, ...) et la transmet au système tout en s'identifiant.

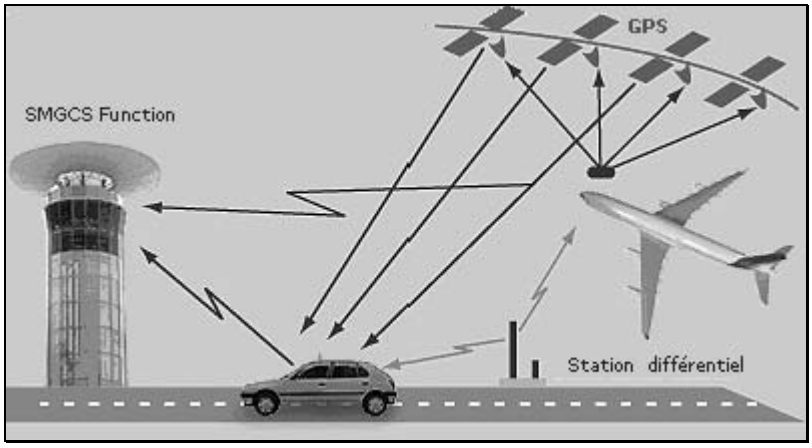

Figure 2.2.2.3d: GNSS, D-GPS, image extraite de [VALLEE, 2001].

La fusion des données est un élément clé des systèmes A-SMGCS. Cette fusion peut être obtenue par la combinaison de tous les senseurs. Ces données peuvent être fournies tant de manières synchrones, qu'asynchrones. Elles peuvent être complètes, partielles, erronées. Aussi, le module s'occupant de la fusion de ces données devra non seulement traiter ces informations multiples, contradictoires, incomplètes et en retirer une vision la plus cohérente avec la réalité et cela rapidement, en temps réel ! EUROCONTROL a normalisé le format de ces informations fusionnées avec le protocole ASTERIX 11[5].

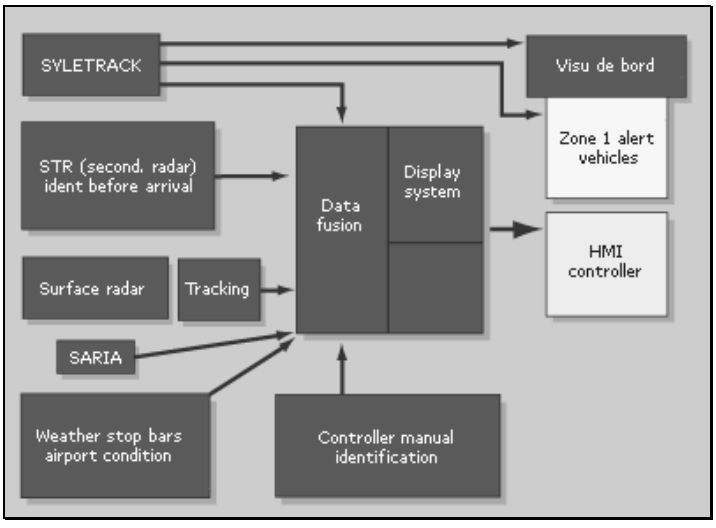

Figure 2.2.2.3e: A-SMGCS - Fusion des données, image extraite de [VALLEE, 2001]

[5] Pour plus d'information sur les concepts d'ASTERIX :

http://www.eurocontrol.int/projects/eatchip/asterix

Tout au long de cette présentation, il apparaît donc qu'une des craintes principales des responsables d'aéroports soit la collision au sol. Aussi, la plupart des aéroports internationaux favorisent la mise en place de systèmes permettant de prévenir et d'éviter ces collisions dans le domaine du A-SMGCS. Or, le développement de modules anticollisions implique au préalable de bons systèmes de surveillance qui seront utilisés par ces modules.

Ce type de module peut générer deux types de messages :

- A moyen terme : représentation de situations potentiellement dangereuses ou problématiques
- A court terme : alarmes impliquant des actions immédiates.

Actuellement, des modules anti-intrusions commencent à bien fonctionner sur plusieurs aéroports.

L'idéal à atteindre dans un futur proche est d'arriver à une intégration complète de toutes les données capturées dans l'enceinte des aéroports, et la mise sur pied de systèmes totalement intégrés, c'est à dire tant le routage, que le guidage, que la transmission d'informations à tous les mobiles terrestres comme aériens.

2.3 Objectifs à atteindre

Comme cette approche sommaire nous l'a montré, la gestion du trafic au sol dans un aéroport est une problématique très importante dans le chef des responsables des différents métiers de l'aviation civile. Dans ce travail, nous souhaiterions proposer une méthode d'analyse et de conception permettant de modéliser un système répondant, entre autres, aux exigences du A-SMGCS. Bien loin de nous l'idée de concevoir un tel système opérationnel. L'idée principale est d'éprouver une méthodologie de développement et de tester un langage de modélisation très à la mode actuellement (UML), de critiquer cette approche et éventuellement d'y apporter des compléments.

Le système à modéliser devra répondre d'une part à des exigences « temps réelles » et d'autre part, certains composants du système devront pouvoir être embarqués dans des mobiles, ou par des utilisateurs humains se déplaçant parfois hors des sites. A cela s'ajoute le fait que ce système devra collaborer avec tous les systèmes déjà éprouvés depuis de nombreuses années dans le microcosme du secteur du transport aérien civil.

Ainsi, les critiques que nous formulerons, positives et négatives, viseront à percevoir tout ce qui peut être relevant, intéressant et utile dans les outils de modélisation que nous utiliserons. Les limitations, inconsistances, et freins devront également être mentionnés. Enfin, nous essayerons de proposer, si cela est possible, des alternatives face à ces limitations !

3 Approche méthodologique

3.1 Introduction

Après avoir planté le décor de notre cas d'analyse et évoquer les objectifs, il convient de présenter la démarche méthodologique suivie pour canaliser notre travail. Dans un premier temps nous proposerons un aperçu des concepts de « systèmes coopératifs temps réels embarqués». Celle-ci nous permettra de saisir les notions clés, les contraintes et les spécificités de tels systèmes. Dans un deuxième temps, nous présenterons des méthodes utilisées pour encadrer les développements de tels systèmes. Enfin, nous présenterons la démarche personnelle que nous souhaitons suivre pour aboutir à modéliser et développer un système d'aide au guidage de mobile terrestre dans un aéroport.

3.2 Systèmes coopératifs temps réels embarqués

Plusieurs concepts différents sont associés pour caractériser le type de système que nous souhaitons mettre en place, la coopération entre systèmes, les contraintes temps réelles sur ceux-ci et la mobilité des composants.

La **coopération** fait essentiellement référence à des systèmes dans lesquels plusieurs composants de natures et de technologies différentes sont appelés à travailler ensemble. Ce travail en commun implique de longues analyses afin de comprendre les processus et les mécanismes d'échanges possibles entre systèmes.

Un système temps réel embarqué doit répondre à deux grandes caractéristiques :

D'une part, il doit être **temps réel**, ce qui signifie qu'il est conçu pour répondre et traiter les informations qu'il reçoit « à ce moment là précis et non avant ou plus tard ». En d'autres termes, un tel système doit répondre correctement à des stimuli externes selon un ensemble de contraintes temporelles [GULLEKSON,2001]. Ainsi, dans un avion, le délai entre la demande d'un pilote pour changer de direction et le changement du gouvernail doit arriver dans un intervalle de temps acceptable sinon l'avion risque un accident. Cette contrainte contraste fort avec les applications de gestion traditionnelle pour lesquelles des retards peuvent être ennuyeux mais jamais fatals !

S'il est assez simple de définir sommairement ce qu'est un système temps réel, il est plus complexe d'en définir toutes les contraintes imaginables qui peuvent y être liées. Dans un système qui est « non temps réel », le retard est acceptable. A l'inverse, dans les systèmes temps réels, le retard est à bannir. Bien plus, un système temps réel n'est pas forcément un système rapide, mais il doit avant tout être prévisible dans le temps et déterministe.

Il est également possible de différencier les systèmes temps réels en deux types : Le système temps réel « dur » et le système temps réel « doux ».

Le système temps réel « dur » signifie que les réponses à un événement ou que les actions prises suite à une requête doivent se terminer dans un délai précis [DOUGLASS, *Capturing Real-Time Requirements*,2001].

> *Par exemple, un système d'assistance médicale doit prévenir immédiatement (dans un délai de moins de 1 seconde) le corps médical après que la température d'un patient hospitalisé soit descendue en dessous de 35°.*

Le système temps réel « doux » est plus laxiste. Pour ces systèmes, les réponses à un événement devraient être traitées endéans un certain temps. Un taux fixé d'erreur est accepté,.... [DOUGLASS, *Capturing Real-Time Requirements*,2001]

> *Par exemple, dans un système d'ascenseur, il serait bon, que la cabine s'arrête dans les 2 minutes qui ont suivi l'appui sur un bouton d'appel.*

Les mathématiques qui doivent être utilisées pour analyser les systèmes « soft » sont souvent plus complexes que pour les cas temps réels « hard ». Aussi, il est habituel d'analyser les « soft » de la même manière que les « hard ».

D'autre part, un **système embarqué** a pour caractéristique d'être « autonome » ou plutôt que la puissance de calcul soit construite dans le système lui-même [BESSIN,2003]. Cependant, la multiplication des systèmes modulaires communiquant entre eux permette de concevoir des systèmes embarqués pour lesquels soit la puissance de calcul est répartie entre différents sous-systèmes, soit l'échange d'informations entre sous-systèmes est essentiel pour la bonne utilisabilité du produit. Par exemple, un téléphone portable est autonome, possède un processeur qui peut avoir une capacité de calcul, mais si le téléphone n'est pas lié à un réseau d'antennes, ou s'il est unique sur le marché, l'utilisation de l'outil est caduque.

Les systèmes embarqués temps réels peuvent regrouper un large spectre d'outils sur le marché, allant des machines à laver, aux téléphones portables, PDA, des système de guidage GPS dans les automobiles ou encore des patches médicaux contrôlant des fonctions vitales...

Aussi, l'accroissement de tels systèmes sur le marché est un bon révélateur du « paradoxe de développement logiciel ». En effet, alors que les sociétés sont amenées à concevoir des systèmes de plus en plus complexes ; de plus en plus rapidement, elles doivent également garantir une fiabilité sans faille de ces systèmes si souvent liés à des domaines critiques. Dans le développement du design de tels systèmes il est important de répondre aux nécessités suivantes :

- Disponibilité sans faille
- Supporter la concurrence d'utilisation
- Système répondant à des stimuli externes
- Distribution
- Facilité de maintenance et de mise à jour

Dans le travail qui nous occupe, nous allons essentiellement proposer un système permettant d'échanger des informations dans les espaces ouverts des aéroports civils. Ces échanges d'informations toucheront essentiellement des données sensibles ayant un impact tant sur la sécurité des mobiles au sol ou en vol que des informations destinées à des techniciens souvent éloignés des lieux de production. Cette double exigence de sécurité et de transportabilité des données liées à la multitude de sous-systèmes parfois anciens mis en collaboration nous a donné l'idée de ce concept de « Système Coopératif Temps Réel Embarqué ».

3.3 Moyens à disposition

L'objectif de départ est donc d'éprouver les outils UML existants et des méthodologies associées pour modéliser un système coopératif temps réel embarqué. L'idée d'étudier des systèmes coopératifs à l'aide de ce langage n'est pas neuve et est relativement bien éprouvée. Cependant, si depuis quelques années UML se diffuse comme outil de modélisation de systèmes orientés Gestion, son utilisation pour des systèmes davantage sécurisés n'est pas encore fort répandue. Malgré une abondance d'articles et d'ouvrages sur le sujet, il faut bien reconnaître que pour la plupart des projets nécessitant des développements de composants intelligents, prédictifs et fiables, son utilisation n'est pas encore fort répandu.

Aussi, sur base d'une analyse de cas essentiellement axée sur les spécifications d'un type de système d'aide à la navigation au sol dans les aéroports, nous comptons éprouver certains aspects d'UML, en sortir les cotés positifs, en pointer les faiblesses et éventuellement proposer des solutions alternatives.

Nous commencerons donc cette étude dès la capture des besoins auxquels le système devra répondre. Nous raffinerons notre analyse par étapes successives pour finalement aboutir à des conclusions sur les points forts et points faibles de notre démarche.

3.3.1 UML et Temps Réel[6]

Si UML est un langage de modélisation se voulant universel pour l'analyse et la conception de systèmes selon une orientation objet, l'ambition de ses concepteurs a dès le départ été de couvrir tous les domaines d'application. Mais, si pour la modélisation de systèmes d'information de gestion, cette approche universalisante est sans doute quasi atteinte, il n'en va pas de même pour tous les types de système. Il existe donc des nécessités d'introduire des extensions à ce langage pour tenter de modéliser ces autres systèmes.

Dans le cadre des systèmes temps réels deux extensions sont assez populaires [PERALDI-FRATI, 2002]:

- *UML for Real-Time* (UML-RT) [DEL BIANCO, 2003 ; SELIC, RUMBAUGH,1998 ; LYONS, 1998]

  Adaptée de la méthode ROOM (cf. ci-dessous), elle introduit un stéréotype de classes « *Capsule* » ainsi que les notions de *Port, Protocole* et *Connecteur*, mieux adaptés aux systèmes temps réels que les classes seules.

  ROOM introduit également une nouvelle représentation, « *le diagramme de structure* », qui permet de décrire la structure d'une agrégation d'objets et l'interconnexion entre eux.

- *Real-Time UML* (RT-UML) [DOUGLASS, *Ropes*,1999]

  Adaptation proposée par B.P. Douglass, elle s'appuie sur UML pour modéliser les systèmes temps réels. La méthodologie, fondée sur UML, n'y ajoute que l'extension des « *diagrammes temporels* ».

  Douglass a également proposé une nouvelle méthodologie, améliorant son Real-Time UML : ROPES (*Rapid Object-Oriented Process for Embedded Systems*), plus spécifique pour les systèmes embarqués (cf. ci-dessous).

3.3.2 ROOM : Real-Time Object-Oriented Methodology

Langage de modélisation visuel associé à une sémantique formelle, ROOM a été développé par la société ObjecTime. Il est optimisé pour la spécification, la visualisation, la documentation, l'automatisation et la construction de systèmes temps réels potentiellement distribués, complexes et « orientés-événement ».

---

[6] La liste d'ouvrages et d'articles utilisés pour rédiger cette partie est longue. Nous invitons le lecteur à consulter dans la bibliographie de ce mémoire les ouvrages des auteurs suivants : DOUGLASS ; JACOBSON, BOOCH et RUMBAUGH ; ZHANG ; HILLARY ; TEYSSIE, MAMMERI, CARCENAC et BONIOL ; LAVAZZA ; Mc LAUGHLIN ; ..

La bibliographie de ce travail reprend tous les articles ou livres consultés. Il serait fastidieux de tous les citer.

Cette méthodologie est antérieure à l'apparition de UML. Pour certains [SELIC, RUMBAUGH,1998], UML est suffisamment générique pour modéliser n'importe quelle méthodologie dont ROOM. En fait, l'apparition d'une méthodologie UML-RT « créée » par ces deux auteurs, n'est rien d'autre que UML pour ROOM.

Ainsi, ROOM peut être considéré comme un des « design patterns » architectural tel qu'il en existe. C'est pourquoi, il est parfois utilisable dans certaines problématiques tandis que pour d'autres, il est préférable d'utiliser d'autres méthodes.

ROOM travaille essentiellement avec des Interfaces. Celles-ci sont réifiées dans des classes *Port* et l'interaction complexe entre les objets est transposée dans des classes *Protocol* qui permettent d'arbitrer le comportement entre *Ports*. A l'inverse des interfaces UML, les classes *Port et Protocol* représentent des interfaces bidirectionnelles.

Les objets « à grande échelle » sont appelés *Capsules*. Ils implémentent les comportements spécifiés par des « statecharts ». Des *Capsules* peuvent contenir des sous-capsules. Il est également possible de communiquer des messages d'une *Capsule* à une *sous-capsule* via un *Relay-Port*.

3.3.2.1 Rôles des objets

Ci-dessous, nous synthétisons l'ensemble des classes et des relations existants entre elles.

- *Capsule*

  Classe qui fournit un comportement bien encapsulé. Il est accessible via des « Port »

- *Connector*

  Il connecte deux ports impliqués dans des échanges de messages

- *End Port*

  Indique qu'un message s'arrête à cette Capsule et qu'il invoque un signal dans le « statechart » de cette capsule.

- *Port*

  Représente une interface pour une Capsule. A l'inverse d'UML, les Ports peuvent avoir des attributs et une structure. Les Ports son liés aux Protocols.

- *Protocol*

  Il s'agit d'une spécification de séquences de flux de messages permis ou désirés entre deux Ports partageant une Connection.

- *Relay Port*

  Port qui passe un message à une sous-capsule.

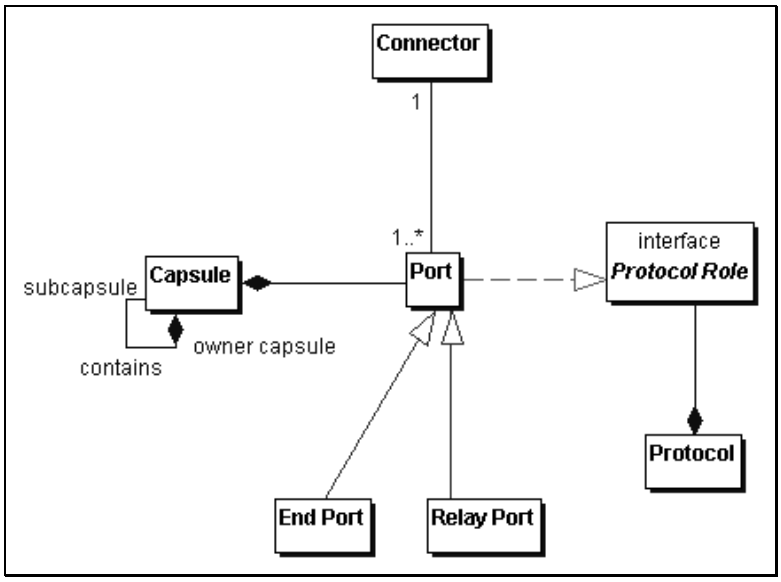

Figure 3.3.2.1: modélisation de ROOM, inspiré de [DOUGLASS, 2003, p.195]

3.3.2.2 Bénéfices – Inconvénients de ROOM

L'un des bénéfices les plus intéressants de ROOM est qu'il modélise les deux côtés d'une interface, tant le client que le serveur. Par ailleurs, l'utilisation de classes *Port*, *Connector* et *Protocol* permet une très bonne définition des interfaces complexes. Les classes *Protocol* sont quasi exclusivement modélisées par des statecharts, ce qui facilite l'utilisation des pré et post conditions des services. De plus, le fait que l'on puisse abstraire les interfaces (UML) en *Port* et classes *Protocol*, permet de leur donner un état et des attributs, ce qui les rend donc plus puissants.

Toutefois, les associations entre les classes sémantiques (*Capsules*) passent par l'intermédiaire d'un ensemble d'autres classes. Une telle structure peut compliquer inutilement les relations entre classes ayant une sémantique simple. Aussi, l'usage de ROOM doit être circonstancié. Il peut être utilisé lorsque des classes ont une sémantique qui est relativement importante et complexe. ROOM est particulièrement approprié quand l'interaction de quelques objets importants est complexe et requiert des significations spéciales pour contrôler et arbitrer des choix.

3.3.3 ROPES – Rapid Object-Oriented Process for Embedded Systems

[DOUGLASS, *Ropes*,1999 ; DOUGLASS, *Real-Time Design Pattern*, 2003]

ROPES est le processus que nous allons éprouver pour encadrer le développement de notre projet. Nous en présentons ci-dessous les principales caractéristiques en nous appuyant sur les articles et ouvrages de B.P. Douglass[7], principal concepteur de cette méthodologie.

Le processus ROPES est basé sur un cycle de vie du développement itératif qui utilise le standard UML et qui encourage la génération automatique de code et ce afin de parvenir rapidement à des prototypes à valider.

[7] Nous invitons le lecteur à consulter tous les écrits de Douglass repris dans la bibliographie ci-dessous.

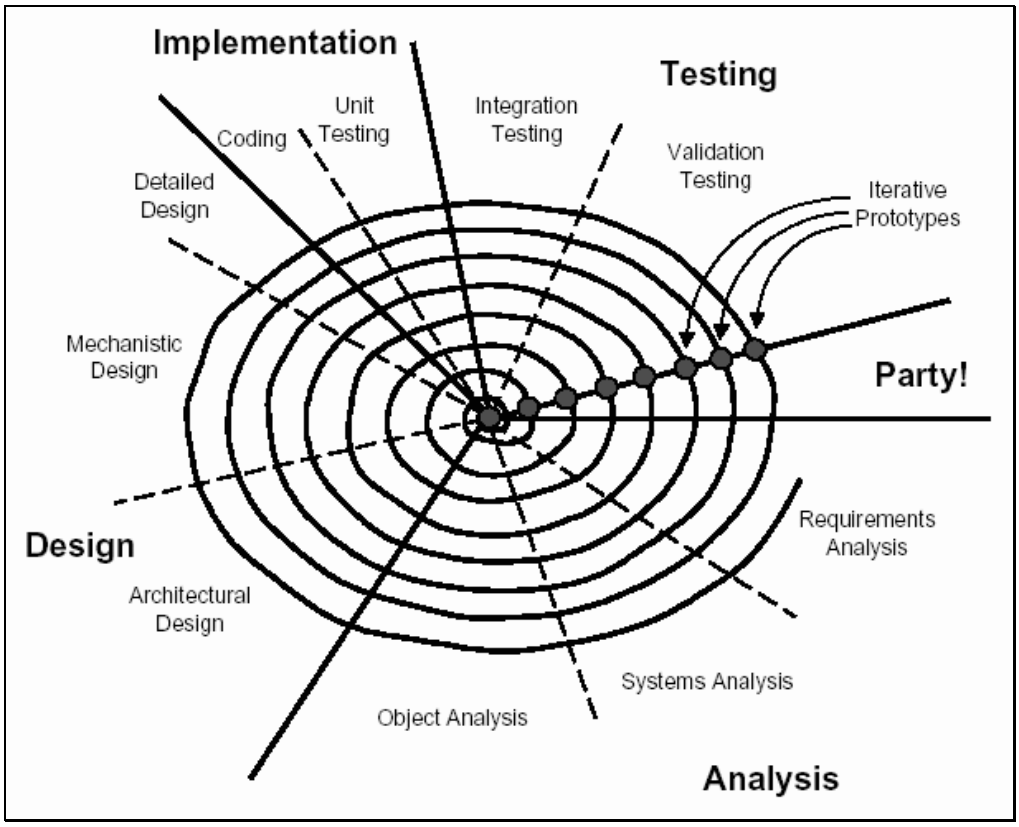

Figure 3.3.3 : ROPES, extrait de [DOUGLASS, *ROPES*, 1999,p.8]

La figure ci-dessus modélise les différentes étapes cycliques de ce processus de développement divisé en quatre grandes phases :

- Analyse
- Design
- Implémentation
- Test

3.3.3.1 Analyse

Cette phase permet d'identifier toutes les caractéristiques du système à développer. Celle-ci peut être découpée en trois sous-phases qui ont chacune leurs spécificités.

- **Analyse des besoins** [DOUGLASS, *Capturing Requirements..*, 2001 & 2003]

Dans cette sous-phase, les exigences et besoins doivent être extraits des clients. Cette analyse doit rester au niveau des vues fonctionnelles du système. ROPES précise qu'au niveau de l'analyse des besoins, aucun objet ne doit intervenir. Cette tâche viendra au moment de la phase de l'analyse objet.

Les activités principales à ce niveau sont :

- Identification des Use Cases et des acteurs associés
- Décomposition des Use Cases et de leurs relations
- Identification et caractérisation des événements extérieurs affectant le système et des messages qui sont échangés.

- Définition de scénarios donnant une vision du comportement dynamique du système.
- Pour des systèmes à hauts risques, recensement des dangers.

Deux types de modèles peuvent être utilisés pour rendre compréhensible cette phase :

- D'une part, les modèles contextuels tels que les diagrammes de Use Cases qui permettent d'embrasser les fonctionnalités principales du système ainsi que les liens entre celles-ci.
- D'autre part, les modèles comportementaux offrant la possibilité de montrer les interactions entre les acteurs et le système par l'échange de messages ; les contraintes (temporelle, performance, « fault tolerance », ...) liées à ces échanges et autres actions prises par les intervenants. Les éléments comportementaux définissent donc comment le système doit se comporter, en tant que « boîte noire ».

Les « diagrammes » utilisés pour cette phase peuvent donc être choisis parmi les propositions suivantes :

- Texte en langue naturelle
- Diagrammes de Use Case
- Diagrammes de séquences
- Diagrammes d'états
- ...

- **Analyse du système – Architecture** [DOUGLASS, *Designing Real-Time Systems,...*, 2004]

L'un des objectifs principaux de cette sous-phase est de proposer une vision architecturale haut-niveau du système. Signalons que cette phase est encore essentiellement une étape fonctionnelle et qu'elle ne nécessite pas de vue « objet » du système.

Les activités principales à ce stade sont :

- Identification d'unités organisationnelles haut-niveau pour les systèmes complexes
- Analyse des spécifications comportementales complexes pour ces unités organisationnelles

Le résultat de cette sous-phase est une architecture fonctionnelle comportant des nœuds « boîtes noires » qui ont chacun des éléments comportementaux propres. Ces composants peuvent donc être analysés en détail indépendamment. Il est nécessaire également de définir les interfaces permettant à ces modules d'interagir. De nouveau, ces interfaces peuvent être décrites à un haut niveau.

Cette analyse comportementale des composants peut être faite au moyen de machines à états finis. Pour les systèmes réactifs, cela signifie qu'il faut construire des statecharts complexes pour chaque composant comportemental. Cependant, UML n'offre pas de moyens pour modéliser les mathématiques continues. Aussi, les algorithmes continus, seront préférablement spécifiés via des équations ou du pseudo code.

Il est à noter que pour les systèmes complexes, il est crucial de pouvoir exécuter le modèle au plus vite afin de s'assurer de sa correction.

- **Analyse objet**

Dans cette sous-phase, les objets et classes essentiels sont identifiés et leurs propriétés importantes sont saisies. Deux approches complémentaires sont associées : les analyses objets structurelles et comportementales. De nouveau, une vision statique des objets est affinée par une vision dynamique de ceux-ci.

Les activités principales dans cette phase sont les suivantes :

- Découverte des objets principaux du système
- Abstraction de ces objets pour en saisir les classes
- Relation entre classes et objets
- Identification des collaborations entre objets
- Comportement de chaque objet
- Identification des opérations et attributs des objets
- Test via des scénarios
- Transposition des contraintes sur le système – composants vers les objets-classes

Les méta-modèles qui peuvent être utilisés ici sont les diagrammes de classes, ainsi que les diagrammes de collaborations et de séquences, dans lesquels des contraintes temporelles peuvent être associées. Il est également possible de représenter certains objets via des diagrammes d'états. Cela peut être intéressant pour des objets dont les comportements sont particulièrement cruciaux et complexes.

3.3.3.2 Design

Tandis que la phase d'analyse identifie la logique d'un système, le design propose une solution particulière unique optimale . Trois sous-phases composent cette étape :

- Le design architectural définit les décisions stratégiques du design qui affecteront les composants software tel que le modèle de concurrence, la distributions des composants.
- Le design mécaniste ajoute le liant entre les composants.
- Le design détaillé définit la structure interne et le comportement de chaque classe, composant, ...

La phase de design consiste majoritairement en l'application de « design patterns[8] » au modèle logique établi à la fin de l'étape d'analyse. Rappelons que tant le modèle de design que le modèle d'analyse représentent des vues différentes d'un même système à un niveau d'abstraction différent. Aussi, est-il fondamental que le design reste conséquent par rapport à l'analyse.

- **Design architectural**

Les activités principales de cette sous-phase sont :

[8] DOUGLASS, B. P., *Real-Time Design Patterns. Robust Scalable Architecture for Real-Time Systems*, Boston, 2003.

- Identification et caractérisation des threads.
- Définition des composants logiciels et leurs distributions.
- Application de « design patterns » pour la gestion des erreurs, la gestion de la sécurité, ...

Si les choix architecturaux peuvent souvent être guidés par l'analyse, cette phase permet d'aborder les architectures d'un point de vue technique et de la faisabilité de certaines demandes.

Les méta-modèles utilisés dans cette phase sont identiques aux modèles précédents (collaborations, classes, objets, composants, états finis, ...). Le raffinement successif permet de descendre graduellement dans la complexité d'un système.

- **Design mécaniste**

Niveau médian du design, cette étape est similaire à la précédente mais le raffinement des objets permet d'ajouter des notions plus techniques telles que les contrôleurs associés aux objets, les définitions de queues, les listes, ...

De nouveau cette étape utilise les même diagrammes que les précédents mais en les précisant davantage.

- **Design détaillé**

Stade ultime du design, au terme de cette phase, la structure interne de chaque classe, message, queue, liste, ... est définie, décrite et consolidée.

Si des classes sont des agrégations, compositions, héritages, … alors tout doit être définit.

Pour chaque opération de chaque classe, .. les pré-post conditions, invariants, exceptions lancées, ... doivent être décrits et consolidés.

3.3.3.3 Translation - Implémentation

Le passage de modèles conceptuels successivement affinés à du code concret et efficient caractérise cette phase. ROPES impose aussi qu'à ce stade des unités de test soient développées de manière à pouvoir valider chaque ensemble cohérent de code exécutable.

3.3.3.4 Tests

Cette phase intègre tant les tests d'intégration que ceux de validation. Il est important de concevoir Ces tests pour obtenir des résultats observables.

3.3.3.5 Remarque

La littérature présentant ROPES propose un outil case commercial, Rhapsody, qui permet de suivre les recommandations du processus et de modéliser les schéma UML adéquat. Notre objectif ne sera pas d'analyser cet outil. Cependant, il est toujours intéressant d'avoir conscience d'une réalité économique pouvant être associée à des propositions méthodologiques qui peuvent paraître académiques.

Ce processus est itératif, ce qui implique qu'à chaque étape, les équipes de développement doivent boucler une phase. Ainsi, dès le premier stade de l'analyse, le responsable d'un projet doit être capable de proposer au client un prototype éprouvé et validé, compréhensible par ce dernier. Il est bien entendu, qu'en fonction de l'avancement, les prototypes proposés ne seront pas de même nature. Au début du processus, il n'est pas réaliste de s'imaginer que l'on puisse fournir, dès la première phase un programme autonome. Cependant, sun premier prototype pourrait être un scénario décrivant une séquence d'événements, les interactions possibles entre les acteurs et le système. D'autres techniques de présentation pourraient également être envisagées tel qu'un petit film d'animation mettant en scène un Use Case décrit dans l'analyse, ou une Bande dessinée reprenant ces scénarios.

Par ailleurs, la méthodologie ROPES telle qu'elle est définie par Douglass fait curieusement penser au processus de développement logiciel aujourd'hui proposé par Rational, le RUP [JACOBSON, BOOCH, RUMBAUGH, 1999] (Rational Unified Process). En effet, sans entrer dans les détails de RUP, ce processus est également itératif et est divisé en 4 grandes phases :

- **Inception (début)** : où l'on définit la portée du projet, les besoins éprouvés par un scénario métier et où une première architecture fonctionnelle peut être proposée.
- **Elaboration** : où l'on valide une architecture pratique, on planifie les futurs développements en proposant les analyses techniques les permettant. A cette étape des composants techniques sont choisis pour être éprouvés
- **Construction** : où des composants sont développés, testés et confrontés avec les besoins définis préalablement.
- **Transition** : où un prototype est proposé aux clients. Ce prototype doit proposer des fonctionnalités souhaitées par les clients et pouvoir s'intégrer avec les composants existants.

L'analyse montre donc que ROPES s'inspire clairement de ce processus unifié tellement à la mode. La méthodologie proposée n'est donc en rien spécifique à des développements de systèmes embarqués. Les spécificités du temps réel viendraient donc davantage de la rigueur, des méthodes d'analyse, des outils de développement, de la robustesse et de la précision des langages d'analyse et de développement, que du processus de développement utilisé.

3.4 Approche personnelle

Au vu des liens entre ROPES et RUP, nous avons donc montré que ce « nouveau » processus destiné aux développements d'applications temps réelles embarquées n'est pas spécifique à ces types de développements et que d'autres méthodologies pourraient être employées. Cependant, vu l'aura que Bruce Powel Douglass[9] peut avoir sur la communauté des développeurs de tels systèmes, il nous semble utile d'appliquer sa méthodologie à notre cas d'étude afin de l'éprouver.

Les activités proposées pour chaque étape de développement seront appliquées et par la suite critiquée. Si des points forts ressortent à certaines étapes du processus, celles-ci seront mises en avant. De même si des incohérences ou des manques apparaissent, ils seront également

[9] Signalons que Bruce Powel Douglass accompagne la signature de tous ses écrits du qualificatif de « Chief Evangelist, I-Logix».

expliqués. Enfin, si dans le cours de notre étude, nous trouvons que des apports peuvent être ajoutés à ce processus, nous en expliquerons les intérêts et limites.

A chaque étape de notre analyse, nous présenterons succinctement les technologies utilisées et les diagrammes employés afin de ne pas surcharger cette partie méthodologique. En effet, les habitués de UML pourront aisément survoler ces passages. En plus de l'étude de ROPES, nous éprouverons aussi UML en tant qu'outil de modélisation de systèmes critiques. Si des manques sont soulevés, nous les signalerons et proposerons le cas échéant des pistes de réflexions pour résoudre ces problèmes.

3.4.1 Limites préalables

Si ROPES offre un processus unique englobant toute la chaîne du développement logiciel, nous nous limiterons au processus de l'analyse fonctionnelle. En effet, notre objectif n'est pas d'arriver à produire un prototype de notre système.

D'une part l'ampleur de la tâche rendrait cette mission impossible dans le cadre d'un mémoire de licence. D'autre part, il peut être dangereux de laisser le développement à la personne qui est déjà responsable de l'analyse. En effet, de même que les phases de test doivent être impérativement réalisées par des personnes n'étant pas intervenues dans le processus de développement pour éviter tout « préformatage » de la pensée du testeur, il en va tout autant pour les analystes – développeurs. Si une personne sait qu'elle va devoir développer ce qu'elle est en train d'analyser, son objectivité et l'abstraction dont elle doit faire preuve dans la phase d'analyse se voient mises à mal. Aussi, le risque est donc important que l'analyse soit orientée vers une solution technique et que lors de l'implémentation, le programmeur travaille avec de œillères. Il arrive très souvent lors des processus de développement qu'un programmeur constate des incohérences, des zones d'ombres dans une analyse qu'il ne remarquerait pas si il en était le rédacteur.

3.4.2 Analyse

3.4.2.1 Capture des besoins

Dans un premier temps nous tenterons de saisir les buts des différents acteurs, leurs besoins face au système.

