the 20 000 585

50376
1999
307

Numéro d'ordre : 2673

UNIVERSITÉ DES SC. ET TECHN. BIBLIOTHÈQUE UNIVERSITAIRE LILLE

THESE

Présentée à

L'Université des Sciences et Technologies de Lille

en vue de l'obtention du grade de

DOCTEUR DE L'UNIVERSITE

Spécialité : Productique, Automatique et Informatique Industrielle

par

Etienne LEMAIRE

Maître E.E.A.

Spécification et Vérification Fonctionnelle et Comportementale d'un Equipement Intelligent

Soutenue publiquement le 14 décembre 1999 devant la commission d'examen :

Président :	E. Craye	Professeur, LAIL, Ecole Centrale de Lille
Rapporteurs :	M. Robert	Professeur, CRAN ESSTIN, Université Henri Poincaré Nancy I
	J.M. Faure	Professeur, CESTI-ISMCM, Toulon
Examinateurs :	M. Bayart	Professeur, LAIL, USTL (Directrice de Thèse)
	P. Desodt	Maître Assistant, Ecole des Mines de Douai
	M.A. Péraldi-Frati	Maître de Conférences, INRIA, Université de Nice Sophia-Antipolis
	M. Staroswiecki	Professeur, LAIL, USTL.

Cette thèse a été préparée au Laboratoire d'Automatique et d'Informatique Industrielle de Lille CNRS UPRESA 8021, USTL.

Remerciements

Le travail présenté dans ce mémoire a été effectué au sein du Laboratoire d'Automatique et d'Informatique Industrielle de Lille (LAIL) à l'université des Sciences et Technologies de Lille sous la direction scientifique de Madame Mireille Bayart, Professeur à l'Ecole Universitaire d'Ingénieur de Lille. Je tiens à la remercier vivement pour m'avoir accueilli dans l'équipe Analyse et Surveillance des Processus Industriels Complexes (ASPIC). Tous les conseils pertinents et surtout la patience qu'elle a su me témoigner ont permis l'aboutissement de ce travail.

Une partie de ce travail a été fait en collaboration avec le laboratoire des Systèmes Industriels Distribués (SID) du département Productique de l'Ecole des Mines de Douai. Je tiens par conséquent à remercier Monsieur Georges Cuiengnet, directeur du département, pour son accueil. Je remercie aussi chaleureusement tous les membres du laboratoire SID et plus particulièrement Monsieur Philippe Desodt pour leur aide et collaboration ainsi que pour la sympathie qu'ils m'ont témoignées.

Je tiens également à remercier les membres du jury qui me font l'honneur de participer à l'examen de ce travail :

Monsieur le Professeur Etienne Craye de l'Ecole Centrale de Lille pour avoir accepté la lourde et ingrate tâche de Président du jury,

Monsieur le Professeur Jean-Marc Faure de l'Institut Supérieur des Matériaux et de la Construction Mécanique et Monsieur le Professeur Michel Robert de l'Université Henri Poincaré Nancy I qui ont accepté la lourde charge de rapporteur,

Madame Marie-Agnès Péraldi-Frati, Maître de Conférences à l'Université de Nice Sophia-Antipolis, et Monsieur Marcel Staroswiecki, Professeur à l'Ecole Universitaire d'Ingénieur de Lille, pour l'intérêt qu'ils portent à ce travail en acceptant de participer au jury.

Je remercie tous les membres de l'équipe ASPIC passés et présents, pour leur gentillesse et la bonne ambiance qu'ils ont su maintenir au sein du laboratoire.

Je tiens aussi à adresser un remerciement tout particulier à Monsieur le Professeur Jean-Marc Toulotte, responsable de la Maîtrise EEA, pour le soutien qu'il a su m'apporter en fin de maîtrise et sans qui ce mémoire n'aurait jamais vu le jour.

Je remercie enfin tous mes amis et ma famille pour leur soutien qui m'a accompagné toutes ces années. Je remercie spécialement mes parents qui ont accepté d'assurer une part importante de mes charges pour permettre la conclusion de ce travail. Et pour finir, j'adresse une pensée affectueuse à Nathalie qui partage ma vie.