Pour ce faire, nous utiliserons différentes méthodes complémentaires, telle qu'une arborescence de buts des acteurs, un texte structuré reprenant les besoins auxquels le système devra répondre une série de Use Case présentants quelques scénarios.

Pour modéliser ces Use Cases nous utiliserons le diagramme de Use Case fournit par UML. Ces Use Cases seront présentés au moyen d'un formulaire permettant de décrire les flux d'informations échangés :

Description :	
Acteur(s) primaire(s):	
Acteur(s) de support	
Extensions	
Inclusions	
Scénario principal	
Précondition	
Action de(s) acteurs	**Responsabilité du système**
Postcondition	
Flux d'exception 1	
Action de(s) acteurs	**Responsabilité du système**
Contraintes	

Ce formulaire reprend donc les acteurs impliqués, les relations entre ce Use Case et d'autres Use Case (inclusion, extension), les pré et post conditions de la bonne exécution du Use Case, le flux de base ainsi que la possibilité d'ajouter des flux d'exceptions. Enfin, une rubrique « contraintes » nous permet de forcer certaines contraintes difficilement exprimables dans un flux de données. Chaque Use Case sera identifié par un numéro unique, ce qui permettra de faciliter les références entre eux.

Afin de pouvoir consolider et valider cette première étape d'analyse, nous proposerons un prototype représentant un scénario intégré. Ce scénario sera celui d'un incident ayant lieu au sein de l'aéroport. Ce scénario permettra de montrer que les différents Use Cases décrits sont utilisables, et que les contraintes temps réelles sont respectées.

Enfin, afin de modéliser les échanges entre les acteurs et le(s) système(s) nous utiliserons des diagrammes de séquences pour marquer plus formellement les contraintes temps réelles.

3.4.2.2 Analyse du système

Dans une seconde partie de cette phase d'analyse, nous tenterons de saisir les composants qui interagissent dans ce système.

Sur base des besoins, nous modéliserons statiquement la réalité d'un aéroport. Chaque élément du système pourra être modélisé par des classes et les liens existants entre ces classes seront signalés. Si ROPES ne juge pas cette approche comme relevante, nous avons jugé qu'il était utile de formaliser davantage notre analyse avant de passer au stade de la décomposition en « composants ».

En effet, parler de composants d'un système dont on n'a pas encore de vue statique des données utilisées et traitées par ces futurs composants peut paraître inepte. Par ailleurs, l'expérience nous a montré que la conception de diagrammes statiques permet de passer plus facilement à l'étape dynamique d'échange d'informations. Embrasser une réalité dans sa globalité permet de mieux percevoir les relations entre les éléments de cette réalité.

Dans un seconde étape, un diagramme de composants nous permettra d'avoir une vision globale des modules à l'oeuvre dans notre système.

Par un effet de loupe, nous entrerons dans un composant global (A-SMGCS) pour en analyser les sous composants. Chacun sera étudié et formalisé par des diagrammes d'états finis. Grâce à ces machines à états finis, nous pourrons plonger dans les contraintes temporelles plus fines. Les exigences mentionnées dans l'étape préalable seront prises en compte et formalisées.

Enfin, nous pourrons assembler chacun de ces composants pour en montrer les interactions et les éventuels conflits. Repérer ces conflits nous permettra de mettre le doigt sur des lacunes non repérables dans les phases préalables. En effet, il peut arriver que deux exigences mentionnées au stade de l'analyse des besoins se révèlent incompatibles par la suite car elles occasionnent des conflits. Sans un certain formalisme, ces conflits sont difficilement repérables.

3.4.2.3 Analyse Objet

Cette troisième étape dans notre processus d'analyse sera d'affiner les principaux objets déjà rencontrés dans l'étape précédente. Si les objets « métiers » ne seront que très peu changés, il n'en ira pas de même pour les objets liés au système à mettre en place (A-SMGCS). De boîte noire, ils sont devenus des composants autonomes ayant des comportements biens définis.

Ainsi, de composants, ces modules doivent devenir un ensemble d'objets/classes en interactions qui s'échangent de l'information. Les diagrammes de classe et de collaborations pourront nous aider dans cette modélisation.

3.4.3 Design

Si nous n'abordons pas précisément cette étape dans notre travail, nous proposerons une piste de recherche intéressante qu'il faudrait approfondir. En effet, une ébauche de solution permettant d'intégrer les différents services liés à la surveillance dans le monde de l'aéronautique a été proposée par Laurent Honet [HONET, 2002]. Il propose d'intégrer les services existants dans une architectures distribuées basée sur les technologies JAVA et JINI. Une étude spécifique à cette proposition technique devrait par la suite faire l'objet d'une étude plus approfondie.

3.4.4 Traçabilité

Aussi étonnant que cela puisse paraître, il n'est nulle part fait mention dans ROPES d'une quelconque volonté de traçabilité transversale tout au long du processus de développement. Cette absence est d'autant plus étrange que l'on connaît toute l'attention que peuvent porter les concepteurs du Processus Unifié[10] à cette problématique. Par ailleurs, dans des systèmes critiques, cibles idéales pour ROPES, il est fondamental de pouvoir retracer une histoire du développement. Chaque décision prise, chaque abandon a une signification et une répercussion sur le produit fini qui devra être livré. En effacer les traces revient souvent à complexifier toute évolution ou validation du système.

3.4.5 Critique générale

Une fois notre étude terminée, nous reprendrons les différentes critiques et remarques que nous aurons relevées tout au long de ce travail pour les synthétiser dans un chapitre dédié à l'analyse globale de notre démarche. Tant les points positifs que négatifs seront étudiés. Toute approche critique ne peut se terminer sans l'apport de solutions ou de pistes pour résoudre les problèmes soulevés. Nous tenterons donc de résumer les recherches actuelles qui tentent d'affiner les méthodologies de développement de systèmes temps réels et de l'utilisation de UML.

[10] Pour se faire une bonne idée

4 Analyse des besoins

4.1 Introduction

Dans cette partie de notre mémoire nous allons capturer les buts des différents acteurs qui seront appelés à utiliser le système, ainsi que les besoins de celui-ci. Après une présentation des notions de besoin et de contrainte pour des systèmes temps réels, nous caractériserons les principales exigences définies pour notre système.

Dans un premier temps, nous décrirons ces besoins sous forme d'un texte en langage naturel qui correspondrait dans un processus réel au résultat d'une série d'enquêtes réalisées auprès des acteurs d'un aéroport. Cette partie débute par la présentation des acteurs, suivie d'un diagramme modélisant leurs buts. Elle se poursuit par un résumé des fonctionnalités attendues et des contraintes.

Dans un deuxième temps, nous extrairons de ces besoins les éléments clés que nous modéliserons au moyen de Use Case permettant de montrer les interactions des acteurs avec le système.

Enfin, pour présenter un prototype qui puisse démontrer la validité de notre analyse et de notre démarche, nous imaginerons un scénario global illustré par un événement survenant dans l'aéroport, à savoir l'accident d'un camion sur une voie. Nous affinerons les Use Cases au moyen de diagrammes de séquences. Plus « formel », ce type de diagrammes permet d'une part de modéliser les échanges de messages entre composants et acteurs et d'autre part de pointer des contraintes temporelles fortes.

4.2 Types de besoins

Trois caractéristiques essentielles permettent de définir un système « temps réel » embarqué. D'une part, celui-ci doit répondre correctement à des stimuli. D'autre part, des contraintes temporelles entrent en jeu lors du traitement d'une requête. Enfin, un tel système embarqué impose que la puissance de calcul soit incluse dans le système.

Pour définir un système à temps réel, il existe deux types de besoins dont il faut tenir compte, les besoins de type fonctionnel d'une part, et les exigences de type « qualité de service » (QOS) d'autre part [DOUGLASS, *Capturing Real-Time Requirements,* 2001].

Les **besoins fonctionnels** regroupent toutes les exigences traitant de ce que doit faire le système et comment il devra réagir dans un ensemble de circonstances, par exemple :

- Le système devra prévenir le pilote d'un avion qu'un mobile se trouve sur la piste sur laquelle il va atterrir.
- Le système interdira tout décollage en cas d'accident sur la piste de décollage.
- ...

Les **besoins QOS** spécifient comment une exigence fonctionnelle doit être accomplie. Dans des systèmes temps réels ou embarqués, ces besoins de QOS peuvent préciser des propriétés du système (vitesse, sécurité, variations de données, fiabilité, ...).

Les besoins QOS sont souvent induits par les contraintes imposées par/sur les besoins fonctionnels.

- Le pilote d'un avion doit être prévenu plus d'une minute avant l'atterrissage prévu qu'un mobile se trouve sur la piste.

- Le système préviendra les secours d'urgence d'un accident avant de prévenir les techniciens.
- ...

L'approche utilisée dans cette étude se basera essentiellement sur la méthodologie proposée par Bruce Powel Douglass. Aussi, les besoins fonctionnels seront-ils modélisés par l'usage de Use Cases, de spécifications descriptives et des séquences de messages. Par ailleurs, les besoins QOS seront considérés comme des contraintes associées à une ou plusieurs exigences fonctionnelles.

4.3 Modélisation des besoins au moyen de Use Case[11]

Suite à la diffusion massive de UML, le modèle de Use Case a été largement adopté pour saisir les besoins « comportementaux » qui doivent être rencontrés par les systèmes. Ce modèle saisit les séquences d'interactions entre le système et les utilisateurs externes. Un Use Case décrit une voie spécifique à suivre par les utilisateurs externes, pour bénéficier/profiter du système.

Un Use Case peut être défini comme une collection cohérente de besoins apparentés organisée autour d'une capacité du système. Les caractéristiques d'un bon Use Case incluent :

- Le point de vue des acteurs – utilisateurs du système
- Un ensemble d'exigences apparentées
- Le retour du résultat à un ou plusieurs acteurs
- L'opacité de la structure interne du système
- L'indépendance vis-à-vis des autres Use Cases qui peuvent même être concurrents

4.4 Définition des acteurs

Un acteur peut être vu comme une entité externe au système qui interagit avec celui-ci. Typiquement, il peut stimuler le système par des inputs, ou être un récepteur d'événements émanant de ce système.

Le domaine étudié est à ce point vaste qu'il convient dans un premier temps de différencier les intervenants – acteurs qui pourront être regroupés selon des besoins spécifiques. En fonction de ces acteurs, nous élaborerons différents « Use Case » que nous confronterons entre eux. En effet, le risque existe que les besoins spécifiques des acteurs soient antagonistes. Ainsi, un pilote souhaitera faire atterrir son avion au plus vite alors qu'un pompier préférera intervenir sur les lieux d'un accident.

4.4.1 Acteurs

Chauffeur au sol : toute personne qui conduit un mobile dans l'enceinte de l'aéroport, sur les voies réservées à cet usage et/ou sur les pistes si besoin est.

[11] La capture des besoins au moyens des Use Case, pour les systèmes temps réel a été abordée dans de nombreux articles. Citons entre autres :[DOUGLASS, 2001 & 2003; ZHANG, 1999 ; HILLARY, 2003]

Deux possibilités existent quant à la définition de l'acteur « chauffeur ».

D'une part, il est possible de définir un seul acteur « Chauffeur au sol » et de spécifier tous les Use Cases imaginables pour cet acteur générique et éventuellement les « spécialiser » en fonction des rôles que celui-ci jouerait. Les antagonismes entre les objectifs des différents types de chauffeurs n'apparaîtraient donc pas directement dans les Use Case globaux de ces chauffeurs, mais ils pourraient être mis en lumière lorsque l'on définirait des scénarios plus spécifiques. Ainsi, le guidage au sol et l'attribution de priorité entre mobiles est identique pour le pompier comme pour le technicien. Cependant, la manière dont on prendra en compte les priorités entre ces deux « acteurs » variera en fonction de circonstances externes. En cas d'incendie, le système attribuerait un niveau de priorité supérieur au pompier. Par contre, si un avion est signalé en panne sur une voie de traverse de l'aéroport, le technicien aéronautique aura une priorité plus élevée.

D'autre part, la spécialisation directe des acteurs permet de les différencier et de saisir plus adéquatement leurs rôles, fonctions, objectifs et buts au sein du système.

Dans ce travail nous mélangerons les deux options. En effet, l'utilisation d'acteurs génériques permet de repérer rapidement les points communs entre les différents acteurs d'un système. Cependant, une spécialisation rapide des acteurs permet de rendre la première phase d'analyse plus compréhensible pour le donneur d'ordre et le client pour qui l'abstraction face à une réalité métier n'est pas immédiate. Aussi est-il important de conserver cette double approche comme un des fils conducteurs lors de notre étude.

Parmi tous les « chauffeurs au sol » nous avons relevé les catégories reprises ci-dessous. Il est évident que cette liste n'est pas exhaustive et que d'autres types de chauffeurs pourraient apparaître selon les circonstances, les aéroports, ...

Par ailleurs, les être humains ne sont pas « mono-fonctionnels ». Aussi, certains acteurs peuvent jouer plusieurs rôles. Dans notre étude, nous les considérerons comme « mono-fonctionnels ». Une personne qui est à la fois secouriste et technicien devra jouer un rôle bien précis à chaque instant.

> *Secouriste* : Un secouriste est tout individu, groupe d'individus, qui peut intervenir dans l'enceinte d'un aéroport quand un incident est signalé. Par secouriste on entend médecin, ambulancier, pompier, agent de la protection civile. Selon les cas, les types d'incidents, certains secours peuvent avoir une priorité par rapport à d'autres. Ainsi, si un incident implique des mobiles comportant des produits chimiques dangereux sans qu'il n'y ait eu de blessés graves, la protection civile et les pompiers ont priorité sur les médecins et ambulanciers. A l'inverse, si un accident entre deux mobiles au sol a fait des blessés sérieux sans que l'infrastructure de l'aéroport et/ou que la sécurité de celui-ci ne soit affectée, les ambulanciers et médecins auront priorité sur les pompiers.
>
> *Technicien au sol* : Toute personne qui s'occupe de l'entretien des installations de l'aéroport et des avions (plein de carburant, chargement-déchargement de cargaison, ...). Ces techniciens peuvent être appelés à se déplacer dans toute l'enceinte de l'aéroport. De nouveau, des systèmes de priorité peuvent être attribués à certains techniciens par rapport à d'autres.
>
> *Forces de l'ordre* : Toute personne ayant reçu un mandat pour faire respecter la loi et pouvant agir dans l'enceinte de l'aéroport : douanier, policier (civil ou militaire), services de sécurité officiels, ...Il faudrait pouvoir élargir cette catégorie en fonction des situations spécifiques. En effet, selon certaines occasions, certaines catégories peuvent également endosser ces rôles. Ainsi, des militaires peuvent jouer de telles fonctions en cas de guerre ou de menace.

Transport de passagers : Dans certains aéroports certains passagers sont amenés à leur avion via un système de navettes de bus. Ces bus doivent donc traverser certaines installations de l'aéroport depuis le hall d'embarquement jusqu'à l'avion lui-même.

Technicien radar : Acteur dont le rôle est de capturer des informations émanant tant des avions que des radars disposés à travers l'aéroport. Il peut en capturer toutes les informations.

Contrôleur : Acteur qui contrôle le trafic aérien et/ou le trafic terrestre sur base des informations qui lui sont transmises par différents systèmes. Il peut attribuer des routes, dévier des avions, ou d'autres véhicules, de leur trajectoire et informer les autres acteurs du système de l'état du ciel , du sol,.... Ces autres acteurs sont tout autant les pilotes, que les chauffeurs au sol, les responsables d'autres aéroports, ...

Parmi les contrôleurs, certains pourront avoir des responsabilités plus importantes que d'autres. Ces « super contrôleurs » auront les capacités de décisions finales en cas de doute. Ainsi, si le système A-SMGCS décide de prendre son autonomie ce contrôleur peut, d'initiative, reprendre la main sur le système. En effet, dans tout système automatisé d'aide à la décision, le choix final doit encore rester à l'être humain.

Système de contrôle aérien : Ce « legacy system » est celui qui permet aux contrôleurs aériens de gérer le contrôle aérien, de planifier les décollages et atterrissages au sein d'un aéroport, et de planifier les vols des avions. Ce système est complexe et est soumis à des contraintes de sécurité et de fiabilité très élevées. Il devra donc interroger le système de gestion du trafic terrestre afin de pouvoir planifier les différents mouvements d'avions.

Pilote : Toute personne qui pilote un avion. Le pilote est celui qui dirige cet avion, que ce soit en vol ou au sol. Pour un même avion, un pilote peut changer en cours de vol (pilote, copilote, pilote automatique,...). Une seule personne à la fois peut diriger un avion.

Système de senseurs coopératifs et dépendants[12] : composants autonomes associés à tous les mobiles devant circuler dans l'enceinte de l'aéroport et/ou dans les airs (GPS, Transpondeurs, ...). Ces composants permettent essentiellement de donner la position du mobile, sa direction, sa vitesse à tout instant. Ces composants sont soit interrogés par un système, soit ils donnent eux-mêmes les informations à un système. Ceci permet d'identifier les mobiles se déplaçant dans l'enceinte d'un aéroport.

Système de senseurs non coopératifs : composants autonomes quadrillant l'aéroport et réceptionnant des informations sur des mobiles, leur position, leur vitesse, leur direction. A l'inverse du système des transpondeurs, ces senseurs ne permettent pas d'identifier les mobiles dans l'enceinte de l'aéroport. Il permettent juste de repérer leur présence. Ces radars[13], senseurs, peuvent être isolés, indépendants ou regroupés en cluster, en cluster de clusters...

---

[12] Pour plus d'informations sur les différents senseurs (coopératifs, non coopératifs ou dépendants) : [VALLEE 2001].

[13] Pour une meilleure approche de ces concepts de transmissions d'informations par radar (cluster), cf. P.BARDET, *Software requirements document, cluster controller demonstrator*, Bruxelles, 2003 (Document confidentiel appartenant à la société THALES et donc non disponible).

Il est à noter que ces deux derniers acteurs (transpondeur – Système de senseurs) ne feront pas l'objet de ce travail. En effet, ils entrent en compte comme faisant partie du « legacy system ». Ce sont des acteurs techniques déjà intégrés dans le système complexe d'échange d'information dans un aéroport. Il n'est pas prévu de modéliser les systèmes complexes mis en place pour capturer les informations et les échanger. Ils sont mentionnés ici afin de savoir que ces deux mécanismes de « tracking » des mobiles existent. Seules les données relatives à l'état du trafic seront prises en compte, sans se soucier de leur provenance et de leur capture.

Cependant, il est à noter que le système de saisie des informations bruts par les différents capteurs est également un système temps réel très sensible. Le déterminisme de ce système en fait un nœud critique pour la sécurité et la gestion des risques dans un aéroport. Notre système considérera donc que les informations concernants tous les mobiles sont bien à jour et que le système qui les récupère et les sauve dans un repository adapté est efficient.

Il apparaît donc que le milieu dans lequel devra être utilisé le système de gestion du contrôle du trafic terrestre fourmille d'acteurs différents aux fonctions parfois différentes et aux intérêts parfois antagonistes.

4.4.2 Buts des acteurs

Afin de bien définir les fonctionnalités auxquelles devra répondre notre système, il convient de décrire quels sont les buts-objectifs de ce système pour les différents acteurs et d'indiquer parmi ces buts quels sont ceux qui sont antagonistes, ou ceux qui à l'inverse se recoupent. La méthode utilisée pour cette description est celle du diagramme des buts. En partant des buts généraux du système, nous les raffinerons pour finalement pointer les quelques buts concrets qui devront sans cesse nous aider dans les étapes de l'analyse.

La page suivante reprend ce diagramme.

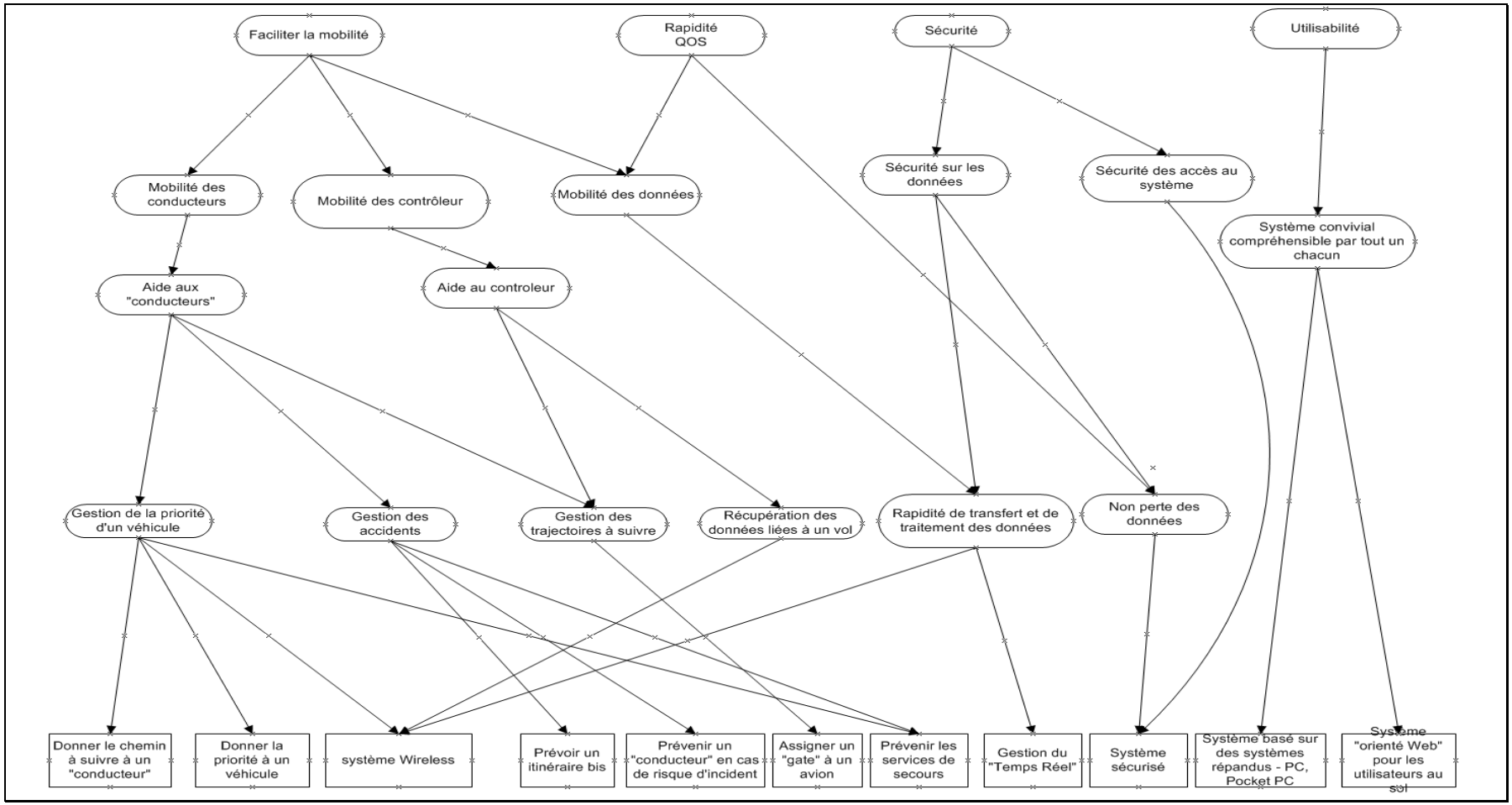

Figure 4.4.2 : Diagramme de but des acteurs

Le diagramme ci-dessus ne présente pas exhaustivement les buts des acteurs, il est davantage là à titre exemplatif afin de montrer qu'il est possible de les modéliser par une arborescence allant des buts les plus abstraits à des buts plus concrets.

Le système A-SMGCS sera plus complet et les buts ci-dessus constituent une infime partie des attentes des acteurs. Notre analyse n'abordera donc pas tous les thèmes.

4.5 Use Cases

Les interactions entre les différents acteurs et le système sont tellement nombreuses que nous ne souhaitons pas compliquer les schémas des Use Cases en voulant être exhaustif du point de vue du système. Ainsi, est –il souvent plus judicieux de découper les schémas des Use Cases par types d'acteurs, ou types de fonctionnalités repérées. Il peut arriver que certains Use Case concernent plusieurs acteurs. Dans ce cas, nous les mentionnerons pour chaque acteur!

Nous pourrons ainsi saisir rapidement les contraintes spécifiques pour chaque acteur et les confronter par la suite aux autres parties prenantes et pondérer les conflits - priorités entre ces acteurs.

Par ailleurs, après avoir défini les Use Cases généraux pour tous les acteurs, nous expliciterons quelques scénarios plus précis et plus proches d'une réalité au quotidien. Ces scénarios montreront comment des Use Cases « de base » peuvent s'imbriquer pour modéliser la réalité. Ainsi, nous verrons comment un incendie (ou tout autre accident) peut être scénarisé en utilisant plusieurs Use Cases prédéfinis et en les adaptant adéquatement, en fonction des acteurs impliqués.

4.5.1 Besoins globaux et contraintes du système

4.5.1.1 Présentation globale des besoins du système[14]

Le système devra rendre des services à classer dans deux grandes familles. D'une part, il devra faciliter la gestion du trafic terrestre dans l'enceinte des aéroports qui l'utiliseront. D'autre part, le système à développer devra également faciliter la capture d'informations des différents radars – capteurs et rendre les échanges plus aisés pour les techniciens radar.

Identification

- **Identification des personnes - acteurs**

Tous les acteurs qui voudront utiliser le système d'échange d'informations (transfert d'informations à distance et guidage au sol) devront s'identifier au système. Pour cela le système devra connaître chaque acteur préalablement. Chaque acteur pourra jouer un ou plusieurs rôles, mais un seul à la fois. Aussi, chaque utilisateur devra indiquer au système le rôle dans lequel il agira lors de son identification. En effet, une même personne pourrait éventuellement jouer plusieurs rôles (Pompier, ambulancier, ...).

Deux possibilités existent. D'une part, le système ne connaîtrait pas le(s) rôle(s) des acteurs, mais connaîtrait tous les rôles existant(s), et l'acteur choisirait parmi cette « liste » de rôle celui qu'il voudrait jouer. A l'inverse, il est imaginable que le système connaisse également le(s) rôle(s) que chaque utilisateur pourrait jouer. Lors de sa connexion, le système

[14] Pour plus d'information sur les besoins spécifiques des aéroports, des systèmes A-SMGCS, nous vous renvoyons aux articles traitant de ces sujets [VALLEE, 2001 ; LEMOINE, 2000 ; STOICA, 2002 ; BLAESS, 1996 ; CHENG, 2001 ; EUROPEAN ORGANISATION FOR THE SAFETY OF AIR NAVIGATION, *A-SMGCS*, 2003 ; EUROPEAN ORGANISATION FOR THE SAFETY OF AIR NAVIGATION, *Overall Architecture for EATMP*, 2002 ; PAPPAS, 1997 ; HONET, 2002]. Nous nous sommes également inspiré de documents de travail THALES, qui doivent rester confidentiels

proposerait à l'utilisateur de choisir un rôle parmi ceux qu'il peut jouer. C'est cette dernière solution qui a été choisie. En effet, il a été jugé trop risqué de laisser l'entière liberté à l'utilisateur de choisir le rôle qu'il veut jouer. Ainsi, on empêcherait à un technicien radar de se présenter comme force de l'ordre. Enfin, un utilisateur ne pourra pas changer de rôle quand il le souhaite. Il devra explicitement le demander et cette demande fera l'objet d'une vérification par le système qui jugera de l'opportunité de ce changement.

- **Identification des mobiles**

Parallèlement à l'identification des acteurs, le système devra également identifier les mobiles circulant dans l'enceinte de l'aéroport. Deux catégories de véhicules sont à distinguer. D'une part, les véhicules assurément identifiables sont ceux qui sont connus par le système. D'autre part, des véhicules non identifiés, intrus, qui peuvent toujours circuler dans l'enceinte de cet aéroport. Le système devrait pouvoir repérer tous ces véhicules.

Les mobiles « identifiables » seront tous munis d'un mécanisme automatique qui permettra à tout instant de les positionner dans l'enceinte. Ils seront donc tous munis d'un transpondeur qui communiquera régulièrement avec le système, lui envoyant des informations. Celles-ci permettront au système de connaître la vitesse du mobile, sa situation, sa destination, les passagers du mobile, le rôle joué par ce mobile....

Les mobiles non identifiables devront être repérés et suivis par le système de contrôle du trafic terrestre. Le système devra prévenir les différents acteurs qu'un véhicule intrus circule dans l'enceinte de l'aéroport. Des procédures d'encadrement, d'intervention devront être mises au point pour gérer ce mobile.

Tout véhicule non identifié par le système devra être pris en compte par ce système de suivi de mobile « extérieur ». Ainsi, si un transpondeur venait à tomber en panne, un véhicule précédemment connu du système serait encore « traqué » par celui-ci.

Cette traque de tous les mobiles au sol permettra au système de connaître à chaque instant l'état de l'aéroport. Il pourra donc le communiquer aux utilisateurs qui le souhaitent. Par ailleurs, un historique de tous ces mouvements devra pouvoir être conservé, et ce afin de pouvoir reconstituer l'état de la « circulation » à un moment donné ou lors d'une période donnée.

Association d'acteurs à des mobiles

Puisque tous les acteurs et que les mobiles « internes » seront identifiés et joueront un rôle précis, il sera possible d'associer des mobiles à des utilisateurs. Une des exigences est que chaque acteur puisse communiquer avec le système au moyen d'interfaces multiples et parfois mobiles (PDA, PC portables, GSM, ...). Aussi, les interfaces de communication entre les acteurs et le système seront associées aux acteurs. Pour des chauffeurs qui devront être avertis de la situation de l'aéroport, l'information sera donc envoyée sur son outil de communication et non pas au véhicule qu'il conduit. Aussi est-il indispensable que pour chaque vésicule en circulation, il soit possible de retrouver le chauffeur qui le conduit et par extension son outil de communication. A chaque instant un conducteur ne pourra être associé qu'à un seul véhicule.

Alerte

Dans un aéroport, tous les intervenants (acteurs connus ou inconnus et systèmes autonomes) peuvent déclencher des alertes en cas de danger. Le système à mettre en place devra être particulièrement réactif à tous les types d'alerte qui risquent de survenir. Un système autonome de la gestion des alertes est considéré comme préexistant. Le système qu'il faut mettre en place devra essentiellement s'occuper de la gestion des déplacements de mobiles en cas d'alerte. Cependant, qui dit indépendance et préexistance d'un système global de gestion des alertes, imposent un interfaçage avec le nouveau système de gestion du trafic terrestre.

Cet interfaçage peut se faire de deux manières :

- Soit le système d'alerte peut être vu comme un service qui est offert à tous. Tous les autres systèmes et sous-systèmes peuvent s'abonner à ce service d'alerte.
- Soit, le système d'alerte est un système global que l'on peut interroger, mais cette interrogation doit être laissée à la liberté des autres sous-systèmes.

Il conviendrait de prévoir un système qui puisse répondre à ces deux types. Cependant, dans notre analyse, nous favoriserons la première version. Ainsi, le système de gestion du trafic terrestre sera averti par le système d'alerte global lorsqu'un incident surviendra dans l'enceinte de l'aéroport.

La spécificité de chaque aéroport et de chaque organisation implique une flexibilité quant au traitement à prendre en cas d'alerte. Ainsi, chaque organisation devra définir elle-même les actions à prendre pour chaque type d'alerte. Il conviendrait de répertorier chaque type d'événements, de les classifier et de les identifier. Pour chaque alerte, un ensemble d'actions à prendre sera également précisé.

Lorsqu'une alerte est déclenchée, il est important que le système sache de quel type d'alerte il s'agit, de l'endroit concerné, des victimes éventuelles, ...

Aussi, a-t-il été décidé que le déclenchement d'alertes soit systématiquement accompagné d'informations minimales :

- Type d'alerte (ensemble de code répertoriant les incidents possibles)
- Urgence
- Lieu ou l'incident se produit
- Datation de l'incident
- Acteurs impliqués (connus ou non)

Le système devra donc prévenir les contrôleurs du trafic au sol ainsi que les différents acteurs de l'incident. En fonction du type d'alerte, différents scénarios devront être exécutés.

Il est évident que lorsqu'une alerte est déclenchée, le fonctionnement normal d'un aéroport s'en trouve modifié et que les réactions à prendre le sont au cas par cas. Aussi sera-t-il indispensable de planifier une analyse complète des scénarios possibles et des mécanismes d'interventions adaptés à mettre en oeuvre pour résoudre tout type d'alerte.

Définition de l'espace dans un aéroport

Un aéroport peut être vu comme un espace clos dans lequel évoluent des personnes et des véhicules sur des voies qui leurs sont dédiées et parsemé de bâtiments aux fonctions définies. Cet espace est découpé en zones cartographiées. Chaque endroit d'un aéroport doit donc pouvoir être situé précisément. Par ailleurs, les aérodromes actuels sont quadrillés par des radars, trackers, capteurs qui devront permettre de positionner chaque mobile évoluant dans l'enceinte.

Circulation dans un aéroport

- **Politique de déplacement**

Plusieurs politiques différentes sont possibles pour la gestion de la circulation des véhicules dans un aéroport. Ci-dessous nous exposerons deux grandes possibilités qui ont été évoquées et nous expliciterons les raisons qui nous ont poussé à privilégier la solution choisie.

La première politique qui se rapprocherait le plus de celle qui est en cours dans la vie courante serait de favoriser une certaine liberté de circulation. Liberté, qui serait contrainte par certaines règles. Ainsi, on pourrait établir un code de la circulation au sein de l'aéroport

auquel les véhicules seraient soumis. Les mobiles pourraient se déplacer librement, en respectant ce code, tant que le système (automatiquement ou via un contrôleur) ne les y aurait pas formellement interdit. Certaines pistes - voies seraient réservées à certains types de véhicules. Un ensemble de panneaux ou instruments de régulation de trafic fonctionnerait pour réguler, autoriser celui-ci. Des feux, panneaux, barrières ou tout autre type d'instrument permettraient de réguler et de contrôler le trafic.

L'autre politique qui a été retenue est de conditionner toute circulation à une autorisation de circuler. Une fois qu'un acteur - chauffeur de mobile est identifié et associé à un véhicule, il peut demander l'autorisation de circuler dans l'enceinte de l'aéroport. Avant tout déplacement d'un point à un autre il est impératif qu'une autorisation lui ait été donnée de se déplacer de ce point à cette destination via un chemin autorisé par le système.

Ainsi, vu l'objectif de prédictabilité souhaité par le A-SMGCS, il a été décidé de contraindre la circulation au maximum et de conditionner toute circulation à une autorisation préalable.

Avant tout déplacement dans l'enceinte de l'aéroport, une procédure de demande d'autorisation au système devra être introduite. Le système, selon les recommandations du A-SMGCS, devra soit décider seul de cette autorisation, soit transmettre la demande à des contrôleurs qui accorderont ou non l'autorisation.

Ainsi, pour chaque déplacement un accord devra être donné par le système-contrôleur. Cet accord sera accompagné par un « plan de circulation » qui imposera la route à suivre par le mobile. A tout instant le système et/ou le contrôleur pourra modifier le plan de route et le communiquer au conducteur du mobile.

Il est bien évident que cette demande de circulation doit être rendue la plus simple, transparente et rapide pour éviter des situations ubuesques dans lesquelles les mobiles seraient bloqués. Aussi, vu l'importance de certains déplacements et pour éviter des procédures trop longues, ces demandes d'autorisation devront être le plus simple possible, rapides et automatisées. En cas d'urgence, des priorités pourront être automatiquement données à certains types de véhicules.

De nouveau, ces procédures d'autorisation devront faire l'objet d'une étude particulière afin de relever tous les besoins des différents acteurs, des priorités de chacun, des urgences liées aux fonctions de chacun.

- **Connaissance de la situation de l'aéroport et prévision des incidents**

A chaque instant le système connaîtra la situation de chaque véhicule dans l'enceinte de l'aéroport. Il connaîtra également la destination et la vitesse de chaque véhicule identifié.

Enfin, le système devra vérifier que des véhicules circulant dans l'enceinte de l'aéroport ne risquent pas de se rencontrer et de provoquer un incident. Le système doit donc être prospectif. Un délai raisonnable avant « impact » pour éviter un accident devra être décidé par les différents intervenants. Ainsi, le système devrait sans cesse calculer les « impacts » possibles en fonction de l'emplacement de chaque mobile, de leur vitesse et de leur direction et de leur « plan de route ». Pour les mobiles « intrus », ce calcul sera rendu plus complexe, car le système devra « imaginer » la direction et la vitesse en fonction des informations qu'il aura pu glaner les instants auparavant (direction et vitesse au moment x). Par exemple, si un véhicule circule vers un bâtiment à une certaine vitesse depuis 10 secondes, il est probable qu'il se rende à cet endroit en conservant la même vitesse. En cas d'incident potentiel, le système devra signaler le danger à tous les acteurs impliqués (chauffeurs des mobiles impliqués et contrôleurs). La fréquence en mode automatique est toutes les 1/2 secondes. Par ailleurs, lorsqu'un contrôleur intervient également, des avertissements « humains » sont laissés à la liberté du contrôleur. Le système (automatique ou via un contrôleur) pourra intimer l'ordre à des chauffeurs de changer de direction, ou de s'arrêter à un endroit précis, pour éviter tout incident.

Il a été évoqué que le système devrait pouvoir prendre le contrôle de certains véhicules pour les forcer à stopper si les chauffeurs ne suivent pas les ordres qui leur sont intimés. Cependant, dans l'état actuel, il a toujours été jugé bien plus sûr de laisser à l'appréciation des chauffeurs humains des actions à prendre dans des cas d'urgence. Ainsi qu'il est signalé dans les recommandations du A-SMGCS, les conducteurs de mobiles doivent conserver une large part d'autonomie et de réactivité. Si le système ne parvient pas à tout prévenir, ou si les chauffeurs n'ont pas suffisamment de temps pour réagir, il faut laisser à l'appréciation des « humains-chauffeurs » la possibilité d'adapter eux-mêmes leur conduite.

Le système doit donc être vu davantage comme un outil facilitant la fluidité de la circulation et aidant les différents acteurs à se faire une idée de la situation globale de la circulation dans l'aéroport à chaque instant.

Gestion des atterrissages et décollages

Parallèlement au système de gestion du trafic terrestre, d'autres systèmes existent déjà et sont éprouvés depuis des années dans le cadre de la gestion du contrôle aérien et de la gestion des atterrissages, décollages au sein d'un aéroport. Il n'entre pas en compte dans notre étude de refondre la gestion de ce trafic aérien. Cependant, la gestion du trafic au sol aura une répercussion sur la gestion du trafic aérien, pour le moins en ce qui concerne les phases communes de vol - atterrissage/décollage.

Plusieurs possibilités existent pour le partage des responsabilité dans la gestion des mobiles entre les différentes phases :

- D'une part, le « legacy system » ATC (Air Traffic Control) pourrait s'interfacer avec le nouveau système de trafic au sol.
- D'autre part, le nouveau système pourrait être subordonné aux décisions prioritaires du système de trafic aérien préexistant.
- Enfin, les deux systèmes pourraient être indépendants et ne pas communiquer entre eux.

Les phases de décollages et d'atterrissages sont reconnues comme les plus dangereuses du cycle de vol d'un avion, il convient donc de donner la priorité aux avions qui sont dans cette phase. Aussi a-t-il été décidé de rendre le nouveau système de « contrôle du trafic au sol » dépendant du « legacy system » de gestion du trafic aérien, des décollages et atterrissages. Ainsi, le système de contrôle aérien décidera de la faisabilité ou non d'un atterrissage et aura la priorité sur l'affectation des pistes. Le nouveau système de contrôle terrestre devra prendre en charge les avions ayant atterris afin de les mener à leur bonne destination au sol. Ils seront en effet devenus de simples mobiles terrestres au même titre que les autres véhicules. Parallèlement, les avions qui devront décoller seront menés sur la piste de décollage et le « legacy system » de contrôle du trafic aérien gérera les décollages.

4.5.1.2 Contraintes

Fiabilités

Il est évident que le système que nous devons mettre en place ne peut jamais être défectueux. Il devra être fonctionnel à tout instant. Tous les scénarios devront donc pouvoir se réaliser. Si des défaillances surviennent soit dans le système, soit chez les acteurs, il est important que des mécanismes de rechange puissent être mis en place sans que les utilisateurs humains ne se rendent compte de ces défaillances. Par ailleurs, si des utilisateurs ne peuvent plus bénéficier des aides du système, alors un ensemble de procédures d'urgence à suivre seront décrites spécifiquement pour chaque type d'acteur. Ces procédures devront donc être décrites par les responsables spécifiquement pour chaque aéroport.

Par ailleurs, en plus de sa disponibilité continue, chaque contrainte décrite dans ce document devra être respectée. Ainsi, les délais maximum et minimum autorisés pour l'accomplissement des tâches ne pourront jamais être dépassés.

Systèmes existants et règles internationales

Le nouveau système de guidage des mobiles au sol devra répondre aux exigences imposées par la normalisation A-SMGCS[15] tel que décrit auparavant dans ce document.

Par ailleurs, le système qui doit être mis en place devra communiquer avec plusieurs systèmes déjà opérationnels. Ainsi, il faudra qu'il satisfasse les systèmes de contrôle aérien, les systèmes de « tracking » des mobiles au sein de l'aéroport et les systèmes de sécurité déjà implémentés.

Le postulat que nous suivrons est que les systèmes de sécurité avec lesquels notre système devra communiquer sont des systèmes qui envoient des informations lors d'une alerte. Ainsi, si une alerte est enclenchée, le système de sécurité prévient les autres sous-systèmes de cette alerte. Nous ne tenons pas à préjuger des méthodes utilisées.

Facilité d'utilisation du système et mobilité des acteurs

Vu la mobilité de nombreux acteurs et leur diversité, il est souhaitable que le système à mettre en place soit très aisé d'utilisation. Du point de vue des utilisateurs, les interfaces d'utilisation doivent être à la fois mobiles et fixes et d'une utilisabilité naturelle pour toute personne déjà familiarisée avec l'usage d'outils bureautiques traditionnels. Aussi, les donneurs d'ordres souhaiteraient que ce système puisse communiquer avec les utilisateurs sur tout type de plate-forme sur lesquelles tournent les programmes « bureautiques » déjà utilisés (PC - tout OS, Station UNIX, Mac, PDA, ...). Enfin, la mobilité des acteurs et des mobiles implique celle des outils de communication. Aussi les acteurs doivent pouvoir rester en contact avec le système de guidage - contrôle du trafic à tout moment et à chaque endroit de l'aéroport.

Liberté d'appréciation des chauffeurs

Bien que le système à mettre en place doit entre autres aider les chauffeurs de mobiles au sol à se déplacer, il est important de constater que les normes du A-SMGCS laissent encore une grande liberté d'appréciation à ces chauffeurs. Ainsi, la décision finale de suivre les injonctions émanant du système (soit automatisé soit via les ordres des contrôleurs au sol) reste du ressort des chauffeurs. Le système ne pourra donc pas prendre en main le guidage à distance des mobiles.

Cependant, le système à mettre en place doit être suffisamment évolutif que pour permettre l'adaptation vers de telles fonctionnalités dont les besoins ne sont pas encore définis.

4.5.2 Présentation des Use Cases

Nous avons décidé de représenter certains besoins auxquels le système doit répondre. Ceux que nous traitons dans ce travail sont essentiellement induits par les recommandations du A-SMGCS. En effet, d'autres types de besoins ont été évoqués mais il ne feront pas partie de ce travail. Nous avons donc décidé de ne pas représenter tous ces scénarios. Pour certains besoins, plusieurs possibilités de « traitement » étaient souvent possibles, et nous avons donc parfois opté pour des solutions particulières.

[15] A-SMGCS :

Ci-dessous, sont repris les Use Cases que nous étudierons. Ceux-ci sont répartis en deux catégories, les Use Cases globaux qui décrivent des interactions identiques par différents acteurs et les Use Cases davantage liés aux chauffeurs. Si nous devions étudier le système au complet, nous élargirions ce tableau aux autres acteurs (Contrôleurs, système d'alerte, système de tracking, ...).

Chaque Use Case est identifié et nommé. Cette identification permet par la suite de faciliter leur modélisation et les référencement entre eux.

Identifiant	**Nom**
UC. Globaux	
UC. 1	Identification d'un acteur
UC.2	Identification d'un mobile connu du système
UC.3	Association Acteur – Mobile
UC.4	Alerte
UC.5	Repérage mobile intrus
UC.6	Alerte danger imminent
UC.7	Vérification de l'autorité du système
UC. Chauffeurs	
UC.100.1	Autorisation de circuler
UC.100.2	Attribution priorité
UC.100.3	Guidage au sol
UC.100.4	Vérification situation aéroport

Tableau 4.5.2 : Tableau des Use Cases

Ces Use Cases décrivent donc des scénarios de base auxquels le système devra répondre pour rendre les services qui lui sont demandés. Il est bien entendu que des contraintes bien plus strictes que celles définies dans les Use Cases devront être précisées lors de l'élaboration des scénarios plus concrets. Il faut davantage voir ces Use Cases comme étant des contraintes de base qui lorsqu'elles seront traduites dans des plans d'actions plus vastes devront être « surchargés » en terme de contraintes. Ainsi, la gestion d'un trafic terrestre en temps normal n'est pas identique à la gestion de ce même trafic lorsqu'un incendie grave éclate ou si une collision entre plusieurs mobiles provoquent des dysfonctionnements.

A la suite de la présentation de ces cas, nous présenterons un scénario plus « réel » qui permettra de montrer comment ces cas interagissent entre eux pour rendre un service plus global. Ainsi, si un incendie se déclare dans l'enceinte de l'aéroport, les différents acteurs repérés devront utiliser le système adéquatement pour résoudre ce problème au plus vite et en suivant les règles établies.

Chaque Use Case sera présenté dans un formulaire dont l'explication a été donnée dans la partie méthodologique de ce travail.

4.5.3 Use Cases globaux au système

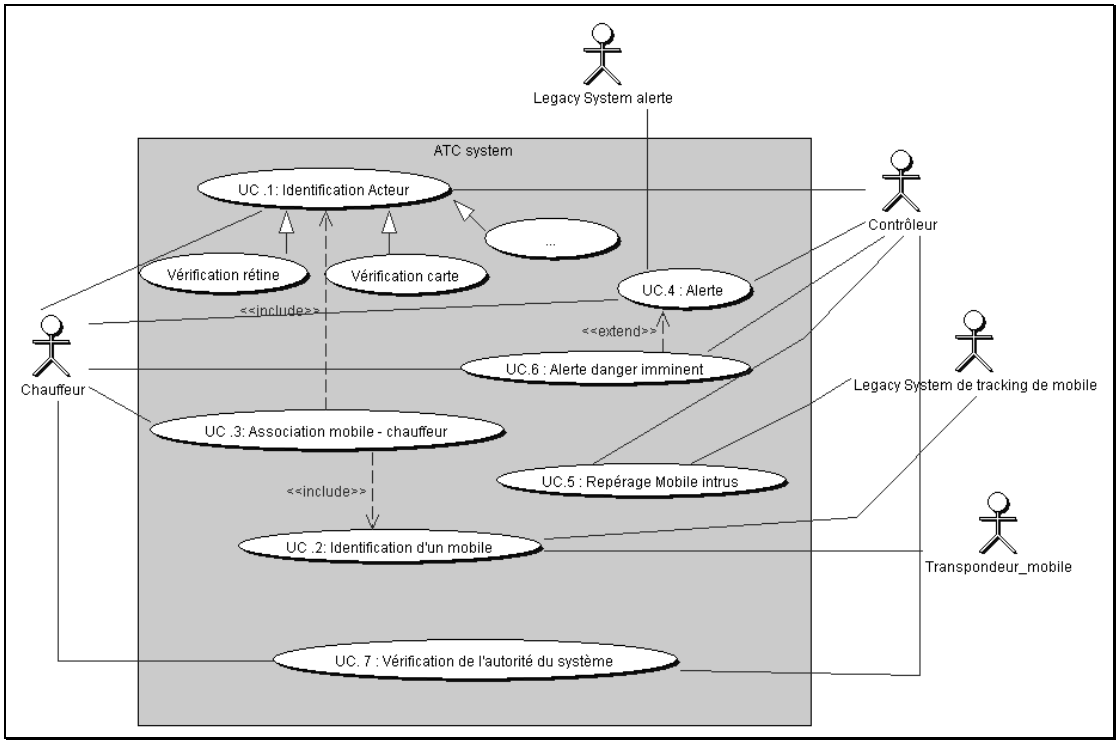

Figure 4.5.3.: Diagramme des uses cases communs

4.5.3.1 UC. 1 Identification d'un acteur

Description : Ce Use Case présente l'identification d'un acteur au système
Acteur(s) primaire(s):
Tous les acteurs «Personne Physique »
Extensions
/
Inclusions
/
Scénario principal
Préconditions
L'acteur est enregistré dans le système. Le système connaît tous les rôles que cet acteur peut jouer. Un acteur possède au moins un rôle dans le système. Rôles : Secouriste Force de l'ordre Technicien au sol Transporteur de passagers Technicien radar

Pilote de ligne .... (à définir par les différents acteurs de l'aéroport)	
Action de(s) acteurs	**Responsabilité du système**
1. L'acteur s'identifie auprès du système.	2. Le système vérifie l'identité de l'acteur. 3. Le système retrouve tous les rôles qu'un acteur peut jouer dans le système. 4. Le système demande à l'acteur quel est le rôle qu'il souhaite jouer
5. L'acteur choisit le rôle qu'il veut jouer parmi la liste qui lui est proposée.	6. Le système enregistre le choix et enregistre l'identification de l'acteur (identifiant unique, date, rôle)
Postcondition	
L'acteur est identifié dans le système. Celui-ci connaît le rôle qu'il joue.	
Flux d'exception 1	
Ce flux d'exception peut survenir si des informations envoyées au système ne sont pas correctes, si une personne possède des informations sur une personne ne travaillant plus dans l'aéroport.	
Action de(s) acteurs	**Responsabilité du système**
5. L'acteur ne peut « utiliser » le système.	3. Le système ne connaît pas cet acteur 4. Le système refuse l'identification et rejette l'acteur
Contraintes	
Chaque acteur humain doit rester en contact permanent avec le système tant dans ses déplacements dans l'enceinte de l'aéroport, que lorsqu'il est amené à rester au même endroit pendant son activité. Aucune contrainte forte d'un point de vue « temps réel » n'est associée à ce Use Case. L'important est que cette association se fasse dans un délai raisonnable. Un maximum de 5 secondes est accepté comme temps de réponse aux inputs de l'acteur pour que celui-ci soit identifié ou rejeté.	

4.5.3.2 UC. 2 : Identification d'un mobile connu du système

Description : Ce Use Case présente l'identification d'un mobile au système
Acteur(s) primaire(s):
Transpondeur (émetteur – récepteur)
Extensions

/
Inclusions
/
Scénario principal
Précondition
Senseur (transpondeur, GPS, ..) est connu du système et est actif. Il est associé à un mobile. Tous les mobiles qui sont amenés à rouler dans l'enceinte de l'aéroport ont un transpondeur actif. Par actif, cela signifie que le senseur est en contact continu avec le système de tracking(à intervalle régulier à définir selon les types de mobiles). Si le mobile est conduit par un chauffeur, celui-ci est associé au mobile. Tous les acteurs « passagers du mobile » sont également associés au mobile.

Action de(s) acteurs	**Responsabilité du système**
1. Le senseur associé à un mobile envoie un signal au système comprenant les informations suivantes. « Identité » : identifiant unique par mobile. Situation dans l'espace Moment : date – heure – minute – seconde Vitesse de déplacement Destination prévue (s'il y en a une de définie) 4. Le senseur associé au mobile acte de cet enregistrement et se met en attente jusqu'à sa prochaine identification	2. Le système reçoit ces signaux, les enregistre et les vérifie 3. Le système confirme au senseur la validité de sa requête et lui signale qu'il est bien enregistré dans le système. Le système connaît donc à cet instant l'emplacement du mobile au sein de l'enceinte de l'aéroport, sa direction, sa vitesse et sa destination, rôle. Il tient à jour la situation globale de l'aéroport.
Postcondition	

Le mobile est identifié et situé dans l'espace et le temps.
Flux d'exception 1
Cette exception peut survenir en cas de panne temporaire du système de senseurs

Action de(s) acteurs	**Responsabilité du système**
	3. Le système ne reconnaît pas le mobile. 4. Le système impose un identifiant connu au mobile
5. le mobile accepte et le communique au système 7. le mobile enregistre cet acceptation	6. Le système enregistre cet identifiant et le signale au mobile

Contraintes
Le système d'identification d'un mobile doit se faire en continu. Une fois qu'un cycle normal

est terminé, un nouveau cycle d'identification doit être relancé dans la demi seconde qui suit. Un cycle d'identification ne doit jamais dépasser un dixième de seconde entre le moment où le transpondeur d'un mobile envoie son signale d'identification et l'enregistrement par le transpondeur que tout s'est bien passé.

4.5.3.3 UC. 3 : Association Acteur – Mobile

Description : Ce Use Case présente les étapes de l'association d'un acteur à un mobile	
Acteur(s) primaire(s):	
Chauffeur au sol Pilote de ligne Transpondeur	
Extensions	
/	
Inclusions	
UC.1. Identification d'un acteur UC.2. Identification d'un mobile connu du système	
Scénario principal	
Précondition	
Les acteurs sont tous identifiés auprès du système. L'acteur humain joue un rôle bien précis. Tous les acteurs qui souhaitent être véhiculés par un mobile (dans le cadre normal) doivent demander à être associés à ce mobile. Les mobiles sont tous connus du système, identifiés et positionnés. A chaque mobile est associé un ensemble de rôles que celui-ci peut jouer (mobile de pompier – ambulance –force de l'ordre-....)	
Action de(s) acteurs	**Responsabilité du système**
1. Un acteur demande au système pour utiliser un mobile connu du système comme conducteur. Il donne au système son identifiant et l'identifiant du mobile	2. Le système vérifie l'identité de l'acteur et du mobile 3. Le système vérifie que le rôle joué par l'acteur est compatible avec l'un des rôles possibles du mobile. 4. Le système associe l'acteur avec le mobile. Il attribue le rôle adéquat au mobile, i.e. le rôle de l'utilisateur.
5. D'autres acteurs peuvent s'associer à ce mobile. Chaque acteur donne son identifiant et l'identifiant du mobile.	6. Pour chaque acteur, le système enregistre le couple acteur –mobile.
Postcondition	
L'acteur est associé à un mobile et le rôle du mobile est identique à celui de l'acteur. L'acteur	

est considéré comme le chauffeur. Il conduit donc un mobile au rôle bien défini.	
Flux d'exception 1	
Cette exception peut survenir si un mobile est déjà associé à un chauffeur, ou que le chauffeur ne peut conduire ce type de mobile.	
Action de(s) acteurs	**Responsabilité du système**
	3. Le mobile ne peut être associé à cet acteur, car le rôle de l'acteur ne correspond à aucun rôle possible du mobile. 4. Le Système refuse le droit à l'acteur d'être le chauffeur de ce mobile.
5. L'acteur est averti qu'il ne peut utiliser ce mobile en jouant le rôle qu'il a choisi.	
Flux d'exception 2	
Cette exception peut survenir si le chauffeur est déjà associé à un autre véhicule.	
Action de(s) acteurs	**Responsabilité du système**
	2. Le système constate que l'acteur est déjà associé à un autre mobile. Il ne peut donc être chauffeur de deux mobiles simultanément. 3. Le système refuse le droit à l'acteur d'être le chauffeur de ce mobile.
4. L'acteur est averti qu'il ne peut être le chauffeur de ce mobile. Il doit d'abord se délier de l'autre mobile et réintroduire une demande.	
Contraintes	
Aucune contrainte forte d'un point de vue « temps réel » n'est associée à ce Use Case. L'important est que cette association se fasse dans un délai raisonnable. Un maximum de 1,5 seconde est accepté comme temps de réponse aux inputs de l'acteur. Par ailleurs tout association à un mobile doit être unique. Pour pouvoir s'associer à un autre mobile il est indispensable de se « dés-associer » du mobile avec l'acteur est lié.	

4.5.3.4 UC. 4 : Alerte

Description : Ce Use Case présente les actions du système en cas d'alerte.
Acteur(s) primaire(s):
Contrôleurs, chauffeurs, pilotes, système global de gestion des alertes,... tout acteur ayant accès au système de déclenchement d'alerte – tout être humain qui pourrait actionner une des alarmes disposées dans l'enceinte de l'aéroport. Tout mécanisme automatique ...
Acteur(s) de support
Extensions

Scénario principal	
Précondition	
Un problème est perçu par un acteur. Un système global et indépendant de gestion des incidents est prévenu par l'acteur.	
Action de(s) acteurs	**Responsabilité du système**
1. Un acteur signale un « problème » au système de gestion des incidents. Ce « legacy system » prévient le système de gestion du trafic au sol et signale le problème Ce problème est : Qualifié (Incendie, accident, problème technique,....) par un code[16] - Incendie d'un mobile transportant des produits dangereux - Incendie d'un mobile transportant des personnes - Incendie d'un mobile transportant – une cargaison non dangereuse - Incendie d'un bâtiment abritant des produits dangereux - .... Quantifié :non urgent, urgent, très urgent, crucial, ... Situé Daté (date, heure) Acteurs impliqués identifiés (si mobiles – identifiants des mobiles, nombre de passagers...)	2. Le système est averti du problème. 3. Le système enregistre les données du problème et avertit les acteurs concernés (en fonction de la qualification et des quantifications de l'alerte). Tous les contrôleurs au sol sont avertis ! 4. Le système désigne un contrôleur chargé de s'occuper de la gestion de cette crise.
5. Les acteurs concernés sont avertis de l'alerte 6. Les acteurs appliquent les procédures à suivre dans chaque cas particulier.	
Postcondition	
Le système a enregistré l'alerte et a désigné un contrôleur qui doit s'occuper de la gestion du	

[16] Une liste reprenant les problèmes potentiels sera fournie. Cette liste reprendra les problèmes et leur code identifiant.

trafic au sol par rapport à cette alerte. Le système global de gestion des alertes est prévenu et a prévenu les différents services qui doivent s'occuper de régler cette alerte. Les actions sont prises en conséquence, en fonction du type d'alerte..
Contraintes
Vu l'importance de cette prise en charge de l'alerte, il convient que le délai entre la déclaration du problème et l'alerte donnée soit immédiate (de l'ordre du dixième de seconde).

Il apparait donc important pour la bonne mise en place des fonctionnalités liées à ce Use Case de prévoir tous les scénarios possibles pour chaque type d'alerte.

Le travail des différents experts en sécurité aéroportuaire serait donc de relever tous les incidents potentiels, de définir clairement les processus de traitement en cas de problème, les acteurs impliqués et les procédures d'urgence à appliquer selon les cas.

Notre travail, n'ayant pas comme objectif de définir exhaustivement tous les événements qui peuvent survenir dans un aéroport, nous étudierons le fonctionnement d'un seul cas, l' incident d'un camion sur une voie de circulation de l'aéroport. Par cet exemple, nous tenterons de montrer les intérêts et les limites de la technique d'analyse adoptée. Nous proposerons enfin des pistes permettant d'améliorer ce type d'analyse.

4.5.3.5 UC. 5 : Repérage mobile intrus

Description : Ce Use Case présente les impacts du repérage d'un mobile intrus dans l'enceinte d'un aéroport.	
Acteur(s) primaire(s):	
Système de Senseurs (indépendants)	
Acteur(s) de support	
/	
Extensions	
/	
Inclusions	
/	
Scénario principal	
Précondition	
Le système autonome des senseurs à l'écoute des mobiles évoluant dans l'enceinte de l'aéroport. Ce système ne fait aucune différence entre les mobiles identifiés et les mobiles non identifiés. Le système a repéré un mobile évoluant dans l'enceinte de l'aéroport.	
Action de(s) acteurs	**Responsabilité du système**
1. Senseur signale au système qu'un mobile est repéré à un endroit dans l'enceinte de l'aéroport. Il signale au système, la date, sa position et sa direction.	2. Le système enregistre cette information. Et signale au système de senseurs qu'il a bien enregistré l'information. 3. Le système vérifie que ce mobile n'a pas déjà été identifié (système

	d'identification des mobiles) 4. Si le mobile est déjà identifié alors le système ignore l'information. Si le mobile est inconnu, le système enregistre ce nouveau mobile (situation, date, direction) et lui donne un identifiant temporaire. Il informe les contrôleurs au sol, ainsi que les services d'ordre qu'un mobile inconnu se déplace dans l'enceinte de l'aéroport (UC Alerte).
Postcondition	
Le système de senseurs continue son observation de l'enceinte de l'aéroport. Il agit de manière autonome et signale	
Contraintes	
Tout véhicule non identifié circulant dans l'enceinte de l'aéroport doit être repéré et enregistré dans le système dans les 30 secondes qui suivent son intrusion dans cette enceinte	

4.5.3.6 UC.6 : Alerte danger imminent

Description : Ce Use Case présente les actions que doit prendre le système s'il repère qu'un danger est imminent.	
Acteur(s) primaire(s):	
Chauffeurs au sol, contrôleur	
Acteur(s) de support	
Transpondeur	
Extensions	
UC.4 Alerte UC.100.3 : Guidage au sol UC. 100.4 : Vérification situation aéroport	
Inclusions	
Scénario principal	
Précondition	
Les mobiles circulant dans l'enceinte de l'aéroport sont tous repérés, tant les identifiés que les intrus. Le système connaît la destination, l'emplacement et la vitesse de tous les véhicules identifiés. Le système connaît l'emplacement des véhicules « intrus ». Il anticipe leur direction et leur vitesse sur base de l'historique.	
Action de(s) acteurs	**Responsabilité du système**
	1. Le système constate que plusieurs mobiles risquent de se rencontrer et de

	provoquer un incident. Ce risque est calculé. Exemple : « Si les véhicules continuent à évoluer dans les même conditions, ils se rencontreront dans les 5 secondes ». 2. Le système tente de recalculer une route pour les véhicules concernés. 3. Le système prévient les chauffeurs des véhicules concernés (si identifiés) et les contrôleurs au sol ; et si une nouvelle route a été trouvée il la transmet.
4. Les contrôleurs sont prévenus du danger. Ils peuvent prendre des mesures en fonction des informations dont ils disposent (nouvelle route, situation de l'aéroport à ce moment, ..). 5. Les chauffeurs sont avertis du danger. Par défaut, ils doivent suivre les ordres qui leur ont été envoyés (arrêt, changement de route, ...). Cependant, vu leur autonomie de décision, ils peuvent apprécier ce danger et adapter leur vitesse, et s'arrêter sur les bas-côtés ou réaliser toute manœuvre d'évitement.	
Postcondition	
Tous les véhicules concernés sont prévenus et si une route de rechange a été calculée elle leur a été transmise. Tous les contrôleurs sont prévenus.	
Flux d'exception 1	
Cette exception peut survenir si des mobiles intrus sont dans l'enceinte et concernés par l'incident. Elle peut également survenir si le lien est coupé avec les chauffeurs des mobiles concernés.	
Action de(s) acteurs	**Responsabilité du système**
	3. Le système ne parvient pas à prévenir tous les chauffeurs des mobiles concernés car certains mobiles sont des « mobiles intrus ». Il prévient le système autonome d'alerte en donnant les infos concernant cette alerte.
4. Le système d'alerte lance une alerte (reprise du UC. 4) 5. Les contrôleurs sont avertis avant qu'une alerte officielle ne soit lancée. Ils peuvent donc prendre des actions autonomes. 6. Les chauffeurs de mobiles qui ont réussi	

à être prévenus suivent les instructions qu'ils ont reçues.	

Contraintes
Toutes les demi-secondes, le système doit pouvoir calculer la situation de l'aéroport. L'évaluation des risques d'un incident doit être terminée en moins de 5 millisecondes. Pendant ce laps de temps, le système doit avoir prévenu tous les acteurs impliqués. Toute une série de contraintes bien plus complexes doivent diriger les calculs complexes de « recalcules » d'un chemin. De même, un ensemble de contraintes mathématiquement complexes permettront au système d'évaluer les risques de collision.

4.5.3.7 UC. 7 : Vérification de l'autonomie du système

Description : Ce Use Case présente le contrôle permanent que doit faire le système pour vérifier s'il doit être autonome.	
Acteur(s) primaire(s):	
Contrôleurs au sol	
Acteur(s) de support	
Repository de l'aérodrome.	
Extensions	
Scénario principal	
Précondition	
Le système est initialisé et est toujours en activité. Les acteurs sont tous identifiés.	
Action de(s) acteurs	**Responsabilité du système**
1. Un contrôleur considère que la situation n'est plus gérable pour lui et demande au système de prendre la main quant aux prises de décision concernant la gestion du trafic au sol pour une zone précise ou l'aéroport entier, pour une durée déterminée ou indéterminée)	2. Le système prend acte de la décision du contrôleur. Il lui demande confirmation de cette demande.
3. Le contrôleur confirme ou infirme (si infirme alors, le UC s'arrête)	4. Le système acte la décision du contrôleur et prend la main sur la gestion du contrôle du trafic et des décisions à prendre en cas de conflit, de demande d'autorisation, ... 5. Le système analyse chaque situation (alerte, demande d'autorisation, risque d'accident, ...) et choisit une solution selon un ensemble de règles.

	6. Pour chaque décision prise, le système avertit les acteurs concernés (chauffeurs de mobiles,...) et les contrôleurs. Chaque décision prise par le système est enregistrée par le système. Celui-ci garde son autonomie aussi longtemps qu'un contrôleur ne décide de reprendre la main.
7. A chaque instant, un contrôleur peut décider de reprendre le contrôle sur les prises de décisions concernant la gestion du trafic terrestre, pour autant que la cession de contrôle ait été déléguée par un contrôleur. Il en informe le système.	8. Le système perd son autonomie et le signale à tous les acteurs concernés.

Flux alternatif

Il s'agit d'un autre flux possible, non lié à une exception.

Action de(s) acteurs	**Responsabilité du système**
	1. Le système vérifie à intervalle régulier s'il doit prendre la main sur la gestion du contrôle du trafic. 2. Le système calcule cette « autonomie » sur base de multiples données accessibles (incident, visibilité, ...). 3. Le système décide de prendre la responsabilité de la gestion du trafic pour une zone précise ou pour plusieurs zones. Il prévient tous les acteurs concernés (chauffeurs de mobiles présents dans cette (ces) zone(s) ou devant y passer, contrôleurs,...).
4. Les acteurs sont prévenus de cette prise d'autonomie par le système.	5. Le système vérifie à intervalles réguliers que cette autonomie est encore d'actualité. Si oui, le système conserve cette autonomie. Si non, elle l'abandonne et rend la main au(x) contrôleur(s). 6. Si le système abandonne son autonomie, il le communique aux contrôleurs.
7. A tout instant, un super-contrôleur peut annuler l'autonomie du système pour une durée limitée (à spécifier : 5 min, 30 min, 1h, ...).	8. Si le système reçoit cette « annulation d'autonomie pour x temps », il rend la main aux contrôleurs. Il continue à « contrôler » s'il doit prendre la main.
Postcondition	

Le système conserve son autonomie tant que celle-ci ne lui est pas explicitement enlevée ou que le système considère que la situation n'en vaut plus la peine.

Le système continue à rendre les autres services qu'il rendait auparavant.

A tout instant, le système peut abandonner son « autonomie » soit en fonction des données

reçues et analysées, soit suite à une injonction d'un contrôleur.
Contraintes
- La décision de prise de contrôle du système doit être prise en fonction de critères précis à définir (visibilité, conditions climatiques, incidents dans l'aéroport, ...) - Il faut éviter tout blocage. Ainsi, un super-contrôleur doit toujours être disponible immédiatement (dans la minute) pour enlever le pouvoir au système.

4.5.4 Use Cases relatifs aux chauffeurs au sol – Tronc commun

Ainsi qu'il a été indiqué plus haut, la stratégie qui a été décidée pour définir les chauffeurs au sol est de les répertorier exhaustivement[17] selon leur « rôle » dans le système et de définir quelques Use Cases particuliers permettant de saisir les besoins et contraintes pour ces différents acteurs.

Cependant, la quasi totalité des Use Cases de base sont communs à tous les acteurs « chauffeurs ». Ce sont les circonstances réelles dans lesquelles ils sont « appelés » qui définiront les priorités, tâches et obligations respectives des différents acteurs. Ainsi, le Use Case définissant le guidage au sol sera le même pour un pompier et pour un technicien. Seules les priorités et les autorisations spéciales varieront en fonction des circonstances.

Lorsqu'un technicien devra se rendre auprès d'un avion en panne, il aura sans doute priorité sur un mobile de pompiers qui se rend à la réserve de carburant pour faire un plein. A l'inverse, en cas d'incendie dans l'aéroport, les pompiers seront prioritaires par rapport à des techniciens.

Toutes ces règles de priorités liées aux différents événements potentiels devront être décrites par les responsables des aéroports et être intégrées au système mis en place.

Dans cette section, nous exposerons donc le tronc commun des besoins pour les utilisateurs « chauffeurs au sol ». Enfin, afin de faciliter la compréhension des Use Cases qui suivent il est utile de rappeler sommairement les buts principaux de ces chauffeurs concernant l'utilisation de ce système de guidage :

- Gestion des autorisations de circuler dans l'enceinte de l'aéroport
- Guidage au sol des chauffeurs de mobiles terrestres
- Gestion des priorités entre mobiles
- Prévenir les différents chauffeurs en cas d'accident potentiel entre mobiles
- Prévenir les différents chauffeurs qu'un incident s'est commis dans l'enceinte de l'aéroport

Le schéma ci-dessous reprend donc tous les scénarios que nous avons pu élaborer à partir des descriptions fonctionnelles.

[17] Nous entendons ici l'exhaustivité au point de vue du sous-système limité que nous étudions.