Sommaire

Introduction Générale..... 4

Chapitre I : La Problématique de la Conception..... 6

Chapitre II : Langages Utilisés dans la Conception..... 38

Chapitre III : Le Modèle Externe..... 62

Chapitre IV : Un Modèle Comportemental associé au Modèle Externe..... 106

Chapitre V : Application à un Régulateur de Vitesse de Croisière..... 133

Conclusion Générale 178

Annexe I : Présentation de la spécification sous Excel97..... 180

Annexe II : Présentation l'analyse de la spécification sous Excel97..... 188

Annexe III : Feuilles du Régulateur de Vitesse..... 194

Introduction Générale

Les progrès réalisés tant en électronique qu'en micro-électronique ont considérablement augmenté les capacités de traitement des outils informatiques. Ainsi, l'ère d'avancée technologique que nous vivons nous conduit à imaginer et à fabriquer des équipements de plus en plus complexes en incorporant aux équipements de base des moyens leur permettant d'effectuer des calculs plus ou moins élaborés et de communiquer à "l'extérieur" les résultats. Ces équipements capables de traitements locaux et de communication avec l'environnement deviennent alors "intelligents". Si "l'intelligence" introduite vise généralement à améliorer les rendements, elle tente aussi de sécuriser les applications tant au niveau du personnel qu'au niveau des installations en elles-mêmes. Les équipements intelligents sont utilisés, en particulier, dans deux domaines : l'industrie du process et les systèmes embarqués. Ces deux domaines possèdent, en général, de fortes contraintes d'environnement.

Ces progrès ont aussi amené leurs lots de problèmes. En effet, le fait de pouvoir communiquer avec son environnement et différents utilisateurs a introduit des problèmes d'adaptation dans l'environnement et de cohérence de l'équipement vis-à-vis de l'ensemble. Le coût de telles incohérences au sein d'une application peut être catastrophique tant au niveau fonctionnement qu'au niveau financier, voire au niveau de la sécurité des personnels.

Les problèmes qui apparaissent au cours du fonctionnement sont souvent dus à une mauvaise prise en compte de l'application lors de sa conception et notamment dans la rédaction du cahier des charges et dans sa spécification. Le manque de compréhension respective de la part des deux parties que sont les clients et le concepteur-réalisateur de l'application est un des problèmes qui apparaît le plus souvent et qui amène à des solutions de réalisation qui sont incorrectes du point de vue du client.

Différentes aides à la conception sont aujourd'hui disponibles sous forme de logiciel. Elles sous-entendent une spécification préalablement établie et validée.

Pour pallier les problèmes de la conception d'équipements intelligents, nous donnons dans ce mémoire une méthode afin de modéliser correctement un équipement intelligent avant sa conception-réalisation. Le modèle obtenu diminue les erreurs qui auraient pu s'introduire lors de la conception du fait d'une modélisation incorrecte. C'est également le cahier des charges de l'étape de conception.

Le mémoire est structuré en cinq chapitres.

Le premier chapitre donne une présentation de la problématique générale de la conception que nous appliquons ensuite aux équipements intelligents. On y consacre une attention toute particulière à la formation du cahier des charges destiné à la conception d'un équipement intelligent.

Le second chapitre est consacré à une étude des diverses méthodes utilisées dans la conception d'équipements intelligents. Leurs avantages et leurs inconvénients respectifs y sont précisés.

Au troisième chapitre, le modèle externe est développé comme méthode de spécification fonctionnelle d'un équipement intelligent. On y introduit les différentes étapes nécessaires à la démarche proposée. Un outil développé sous Excel permet la mise en œuvre de la procédure. Le modèle alors obtenu décrit complètement un équipement intelligent d'un point de vue fonctionnel, tout en lui assurant une cohérence vis-à-vis du cahier des charges fourni.

Une fois le modèle fonctionnel obtenu et validé, le quatrième chapitre offre la possibilité d'en modéliser le comportement en passant au modèle SYNCCHARTS. Une méthode est alors donnée afin de passer du modèle externe obtenu à son modèle équivalent SYNCCHARTS. Le syncChart de l'équipement permet alors au concepteur de vérifier et de valider le modèle comportemental de l'équipement intelligent.

Enfin, dans le dernier chapitre, les démarches basées sur le modèle externe et sur le modèle SYNCCHARTS sont appliquées au cas d'un régulateur de vitesse d'un véhicule. Après avoir fourni le cahier des charges de ce dernier, nous obtiendrons un modèle fonctionnel cohérent et validé, puis le modèle comportemental correspondant sera simulé, vérifié et enfin validé. Cet exemple permettra de démontrer l'efficacité de la démarche que nous proposons.