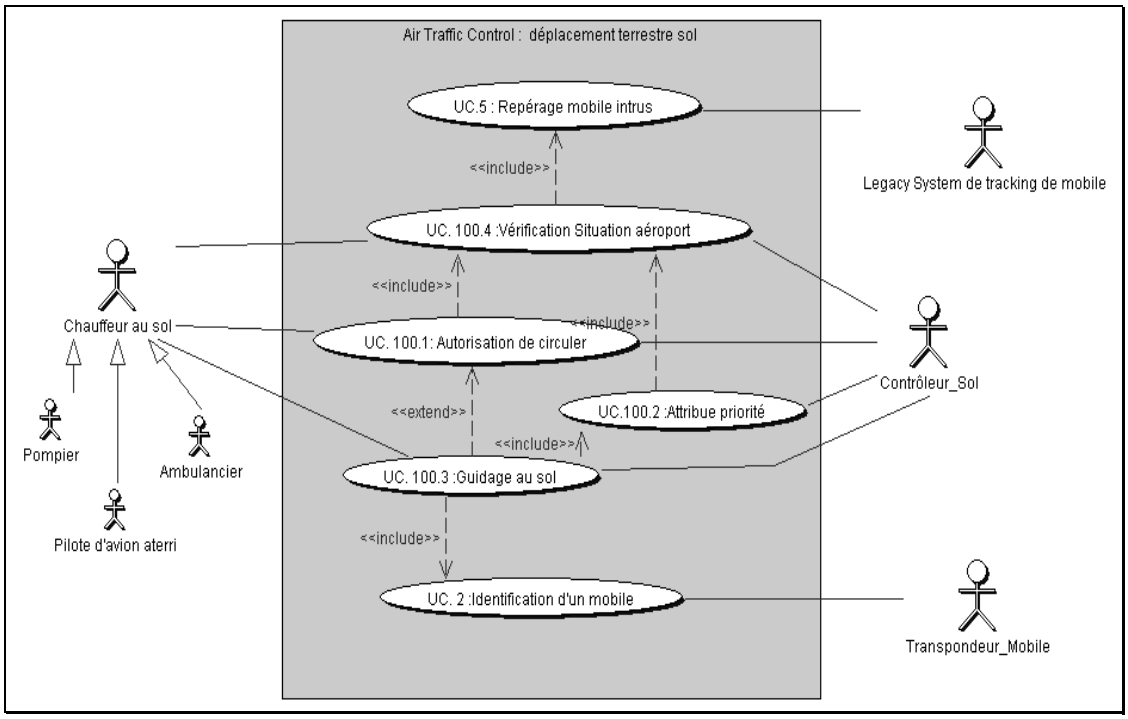

Figure 4.5.4 : Diagramme des uses cases chauffeurs

4.5.4.1 UC. 100.1 : Autorisation de circuler

Description : Ce Use Case présente les interactions entre le systèmes et les acteurs pour demander l'autorisation de circuler.
Acteur(s) primaire(s):
Chauffeur au sol, contrôleur au sol.
Acteur(s) de support
Transpondeur (senseurs)
Extensions
Inclusions
UC. 100.4. Vérification situation aéroport UC.1. Identification d'un acteur UC.2. Identification d'un mobile connu du système
Scénario principal
Précondition
Le chauffeur est identifié auprès du système et est associé à un mobile. Le rôle joué par le chauffeur et par le mobile est déterminé. Le système connaît le mobile dans lequel il compte circuler. Ce mobile est également identifié auprès du système.

Action de(s) acteurs	Responsabilité du système
1. Le chauffeur demande l'autorisation au système pour circuler dans l'enceinte de l'aéroport. Il indique au système l'endroit où il se trouve et la destination où il souhaite se rendre, ainsi que la « durée de vie de sa demande ». 3. Le contrôleur vérifie que le mobile peut se déplacer dans l'enceinte de l'aéroport, en vérifiant l'état de celui-ci. Il signale au système son autorisation ainsi que le plan de routage. 5. Le chauffeur est averti de la décision et l'acte au système	2. Le système vérifie que ce chauffeur et ce mobile sont connus et identifiés. Il vérifie que l'environnement est suffisamment stable pour qu'un contrôleur puisse décider de cette permission. 4. Le système enregistre ces informations. Il donne l'autorisation au chauffeur de se rendre à cet endroit avec son mobile en suivant la route décidée par le contrôleur.
Postcondition	
Le chauffeur a reçu un accord ou un refus de circuler dans l'enceinte pour se rendre à un endroit précis. Le système a enregistré cette décision et la conserve (chauffeur, mobile, from, where, date) Le chauffeur doit demander au système de le guider (lui donner un plan de circulation) avant de pouvoir circuler.	
Flux d'exception 1	
Exception peut survenir si une région de l'aéroport où devrait circuler le mobile est bloquée, si la circulation est trop dense, si un incident est survenu, si les conditions météorologiques sont mauvaises, ...	
Action de(s) acteurs	**Responsabilité du système**
3. Le contrôleur vérifie que le mobile peut se déplacer dans l'enceinte de l'aéroport, en vérifiant l'état de celui-ci. Il signale au système son refus. 5. Le chauffeur reçoit ce refus. Il peut réintroduire une demande.	4. Le système enregistre ces informations. Il donne l'autorisation au chauffeur de se rendre à cet endroit avec son mobile. Il signale au chauffeur qu'il sera averti le moment venu quand il pourra se déplacer.
Flux d'exception 2	
Survient si le système décide de prendre son autonomie décisionnelle.	
Action de(s) acteurs	**Responsabilité du système**
	2. Le système vérifie que ce chauffeur et ce mobile sont connus et identifiés. Il constate que l'environnement ne permet pas de laisser un contrôleur décider seul de la gestion des autorisations et de routage. Il signale au contrôleur et au chauffeur qu'il prend la main sur ces prises de décisions. Le système

3. Le chauffeur reçoit la décision et éventuellement un plan de route.	analyse la situation (plusieurs paramètres à prendre en compte selon les scénarios) et accorde ou non l'autorisation de circuler. Si oui, alors il donne un plan de route au chauffeur et au contrôleur. Si non, il le signale au chauffeur.
Contraintes	
L'autorisation doit être traitée par le système dans les 5 secondes après l'introduction de la demande.	

4.5.4.2 UC. 100.2 : Attribution priorité

Description : Ce Use Case présente l'attribution de priorité entre mobiles.	
Acteur(s) primaire(s):	
Chauffeurs au sol	
Acteur(s) de support	
Transpondeur	
Extensions	
/	
Inclusions	
UC.2. Identification d'un mobile connu du système	
Scénario principal	
Précondition	
Les mobiles circulant dans l'enceinte de l'aéroport sont tous identifiés, situés et leur destination est connue. L'identification en continu du mobile est active. Le chauffeur associé au mobile et les autres acteurs transportés dans ce mobile sont également connus.	
Action de(s) acteurs	**Responsabilité du système**
1. Le(s) mobile(s) (transpondeur) donne(nt) sa(leur) position dans l'enceinte de l'aéroport	2. Le système calcule la nouvelle situation d'un mobile dans l'enceinte de l'aéroport 3. Le système évalue la situation actuelle au sein de l'enceinte et évalue les priorités entre les différents mobiles évoluant dans l'enceinte. 4. Si un conflit de priorité existe entre plusieurs mobiles, le système communique aux chauffeurs des mobiles concernés les différentes priorités.

5. Les chauffeurs des mobiles sont avertis de ce conflit de priorité et sont tenus au courant de ces priorités.	
Postcondition	
Le système a décidé de la priorité entre les différents mobiles et tous les chauffeurs des mobiles concernés en sont notifiés.	
Contraintes	
Il est indispensable que les réponses envoyées aux mobiles le soient dans un délai inférieur à la demi-seconde à partir du moment où le mobile a envoyé sa position au système. En fonction des résultats des « calculs » (position, vitesse, direction), les véhicules ne peuvent pas être à moins de 5 secondes d'un impact. Ainsi, si deux véhicules continuent leur déplacement tel quel, un délai de 5 secondes minimum est indispensable avant qu'ils ne se rencontrent. En deçà de ce délai, le système avertit les deux véhicules et prend des mesures pour attribuer les priorités, les routes à suivre.	

4.5.4.3 UC. 100.3 : Guidage au sol

Description : Ce Use Case présente comment le système guide les mobiles dans l'enceinte de l'aéroport	
Acteur(s) primaire(s):	
Chauffeur au sol, contrôleur au sol	
Acteur(s) de support	
Transpondeur	
Extensions	
UC. 100.1 : Autorisation de circuler	
Inclusions	
UC. 2. Identification d'un mobile connu du système UC. 100. 2 : Attribution priorité	
Scénario principal	
Précondition	
Le chauffeur est identifié, associé à un mobile et les autres acteurs transportés dans le mobile sont connus du système. Le rôle du chauffeur et du mobile sont connus. Le chauffeur a reçu un accord de circuler dans l'enceinte avec un mobile identifié pour se rendre d'un endroit à un autre. Il a reçu un plan de route pour s'y rendre.	
Action de(s) acteurs	**Responsabilité du système**
1. Le chauffeur signale au système qu'il démarre.	2. Si le système juge qu'un contrôleur peut guider le véhicule, il signale au contrôleur au sol que le véhicule démarre.
3. Le contrôleur demande au système de visualiser l'état de l'aéroport. Il peut ainsi suivre le mobile de seconde en seconde.	4. Le système enregistre cette modification et transmet ce nouveau parcours au chauffeur.

S'il juge qu'un nouveau parcours doit être suivi, il le signale au système. 5. Le chauffeur reçoit ce nouveau parcours et modifie son chemin.	
Postcondition	
Le chauffeur est guidé jusqu'à sa destination. Celle-ci peut avoir évolué à travers le temps si des événements contraignent le système et/ou le chauffeur à changer de destination.	
Flux d'exception 1	
Système décide de prendre son autonomie décisionnelle et prend donc la place des contrôleurs.	
Action de(s) acteurs	**Responsabilité du système**
	2. Si le système juge qu'un contrôleur ne peut pas guider le véhicule, il le signale au contrôleur et prend en charge le guidage du mobile. 3. Le système contrôle en permanence le déplacement du mobile et connaît à chaque instant sa position dans l'enceinte de l'aéroport. Il connaît à chaque instant l'état de la circulation.
Flux d'exception 2	
Système (ou contrôleur) décide que le chemin suivi ne peut plus l'être suite à un événement (risque d'asccident, incendie, restriction d'arrêt à une région de l'aéroport, ...)	
Action de(s) acteurs	**Responsabilité du système**
	4. Le système décide que le chemin suivi ne peut plus l'être. Il recalcule une nouvelle route et le signale au chauffeur.
5. Le chauffeur est averti de cette nouvelle route et accepte cette route. Il signale au système qu'il change de route	6. Le système contrôle en permanence le déplacement du mobile et connaît à chaque instant sa position dans l'enceinte de l'aéroport.
Contraintes	
Le chauffeur et le contrôleur sont sans cesse en relation avec le système et sont avertis immédiatement de chaque action qu'ils doivent accomplir. Il est indispensable que les réponses envoyées aux mobiles le soient dans un délai inférieur à la demi-seconde à partir du moment où le chauffeur a envoyé sa demande. En fonction des résultats des « calculs » (position, vitesse, direction) les véhicules ne peuvent pas être à moins de 5 secondes d'un impact. Ainsi, si deux véhicules continuent leur déplacement tel quel, un délai de 5 secondes minimum est indispensable avant qu'ils ne se rencontrent. En deçà de ce délai, le système avertit les deux véhicules et prend des mesures pour attribuer les priorités, les routes à suivre. Une fois cet espace de 5 secondes dépassé, les chauffeurs de mobiles doivent être prévenus dans le vingtième de seconde. Ils ont donc 4,8 secondes minimums pour éviter l'incident.	

4.5.4.4 UC 100.4 : Vérification situation aéroport

<table>
<tr><td colspan="2">Description : Ce Use Case présente comment le système peut calculer la situation de la circulation de l'aéroport à un moment donné</td></tr>
<tr><td colspan="2">Acteur(s) primaire(s):</td></tr>
<tr><td colspan="2">Chauffeur au sol, Contrôleur au sol</td></tr>
<tr><td colspan="2">Acteur(s) de support</td></tr>
<tr><td colspan="2">Transpondeur, radar</td></tr>
<tr><td colspan="2">Extensions</td></tr>
<tr><td colspan="2">/</td></tr>
<tr><td colspan="2">Inclusions</td></tr>
<tr><td colspan="2">/</td></tr>
<tr><td colspan="2">Scénario principal</td></tr>
<tr><td colspan="2">Précondition</td></tr>
<tr><td colspan="2">Le système a enregistré tous les mobiles qui circulent dans l'enceinte de l'aéroport. Il connaît tous les mobiles en stand-by. Il connaît tous les chauffeurs et passagers de chaque mobile.
Il connaît la position de tous les mobiles leur destination et le chemin suivi.
L'aéroport est divisé en zones. Chaque zone est identifiée uniquement.</td></tr>
<tr><td>Action de(s) acteurs</td><td>Responsabilité du système</td></tr>
<tr><td>1. L'utilisateur (chauffeur ou contrôleur) demande la situation actuelle d'une zone ou de la totalité de l'aéroport.

4. L'utilisateur récupère les données et peut les visualiser sur un écran représentant la (les) zone(s) demandées
5. L'utilisateur peut demander ces</td><td>2. Le système récupère les informations concernant la requête.
3. Le système vérifie les infos et les envoie au demandeur :
➔ Mobiles
- Identité du mobile (si connue)
- Situation (si connue)
- Direction (si connue)
- Destination (si connue)
- Vitesse (si connue)
➔ Evénement
- Type d'événement (incendie, accident, ...)
- Situation
- Date de « création »</td></tr>
</table>

informations toutes les deux secondes.	
Postcondition	
Les utilisateurs connaissent la situation des mobiles et des événements en cours dans l'enceinte de l'aéroport. Le système n'aura rien modifié dans les informations	
Flux d'exception 1	
Données sur la situation de l'aéroport inaccessibles.	
Action de(s) acteurs	**Responsabilité du système**
5. L'utilisateur est averti de l'impossibilité de recevoir ces informations. 6. Il envoie toute les secondes une demande identique.	3. Le système ne parvient pas à récupérer les informations. 4. Il informe l'utilisateur qu'il ne lui est pas possible de récupérer ces informations.
Contraintes	
Le délai pour recevoir les informations demandées ne doit jamais dépasser 0.5 seconde. Et entre chaque demande d'information il faut un délai de 1 seconde maximum.	

4.5.5 Particularités des différents chauffeurs de mobiles.

Tant les pompiers que la protection civile, les secouristes, les forces de l'ordre, les techniciens au sol et autres chauffeurs potentiels ont des besoins et attentes légèrement différents selon les cas. Il serait vain de les prévoir tous dans notre système général de gestion du trafic terrestre dans l'enceinte d'un aéroport. Chacun de ces acteurs devra se plier aux exigences du système pour pouvoir en profiter. A l'inverse, le système devra également tenir compte des exigences que ces acteurs peuvent avoir.

Par ailleurs, il apparaît évident que le système de contrôle du trafic terrestre au sol n'est pas le seul système mis en place pour la gestion d'un aéroport. D'autres systèmes parallèles spécifiques à certains acteurs seront également utilisés. Il n'entre pas en compte dans notre travail de les repérer tous. Ainsi, on peut imaginer qu'un système d'alerte globale soit mis en place et que celui-ci soit directement relié aux services de secours, services d'ordre, ...

Les fonctions générales que le système donnera aux différents acteurs, seront les mêmes et satisferont aux exigences décrites ci-dessus dans les Use Cases généraux. Les variations dans l'exécution de ces Use Cases viendront des circonstances dans lesquelles ces scénarios seront exécutés.

Ainsi, un véhicule de pompier se rendant vers un incendie aura la priorité absolue sur un véhicule de maintenance des infrastructures. A l'inverse, si un véhicule de pompier se rend à

la réserve de carburant pour faire le plein, il ne sera pas prioritaire par rapport à un véhicule de maintenance qui se rend sur une piste d'atterrissage pour réparer d'urgence le bitume.

Pour bien démontrer ces circonstances, nous étudierons le cas concret d'un incendie d'un véhicule dans l'enceinte de l'aéroport.

Si nous souhaitions définir un système complet, il conviendrait d'interroger tous les acteurs gravitant dans le microcosme d'un aéroport, de saisir tous leurs besoins et organiser des rounds de réunion de concertation pour définir les niveaux de priorités pour tous les événements envisageables dans l'enceinte d'un aéroport.

Buts principaux des différents acteurs

Dans cette présentation des Use Cases liés aux chauffeurs, nous les avons tous considérés sous un seul angle. Il convient désormais de différencier certains chauffeurs en fonction de leurs besoins spécifiques. Si les interactions avec le système sont toutes similaires, les conditions dans lesquelles les chauffeurs utiliseront ce système varieront.

A titre d'exemples nous pouvons reprendre ici les buts principaux des différents acteurs-chauffeurs « spécialisés ».

1. Secouristes

S'ils sont tous similaires au tronc commun, quelques précisions s'imposent.

- Faire valoir une priorité par rapport à d'autre véhicules.
- Attribuer une autorisation de circuler dans l'enceinte de l'aéroport en cas d'urgence.
- Donner un chemin à suivre pour se rendre le plus rapidement sur des lieux d'une « catastrophe ».
- Gérer les priorités entre mobiles.
- Prévention en cas de danger entre un mobile de pompier et un autre mobile.

2. Forces de l'ordre

Les chauffeurs de mobiles des forces de l'ordre, à l'instar des véhicules de secours peuvent être « Prioritaires » et par la même exiger du système un niveau de priorité plus important que pour d'autres véhicules évoluant dans l'enceinte.

Les buts sont donc identiques à ceux des services de secours.

3. Pilotes

Les pilotes sont des chauffeurs au sol un peu particuliers car ils entrent dans l'enceinte et sortent de celle-ci, guidé par un autre système d'Air Trafic Control. Ces systèmes sont déjà particulièrement éprouvés et il n'est pas question ici de les remettre en cause.

Le nouveau système à mettre en place devra donc interagir avec ce legacy system pour les décollages et atterrissages des avions. Etant donné que les systèmes d'ATC sont tous implémentés et que l'on ne peut les remettre en cause ex abrupto, le nouveau système devra l'interroger pour calculer l'état de l'aéroport.

Parmi les besoins des pilotes, citons :

- Gestion des autorisations de circuler dans l'enceinte de l'aéroport.
- Gestion des autorisations de circuler suite à un atterrissage.
- Gestion des autorisations de circuler pour se rendre sur la piste de décollage.
- Guidage au sol des avions.

- Gestion des priorités entre avions et mobiles.
- Information permanente des différents pilotes en cas d'accident potentiel entre avions-mobiles circulant dans l'enceinte de l'aéroport.
- Information permanente des différents chauffeurs qu'un incident s'est commis dans l'enceinte de l'aéroport.

On se rend compte que les besoins de ces pilotes sont identiques aux autres chauffeurs de mobiles et qu'aucun « Use Case particulier » ne permettra de mieux formaliser les interactions entre ces chauffeurs-mobiles avec le système. Seuls les types d'informations à transmettre au système peuvent différer. Ainsi, pour pouvoir se rendre à une piste de décollage un pilote devra fournir une « Autorisation » au système de contrôle de trafic terrestre. A l'inverse, pour quitter une piste de décollage, le pilote d'un avion atterri signalera au système qu'il doit être prioritaire pour circuler vu les dangers de contacts avec d'autres avions atterrissant.

4. Techniciens au sol

Ces chauffeurs ont de nouveau des buts et des besoins identiques par rapport aux autres chauffeurs, mais leur « priorité » est par nature moindre.

Ils peuvent cependant également revendiquer une certaine priorité si des entretiens fondamentaux doivent être réalisés et qu'ils doivent s'y rendre urgemment.

5. Administratifs, véhicules de maintenance, ...

Tous ces acteurs ont de nouveaux les même besoins mais ne jouissent pas de niveau de priorité particulier.

6. Conclusions

Il apparaît donc clairement que ce qui différencie les besoins des différents chauffeurs de mobiles terrestres se situe essentiellement au niveau des priorités que ces acteurs ont en fonction des circonstances.

Ainsi, le système de contrôle du trafic terrestre devra tenir compte de ces différents niveaux de priorités lors de l'attribution des routes, de la gestion des conflits entre mobiles.

Pour le reste, les besoins de tous ces pilotes-chauffeurs sont identiques pour leurs rapports avec ce système de gestion de trafic au sol.

4.5.6 Scénario récapitulatif – Prototype conceptuel

Si dans la première partie de la définition des Use Cases nous avons voulu modéliser les besoins du nouveau système à développer par des Use Cases précis et « mono-fonctionnels », nous allons tenter d'éprouver ces Use Cases en les confrontant à une situation critique du réel. Les différents scénarios vont donc s'imbriquer, s'opposer et parfois même risquer de se neutraliser.

Par ce cas d'espèce, nous allons tenter de valider notre méthodologie tout en notant les points faibles de celle-ci. Ceci nous permettra, éventuellement, de proposer des solutions ou des pistes de réflexions pour surmonter les écueils rencontrés.

L'exemple utilisé sera celui de l'incendie d'un véhicule sur une voie de circulation de l'aéroport. Nous verrons comment il est possible de modéliser cet exemple, de saisir les contraintes.

4.5.6.1 Prototype : Incendie d'un camion de transport de fret sur une voie de l'aéroport.

Afin d'illustrer comment le système devra réagir en cas d'un incident concret, l'explication par un prototype peut nous aider à mieux comprendre les contraintes et les besoins du système.

Ce prototype sera expliqué de deux manières :

- En langage naturel
- Via des diagrammes de séquences « hauts niveaux ».

Il est a constater que ce Use Case est davantage une succession « d'événements possibles » dans l'enceinte de l'aéroport. Il conviendra de définir toutes les règles à mettre en place, toutes les procédures à suivre, la classification des types d'alertes pour chaque type d'événements.

Nous reprendrons donc dans ce scénarios les différentes actions qui seront menées pour gérer tant la sécurité des personnes et des biens dans un aéroport.

- **Incident – Incendie d'un camion sur une voie: Langage Naturel**

1. Alerte

Un incendie d'un camion de transport de fret sur une voie de l'aéroport est constaté par le chauffeur du camion qui a pu sortir de son véhicule avec son passager blessé. Ce chauffeur le signale au système d'alerte qui gère l'ensemble de l'aéroport. Il donne des informations concernant cet incident :

- **IncidentInfo**
  - Lieu où il est situé
  - Véhicule(s) impliqué(s)
  - Date (jour heure)
  - Le fret transporté est inflammable mais pas toxique et ne risque pas d'exploser
  - Le passager du camion est blessé mais est hors du véhicule
  - …

➔ Une analyse plus en profondeur permettra d'affiner les informations à transmettre pour chaque type d'alerte.

Le système d'alerte enregistre cette information et lance les mesures à prendre selon les règles prédéfinies. Il est à constater que ce sous-système est autonome. Qu'il soit entièrement automatisé ou non importe peu. Ainsi qu'imposé plus haut, il est de la responsabilité du système de sécurité de prévenir les différents autres sous-systèmes de l'événement.

Ce système d'alerte peut être vu comme un système qui préviendrait tous les autres systèmes qui en auraient marqué la volonté préalable.

Le délai entre le moment où le système d'alerte est averti et où ce système a lancé les mesures à prendre ne peut dépasser 0.1 seconde s'il est automatique et à définir si des humains interviennent. Dans ce deuxième cas des règles de procédures d'urgence strictes doivent être décrites.

2. Information aux différents acteurs

Le système d'alerte prévient le système de contrôle du trafic terrestre en lui indiquant les informations à disposition (UC. 4).

Parallèlement, le système d'alerte prévient également les services de secours, les services d'ordre, ...

Dans le 0.1 seconde qui suit l'avertissement de l'alerte, le système de contrôle du trafic terrestre prévient tous les contrôleurs, et chauffeurs de mobiles qui circulent à ce moment là dans l'enceinte de l'aéroport. Il signale qu'un incident d'une telle nature a eu lieu à tel endroit. Ce signal doit être fait en sorte que les acteurs ne peuvent l'ignorer.

Le système de contrôle du trafic doit décider si la main peut être laissée aux contrôleurs humains pour ce qui est du guidage des mobiles. Cette prise de décision ne peut être faite que sur base d'un ensemble de paramètres analysés selon des règles strictes (visibilité, climat, dangerosité de l'événement, état du trafic, charge de travail des contrôleurs disponibles, ...) Le système considère qu'il peut pour l'instant laisser la gestion du trafic au sol à un contrôleur humain. Dans la seconde qui suit l'avertissement à chacun qu'un incident a eu lieu, un contrôleur au sol est désigné « responsable » pour gérer les problématiques liées à la gestion du trafic au sol.

A partir de ce moment là, la gestion qui suivra sera faite par un (ou plusieurs) hommes en contact avec le système automatisé. Tant que le système l'autorisera, les prises de décisions finales doivent rester humaines, suite à des concertations entre contrôleurs, chauffeurs, services de sécurités, de secours et système automatisé.

Le contrôleur demande au système l'état de la circulation dans l'enceinte de l'aéroport à ce moment (UC. 100.4).

Le contrôleur décide de rediriger tous les véhicules se déplaçant dans la zone où l'incendie a lieu de façon à éviter les accidents supplémentaires (UC. 100.2, 100.3). Il condamne un périmètre autour de cet incident, zone dans laquelle toute circulation sera interdite hormis une autorisation de sa part.

Ainsi, les véhicules qui circulaient dans l'enceinte doivent la quitter. Les véhicules qui devaient passer par ce secteur sont redirigés. Le secteur désigné est signalé comme inaccessible. Seuls les véhicules autorisés peuvent accéder à cette zone. Le contrôleur donne l'autorisation à tous les véhicules de secours, de sécurités et de force de l'ordre dans cette zone.

Il est important que tous ces véhicules qui doivent circuler dans cette zone soient avertis de la situation endéans les délais minimum autorisés.

Le système automatique doit donc calculer chaque seconde l'état de tout l'aéroport, et de la zone concernée en particulier, et le fournir aux chauffeurs des mobiles circulant sur cette zone.

3. Services de secours

Suite à l'alerte les différents services de secours ont été prévenus et les véhicules de pompiers, d'assistance médicale et de service d'ordre doivent se rendre sur les lieux.

- Chaque chauffeur et passager s'identifie au système (UC. 1).
- Chaque chauffeur prend possession d'un véhicule qui lui est attribué et demande à y être associé (UC. 3).
- Chaque chauffeur demande l'autorisation de circuler au système (UC. 100.1). Le système transfère automatiquement cette demande auprès du contrôleur au sol adéquat.
- Le contrôleur donne l'autorisation à ces véhicules de se rendre sur les lieux de l'incident (UC. 100.1)
- Le contrôleur, aidé par le système automatique qui lui donne l'état de la circulation à chaque instant (toutes les secondes) peut guider les véhicules autorisés oralement.
- Les mobiles, conduits par des chauffeurs, se déplacent en suivant les routes qui leur sont imposées. Toute les demi-secondes les mobiles signalent leur position, vitesse, destination, ... au système. Ils peuvent donc être suivis à la trace et le système peut prédire leur position dans un espace de temps de 10 secondes.

➔ Il est à noter que pour la succession de ces événements le délai de réponse du système est de l'ordre du dixième de seconde pour chaque « question ».

- **Incident – Incendie d'un camion sur une voie: Sequence Diagrams**

Ci-dessous nous présenterons une série de diagrammes de séquence grâce auxquels nous avons pu affiner la représentation de nos besoins et exigences.

En effet, il apparaît clairement que la seule représentation des besoins via des Use Cases reste limitée pour des systèmes critiques tels que celui que nous cherchons à définir.

Le premier diagramme représente la séquence d'événements, de messages échangés entre acteurs et sous-systèmes. Il est à noter qu'aucun à priori technologique ou architectural n'est pris en compte à ce stade.

Dans ces séquences, nous allons donc essayer de repérer les sous-systèmes, les acteurs à mettre ensemble. Ainsi, le système de « gestion du trafic terrestre (A-SMGCS)» sera encore considéré comme une boîte noire qui recevra des inputs et produira des outputs. A ce stade, certains éléments tels des « repository » peuvent également être signalés.

1. Incident_sequence_Diagram.

Ce diagramme reprend dans les grandes lignes les actions qui seront prises par les différents acteurs, et le système A-SMGCS pour gérer une alerte qui sera communiquée par le système autonome d'alerte !

Certaines « opérations » seront explicitées plus en détail dans d'autres sous diagrammes que l'on pourrait comparer à des « agrandissements photos ».

2. IncidentTreatment_sequence_Diagram

Diagramme « gros plan » d'une opération réalisée par le système A-SMGCS lorsqu'il est averti d'une alerte !

3. ServiceSubscribing_sequence_Diagram

Montre comment les systèmes s'abonnent à des services offerts par d'autres systèmes

4. AirportSituation_sequence_Diagram

Montre comment la situation de l'aéroport est tenue à jour à tout instant.

5. GroundTrafficControl_sequence_Diagram

Montre comment le système de contrôle de trafic aérien parvient à contrôler l'état de la circulation au sein de l'aéroport.

1. Incident_sequence_Diagram

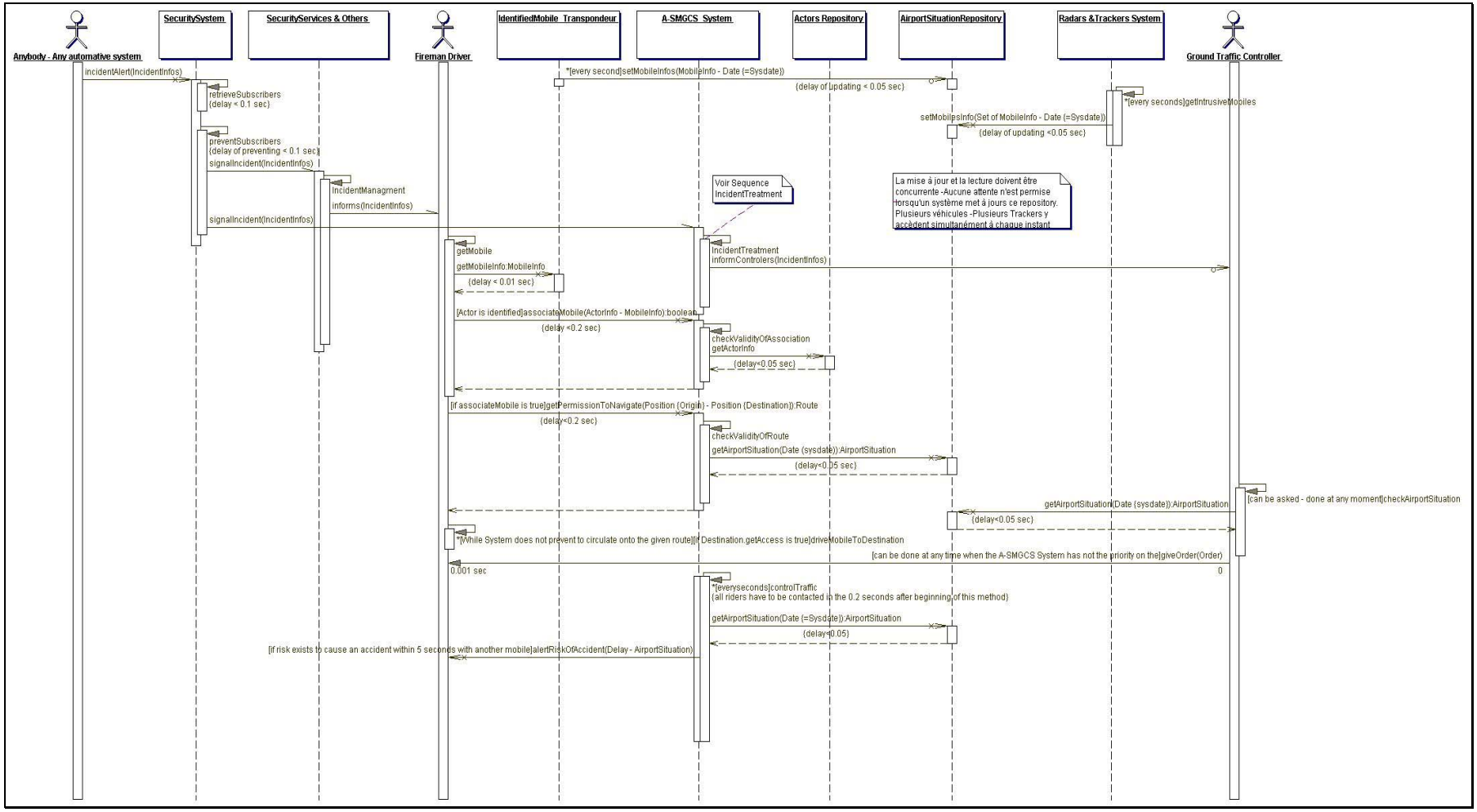

Figure 4.5.6.1a: Diagramme de séquence : Incident

2. IncidentTreatment_sequence_Diagram

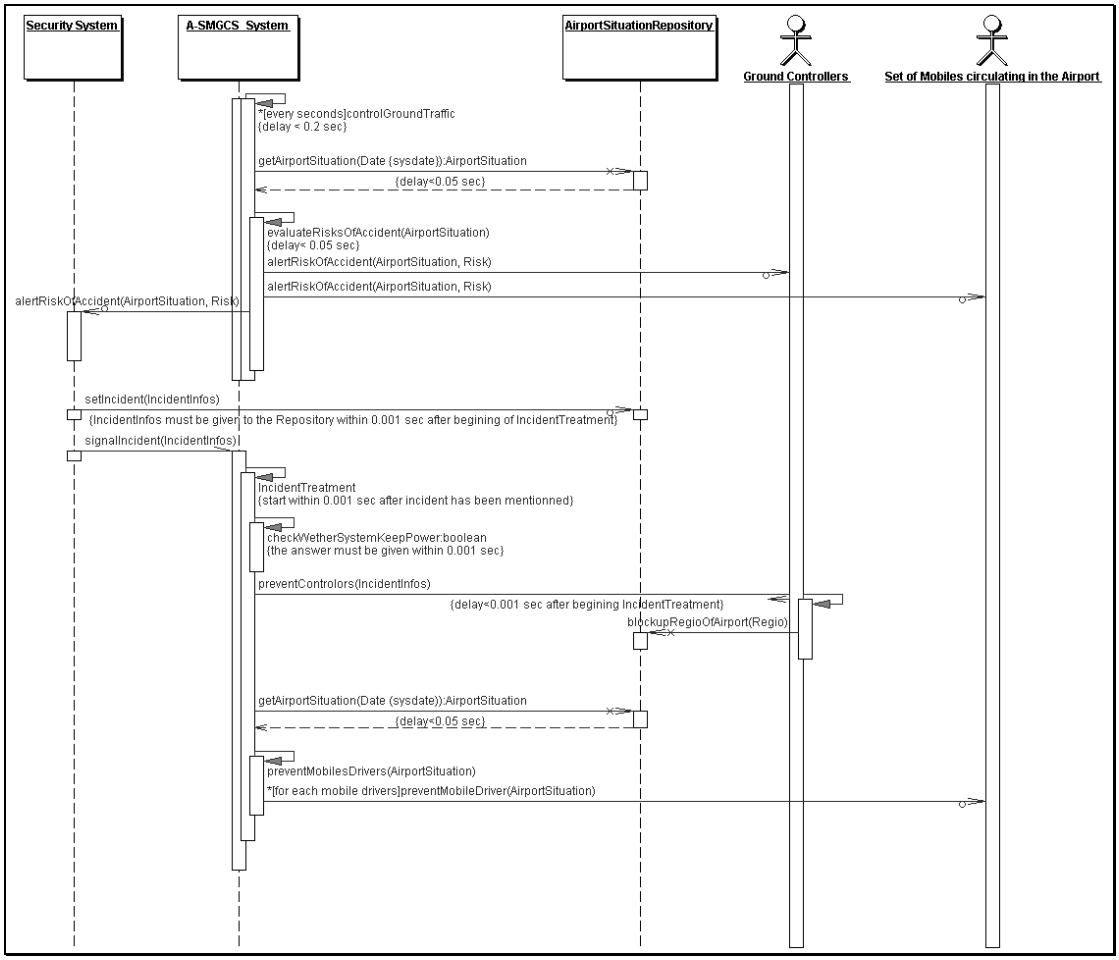

Figure 4.5.6.1b: Diagramme de séquence : Traitement d'un incident

3. ServiceSubscribing_sequence_Diagram

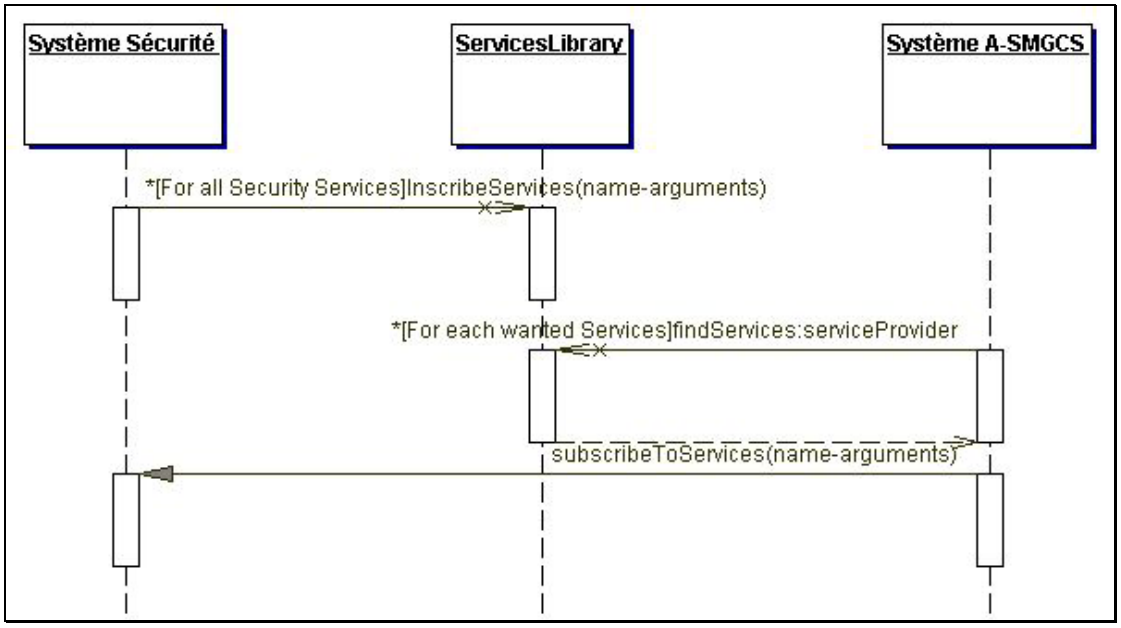

Figure 4.5.6.1c: Diagramme de séquence : Services

4. AirportSituation_sequence_Diagram

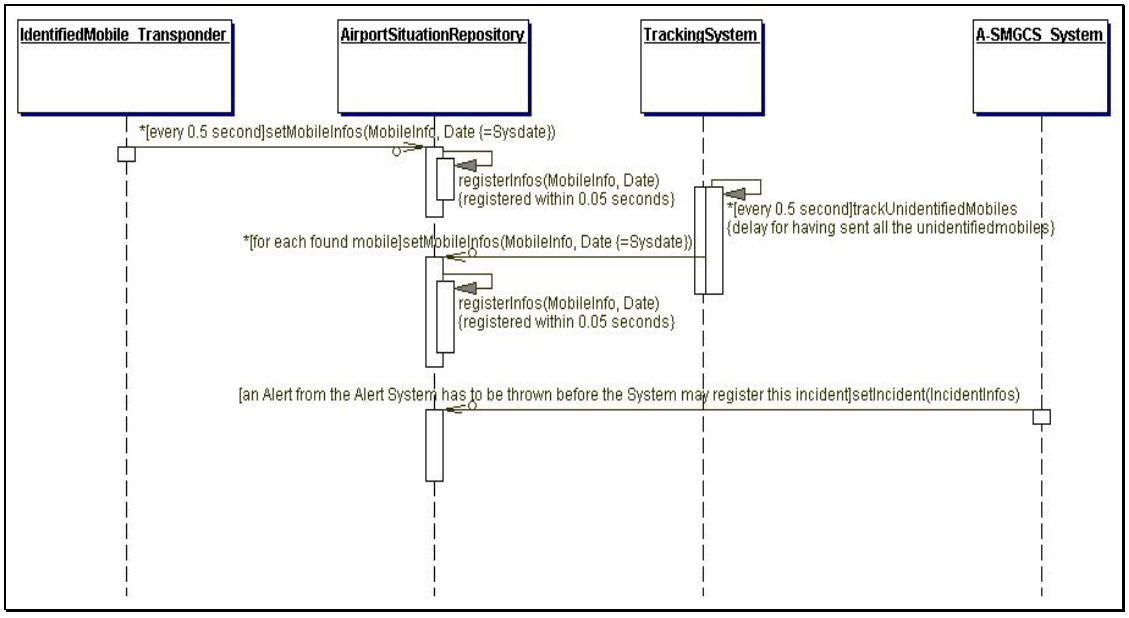

Figure 4.5.6.1d: Diagramme de séquence : Situation d'un aéroport

5. GroundTrafficControl_sequence_Diagram

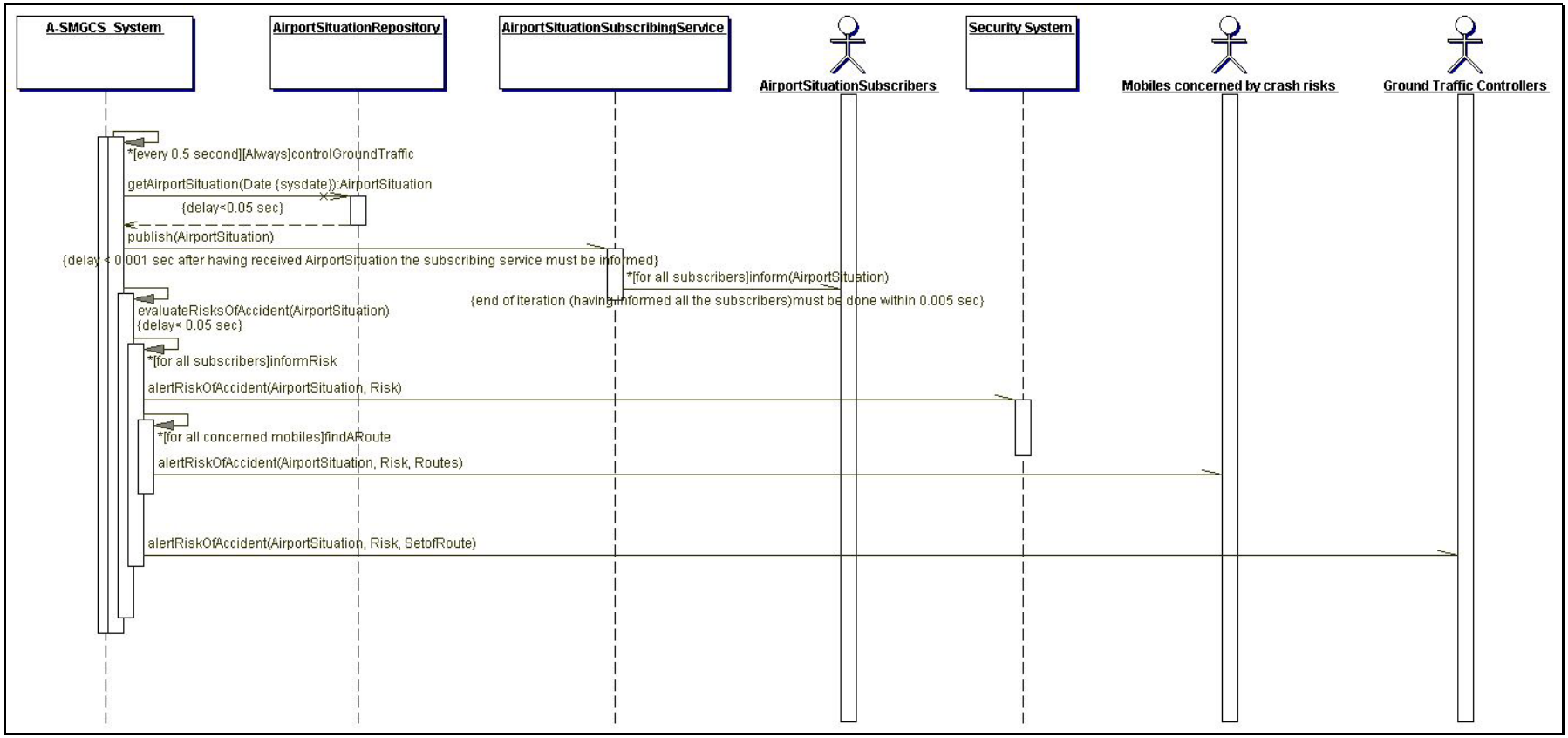

Figure 4.5.6.1e: Diagramme de séquence : Gestion du trafic au sol

6. Autonomie décisionnelle du système ?

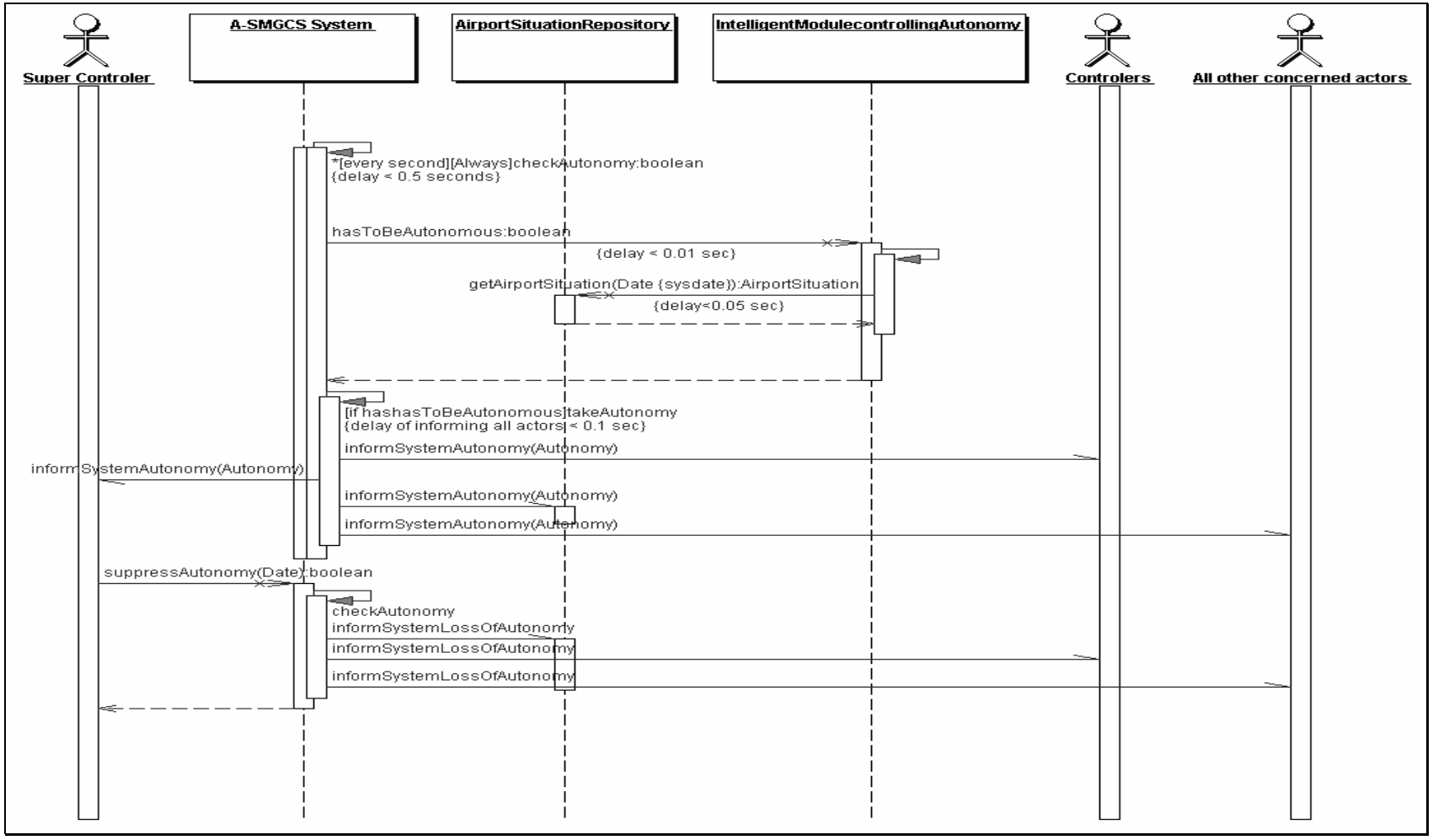

Figure 4.5.6.1f: Diagramme de séquence : Autonomie

4.5.7 Conclusion

Au terme de cette première approche méthodologique qui nous a permis de saisir une partie des besoins et des contraintes auxquels devait répondre notre système, une double constatation s'impose.

D'une part, la capture de besoins complexes mais reflétant une réalité concrète parlante, non seulement pour les acteurs professionnels mais aussi pour les donneurs d'ordre ou les intervenants passifs, est très intuitive et accessible grâce à la méthode des « Use Cases ». La combinaison du langage naturel et des schémas aisément lisibles permet à des quidams de saisir rapidement les besoins hétérogènes d'un système souvent complexe. En effet, quiconque souhaite comprendre les fonctionnalités de base du système que nous souhaitons mettre en place n'a pas besoin de connaissance d'un langage formel inaccessible pour les décrypter.

D'autre part, le risque d'une telle approche est de trop simplifier les exigences et les besoins que doit rencontrer notre système. A trop vouloir rendre cette première étape très simple, le risque est grand de ne pas saisir les contraintes fortes sur ce système. Il est en effet difficile de modéliser les contraintes de sécurité, liées au temps réel par ce mode de spécification. En rester au niveau du langage naturel ne permet donc pas de consolider les spécifications d'un système temps réel tel que celui que nous étudions.

L'utilisation de diagrammes de séquence « haut niveau » permet de peaufiner ces spécifications. En effet, un tel diagramme nous permet de modéliser sommairement les « acteurs » et « systèmes » principaux qui interviennent dans la réalisation d'un scénario. Ces systèmes doivent être vus comme des « boîtes noires » qui seront par la suite décomposés en modules fonctionnels dans lesquels des interactions seront également étudiées.

Ces diagrammes nous permettent donc d'introduire une idée de séquentialité et d'imposer certaines contraintes temps réelles lors de l'exécution d'une action. Ils facilitent aussi la modélisation de sous actions ou d'activités répétitives dans un système. Cependant, des limites sont toujours présentes. S'il est aisé de modéliser des contraintes liées à des délais de réalisation d'une action (temps de réponse maximum pour une question, ou une action), il est difficile de représenter dans de tels diagrammes des contraintes du type :

> *Si une action est lancée et que telle action n'est pas lancée dans les x secondes qui viennent alors, tel événement devra être signalé.*
>
> *Si une condition sur un système est remplie et qu'une autre condition n'est pas remplie sur un autre système dans les x secondes, alors il est nécessaire qu'une troisième condition soit remplie sur un troisième objet sinon, un événement doit obligatoirement survenir :*
>
> ***Si une alerte est lancée suite à un incendie à un endroit , que dans les 120 secondes des pompiers ne sont pas sur place alors, il est obligatoire que des pompiers soient en route et qu'un délai d'une minute soit ajouté, sinon une seconde alerte d'extrême urgence doit être lancée.***

Si les diagrammes de séquence hauts niveaux, nous permettent d'améliorer les représentations des captures des exigences d'un système, ils ne suffisent pas pour représenter toutes les contraintes d'un système critique temps réel embarqué. Cependant, au stade de la simple définition des besoins métiers que devrait rendre le système, est-il judicieux de rentrer profondément dans l'explicitation des contraintes complexes. Ainsi, n'est-il pas suffisant de signaler que des « mobiles ne peuvent entrer en collision et que les chauffeurs de ces mobiles

doivent être prévenus en cas d'accident potentiel » ? A trop vouloir complexifier trop vite un problème, cela oriente parfois les pistes de solutions à adopter.

Si des solutions existent pour consolider les spécifications temps réelles, elles devraient venir d'une utilisation parcimonieuse de langages davantage formels permettant de modéliser ces contraintes. Parmi ces langages et paradigmes de spécification, citons TLA, OCL (Object Constraint Language, ...)

L'apport du diagramme de séquence nous permet de saisir plus aisément les composants/modules fonctionnels qui seront en oeuvre pour notre système. Les exigences fonctionnelles du nouveau système une fois saisie, il convient par la suite d'ouvrir les différentes « boîtes noires » afin d'y relever les différents modules qui interagissent afin de rendre les services demandés. L'étape ultérieure de notre travail sera de répertorier les différents modules dans le système de contrôle de trafic et de tenter de les spécifier plus précisément en utilisant des modèles plus adaptés.

Enfin, ROPES nous incite à proposer fréquemment des prototypes aux clients, afin que celui-ci puisse valider, étape après étape, tout le processus de développement. Au stade de l'analyse des besoins, le danger dans une telle approche serait de choisir de manière univoque le prototype validant nos Use Cases. Aussi, serait-il judicieux de demander aux « clients » de rédiger un scénario qu'ils nous demanderaient d'éprouver et par là même de valider toute notre étude.

5 Analyse des composants du système

5.1 Introduction

Dans la première partie de l'analyse de notre « système coopératif temps réel embarqué », nous avons utilisé la méthode des Use Cases pour saisir et modéliser les fonctionnalités, les besoins et les contraintes de ce nouveau système. Nous en avons extrait les points forts et les points faibles, critiqué les limites et proposé certaines pistes utiles pour combler certains manques. Dans cette deuxième partie nous nous baserons sur les scénarios et le prototype proposés pour affiner notre analyse et passer à une nouvelle étape. En effet, désormais, les acteurs et les systèmes, sous-systèmes, sont bien définis. Leurs rôles respectifs ont été attribués. Aussi, convient-il d'ouvrir chaque système interdépendant pour en étudier leur fonctionnement et leur interconnexion. En nous basant essentiellement sur les diagrammes de séquence, nous avons tenté de relever les composants « fonctionnels » qui sautent aux yeux immédiatement.

Dans un premier temps, nous proposerons une vue statique des composants du système à développer de haut niveau[18]. Nous représenterons ces composants par des diagrammes de classes dans lesquels seront repris les « objets » principaux du système ainsi que les interactions entre eux. Une telle présentation nous permettra donc de comprendre les liens entre les composants et leur nature.

Dans un second temps, nous proposerons une vue davantage dynamique de chaque composant fonctionnel. Après une courte énumération de ceux-ci, nous plongerons plus en détail dans leur composition interne et dans les exigences internes propres à chaque composant. Nous utiliserons de nouveaux diagrammes afin de modéliser toutes les contraintes temporelles et autres sur ces modules indépendants et en interactions mutuelles. Pour ce faire nous les modéliserons via des diagrammes d'états.

Cette étape terminée, nous pourrons retourner à la marche à suivre proposée par ROPES et affiner les diagrammes d'objets.

5.2 Vues statiques du système

Dans cette partie, nous montrerons la plupart des objets qui interviennent dans le système étudié. Ces présentations, par des diagrammes de classes, doivent davantage être vues comme une modélisation du réel et non pas comme une approche technique.

L'avantage de tels diagrammes est qu'il est possible de rester encore à un niveau d'abstraction relativement élevé, ce qui permet aux « non – initiés » de saisir rapidement le domaine étudié.

[18] Cette approche est donc légèrement différente de celle proposée par ROPES qui préconise une approche statique à la suite de l'étude des composants. [DOUGLASS, *ROPES*, 1999]

5.2.1 Acteurs – Chauffeurs - Contrôleurs

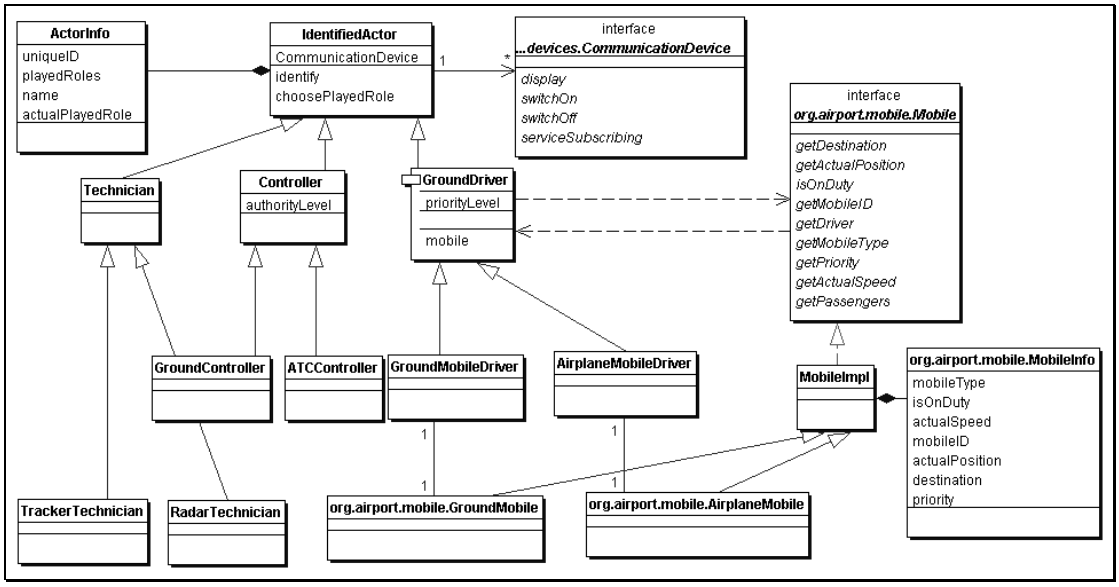

Figure 5.2.1: Diagramme de classe : acteurs

Ce diagramme propose un aperçu statique des acteurs qui peuvent interagir avec le système. Le diagramme montre également les liens entre différentes « interfaces » avec lesquelles ces acteurs peuvent « communiquer ».

Ainsi, grâce à ce diagramme, nous pouvons conclure que tous les « IdentifiedActor » sont en contact avec un ou plusieurs outils de communication (pda, gsm, pc, téléphone, bipper, ...).

Notons aussi que les conducteurs de mobiles au sol sont en contact avec un et un seul mobile.

Il existe une double référence entre le chauffeur et son mobile, ce qui signifie que si l'on connaît un chauffeur, on connaît également le véhicule qu'il conduit à ce moment. Inversement, si l'on connaît un mobile, il est possible de retrouver le chauffeur qui le pilote. Par déduction, si l'on connaît un mobile et que l'on connaît son chauffeur, il est possible de communiquer avec lui car à partir du chauffeur, il est possible de retrouver tous ses outils de communication.

Ainsi, les exigences d'accessibilité à tout instant des chauffeurs par le système sont acquises par ce biais.

5.2.2 Espaces

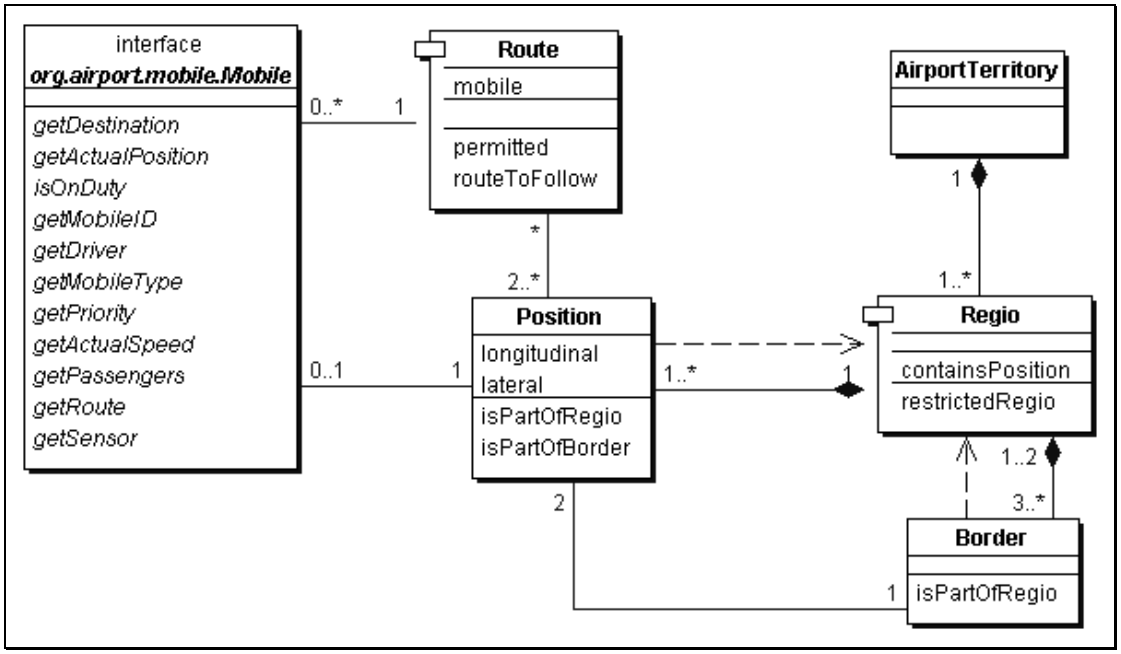

Figure 5.2.2 : Diagramme de classe : Espace dans un aéroport

Vu les besoins du système, il est apparu évident qu'il fallait pouvoir modéliser chaque endroit d'un aéroport, ainsi que les routes suivies par des mobiles et pouvoir déterminer à chaque instant l'endroit où se trouvait chaque mobile.

Par le schéma ci-dessus, ces exigences sont rencontrées :

- Un mobile a une et une seule route (à un moment donné)
- A un instant « t », un mobile ne se trouve qu'à un et un seul endroit et à cet endroit ne se trouve que 0 ou un seul mobile
- Par contre, une route peut être attribuée à plusieurs mobiles en même temps. La certitude existe que deux mobiles ne peuvent se trouver au même endroit au même moment.
- Une route est constituée d'un ensemble de Positions (au moins deux), qui détermine chaque point de passage de la route.
- Une région d'un aéroport est constitué d'un ensemble de « Border », au minimum trois qui doivent former un espace fermé. Une région peut être inaccessible. Si tel est le cas alors, la circulation y est restreinte selon certaines règles.
- Le territoire d'un aéroport est constitué d'un ensemble de régions.

Si la contrainte empêchant que deux véhicules se retrouvent à un même endroit en même temps est bien exprimée, il n'est par contre pas possible par un diagramme statique tel que celui-ci, de symboliser le fait que deux véhicules, en route ne peuvent se rencontrer dans les 5 secondes s'ils continuent leur route tel quel !

5.2.3 AirportSituation

A chaque instant la situation d'un aéroport peut être demandée. La réalité de ce que l'on entend par situation n'est pas expressément définie, mais il s'agit en fait de toutes les informations que l'on juge utiles pour pouvoir prendre des décisions quant aux actions à prendre (routes à donner, région à fermer, ...) soit par des contrôleurs, soit par le système intelligent.

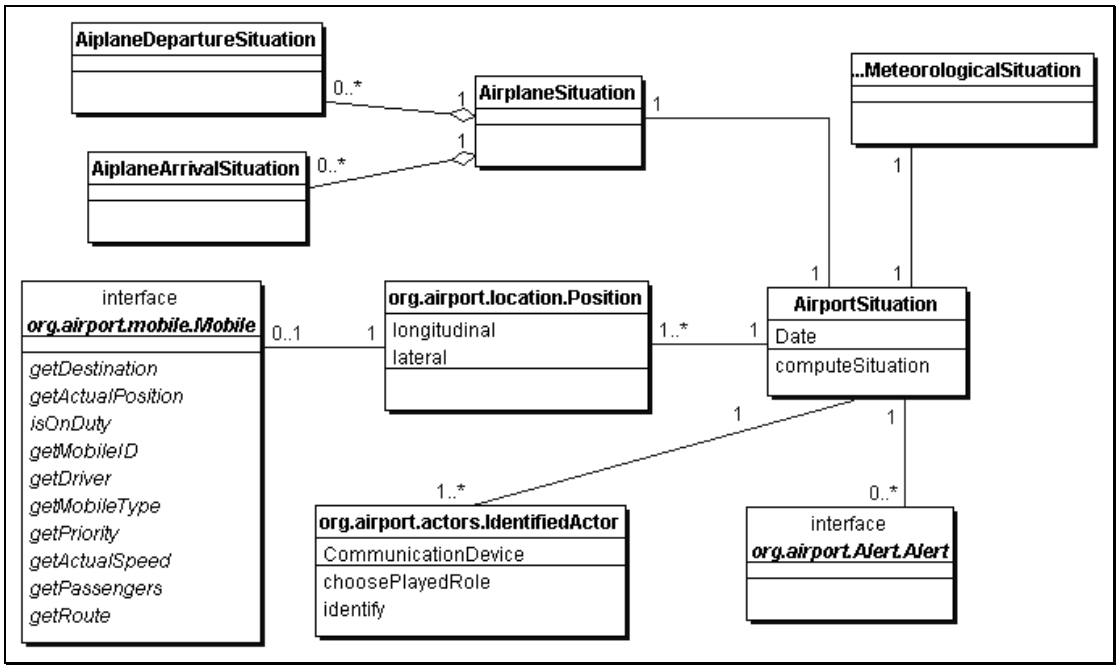

Figure 5.2.3: Diagramme de classe : situation d'un aéroport

Le diagramme ci-dessus montre comment il est possible de représenter la situation d'un aéroport à tout instant. Ainsi, l'objet AirportSituation peut être identifié par une Date qui donnera à un moment donné, la situation d'un aéroport, à savoir :

- L'ensemble de toutes les positions de l'aéroport, et par extension, de tous les véhicules qui s'y trouvent.
- Pour tout service qui utiliserait cette situation, il est possible d'interroger chaque mobile afin de connaître sa route, sa vitesse, son chauffeur (et donc son outil de communication),... et pouvoir faire des prédictions.
- Ensemble de toutes les alertes en cours
- Situation météorologique du moment
- Tous les acteurs identifiés qui œuvrent à ce moment dans l'aéroport,
- ...

5.2.4 Mobiles

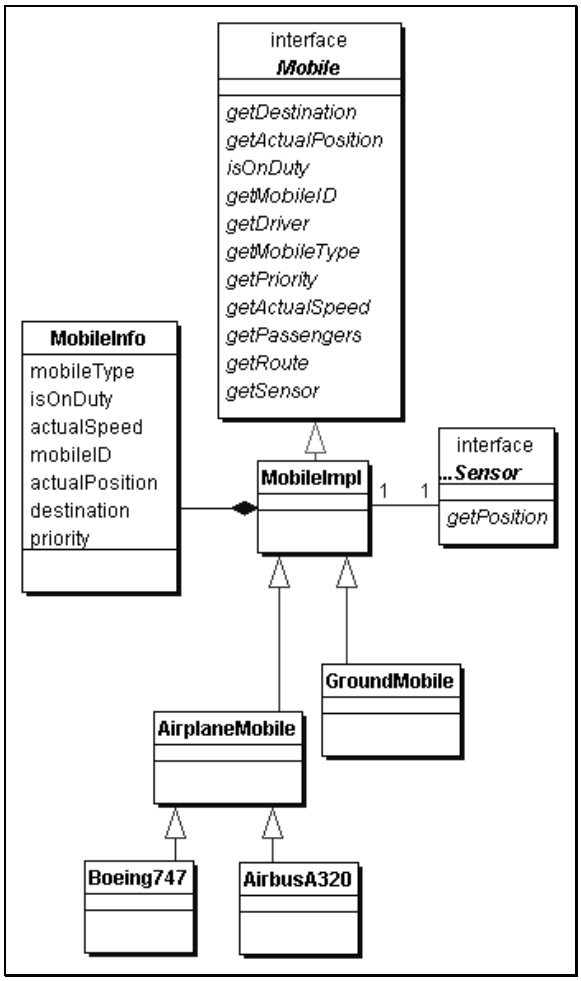

Figure 5.2.4a : Diagramme de classe : Mobiles

Ce diagramme nous permet de modéliser de nouveau tous les mobiles identifiés qui peuvent circuler dans l'enceinte de l'aéroport. Tous ces mobiles sont associés à un système de senseurs qui lui permet de donner sa position à chaque instant. Que ce senseur soit un GPS, un transpondeur ou un radar qui suive ce mobile d'une autre manière, cela a peu d'importance et ne rentre pas dans le cadre de cette étude.