En conclusion, nous positionnerons nos travaux dans le processus global de la conception d'équipements intelligents.

Chapitre I : La Problématique de la Conception

Table des Matières :

I.1- Introduction : **8**

I.2- Cycle de Vie d'un système : **8**

I.3- Le développement : **9**

I.3.1- Le Cahier des Charges : 10

I.3.1.1- Buts d'un cahier des charges : 11

I.3.1.2- Contenu d'un cahier des charges : 11

I.3.1.3- Propriétés d'un cahier des charges : 12

I.3.1.4- Démarche d'expression des besoins : 13

I.3.2- Les cycles de développement : 14

I.3.2.1- Le cycle de développement en cascade (ou Waterfall) : 14

I.3.2.2- Le cycle en V : 15

I.3.2.3- Le cycle en spirale : 16

I.3.3- Activités communes aux cycles de développement : 17

I.3.3.1- Spécifier : 17

I.3.3.2- Concevoir : 18

I.3.3.3- Vérifier, tester, valider : 18

I.3.3.4- Programmer, coder, intégrer : 19

I.4- Application aux équipements intelligents : **19**

I.4.1- Spécificités des équipements intelligents : 20

I.4.1.1- Temps-réel : 20

I.4.1.2- Réactif : 21

I.4.1.3- Domaines d'application des équipements intelligents : 22

I.4.2- Cahier des Charges d'un Equipement Intelligent : 22

I.4.2.1- Environnement du système d'automatisation : 23

I.4.2.2- Besoins du Système d'Automatisation : 24

I.4.2.2.1- Les aides fournies par le système d'automatisation : 24

I.4.2.2.1.1- Les aides fournies aux concepteurs-constructeurs-installateurs : 25

I.4.2.2.1.2- Les aides fournies aux exploitants : 26

I.4.2.2.1.3- Les aides fournies aux démantèleurs : 26

I.4.2.2.2- Format de l'aide : 27

I.4.2.2.2.1- Formats simples : 27
I.4.2.2.2.2- Formats complexes : 28
I.4.2.2.3- Moyens de l'aide : 28
I.4.2.2.3.1- Contraintes environnementales : 29
I.4.2.2.3.2- Attentes externes : 30
I.4.2.2.3.2.1- Aspect Réactif : 30
I.4.2.2.3.2.2- Aspect temps-réel : 31
I.4.2.2.3.2.3- Aspect performances : 31
I.4.2.2.3.3- Méthodologie : 32
I.4.2.2.4- Autres aides possibles : 32
I.4.2.2.4.1- Paramétrage d'une aide : 32
I.4.2.2.4.2- Défaillance d'une aide : 33

I.5- Conclusion : 34

Références : 36

La Problématique de la Conception

I.1- Introduction :

Depuis les années 1980, les systèmes industriels ont considérablement évolué. Les progrès faits, en particulier en électronique, ont permis le développement des traitements numériques qui permettent des calculs plus évolués et plus variés que ceux offerts par les circuits analogiques classiques lesquels, souvent conçus spécialement pour un type d'application donné, apportaient peu de flexibilité. L'informatique industrielle a donc évolué de façon significative durant ces vingt dernières années. Parallèlement, la taille des composants entrant dans la fabrication des systèmes informatiques, notamment les transistors, puis les micro-processeurs, a considérablement évolué et tend vers une miniaturisation de plus en plus importante.

Ces évolutions électronico-informatiques ont permis la prise en compte de systèmes de plus en plus grands. Cependant, la taille de ces systèmes, en termes de données à prendre en compte et à gérer et en termes de phénomènes physiques observés, les rend de plus en plus complexes à aborder d'une part et à maîtriser d'autre part. Ainsi, lorsqu'il s'agit de concevoir (i.e. comprendre – représenter – réaliser) un tel système, plusieurs problèmes surviennent. Sans méthodologie rigoureuse, des incohérences peuvent apparaître et porter de nombreux préjudices au système (lui-même) et à son environnement s'il a été mal conçu et porter aussi atteinte à la bonne marche de l'entreprise ayant conçu le système par les répercussions financières qui s'en suivent. En conséquence, des méthodes de conception de systèmes informatisés ont été développées afin de pouvoir prendre en compte le plus efficacement possible les problèmes posés dans la conception d'un système.

On se propose dans ce chapitre de marquer les différents problèmes que posent la conception d'un système dans sa globalité. On s'intéressera au développement du cahier des charges initial et à la réalisation de celui-ci dans le cadre du cycle de vie du système. Enfin, on s'attachera plus particulièrement au cas du cahier des charges d'équipements intelligents après en avoir explicité leurs spécificités.

I.2- Cycle de Vie d'un système :

Le cycle de vie doit représenter de façon générale la vie d'un produit de sa "naissance" à sa "mort". Ainsi, le cycle de vie d'un produit peut se décomposer en trois grandes phases comme le montre la figure 1.1 ci-après.

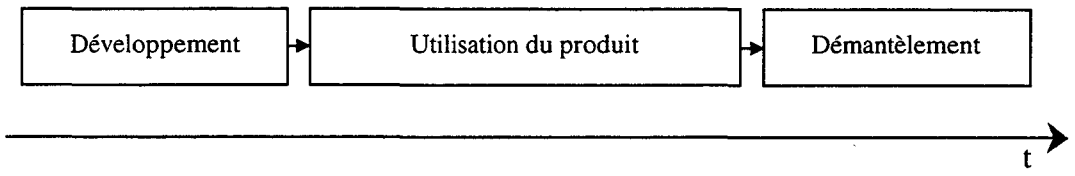


Figure 1.1 : Cycle de vie d'un produit

Ces trois grandes phases sont :

- le développement, qui correspond à la conception, à la réalisation et à la mise en place du produit,
- l'utilisation du produit, qui correspond à l'exploitation et à la maintenance du produit développé dans la phase précédente,
- le démantèlement, qui finit la vie du produit et qui inclut aussi une récupération des divers composants formant le produit.

La première phase est la plus importante car elle influencera tout le reste du cycle de vie du produit. C'est pourquoi nous allons l'étudier plus précisément dans le paragraphe suivant.

I.3- Le développement : [ALA89][CAL90]

Le développement est une des phases les plus importantes dans la vie d'un système. Toute erreur dans cette phase se répercutera inévitablement dans les phases d'utilisation et de démantèlement.

Dans certains cas, le concepteur propose de lui-même un nouveau produit afin de servir au mieux les besoins potentiels de ses clients après une étude de marché adéquate. Dans d'autres cas, le développement se fait à la suite d'une demande directe d'un ou de plusieurs clients. Quelle que soit l'origine du projet, le développement se fera par rapport à des objectifs bien précis en termes de fonctionnalités, de délais et de coûts. En ce sens, en dehors des compétences techniques que le concepteur doit posséder, ce dernier devra résoudre plusieurs problèmes :

- Il doit répondre au mieux aux objectifs à atteindre. Ceci est un point essentiel : le produit final doit recouvrir au plus près les besoins exprimés ou en prenant en compte toutes les données et contraintes (matérielles, financières, normes...) fournies.
- Le concepteur-réalisateur doit respecter les délais et les coûts : un manquement dans les délais de livraison et/ou un dépassement des coûts de développement engendre inévitablement des dépenses supplémentaires voire même une perte de crédibilité sur le marché (et donc des pertes de part de marché).

- Le concepteur-réalisateur doit satisfaire des critères de qualité : non seulement le système final fourni doit bien fonctionner (être fiable), mais il doit aussi être :
  - robuste : résistant aux dégradations liées à son utilisation et à son environnement,
  - convivial : au niveau de la prise en main du produit et de la documentation fournie avec le produit,
  - maintenable : on doit pouvoir facilement corriger les défauts qui apparaîtront au cours de l'utilisation du produit et on doit aussi pouvoir assurer une évolution des caractéristiques du produit.

Il est donc important de disposer d'une méthodologie de développement qui permettra d'assurer la maîtrise de la production, de l'évolution et de l'exploitation du produit qui sera fourni au client. Dans cette méthodologie il faut, notamment, identifier quelles sont les étapes distinctes qui interviennent pour arriver jusqu'à la réalisation finale du produit. Ensuite, il faut définir pour chacune de ces étapes quels sont les résultats à atteindre et quels sont les liens avec l'étape suivante ; les liens entre les différentes étapes donnent ainsi un graphe d'enchaînement des étapes du développement. Chaque étape décrit alors un modèle suivant un point de vue sur le système à concevoir et à réaliser.

Un des points importants que fait ressortir le développement est le fait qu'il s'appuie sur le document exprimant les besoins et les objectifs que s'est fixé le développeur du futur produit ou que le client lui a fourni. C'est sur ce document, couramment appelé cahier des charges, que nous allons nous attarder pour passer ensuite aux divers cycles de développement souvent utilisés lors de cette phase.