A titre d'information ci-dessous une idée de la représentation que l'on peut se faire de ces senseurs :

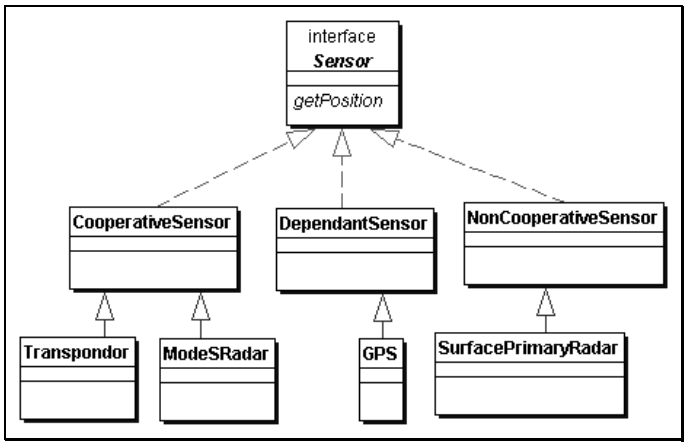

Figure 5.2.4b: Diagramme de classe : Senseurs

5.2.5 Alertes, risques

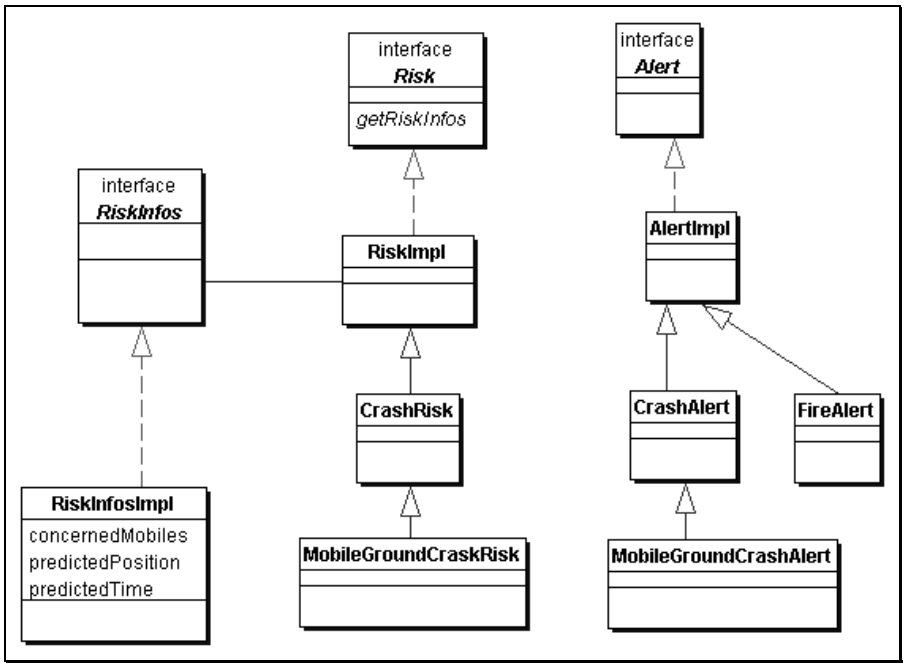

Figure 5.2.5: Diagramme de classe : Alertes et risques

Ainsi qu'il est indiqué dans l'analyse des besoins, le système devra pouvoir traiter les alertes émanant du système autonome de gestion d'alertes. Par ailleurs, il devra également prévenir les acteurs concernés en cas de risques d'incidents, spécialement de risques entre mobiles circulant dans l'enceinte de l'aéroport et il devra enfin prévenir le système d'alerte autonome.

5.2.6 Airport Controlling

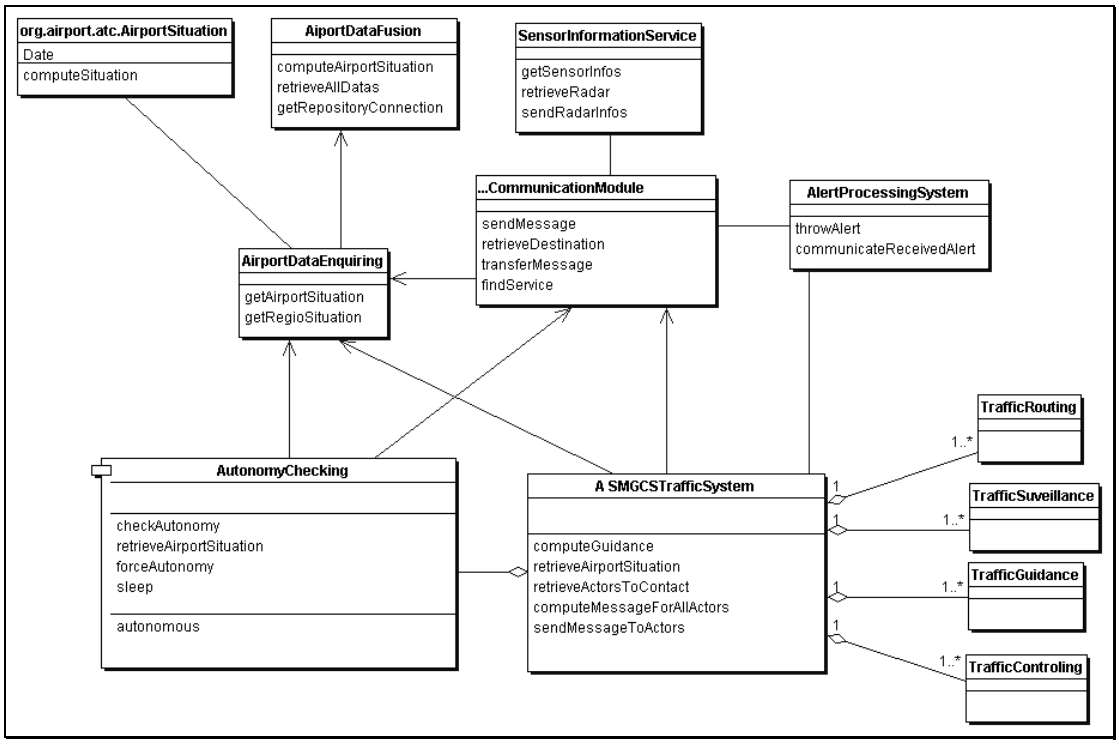

Figure 5.2.6: Diagramme de classe pour le **A-SMGCS**

Ce modèle permet de modéliser succinctement certains objets qui devront rendre les services du système A-SMGCS et des autres services à rendre. Tout système A-SMGCS doit remplir les quatre fonctions de surveillance, routage, guidage et de contrôle du trafic au sol.

Ce diagramme est évidemment incomplet. En effet, le coeur de l'intelligence du système se retrouvera en grosse partie dans ces objets qui devront avoir une puissance de calcul très importante et qui devront pouvoir prendre des décisions importantes.

Les diagrammes présentés auparavant ont davantage pour vocation de modéliser le business du système dans lequel l'outil A-SMGCS devra intervenir.

Il est impossible dans l'état actuel de modéliser plus avant le système. En effet, la complexité des actions à prendre est telle que le nombre d'objets réels qui permettront de le mettre en place risque d'être multiplié par 100.

5.2.7 Remarques

Afin de comprendre le fonctionnement dynamique du système, les diagrammes statiques sont insuffisants. Cependant cette approche nous aide à repérer les composants métiers du système à modéliser. Il est désormais relativement simple de différencier les diagrammes présentant les données du système aux diagrammes modélisant des « objets » métiers appelés à rendre des services. La compréhension de ces « objets » métiers constitue la base de l'étape suivante de notre analyse, à savoir l'étude des composants du système. Telles des poupées gigognes, nous ouvrirons chaque « boîte noire » pour y découvrir des sous-systèmes, sous-sous-systèmes,... En partant du général, nous devrions préciser progressivement les comportements de ces composants.

5.3 Relevé des différents composants

Bien qu'à ce stade de notre analyse, l'intention n'est pas encore de proposer des solutions techniques, il peut parfois apparaître opportun de proposer des idées. En effet, désormais, nous entrons déjà dans une étape de définition d'une architecture dans laquelle des composants « physiques » devront interagir.

Ainsi, il peut arriver que des besoins relevés plus avant s'expriment ici en un choix « philosophique ». Ainsi, il est apparu dans les Use Cases que divers types de services, destinés à des clients tout autant multiples devraient être disponibles à tout instant. Certaines données identiques peuvent être utilisées par des composants différents afin que l'information soit traitée différemment selon les circonstances. Aussi, le choix de présenter le composant « communication » comme un système « d'abonnement » (cf. ci-dessous) pourrait paraître arbitraire car il insinue déjà une solution technique dans le système à mettre en place. Cependant, ce choix s'il propose une piste à creuser n'impose en rien une et une seule voie technologique possible.

Ci-dessous, nous présentons la plupart des modules que nous avons pu relever tant dans notre analyse des besoins que dans la modélisation statique. Nous avons essayé de les regrouper en fonction de leur spécificité et de leur connexion. Cette représentation ne reflète pas forcément une réalité architecturale concrète qui serait déjà implémentée ou à développer.

5.3.1 Composants « communs »

- **Système d'abonnement à différents services** : Il apparaît que la plupart des composants à mettre en place devront communiquer entre eux, pour par exemple demander des infos ou services et fournir des infos et des services. Ce composant permettra donc à tous les autres modules de retrouver les autres modules auxquels des messages doivent être envoyés.
- **Système de gestion des « repository » de l'aéroport** : Il contiendra toutes les informations concernant l'état d'un aéroport à chaque instant donné. Ainsi, tous les mobiles se trouvant dans l'enceinte de l'aéroport, toutes les alertes enclenchées, tous les incidents, des données météo, des données de sécurité. D'autres « repository » seront également accessibles.
- **Système de gestion des alertes – sécurité :** Il gère les alertes qui sont lancées et doit les répercuter vers les systèmes qui ont demandé à être prévenus.

5.3.2 Communication

Nous avons intentionnellement choisi de présenter ce composant alors qu'habituellement il n'intervient que lors des conceptions techniques des systèmes. En effet, dans notre projet, il intervient déjà dans les considérations fonctionnelles. Ainsi, est-il demandé que notre système puisse offrir un système « d'abonnements » pour des clients qui souhaiteraient bénéficier de services. Par ailleurs, la mobilité des acteurs et des composants du système impose une prise en compte rapide de cette problématique de communication.

Ainsi, ce composant est celui qui permet à tous les autres composants de communiquer entre eux. Sans présager d'une solution technique, ce module peut offrir en gros deux types de services :

- Proposer un annuaire de services accessibles et en expliquer le moyen pour l'atteindre et l'utiliser.

- Proposer un service d'abonnement et de publication à des services prédéfinis et connus.

5.3.3 Composants des acteurs (chauffeurs, contrôleurs, techniciens radar,...)

- **Système d'association Mobile – Acteur** : composant permettant d'associer un acteur - chauffeur à un véhicule de manière à ce que le système sache à tout instant quel chauffeur conduit quel véhicule.
- **Identification des acteurs** : composant qui permet d'identifier assurément chaque intervenant autorisé.

5.3.4 Composants du système A-SMGCS de contrôle du trafic

- **Système multi-décisionnel d'autonomie** : composant qui doit s'occuper de décider si le système global de gestion du trafic doit être temporairement autonome ou non.
- **Système de surveillance de l'aéroport** : composant qui fournit aux contrôleurs, et éventuellement aux pilotes (à tous ceux qui sont « abonnés ») l'état de la situation sur l'aire de l'aéroport. Il doit s'occuper de représenter et calculer l'état de l'aéroport à chaque instant. Il doit interroger un repository qui connaît à chaque instant les informations de chaque mobile, les conditions météorologiques, les configurations de l'aérodrome, ..
- **Système de routage au sol et de prévention des incidents** : composant qui attribue les routes pour des mobiles et qui vérifie le bon suivi de ces routes.
- **Système de guidage** : composant qui permet de donner des indications claires aux chauffeurs quant à l'état du trafic.
- **Système de Contrôle** : Système autonome qui contrôle à chaque instant les données concernant l'aéroport, les véhicules circulant et calculant les risques d'incidents entre les véhicules.
- **Système de captures des mouvements de mobiles (trackers, radar, transpondeurs, ..)** : Si ce module est dissocié du composant « A-SMGCS », il en est cependant fort lié. C'est ce composant, divisé en sous-composants, qui permet de suivre les mobiles dans l'aéroport :
    - Suivi des mobiles terrestres identifiés et munis de transpondeurs
    - Suivi des mobiles terrestres intrus et/ou non identifiés.
    - Suivi des déplacements d'avions en vol.
    - Gestion des décollages et atterrissages des avions.

5.3.5 Composant IHM

Ce composant est celui qui réunit tous les traitements liés aux outils utilisés par les acteurs pour utiliser le système, et pour communiquer avec lui. Parmi ces outils, citons les PDA, les PC (ou assimilés), les GSM, les bornes sur aéroport, ...

5.3.6 Une multitude de composants

Sur base d'une rapide analyse des exigences et de la présentation de scénarios d'utilisations mis en oeuvre par des diagrammes de séquence, des grands composants « fonctionnels » ressortent. Cette simple énumération des composants permet de modéliser ces systèmes, sous-systèmes et composants comme suit :

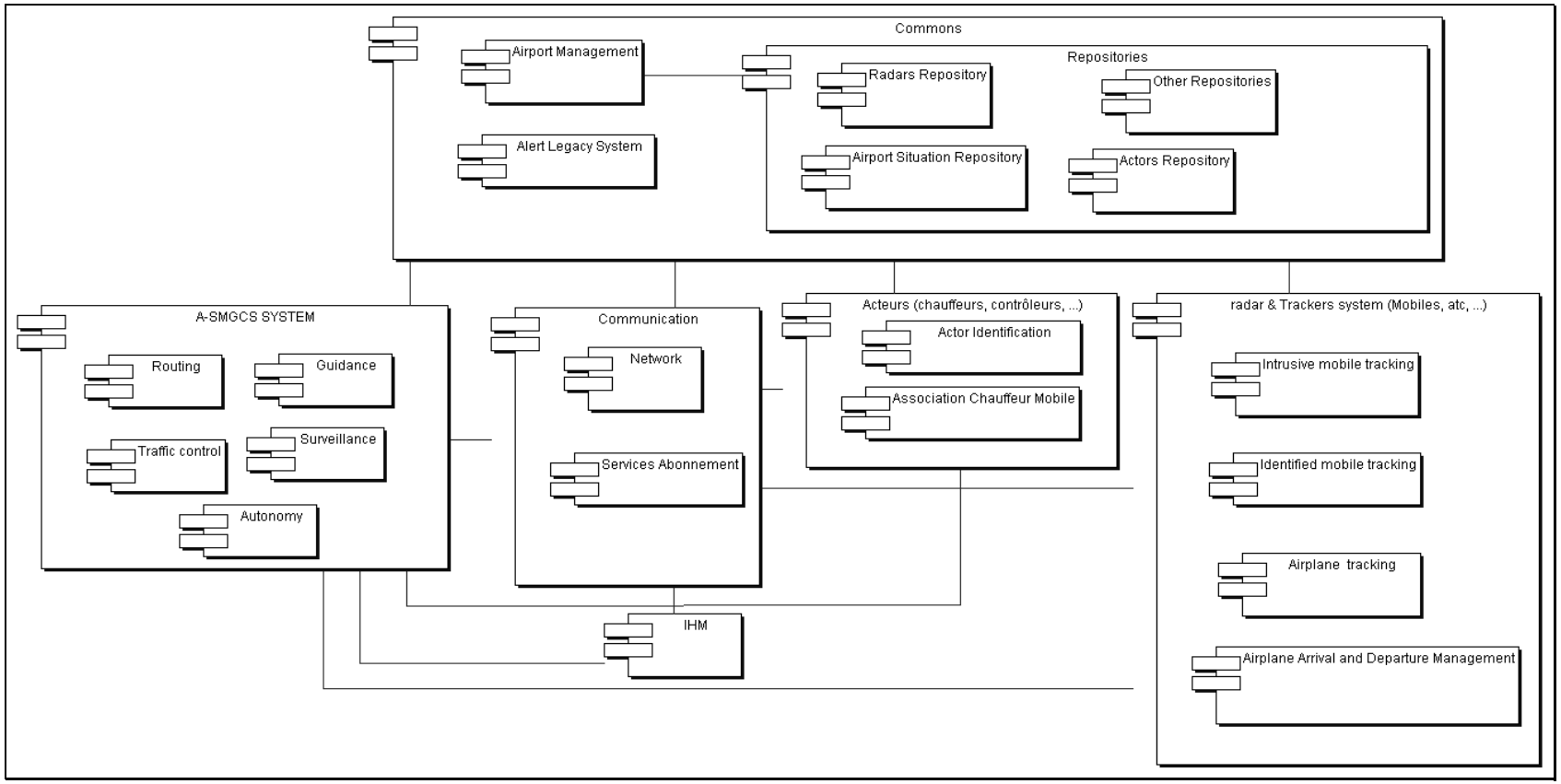

Figure 5.3.6: Diagramme de composants : Composants en interactions

5.4 Analyse de chaque composant

La plupart des composants du système de gestion du trafic et d'échange d'informations peuvent se concevoir comme autant de sous-systèmes autonomes qui sont soumis à des contraintes fortes.

Dans notre volonté de simplifier autant que faire ce long processus de développement logiciel pour notre système, nous allons représenter chaque composant indépendamment les uns des autres de manière à en exprimer les contraintes internes et externes quant aux services qu'ils doivent rendre. Dans un second temps, nous pourrons associer chaque composant comme autant de « boîtes noires » rendant un ensemble de services et complexifier cet assemblage en ajoutant des contraintes sur les interactions entre ces mobiles.

Il sera dès lors plus facile de repérer d'éventuels blocages, incompatibilités ou « deadlock » entre une contrainte « globale » par rapport à des contraintes « locales » à des composants.

Afin de représenter ces composants (sous-systèmes, modules, ...), nous avons opté pour une représentation via un diagramme d'état qui permet de modéliser chaque étape dans le processus d'un traitement dont les contraintes de temps et d'états sont très importantes. La succession des états, les conditions (contraintes) pour passer d'un état à un autre et le type de messages à échanger sont facilement modélisables par ces diagrammes.

Par ailleurs, cette représentation est très efficace pour comprendre le comportement que doit avoir un module pris isolément. Dans des systèmes complexes, il est souvent préférable de pouvoir cerner chaque composant pour que des spécialistes soient en charge de leur conception, maintenance et utilisation. A partir du moment où ces composants définissent clairement les services qu'ils peuvent rendre et les conditions strictes d'exécution, les autres composants du système peuvent se fier au module.

5.4.1 Diagrammes d'états-transitions

Ces diagrammes visualisent des automates d'états finis, du point de vue des états et des transitions. Avant de présenter nos diagrammes, il est utile de présenter succinctement les principaux concepts liés à ce type de modélisation[19].

5.4.1.1 Etat

Chaque objet (système) est à un moment donné dans un état particulier. Les états se caractérisent par la notion de durée et de stabilité. Ainsi un objet est toujours dans un état particulier pendant une durée, définie ou non. Un objet ne peut donc jamais être dans un état inconnu ou non défini.

En UML, les automates peuvent être « déterministes » et ne peuvent pas laisser de place aux ambiguïtés [MULLER, 1999]. Ce sont ces types de diagrammes d'états-transitions que nous retiendrons dans notre approche. En effet, l'une des caractéristiques des systèmes temps réels est cette nécessité du déterminisme. Aussi, faut-il toujours décrire l'état initial d'un système.

---

[19] Pour plus d'information, il convient de se référer aux ouvrages de références sur UML [BOOCH, RUMBAUGH, JACOBSON, 2001 ; MULLER, 1999].

Des articles spécifiques sur l'utilisation des diagrammes d'états-transitions pour la représentation des composants d'un système temps réel peuvent également intéresser le lecteur [HAREL, 1997 ; DOUGLASS, 1998 ; DOUGLASS, 2003 ; LIEBERMAN, 2001 ; DEL BIANCO, 2003]. Par ailleurs, la plupart des ouvrages/articles traitant de UML et du temps réel exposent cette méthode de représentation des composants.

Pour un niveau hiérarchique donné, il y a toujours un et un seul état initial. A l'inverse, il peut y avoir plusieurs états finaux correspondant à des conditions de fin différentes.

5.4.1.2 Transition

Lorsque les conditions dynamiques évoluent, le objets changent d'état en suivant les règles décrites dans l'automate associé à leurs classes. Les diagrammes sont donc des graphes orientés. Les états sont reliés entre eux via des connexions unidirectionnelles, appelées transitions.

Le passage d'un état à un autre se fait lorsqu'une transition est déclenchée par un événement survenant dans le domaine du problème. Le passage d'un état à un autre est immédiat et ne doit pas être pris en compte dans des contraintes temps réelles. En effet, tout système doit toujours être dans un état connu. Ainsi, le passage d'un état à un autre ne peut introduire de l'inconnu dans l'état d'un système !

5.4.1.3 Evénements

Un événement correspond à l'occurrence d'une situation donnée dans le domaine du problème. Contrairement aux états qui durent, un événement est une information instantanée. Un événement sert souvent pour passer d'un état à un autre. Ces passages peuvent dépendre d'une combinaison d'événements et de conditions à remplir.

5.4.1.4 Historique

Par défaut, un automate n'a pas de mémoire. En UML, il est possible de mémoriser le dernier sous-état visité et de le rejoindre lors d'une transition entrant dans le super-état qui l'englobe.

5.4.1.5 Transitions temporisées

Les attentes peuvent être vues comme des activités durant une période déterminée. Puisque l'on a postulé que les transitions sont immédiates, une attente doit être rattachée à un état, qui est interrompu une fois qu'un événement attendu a lieu.

Ainsi est-il possible de modéliser les dépassements de délais dans l'accomplissement d'une activité. Ci-dessous, un exemple d'une transition d'attente qui permet de modéliser le système d'alerte en cas d'incident.

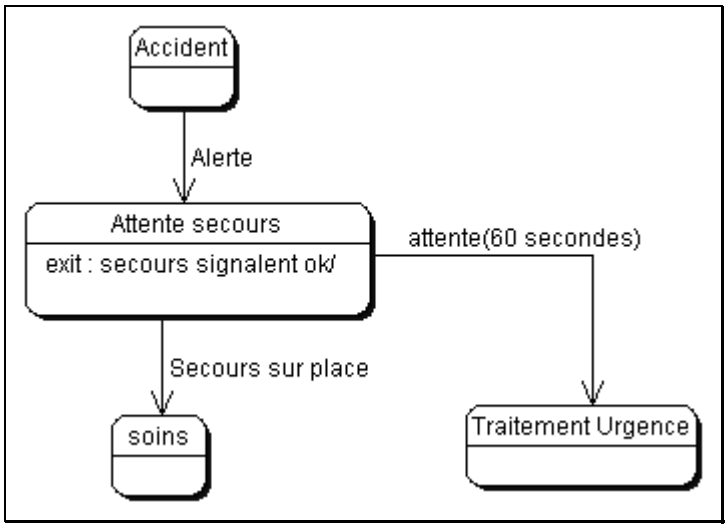

Figure 5.4.1.5: Diagramme d'états: contraintes de temps

Ainsi, si un incident se produit une alerte est lancée. Une attente de 60 secondes est mise en marche. Si après ces 60 secondes les secours n'ont pas signalé qu'ils étaient sur place, alors un traitement d'urgence sera lancé. Sinon, le processus normal suit son cours.

5.4.2 Composants du système de contrôle du trafic

5.4.2.1 Système de communication

Il apparaît sur base de tous les Use Cases que nous avons pointés que les composants fonctionnels devront communiquer entre eux de manière sûre, efficace et rapide. Aussi avant de modéliser chaque composant fonctionnel, nous présenterons une modélisation possible du composant « communication », sans présager de la solution reprise. Si des termes utilisés peuvent éventuellement faire penser à des solutions techniques, il s'agit essentiellement du fait que, dans le langage de la communication, le vocabulaire est restreint.

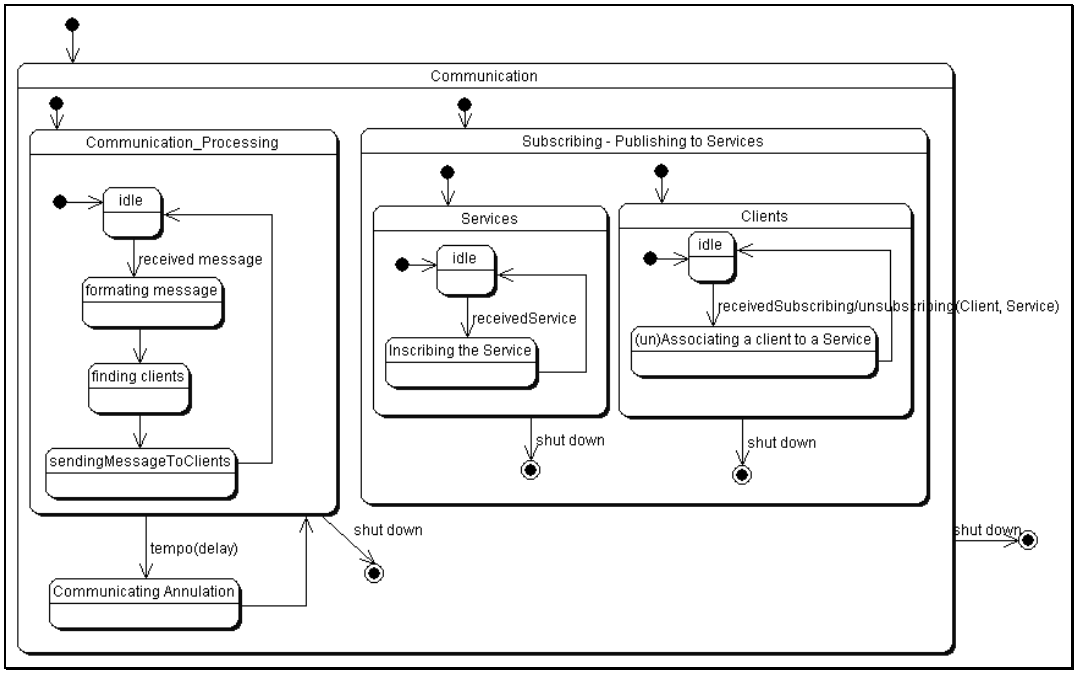

Figure 5.4.2.1: Diagramme d'états: communication

Ce diagramme d'état ci-dessus, haut niveau, permet de montrer que le composant de communication offre plusieurs services :

- **Communication** d'informations, de services, d'un composant à d'autres composants.
- **Inscription de services**, afin de les faire connaître. Ce module permet donc de retrouver tous les services accessibles par tous les composants et/ou acteurs du système
- « **Abonnement » de clients à des services**. Ainsi, des modules-composants pourraient demander de profiter de services. Cet abonnement peut être de différents types. Il peut être limité dans le temps ou dans l'espace, illimité, « one-shot », ...

➔ La notion de service est un concept large. Il peut s'agir d'abonnement à des données, information, applications,...

Au démarrage, deux sous-systèmes seraient lancés et ils tourneraient en parallèle indéfiniment, ne pouvant être arrêtés que suite à une demande explicite d'arrêt.

Ces sous-systèmes devraient donc être « indépendants » et tourner simultanément. Il est à noter que ce composant doit être accessible à tout instant.

Aucune contrainte temporelle ne peut être faite sur ce composant précis, mais ces contraintes devront être exprimées sur la combinaison entre composants. Ci-dessous dans le diagramme « Airport Status », nous verrons comment le modéliser.

5.4.2.2 Système multi-décisionnel de prise d'autonomie

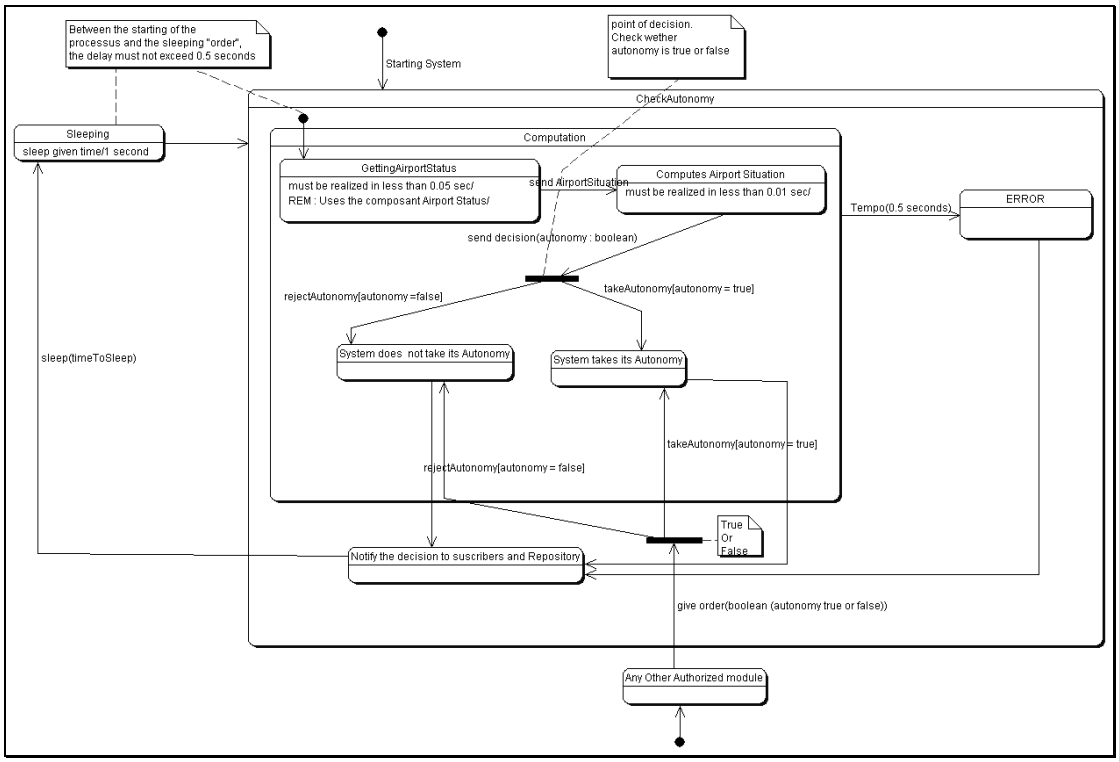

Figure 5.4.2.2: Diagramme d'états: Autonomie

Sur base de ce diagramme d'états, il est possible d'affiner le comportement que doit adopter un composant défini de notre système. Ainsi, le diagramme ci-dessus représente le fonctionnement d'un composant autonome qui doit tourner en permanence et être sans cesse à l'écoute.

Ce module doit donc, toute les x secondes vérifier que le système de gestion du trafic doit prendre/conserver la mainmise sur certaines prises de décision (guidage en cas de visibilité mauvaise, guidage en cas d'accident, ...).

Ce module peut « démarrer » son action de deux manières :

1. Chaque x secondes un processus se réveille et lance le composant en son « point de démarrage.

   Une fois déclenché, ce composant doit obligatoirement finir sa tâche dans la demi seconde qui suit (avoir enclenché un « sleep » de x secondes – 1 seconde)

   Le composant interroge un repository pour demander quel est l'état de l'aéroport. Cet état est un ensemble de données non encore définies, et à définir au cas par cas. On peut citer la visibilité : existe-t-il un incident en cours de traitement , de quel type, une mainmise a-t-elle été explicitement demandée/annulée par un acteur autorisé, ...)

   Il envoie cet « AirportSituation » à un module d'évaluation de ces données. Sur base d'un modèle d'analyse complexe, ce module décide rapidement (délai < 0.01 seconde) si la mainmise du guidage doit être prise par le système ou non. Quelle que soit la décision, un module préviendra tous les sous-systèmes qui doivent être prévenus, mettra à jour les informations dans les repository nécessaires et se mettra en « sleep » pour un temps donné.

2. A tout instant, un acteur autorisé (via une interface) peut demander explicitement au système de prendre/relâcher sa mainmise. A ce moment, il n'est pas nécessaire de vérifier la situation de l'aéroport, ni de traiter ces données. Seuls les états permettant de traiter cette prise/relâche d'autonomie doivent être atteints.

   A ce moment le « sleeping » peut être plus long. On peut imaginer que le « sleeping » soit : « tant qu'aucun ordre externe ne vient le démentir ». La notion de « TimeToSleep » n'est donc pas expressément définie en terme de délai temporel (seconde, minute, millisecondes, ...).

Enfin, si le délai de 0.5 secondes accordé pour ce module à calculer sa décision de prise d'autonomie du système est dépassé, le système passera alors dans un autre état annulant cette démarche.

Cette première approche d'un module nous a permis de constater que ce composant, bien qu'il puisse et doive vivre de manière indépendante, est en relation avec d'autres modules. Que ce soit pour les informer, pour consulter des informations ou pour être lancé de « l'extérieur », les liens avec d'autres modules aux contraintes qui risquent d'être antagonistes sont déjà perceptibles.

5.4.2.3 Système de surveillance de l'aéroport

Ce composant a pour vocation de fusionner et de calculer les informations concernant l'état de l'aéroport, sur la circulation, sur la météorologie, sur les alertes en cours, sur les événements qui se produisent, ...

Toutes ces informations sont présentes dans un système de divers repository qui sont alimentés par tous les autres systèmes : (Météorologique, radars, trackers, alerte, ...).

Ce composant récupère toutes les données et les traite de manière à pouvoir fournir toutes les informations de l'aéroport (d'une partie de l'aéroport, d'une partie des données,...) à un moment donné.

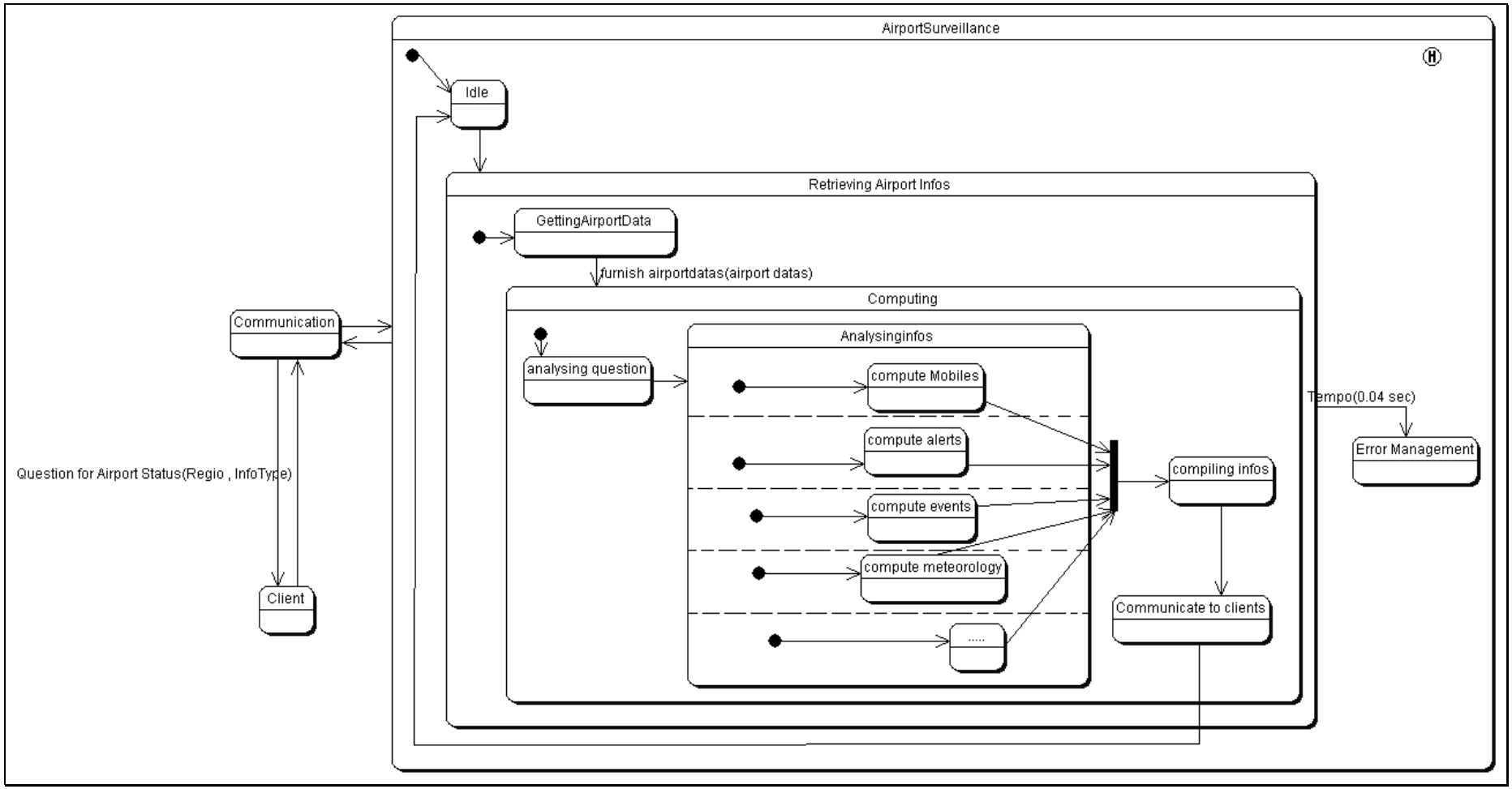

Figure 5.4.2.3a: Diagramme d'états: Surveillance

Une fois la compilation des données de l'aéroport obtenue, le composant doit transmettre les informations au client et s'arrêter.

Il est à noter que l'état général du composant est estampillé du logo H, qui signifie que ce module conserve le dernier sous état actif avant qu'il ne quitte la machine. Ainsi, il est possible de retourner directement dans ce sous-état par la suite.

La contrainte temporelle liée à ce composant qui impose qu'une fois la question posée la réponse doit être fournie dans les 0.05 secondes, implique deux composants séparés. En effet, tant le composant de communication que celui traitant la requête doivent travailler ensemble pour répondre à la contrainte. Aussi, pour modéliser cette contrainte fonctionnelle devons-nous passer par un stade de « méta-diagramme d'état » englobant les deux diagrammes :

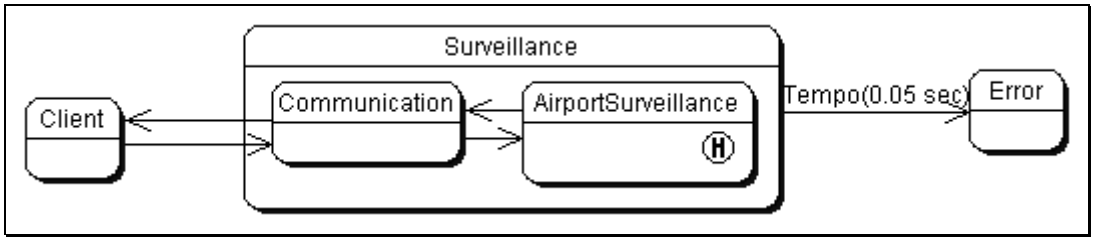

Figure 5.4.2.3b: Diagramme d'états: Surveillance et contraintes de temps

Cette manière de modéliser cette contrainte temporelle pourrait donc être utilisée pour tous les composants étudiés ci-après.

5.4.2.4 Routage

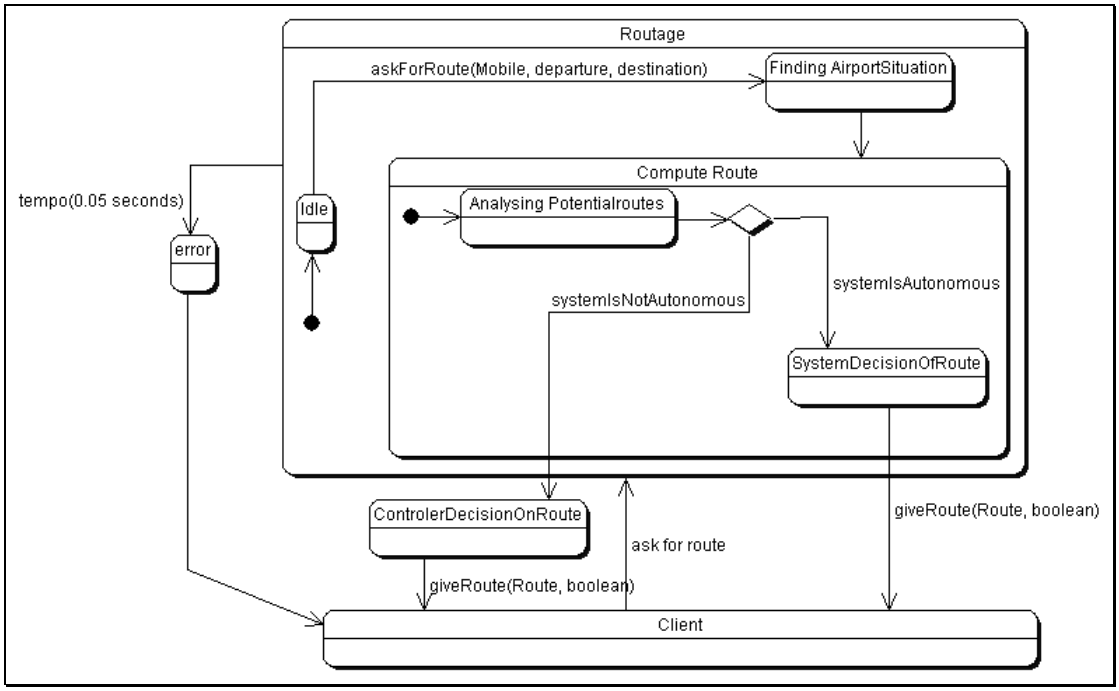

Figure 5.4.2.4 : Diagramme d'états: Routage

Ce diagramme présente le composant de routage. La fonction de ce composant est donc de donner les routes à suivre aux clients qui en font la demande.

L'état « FindingAirportSituation » correspond en fait au composant global de « Surveillance ». En effet, la « Surveillance » de l'aéroport se limite à capturer toutes les

informations liées à l'aéroport, à les fusionner et à les transmettre à qui le souhaite. Sur base de ces informations, le module calcul les routes à suivre.

Il apparaît clairement dans ce diagramme que le module n'est pas « éternel ». Ce composant n'existe que lorsqu'il est appelé. Il doit formuler une réponse au client dans les 0.05 seconde sans quoi une erreur sera lancée et un processus de traitement des erreurs devra être conçu.

5.4.2.5 Système de contrôle, de prévention d'accident et d'autorisation

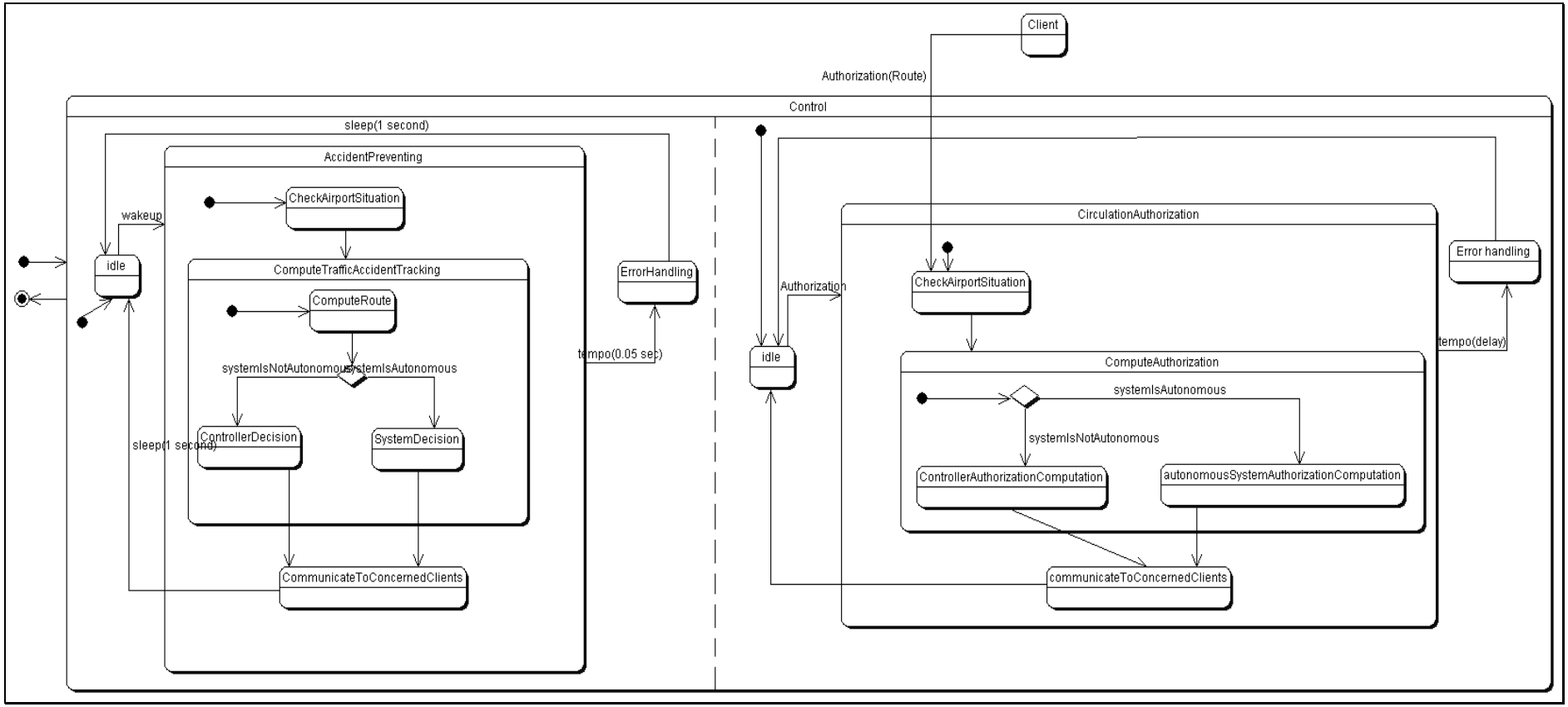

Figure 5.4.2.5: Diagramme d'états: contrôle

Dans ce composant, nous avons souhaité ne plus montrer les interactions entre les composants de communications et les clients avertis par le composant. Ces actions sont toutes prises dans l'état « CommunicateToConcernedClient ». Derrière cette « boîte noire » se cachent donc tous les mécanismes de transfert de données entre acteurs, interfaces, et autres composants interagissants.

Autorisation de circuler

Le premier permet de contrôler en permanence que les autorisations de circuler et de suivre la route donnée sont toujours valides. Il permet aussi d'être interrogé à distance par un client demandant l'autorisation de circuler.

Quand le système, autonome ou via un contrôleur, a donné sa décision, il en prévient les différents clients intéressés (chauffeurs, contrôleurs, composants de routage afin qu'il recalcule une nouvelle route, ...)

Ce composant tourne donc en permanence. Il ne s'arrête jamais et peut « dormir » un laps de temps à définir. En cas d'erreur, une gestion d'erreur est à prévoir. Les clients seront aussi prévenus.

Prévention des accidents

Ce deuxième sous-état permet au système de contrôler en permanence (toute les secondes) si des véhicules ne risquent pas de rentrer en collision.

Il apparaît clairement que ce module sera l'un des plus critiques au sein de notre système. En effet, c'est ce composant qui permettra à tout le système d'assurer une cohérence et une sécurité optimale dans la gestion du trafic au sol.

Ainsi, si les modèles utilisés ne nous permettent pas de spécifier concrètement comment le calcul du mouvement des mobiles devra être fait, notre approche permet tout de même de repérer l'endroit où ces spécifications « bas niveau » devront être faites, et isoler les composants critiques qui auront un impact sur ce module.

5.4.2.6 Guidage

Le composant permettant de guider correctement les chauffeurs des mobiles qui se déplacent dans l'aéroport, et tous ceux qui souhaitent recevoir ces infos (pilotes, contrôleurs, forces de l'ordre, agents de sécurité, ...)

A l'instar du modèle présentant l'état de l'aéroport, il convient de modéliser les contraintes temporelles issues de la spécification des besoins via le même principe de « meta diagramme ».

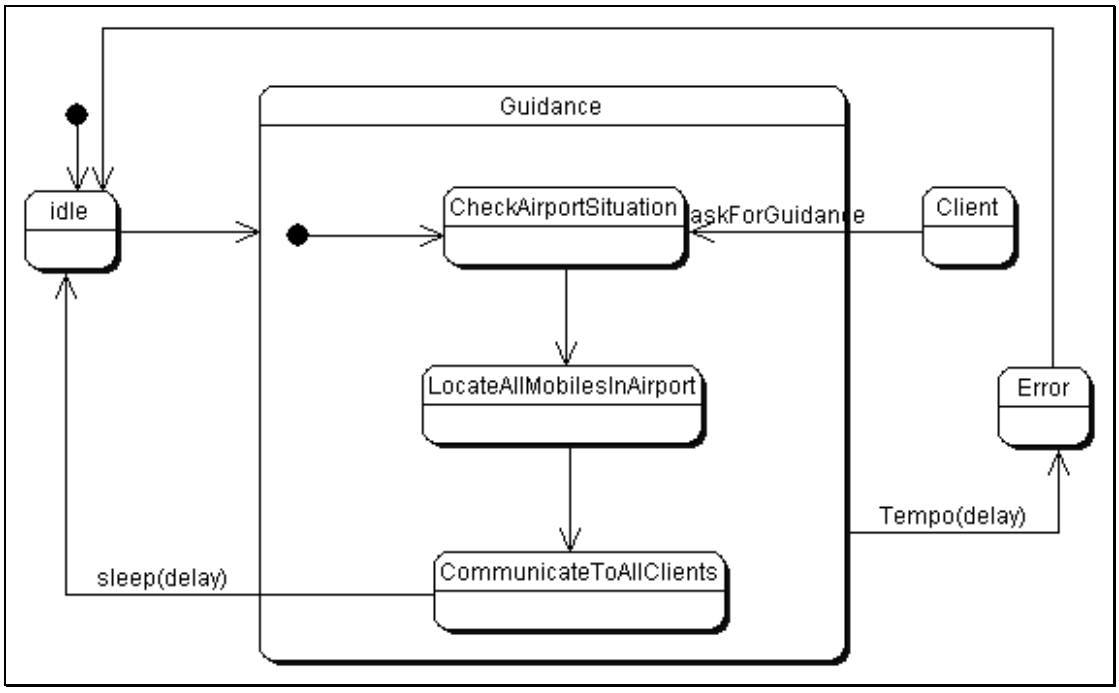

Figure 5.4.2.6: Diagramme d'états: Guidage

Sur ce modèle, il apparaît clairement qu'un blocage peut survenir. En effet, le composant « principal » de ce module pourrait être plus sévèrement contraint qu'un composant « client ». Ainsi, le composant « CheckAirportSituation » (AirportSurveillance) qui doit obligatoirement rendre une réponse dans les 0.04 secondes pourrait être trop lent si la contrainte sur le module de guidage était plus forte.

Dans ce cas, nous n'avons pas explicitement donné de contraintes pour ce module, car celles-ci peuvent être variables selon les conditions d'entrées et les questions posées à ce module. Le délai est donc « paramétrable ».

5.4.2.7 Globalité du système

Après avoir modélisé les différents modules qui composent le système A-SMGCS, il apparaît clairement que des interdépendances existent entre ceux-ci. Certains composants doivent faire appel à d'autres pour pouvoir réaliser la tâche qui est la leur.

Il est donc possible de montrer les dépendances entre les différents services comme suit :

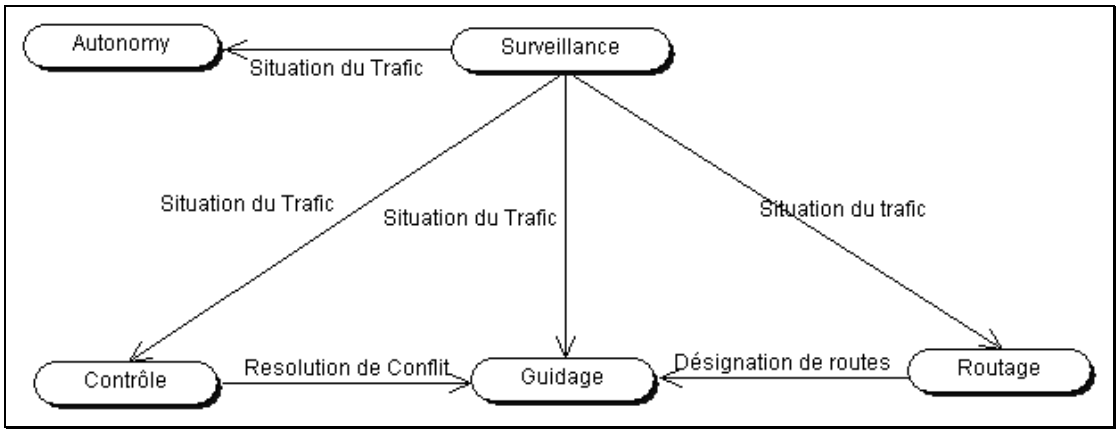

Figure 5.4.2.7a: Diagramme d'états: interdépendance entre composants

La fonction de guidage est celle qui recueille les informations de chaque module pour pouvoir aider les acteurs (chauffeurs, pilotes, contrôleurs, ...) à se guider sur la piste de l'aéroport.

Parallèlement, le module vérifiant l'autonomie du système n'a besoin que de l'état de l'aéroport pour « exister ».

Sur base de cette représentation il est possible de représenter le système A-SMGCS sous forme d'un seul diagramme d'état reprenant les différents composants :

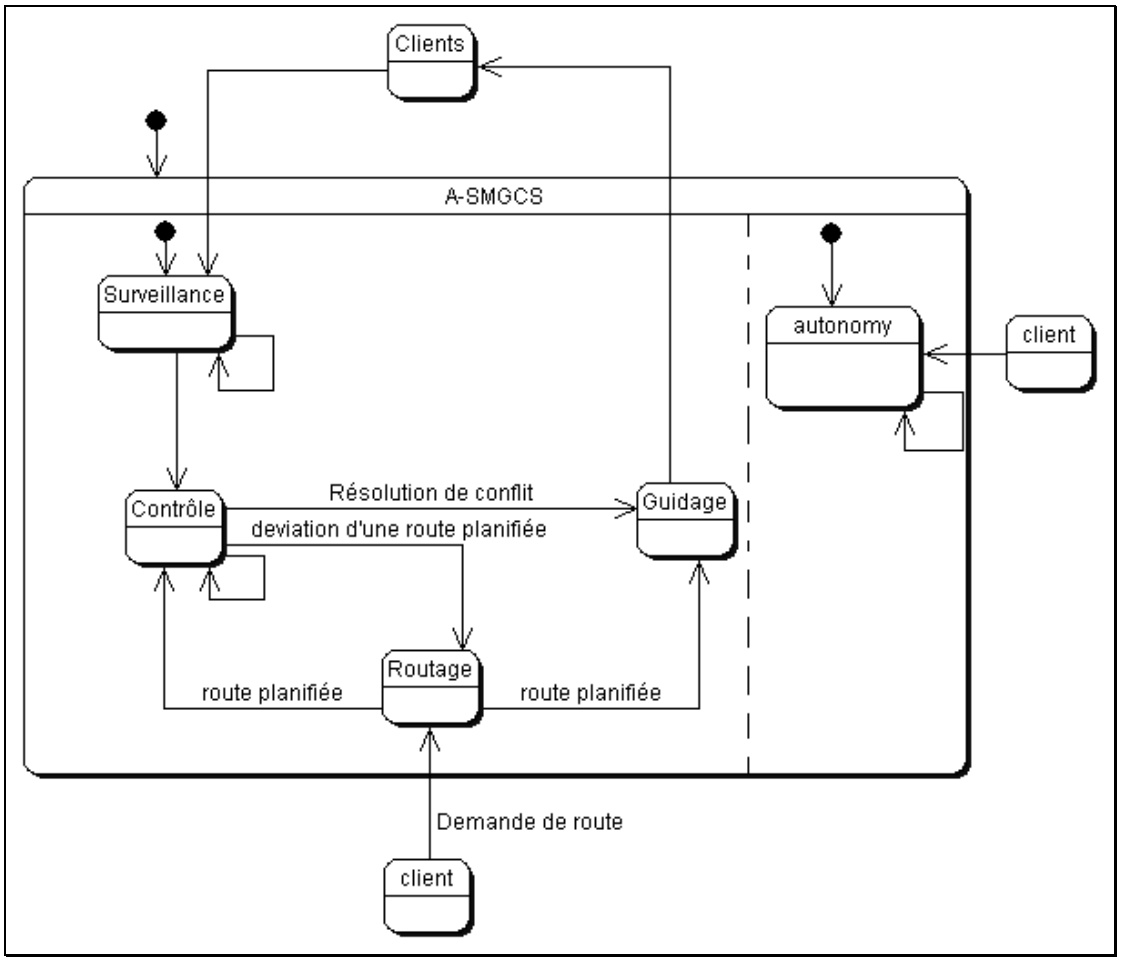

Figure 5.4.2.7b: Diagramme d'états: A-SMGCS global

Cette modélisation ne montre pas toutes les intéractions possibles entre composants aux travers des passages d'états. La réalité peut être plus complexe et l'entrée dans le système A-SMGCS, peut se faire directement à partir de la plupart des états. Les « états clients » peuvent quasi tous s'attendre à se voir « réactivés » à partir d'un état du système. Ainsi, un client peut demander un contrôle et attendre une réponse de cet état. Le concept de client utilisé ci-dessus désigne essentiellement tout état d'un autre système qui souhaiterait lancer un processus dans le système A-SMGCS, en s'attendant à être réactivé par le système A-SMGCS ou non.

5.5 Analyse Objet - précisions

Si selon Douglass [DOUGLASS, 2001], cette étape ne doit intervenir que suite à l'analyse des composants du système, nous l'avons, de notre propre chef, intégrée comme préalable à cette phase d'analyse. Cependant, la modélisation des composants du système nous permet d'avoir une vue des objets en interaction plus précise encore. Si les réalités métiers n'ont pas changé, les objets symbolisant les fonctionnalités du système à développer peuvent être affinés. Ainsi, dans cette partie, nous allons présenter un diagramme de classe plus précis pour le système A-SMGCS. Avec cette présentation statique, nous montrerons le comportement que devraient avoir certains objets en utilisant un exemple de diagramme de séquence.

5.5.1 A-SMGCS : Diagramme d'objets

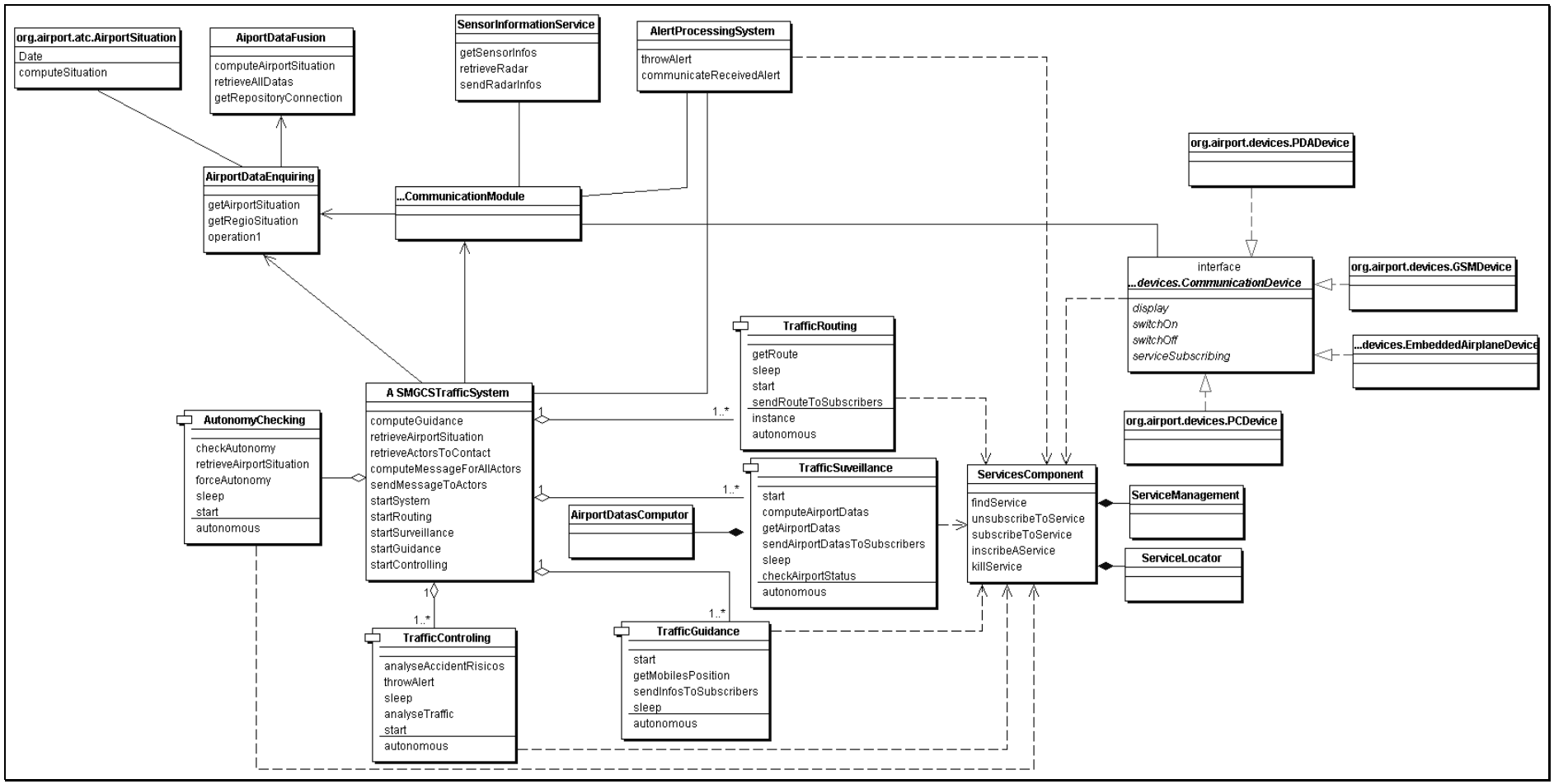

Figure 5.5.1: Diagramme de classe: Système A-SMGCS

Ce diagramme est une extension du premier diagramme de classe. Nous avons précisé les 4 modules propres au A-SMGCS auxquels nous avons ajouté des opérations qui nous ont été révélées lors de la phase précédente.

Par ailleurs, nous avons vu que chaque composant a des besoins similaires en terme d'information. Fournir une route à un mobile et contrôler l'état de l'aéroport nécessite une bonne connaissance de l'état de l'aéroport. Ces besoins peuvent être autant de services que des modules autonomes peuvent fournir. Ces services pouvoir être multiples. Aussi, avons nous modélisé un service général qui permet de mettre tous les autres services à disposition des composants du système.

Ce ServicesComponent permet donc à chaque composant de trouver un service dont il aurait besoin, et également de se faire connaître en terme de service disponible pour d'autres.

Parallèlement à ce « ServicesComponent », tous ces objets sont également reliés entre eux par un module de communication (CommunicationModule) qui offre les services de communication entre objets, plate-forme, ... Ainsi, derrière ce composant se cachent les problématiques liées au réseau, aux protocole de communication, ...

Il est évident que cette proposition n'est pas innocente en terme de choix architectural et technologique futur, mais il s'est imposé petit à petit suite aux raffinements successifs de l'analyse. Dans l'environnement distribué et hétérogène où nous allons devoir implanter notre système, il convient d'avoir un mécanisme le plus générique possible qui nous permette d'utiliser tous les composants déjà existants sans devoir les repérer à l'avance. Dans la partie architecture qui suit, nous tenterons de voir si un tel choix est judicieux et réalisable.

5.5.2 Interaction entre objets

5.5.2.1 Démarrage Système

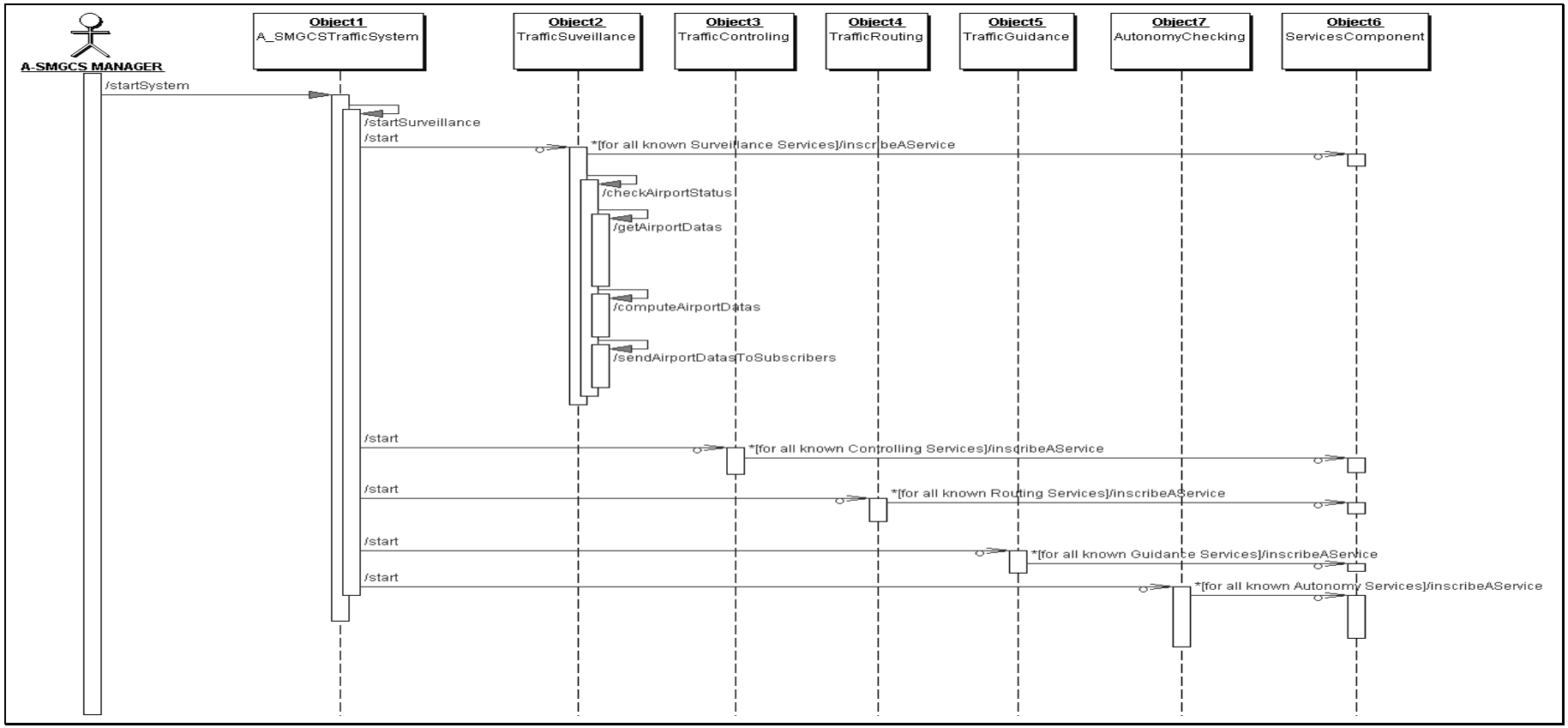

Figure 5.5.2.1: Diagramme de séquence : démarrage du système

En miroir à la modélisation statique, il est judicieux de montrer les interactions entre les objets pour valider notre modèle. Ci-dessus, un diagramme de séquences montre comment les objets interagissent entre eux pour des actions bien précises.

Le diagramme ci-dessus montre comment le système A-SMGCS lance tous les processus « autonomes » de surveillance, contrôle, guidage, routage et d'autonomisation. Ces services au démarrage inscrivent tous les services qu'ils peuvent rendre au système central de gestion des services.

Il est à noter que dans cette partie nous ne nous soucions pas encore des problématiques liées aux threads concurrents ou non qui peuvent tourner simultanément. Quand un composant appelle un service d'un autre composant, il est possible que ce service soit disponible simultanément pour plusieurs clients ou que chaque client doive attendre son tour. Il est bien entendu que pour notre système, une trop longue attente n'est pas permise et qu'une gestion de ces accès aux services est indispensable. Cette approche devrait être abordée dans les parties davantage techniques du processus de développement.

5.6 Conclusion

La double approche, statique et dynamique, bien qu'elle ne soit pas proposée par ROPES [DOUGLASS, *ROPES*, 1999, pp. 24-29], nous semble pourtant judicieuse dans le domaine qui nous intéresse. En effet, le passage de la modélisation des besoins et des contraintes du système à la conceptualisation de composants autonomes relativement bien définis n'est pas automatique. Au sortir de l'analyse des besoins, le système à développer apparaît comme une « boîte noire » aux fonctionnalités définies, mais également comme un système dont la répartition des responsabilités n'est pas encore claire. Afin de passer du stade de « boîte noire » à celui de système modulaire aux responsabilités définies, nous sommes donc passés par un stade intermédiaire dans lequel nous avons modélisé le système au moyen d'un diagramme de classes.

Ce modèle offre l'intérêt de pouvoir passer des réalités métiers et fonctionnelles à une représentation davantage formelle. Par ailleurs, des contraintes exprimées sous forme textuelle peuvent déjà se concrétiser dans ce modèle. Enfin, les relations, dépendances, communication, .. entre objets du réel peuvent apparaître.

Ce stade intermédiaire nous a donc permis de « débroussailler » la complexité du système, des messages, des réalités du terrain en isolant chaque réalité dans des diagrammes aisément lisible. Cette modélisation simplifiée ouvre également la voie sur la représentation des composants indépendants qui devront interagir.

Cette étape franchie, chaque composant fonctionnel a pu être modélisé par des diagrammes d'état. Cette technique offre l'avantage de montrer le comportement de chaque composant, indépendamment des autres. Il est également possible de contraindre le comportement de ceux-ci en y ajoutant des exigences temporelles, de sécurité, ... Chaque diagramme peut par la suite être vu comme une pièce d'un grand puzzle qu'il suffit alors d'assembler pour aboutir au « supra système » demandé.

Cependant, il apparaît également que des contraintes plus complexes, liées aux données qui doivent être traitées (analyse du déplacement des véhicules, gestion des alertes, ...) sont de nouveau difficilement modélisables par ce seul outil. Le composant de « contrôle du trafic » s'il doit bien faire son travail en « x secondes » ne dit rien de comment traiter les données reçues, comment repérer les éventuels accidents potentiels. Cette problématique liée à la limitation des modèles UML pour représenter les contraintes fortes de temps réel a bien été montrée dans l'article de Lavazza, Quaroni et Venturelli [LAVAZZA, QUARONI, VENTURELLI, 2001]. En utilisant l'exemple traditionnel de la gestion d'un passage à

niveau (GRC), ces auteurs ont proposé des solutions pour arriver à modéliser plus sûrement ces types de problématiques en combinant UML et d'autres langages plus formels.

Cependant, le passage par les modèles utilisés peut être très éclairant dans le cadre d'un projet tel que le notre. En effet, la décomposition en composants, eux-mêmes divisés en états, permet de repérer très aisément les sous-états critiques et les états-composants, qui y sont liés. Aussi, dans les étapes de spécifications plus raffinées, ces modules devront être particulièrement choyés et privilégiés. Il revient aux chefs de projet de donner les ressources les plus appropriées pour spécifier concrètement les algorithmes de traitement des données (évaluer les dangers d'accident,...) ou de capture de celles-ci (fonctionnement des radars, transpondeurs, ...).

Par ailleurs, ces modèles permettent aussi de pointer les composants souvent utilisés par d'autres. Ces différentiations seront utiles lorsqu'il faudra songer aux solutions architecturales concrètes à mettre en œuvre. Un composant isolé ne devra pas forcément tourner sur une plate-forme accessible par d'autres. A l'inverse, un composant souvent appelé devra être aisément accessible et d'une disponibilité sans faille.

6 Critique de la démarche suivie

6.1 Introduction

Séduisante, la méthodologie ROPES éveille en nous des sentiments mitigés. En effet, après usage, il apparaît que plusieurs zones d'ombres subsistent dans ce processus de développement utilisant presque exclusivement UML comme langage de spécification et de modélisation. Cependant, il est important de constater que ROPES nous a permis d'enchaîner naturellement les étapes de notre analyse en affinant successivement les modèles que nous produisions. Dans cette partie, nous allons donc synthétiser les critiques tant positives que négatives inspirées par l'utilisation de ROPES et UML dans notre étude s'intéressant à la modélisation des besoins et des contraintes d'un système temps réel embarqué. Nous proposerons également des pistes pour résoudre certaines imperfections. Sans être exhaustif, nous avons parcouru la littérature s'exprimant sur le sujet et nous évoquerons des idées de solutions relevées dans ces articles.

6.2 Apports et limites de ROPES et d'UML

Présenté comme un processus particulièrement adapté au développement des systèmes temps réels embarqués, ROPES n'est en fait qu'une adaptation du Processus Unifié (RUP) originellement imaginé par Jacobson, Booch et Rumbaugh. Il n'est donc pas particulièrement révolutionnaire et adapté aux seuls développements temps réels. Par ailleurs, la littérature accompagnant ROPES propose évidemment un outil case commercial qui permet de suivre à la lettre les recommandations du processus de développement. Si nous n'avons pas eu l'opportunité de tester cet outil, le lien commercial entre ROPES et un outil permet toutefois de relativiser l'éventuel désintéressement académique qui sous-tendrait la diffusion de cette méthodologie.

Itérative, la méthodologie suivie est essentiellement axée sur la recherche de résultats concrets rapides et visibles. En effet, l'un des objectifs est de pouvoir montrer aux clients et aux intervenants d'un projet que la direction suivie est la bonne. La réalisation d'un prototype à chaque itération du processus permet de valider ce qui a été fait. Si dans les phases d'analyse, des prototypes ayant des comportements techniques observables sont difficiles à concevoir, l'utilisation de modèles UML facilement lisible pour des non techniciens offre une solution pour proposer un prototype compréhensible et exploitable dans les phases ultérieures du projet. Cependant, à ce stade, les contraintes liées au temps réel, à la sécurité,... sont difficilement exprimables en termes clairs et précis. Les Use Cases utilisés pour préciser les échanges d'informations entre les acteurs et le système restent souvent de très haut niveau. Sans doute faudrait-il associer à cette représentation, une méthodologie spécifique à la capture des besoins et des exigences. Citons à titre d'exemple ***I**** qui pourrait être utilisée dans cette phase afin de définir plus précisément les besoins de notre système critique [KU,97].

Au terme de la rédaction de ces Use Cases, il est donc fondamental de proposer un ou plusieurs scénarios qui mettent en scène ces Use Cases, constituant ainsi un prototype. UML permet de passer facilement d'une présentation textuelle à une représentation davantage formelle, par les diagrammes de séquence où des précisions temporelles peuvent être ajoutées. Cependant, ces spécifications temporelles sont essentiellement limitées à des contraintes liées à la « rapidité » du système. Il est en effet très difficile de pouvoir modéliser des contraintes complexes impliquant des conditions temporelles imbriquées et/ou associées à des exigences d'occurrence d'événements. Nous sommes toujours en attente d'un raffinement de la méthodologie, ou de UML, qui nous forcerait à utiliser un formalisme plus avancé, nous permettant de proposer par la suite un prototype davantage solide, pour lequel des contraintes plus complexes pourraient être exprimées et formellement validées.

Nous en sommes réduits aujourd'hui à saisir ces contraintes sous forme textuelle et à indiquer par exemple, qu'en cas d'alerte, si les secours n'arrivent pas dans un délai raisonnable alors une procédure d'urgence sera lancée. Ces contraintes sont cependant exprimables par certaines techniques formelles, non souhaitées pour ROPES !

Une fois la capture des besoins bouclée, ROPES propose de passer à la modélisation des composants du système à mettre en place et de laisser à plus tard la représentation « objet » de la réalité du système. Nous avons préféré intervertir ces deux phases. En effet, les diagrammes de séquence du prototype ne nous permettaient pas de définir suffisamment les grands composants fonctionnels du système à développer. Aussi, nous pensons que les étapes proposées par Douglass doivent être plus un canevas adaptable dont on peut s'inspirer plutôt qu'une règle stricte à suivre aveuglément. Nous avons donc privilégié une approche statique pour préparer l'analyse dynamique du système.

La modélisation statique du système autorise également la spécification de certaines contraintes à la fois fonctionnelles et aussi de sécurité. Cependant, cette représentation, à ce stade, est essentiellement destinée à clarifier les différents composants du système et leurs interactions entre eux.

Une fois cette étape passée, la modélisation des composants métiers du système est grandement facilitée. Nous avons donc pu, grâce aux diagrammes d'états, isoler tous les sous-composants qui interagissent entre eux. De nouveau, nous nous sommes très vite retrouvés face à des limitations de UML pour spécifier des contraintes complexes. Ainsi, il est possible de modéliser les comportements « extérieurs » de chaque composant pris séparément, mais il est impossible de préciser le comportement du système globalement lorsque les modules sont en interaction. Cependant, cette approche en décomposition modulaire peut nous éclairer sur la complexité des modules à concevoir. Très vite, nous nous sommes rendus compte que certains états dans les machines à états finis représentaient des nœuds fondamentaux du système. Si nous n'avons pas les outils suffisants pour préciser ces contraintes par des spécifications formelles, nous pouvons, grâce à notre approche, circonscrire les parties critiques du système. La complétude de l'analyse devra venir de l'approfondissement de ces parties par une étude plus fouillée et des spécifications plus précises.

De tels projets temps réels impliquent souvent des contraintes liées au processus de développement lui-même. La traçabilité transversale tout au long du projet doit être assurée. Pour chaque décision prise, chaque besoin exprimé, il est important que l'on puisse retrouver dans les toutes les étapes du processus de développement les dépendances liées à ce besoin. De cette exigence de traçabilité, aucune mention n'est faite dans la littérature sur ROPES. C'est sans doute l'un des autres points faibles de cette méthodologie. Sans cette rigueur méthodologique qui imposerait une politique de traçabilité, aucun développement critique tel que celui que nous étudions ne peut débuter ! Il en va tout autrement pour des petits systèmes embarqués non critiques, mais est-ce dès lors nécessaire de concevoir une telle méthodologie qui ne pourrait être utilisée que pour des projets sans grands risques humains !

Il apparaît donc que Douglass, dans son utilisation quasi exclusive de UML pour modéliser des systèmes temps réels, n'a pas réussi à résoudre les problèmes liés au manque de sémantique formelle de UML. Les problèmes complexes et critiques où l'imprévu, l'inhabituel, doivent être pris en compte ne peuvent être résolus par la simple utilisation exclusive des modèles proposés par UML. Dès la phase d'analyse des besoins, il paraît indispensable de pouvoir rapidement exprimer formellement des contraintes. Pour des systèmes critiques, il importe de ne pas laisser des zones d'ombres. Aussi, serait-il intéressant d'élargir les méthodes d'analyse sans se limiter à un seul langage de modélisation, fut-il unifié !

6.3 Améliorations et extensions possibles

Tout au long de ce travail, nous avons donc constaté que nous étions quelque peu limités par UML pour représenter des besoins complexes liés à des contraintes temps réelles. Par ailleurs, au temps réel, s'ajoute la nécessité de pouvoir modéliser les réalités du système que nous étudions. Ainsi, représenter la mobilité de véhicules et l'espace dans lequel ceux-ci évoluent est relativement complexe et nécessite une approche complémentaire à celle utilisée. Notre volonté dans cette partie n'est pas de combler les manques que nous avons pu relever, mais de proposer des pistes de réflexions permettant d'affiner les futurs processus d'analyse de tels systèmes.

6.3.1 Une combinaison de UML et de notations formelles[20]

Ayant bien résumé les faiblesses de UML pour modéliser les systèmes temps réels, Luigi Lavazza [LAVAZZA, QUARONI, VENTURELLI, 2001 ; LAVAZZA,2001] propose une double approche pour les développements de systèmes temps réels. Dans un premier temps, UML sera utilisé pour modéliser les composants du système, les interactions entre eux et les contraintes simples qui y sont associées. Cette étape devrait être assez simple, économique et accessible pour tous les intervenants du projet. Dans un deuxième temps, il propose d'y associer des concepts davantage formels. L'idée ultime serait de pouvoir convertir ces modèles « automatiquement » dans des notations plus formelles permettant les vérifications des propriétés, les preuves de corrections, les vérifications de sécurité, ...

- Notations déclaratives adaptées pour les preuves formelles
- Notations opérationnelles qui permettent la simulation
- TRIO[21]
- ...

L'idée serait d'associer à UML le langage TRIO et de concevoir des outils permettant ces conversions. TRIO est un langage formel et une méthode pour les spécifications, l'analyse et la vérification de systèmes critiques temps réels. Basé sur une extension métrique de la logique temporelle, TRIO exploite également les concepts de l'Orienté Objet. La méthode proposée offre également la possibilité de valider le travail réalisé par des ensembles de tests.

Cette approche est intéressante dans le sens où elle encourage l'utilisation de UML en association avec d'autres types de notations. Par l'utilisation de UML nous pouvons garantir la facilité de communication entre les intervenants d'un même projet et la possibilité de circonscrire les composants complexes des autres. Par l'ajout de langages formels, nous garantissons la solidité du système et la correction de celui-ci. Il serait donc très utile d'étudier et de critiquer plus avant cette proposition.

D'autres auteurs ont également proposé des alternatives permettant d'associer des méthodes plus formelles pour contrebalancer les limites de UML, mais notre objectif ici n'est pas de les étudier toutes[22].

---

[20] Parmi les articles sur lesquels nous avons basé notre réflexion, citons la bonne synthèse de ce que devra être tout développement de système embarqué dans la décennie qui vient par Hermann Kopetz [KOPETZ, 2000] .

[21] Pour plus d'informations, sur TRIO : http://www.elet.polimi.it/res/TRIO/ ce site propose des liens intéressant sur TRIO ainsi qu'une bonne bibliographie et de outils (prototypes) pour utiliser ce langage.

[22] Parmi ces études citons l'introduction des Message Sequence Charts et des Live Sequence Charts pour aider à la modélisation des échanges de messages entre objets d'un système temps réel. Cet apport pourrait aider à une meilleure capture des besoins [DAMN, 2000]. Citons aussi les associations entre

6.3.2 Modélisation du trafic au sol

Qui dit mobiles se déplaçant sur des routes bien définies, dit aussi modélisation complexe de ces véhicules. UML, comme langage de modélisation d'une réalité statique est très utile et relativement simple à aborder. Il en va tout autrement pour représenter une réalité dynamique. Ainsi, dans le cas qui nous occupe, un court article propose de modéliser la réalité d'un aéroport et des services A-SMGCS par des diagrammes UML [LEMOINE, 2000]. Cependant, s'il est possible de modéliser statiquement l'organisation d'un tel milieu, de préciser qu'un véhicule se trouve à un endroit à un moment, il est relativement complexe de modéliser ses déplacements, ses interactions avec les autres véhicules et les besoins liés à la prédictabilité de là où il se trouverait dans x secondes. De telles exigences font appel à des concepts d'intelligence artificielle et de théorie des graphes, domaines difficilement intégrables dans les méta-modèles UML. Afin de pallier à ces manques, nous jugeons utile ici de présenter deux approches différentes qui ont pour but de modéliser et de spécifier ces exigences.

Dans son article sur la gestion de la circulation des avions au sol, Dragos Stoica [STOICA, 1999] présente une manière de modéliser ces déplacements au sol en associant plusieurs méthodes :

- La simulation numérique (simulation multi-agents et multi-échelles de temps)
- Théorie des graphes (réseaux de Pétri et diodes, files d'attente)
- Techniques d'optimisation mono et multicritères
- Méthodes d'analyse et de diagnostic de l'Automatique Symbolique.

En représentant la structure d'un aéroport par un graphe orienté et les avions (adaptables pour les autres mobiles) par des jetons, il opte pour un choix de modélisation très formelle. Par une approche mathématique relativement complexe, il tente d'optimiser la circulation des avions au sol par la programmation linéaire.

Ce type de modélisation/spécification est à ce point formelle qu'il est très difficile pour des personnes non initiées de la comprendre. Cependant, le domaine étudié est complexe et il est sans doute vain de penser que la simple approche UML permettra de couvrir l'entièreté de la problématique.

Enfin, des chercheurs spécialisés dans les problématiques du A-SMGCS, ont dès 1996 conçu un prototype pour une application d'Intelligence Artificielle d'aide à la circulation au sol de mobiles dans un aéroport [BLAESS, 1996]. Ce système expert n'est qu'un composant d'un plus vaste système de gestion de trafic. L'ensemble des règles, de la base de connaissance, et des contraintes n'est modélisable qu'avec des techniques d'un formalisme certes hermétique pour les clients et pour certains informaticiens, mais pour de tels systèmes, cette approche est inévitable et doit aussi être abordée dans le processus de développement, et UML est actuellement totalement inadapté pour ces besoins.

6.3.3 Design : Proposition d'une solution technique

Si nous n'avons pas proposé de solution technique pour notre étude, il est à noter qu'une proposition basée sur les technologies JAVA et JINI a été réalisée par Laurent HONET [HONET, 2002]. Ce dernier propose en effet d'organiser la plupart des services liés à la surveillance aéronautique dans un système dont l'architecture serait basée sur la technologie

---

SDL et UML [ALKHODRE, 2001 ; MAMMERI, 2002]. Enfin, un court article résumant différentes attitudes face à UML et temps réel permet aussi de trouver des pistes pour améliorer ces processus de développement logiciel [MARTIN, 2001]

JINI[23]. Une étude approfondie de cette piste constituerait un prolongement naturel de notre étude. Il est à noter que notre système s'intégrerait sans aucun problème dans la solution proposée par Laurent Honet. En effet le système JINI devrait permettre à tous les acteurs définis dans notre travail de bénéficier de tous les services offerts par le système (A-SMGCS) où qu'ils soient dans l'enceinte de l'aéroport et sur n'importe quel type de plate-forme. Un contrôleur derrière sa station SUN pourrait contrôler le trafic, tandis qu'un chauffeur derrière son volant pourrait consulter son PDA et être continuellement prévenu par ce même système. Aussi, il serait très intéressant d'éprouver cette approche méthodologique dans un travail davantage technique.

6.3.4 Traçabilité

Ce concept de traçabilité est relativement bien étudié dans le Processus Unifié [JACOBSON, BOOCH, RUMBAUGH, 1999] et il serait intéressant de s'y référer pour des projets ayant des exigences de sécurité, de fiabilité et de rapidité telles que celles que notre projet présente. Nous n'avons pas jugé utile d'approfondir cette question qui a déjà été largement étudiée et traitée dans la littérature[24]. Le plus important, nous semble-t-il est d'être conscient de cette problématique et de l'intégrer au démarrage de tout projet !

6.4 Conclusion

Au terme de cette critique nous constatons que les portes sont encore grandes ouvertes pour améliorer le processus de développement ROPES que nous avons utilisé dans ce mémoire. Les points forts du processus et de UML devraient être conservés et consolidés par l'apport d'autres méthodes d'analyse. Pour des problématiques aussi complexes que celle étudiée ici où des vies humaines sont en jeu et où la sécurité est au centre du système, il convient de minimiser les risques d'imprécision. Ainsi, un développement suivant les seules recommandations telles que proposées par B.P Douglass ne suffit pas. S'il est très intéressant de suivre la méthodologie ROPES, il convient d'y apporter des compléments. Ceux-ci seront d'une part d'ordre méthodologique pour ce qui est de la traçabilité des activités tout au long du processus. D'autre part, d'un point de vue technique il convient également d'élargir les outils de spécification et de modélisation pour couvrir les domaines non rencontrés par UML.

[23] Pour une bonne approche de JINI, nous ne pouvons que conseiller la visite du site de SUN Microsystems : http://java.sun.com .

[24] Il suffit pour s'en convaincre de parcourir l'application Rational Unified Process inclue dans Rational Rose 2000. Conçue comme un ensemble de pages Web d'aide à l'application de RUP, plusieurs liens insistent sur l'importance de la traçabilité dans le processus de développement.

7 Conclusion générale

De l'objectif de départ de ce travail, que pouvons nous retirer de la démarche que nous avons suivie ? De notre volonté de concevoir un système coopératif temps réel de gestion du trafic dans un aéroport, quels sont les écueils que nous avons rencontrés ? Qu'avons nous fait pour sortir d'une impasse dans laquelle nous nous serions fourvoyés? Pourquoi avons-nous préféré un choix plutôt qu'un autre ? Telles sont les principales questions qui se sont posées tout au long de ce travail et que nous allons synthétiser ici. D'un objectif idéal ambitieux, nous avons dû limiter nos ardeurs et canaliser notre recherche.

Préalablement avant toute volonté de conception d'un système d'information, qu'il soit temps réel ou non, il importe de définir clairement la démarche méthodologique la mieux adaptée au problème étudié. Dans le cas présent, l'un des gourous de la technologie temps réel embarquée, B.C. Douglass, propose une méthodologie de développement qui laisse supposer une adéquation parfaite à notre problématique : ROPES. C'est dans l'idée d'éprouver et de valider cette méthodologie que nous avons donc fondé notre démarche. Or, après une analyse de ce processus de développement, il est très vite apparu que tant l'innovation que l'adaptation aux développements de systèmes embarqués étaient sans doute exagérées. Largement inspiré de RUP, la méthodologie suivie est donc dans la lignée des processus itératifs à la mode depuis la fin des années quatre-vingt dix qui utilisent presque exclusivement UML comme langage de modélisation et de spécification.

Tout au long de ce travail nous avons donc suivi les recommandations de ROPES pour analyser le système de gestion de trafic au sol d'un aéroport, suivant les principes du A-SMGCS. Partant des besoins définis dans la littérature, nous avons synthétisé les besoins et contraintes de ce système idéal. Ces exigences, nous avons tenté de les modéliser par la technique des Use Cases pour finalement aboutir à un prototype conceptuel à proposer au client pour validation. Si cette technique offre l'avantage d'être particulièrement naturelle pour l'être humain, elle est cependant limitée pour la représentation des contraintes temps réelles inhérentes au type de système étudié. Aussi, est-elle davantage recommandée pour les phases d'analyse haut-niveau dans lesquelles il convient de se mettre rapidement d'accord sur les fonctions de base du système à concevoir. Vouloir représenter des scénarios trop complexes impliquant des contraintes déterministes trop sophistiquées serait vain ! Pour décrire de tels besoins, il serait intéressant d'associer assez tôt au langage naturel une méthode de modélisation plus formelle dans laquelle certains comportements critiques du système et des acteurs seraient spécifiés. Cette double approche satisferait ainsi les clients tout autant que les intervenants techniques du projet.

Une fois cette étape de la définition des besoins du système réalisée, ROPES propose d'extraire de nos modèles et autres diagrammes les grands composants métiers de celui-ci. Nous avons jugé qu'au stade où nous nous trouvions il était prématuré de proposer des composants et de spécifier leur comportement dynamique. C'est pourquoi nous avons préféré présenter, dans un premier temps, une approche statique du système en adoptant une « analyse objet ». Cette phase de notre analyse nous a permis de modéliser certaines contraintes par de simples relations et contraintes liées à ces relations. Par un simple diagramme de classe nous avons pu imposer au système que deux véhicules ne pouvaient pas se trouver simultanément au même endroit ! De nouveau, cette modélisation est venue d'une manière toute naturelle à partir de l'étape d'analyse précédente. De cette étude statique, les composants du système sont eux aussi apparus avec évidence. Leur modélisation dynamique n'en est devenue que plus simple. Cependant, à l'instar de notre critique des Use Cases, il est vite apparu que l'analyse des composants telle que proposée par ROPES était insuffisante. Basant exclusivement la représentation de ces modules par des diagrammes d'états, certaines contraintes, sur des traitements à réaliser sont difficilement modélisables. L'apport de cette démarche n'est toutefois pas négligeable. En effet, nous avons pu mettre le doigt sur les

composants les plus critiques. Nous avons réussi à isoler les modules qui nécessiteraient une attention particulière lors des phases de spécification ultérieure. Ainsi, il serait donc intéressant de compléter à nouveau les outils à notre disposition pour pouvoir consolider les modèles produits par des spécifications plus formelles.

Au delà de la phase d'analyse fonctionnelle que nous avons proposée dans notre mémoire, ROPES propose une marche à suivre pour continuer les analyses techniques, les architectures, les phases de programmation et de test. Comme nous l'indiquions en début de notre conclusion, nous n'avons pas vocation à la conception complète d'un système autonome. Ainsi, nous nous sommes limités à analyser ROPES pour la première phase de ce processus.

Des pistes que nous avons suivies, nous avons soulevé de nouvelles questions qui restent encore ouvertes. Des études complémentaires seraient donc les bienvenues pour proposer des solutions innovantes afin d'améliorer la capture et la spécification des besoins. L'association à UML de langages et de processus plus formels devrait à terme permettre d'utiliser à bon escient la force de modélisation qu'UML possède, tout en reconnaissant ses limites.

Pour combler les manques de notre étude, les pistes ne manquent pas, les techniques non plus. Faisons confiance à la curiosité et à la sagacité des chercheurs qui finiront sans doute par proposer des solutions aux questions qui sont encore en suspens et associer à UML des techniques de spécifications mieux adaptées pour modéliser des systèmes temps réels.

8 Dictionnaire

A-SMGCS	Advanced – Surface Movement Guidance and Control System
ASTERIX	Format d'échange d'informations issues de radar dans les aéroports. http://www.eurocontrol.int/projects/eatchip/asterix
ATC	Air Traffic Control
GPS	Global Positioning System
Mobile	Tout véhicule qui circule dans l'enceinte d'un aéroport. Un Mobile peut être un camion, un tracteur, une voiture, un avion, … pour autant qu'il circule sur le sol.
Mode S Radar	Mode Select SSR. Type de radar propre au contrôle de mobiles (http://www.eurocontrol.int/mode_s)
QoS	Quality of Service
RADAR	Système ou appareil de détection qui émet un faisceau d'ondes électromagnétiques très courtes et en reçoit l'écho, permettant ainsi de déterminer la direction et la distance d'un objet.
ROPES	Rapid Object-Oriented Process for Embedded Systems
RUP	Rational Unified Process
Senseur	Tout outil qui permet de repérer les mobiles dans l'enceinte d'un aéroport (humain, GPS, radar, transpondeur, ...)
Taxiway	Liaison empruntée par les avions entre les parkings et les pistes. L'avion y roule à basse vitesse et souvent en mode de pilotage manuel.
Transpondeur	Appareil émetteur-récepteur qui répond automatiquement à un message d'identification au signal d'un radar.
UML	Unified Modeling Language
OACI	Organisation de l'Aviation Civile Internationale http://www.icao.int
EUROCONTROL	Organisation Européenne pour la sécurité de la navigation aérienne http://www.eurocontrol.int
JAA	Joint Aviation Authorities http://www.jaa.nl/
EUROCAE	Organisation Européenne pour l'Equipement de l'Aviation Civile http://www.eurocae.org/
CEAC	Conférence Européenne de l'Aviation Civile http://www.ecac-ceac.org/
BELGOCONTROL	Belgocontrol est une entreprise publique autonome ayant pour objet, entre autre, d'assurer la sécurité de la navigation aérienne

	dans les espaces aériens dont l'Etat belge est responsable. http://www.belgocontrol.be/

9 Bibliographie et illustrations

9.1 Ouvrages et articles

ALKHODRE A., BADAU J.-P. et SCHWARZ J.-J., « Méthodologie de développement des systèmes embarqués temps réel basée sur le langage SDL, in *3ème colloque de CAO de circuits et de systèmes intégrés*, (10-04-2004, URL : http://www-asim.lip6.fr/gdrcao/articles/0042-BFGT.pdf).

BLAESS, C., TSIAMPALIDIS, C. and VALLEE, J.-C., « An application of Artificial Intelligence for the Safety in the Neighbourhood of Airport Runways », in *Proceedings of the IEEE ICTAI'96 Workshop on Artificial Intelligence for Aeronautic and Space*, November 16, 1996, Toulouse, France.

BOOCH, G., RUMBAUGH, J., and JACOBSON, I., *The Unified Modeling Language User Guide*, Boston, 2001.

DAMM, W. and HAREL, D., « LSCs : Breathing Life into Message Sequence Charts » , in *Formal Methods in System Design*, September 2000.

DEL BIANCO, V., LAVAZZA, L., MAURI, M. and OCCORSO, G., « Towards UML-based formal specifications of component-based real-time software » , in *ETAPS 2003 Workshop FASE*, Warsaw, Poland, April 9-11, 2003.

DOUGLASS, B. P., « Capturing Requirements for Real-Time and Embedded Systems » , in CD-Rom of DOUGLASS, B. P., *Real-Time Design Patterns. Robust Scalable Architecture for Real-Time Systems*, Boston, 2003.

DOUGLASS, B. P., « ROPES : Rapid Object-Oriented Process for Embedded Systems » , 1999, in CD-Rom of DOUGLASS, B. P., *Real-Time Design Patterns. Robust Scalable Architecture for Real-Time Systems*, Boston, 2003.

DOUGLASS, B. P., « Safety-Critical Systems Design » , in CD-Rom of DOUGLASS, B. P., *Real-Time Design Patterns. Robust Scalable Architecture for Real-Time Systems*, Boston, 2003.

DOUGLASS, B. P., *Real-Time Design Patterns. Robust Scalable Architecture for Real-Time Systems*, Boston, 2003.

EUROCONTROL, *Stratégie de gestion de la circulation aérienne pour les années 2000+*, (10-04-2004, URL : http://www.eurocontrol.int/dgs/publications/brochures/v2_year2000_fr/main.html).

EUROPEAN ORGANISATION FOR THE SAFETY OF AIR NAVIGATION, *A-SMGCS Project Strategy*, 30-09-2003, (10-04-2004, URL : http://www.eurocontrol.int/airports/downloads/asmgcs_project_strategy.pdf).

EUROPEAN ORGANISATION FOR THE SAFETY OF AIR NAVIGATION, *Eurocontrol Standard Document for Surveillance Data Exchange. Part 1. All Purpose Structured Eurocontrol Surveillance Information Exchange (ASTERIX)*, December 2001, (10-04-2004, URL : http://www.eurocontrol.int/asterix/documents/Part%201/pt1ed129.pdf).

EUROPEAN ORGANISATION FOR THE SAFETY OF AIR NAVIGATION, *Overall Architecture for EATMP*, May 21, 2002, (10-04-2004, URL : http://www.eurocontrol.int/eatmp/library/documents/oata/OATA_Phase_1_Mngt_Oview%20_1.4.pdf.

HAREL, D. and GERY, E., « Executable Object Modeling with Statecharts » , in *Proceedings of the 18th international conference on Software engineering*, July 1997, IEEE, pp. 31-42.

HILLARY, N., « Bridging the Gap between Requirements and Design with Use Cases and Scenarios », in CD-Rom of DOUGLASS, B. P., *Real-Time Design Patterns. Robust Scalable Architecture for Real-Time Systems*, Boston, 2003.

HONET, L. , *A Jini-based Surveillance Data Distribution and Radar Quality Monitoring System. A way of Building a Surveillance Grid*, 2002 (Article confidentiel propriété de THALES Information System)

JACOBSON, I., BOOCH, G. et RUMBAUGH, J., *Le processus unifié de développement de logiciel*, Paris, 1999.

KU, E. S. K., « Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering » , in *Proceedings of the 3rd IEEE Int. Symp. on Requirements Engineering (RE'97)*, January 6-8, 1997, Washington D.C., USA.

LAVAZZA, L., QUARONI, G. and VENTURELLI M., « Combining UML and formal notations for modelling real-time systems » , in *Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering*, 2001, Vienna, pp. 196-206.

LEMOINE, M., « Managing the Airport Complexity : Contribution of the Object-Oriented Approach », in *Proceedings of ICAS 2000, International Congress on Aeronautical Sciences*, Harrogate, 27 Aout - 01 Septembre 2000.

MULLER, P.-A., *Modélisation objet avec UML*, Paris, 1999.

NIEVA, T., FABRI, A. and BENAMMOUR, A., « Jini Technology Applied to Railway Systems » , in *Proceedings of International Symposium on Distributed Objects and Applications*, September 21 - 23, 2000, Antwerp, Belgium (10-04-2004, URL : http://csdl.computer.org/comp/proceedings/doa/2000/0819/00/08190251abs.htm).

PAPPAS, G., TOMLIN, C., LYGEROS, J., GODBOLE, D. and SASTRY, S., « A next generation architecture for air traffic management systems, » in *IEEE Conference on Decision and Control,* San Diego, California, USA, December 10-12, 1997, pp. 2405-2440.

PERALDI-FRATI, M.-A., ANDRE, C. et RIGAULT, J.-P., UML et le paradigme synchrone : application à la conception de contrôleurs embarqués », in *Projet SPORTS. Rapport de recherche I3S/RR-2002-02-FR*, Février 2002, Sophia-Antipolis.

RACKL, G., LUDWIG, T., DE STEFANI,F., PASQUARELLI,A. and HÉRAN , F. « Distributed Airport Simulation using CORBA and DIS » , in *Proceedings of International Symposium on Software Engineering for Parallel and Distributed Systems*, May 17 - 18, 1999, Los Angeles, California.

STOICA, D., ACHAIBOU A. et MORA-CAMINO F., « Analyse, représentation et optimisation de circulation des avions au sol. Approche par les systèmes multi-agents », in

Rapport LAAS No02616,3ème Congrès des Doctorants de l'Ecole Doctorale Systèmes, Toulouse (France), 22-23 Mai 2002, 6p.

TEYSSIE, C., MAMMERI, Z., CARCENAC, F. et BONIOL ,F., « Etude comparative SDL et UML pour la modélisation de systèmes temps réel », in *11ème Conférence Internationale sur les Systèmes Temps Réel, RTS'2003*, Paris, France, Mars 2003.

VALLE, J.-C., « L'évolution de la surveillance des mouvements au sol sur les aéroports » , in *Revue Technique*, n°61, Décembre 2001, (10-04-2004, URL : http://www.stna.aviation-civile.gouv.fr/actualites/revues/revue61/61pgarticle2/fr61art2.html).

ZHANG, D.D., « Use Case Modeling for Real-Time Application » , in *Object-Oriented Real-Time Dependable Systems, 1999. Proceedings. Fourth International Workshop on*, January 27-29, 1999, Santa Barbara, CA , USA, pp. 54-64.

9.2 Articles publiés sur sites Internet

BESSIN, G., « Embedded Systems : A Primer », June 23, 2003 (10-04-2004, URL : http://www-106.ibm.com/developerworks/rational/library/806.html).

CHENG, N., J. « An integration Framework for Airport Automation Systems », 2001, (10-04-2004, URL : http://www.mitre.org/work/tech_papers/tech_papers_01/cheng_integration/cheng_integration.pdf).

DOUGLASS, B. P., « Capturing Real-Time Requirements », November 1, 2001 (10-04-2004, URL : http://www.embedded.com/story/OEG20011016S0126).

DOUGLASS, B. P., « Designing Real-Time Systems With UML (Part I-III) », (10-04-2004, URL : http://www.embedded.com/98/9803fe2.htm).

EHRMANNTRAUT, R., « Enabling Air-Ground Integration: Concept Definition for Traffic Information Service in Contract Mode (TIS-C) », Digital Avionics Systems Conference, Indianapolis, Indiana, USA, October 2003, (10-04-2004, URL : http://www.eurocontrol.fr/Newsletter/2003/December/Conferences/DASC/4c5.pdf).

EHRMANNTRAUT, R., « Systems-of-Systems Integration of Air-Ground Telecommunications with the Software Connector » , Digital Avionics Systems Conference Indianapolis, Indiana, USA, October 2003, (10-04-2004, URL : http://www.eurocontrol.fr/Newsletter/2003/December/Conferences/DASC/6a3.pdf).

EHRMANNTRAUT, R., « Enabling Air-Ground Integration: Definition of a Total Information Sharing Protocol », Digital Avionics Systems Conference, Indianapolis, Indiana, USA, October 2003, (10-04-2004, URL : http://www.eurocontrol.fr/Newsletter/2003/December/Conferences/DASC/4b1.pdf).

GULLEKSON, G., « Designing for concurrency and distribution with Rational Rose RealTime » , June 16, 2003, (10-04-2004, URL : http://www-106.ibm.com/developerworks/rational/library/269.html).

GULLEKSON, G., « What Is Real-Time Embedded Software » ,2001 (10-04-2004, URL : http://www-106.ibm.com/developerworks/rational/library/content/RationalEdge/dec00/WhatIsRealTimeEmbeddedSoftwareDec00.pdf).

KOPETZ, H., « Software Engineering for Real-Time : A Roadmap », (10-04-2004, URL : http://www.cs.ucl.ac.uk/staff/A.Finkelstein/fose/finalkopetz.pdf).

LAVAZZA, L., « Guidelines for exploitinting formal methods in the tools to be developed in WP2 » , (10-04-2004, URL : http://www.dess-itea.org/deliverables/ITEA-DESS-D175-V02P.pdf).

LIEBERMAN, B., *UML Activity Diagrams Versatile Roadmaps for Undersatnding System Behavior*, 2001 (10-04-2004, URL : http://www-106.ibm.com/developerworks/rational/library/content/RationalEdge/may01/UsingUMLActivityDiagramsfortheProcessViewMay01.pdf).

LYONS, A., « UML for Real-Time Overview. Technical report » , Rational Rose, April 1998. (10-04-2004, URL : http://pdv.cs.tu-berlin.de/forschung/KUKA_PJ-WS00/documents/uml_rt_overview.pdf).

MARTIN, G., LAVAGNO, L. and GUERIN, J.-L., « Embedded UML : a merger of real-time UML and co-design », March 5, 2001, (10-04-2004, URL : http://www.gigascale.org/metropolis/EmbeddedUML.whitepaper.v7.External.PDF).

McLAUGHLIN, M. J. and MOORE, A., « Real-Time Extensions to UML. Timing, concurrency, and hardware inrefaces » , (10-04-2004, URL : http://www.ddj.com/articles/1998/9812/).

RICHARDSON, M. W., « Object Based Development Using the UML and C » , February 14, 2000, (10-04-2004, URL : http://itpapers.zdnet.com/abstract.aspx?kw=uml&docid=8985).

SELIC, B. and RUMBAUGH, J., « Using UML for Modeling Complex Real-Time Systems » ,1998(10-04-2004,URL:http://www-06.ibm.com/developerworks/rational/library/139.html).

9.3 Liste des illustrations et tableaux

Chaque illustration ou tableau est numéroté en fonction du paragraphe sous lequel il se trouve.