I.3.1- Le Cahier des Charges : [GUE93]

Le cahier des charges est la première étape de toute vie d'un système. Sa rédaction est donc délicate et primordiale pour la suite.

Examinons les buts d'un cahier des charges, ce qu'il doit contenir et les propriétés qu'il doit posséder.

I.3.1.1- Buts d'un cahier des charges :

La norme NF-X 50-151 [AFN91] assigne quatre buts au cahier des charges :

- il sert au demandeur à exprimer les besoins des utilisateurs qu'il représente,
- il sert au demandeur à provoquer chez le concepteur-réalisateur la conception et la réalisation du produit le plus efficient,
- il sert à favoriser le dialogue entre les partenaires,
- il sert à faciliter le dépouillement des propositions.

Ce document qu'est le cahier des charges intervient donc dans les rapports qu'entretiennent les deux intervenants : le demandeur qui a besoin d'un service ou d'un matériel, et le concepteur-réalisateur qui se propose de le lui fournir.

Les deux premiers buts sont les plus importants : ils caractérisent le fait que le cahier des charges ait pour rôle principal de fournir un produit final qui satisfasse au mieux les besoins des utilisateurs de ce dernier. Cela suppose aussi que les besoins exprimés dans le cahier des charges l'ont été de façon correcte : ce sont ceux qui sont attendus et tous les éléments nécessaires à la prise de décision de conception sont disponibles au concepteur-réalisateur. La norme impose la définition, pour chacun des besoins exprimés, d'un critère de satisfaction, du niveau exigé selon ce critère et de la flexibilité de cette exigence.

I.3.1.2- Contenu d'un cahier des charges :

Afin de répondre aux quatre buts assignés ci-dessus et pour permettre la conception du produit qui satisfasse au mieux les différents utilisateurs, le cahier des charges doit contenir différents types d'information ayant trait aux différentes phases du cycle de vie du produit. On a vu que l'on pouvait séparer le cycle de vie du produit en trois grandes phases : conception-réalisation, utilisation et démantèlement. Les deux phases utilisées pour la construction du cahier des charges sont les deux dernières puisque le cahier des charges intervient dans la première et a des projections dans les deux suivantes.

Le cahier des charges doit donc contenir des informations sur les phases d'utilisation du produit et de démantèlement et sur les besoins liés à ces deux phases : i.e. l'exploitation, la maintenance, le démantèlement et la récupération.

Le cahier des charges comprend un exposé du problème à résoudre et non pas la description d'une des solutions. La difficulté est donc de fournir suffisamment d'informations utiles sans en donner qui auraient attrait à une expression d'une solution envisageable particulière. Il n'est malheureusement pas possible de dire, dans le cas

général, quelles sont les informations qui doivent ou ne doivent pas apparaître dans le cahier des charges. Cela dépend, entre autres, du champ des compétences et du niveau de chacun des acteurs qui interviennent dans la conception. Cependant, ces informations peuvent être regroupées en trois grandes catégories :

- *les besoins* : ce sont les informations qui correspondent à l'énoncé du problème à résoudre ou de la question à laquelle il faut apporter une réponse,
- *les données* : ce sont les informations complémentaires qui permettent de résoudre le problème correctement,
- *les contraintes* : ce sont les informations qui fixent des parties de la solution ou des moyens d'y parvenir, soit en termes de technologie utilisée (performances, matériel existant, ...), soit en termes plus vagues tels que le coût de revient maximum du produit.

I.3.1.3- Propriétés d'un cahier des charges :

Pour atteindre le but du cahier des charges qui est la conception et la réalisation du produit qui satisfasse au mieux les besoins des différents utilisateurs, le cahier des charges doit posséder certaines propriétés. Celles-ci expriment le fait que le cahier des charges doit contenir tous les besoins et données nécessaires à la conception et que ce document servira de référence pour le concepteur-réalisateur du produit. Pour tendre vers la perfection, les principales propriétés que doivent posséder les spécifications du cahier des charges sont les suivantes [GUE93] :

- *correction* : les besoins exprimés doivent être ceux réellement requis,
- *utilité* : tous les besoins spécifiés doivent être réellement utiles,
- *complétude* : tous les besoins doivent être exprimés,
- *cohérence* : les besoins ne doivent pas être contradictoires les uns avec les autres,
- *non-ambiguïté* : il ne doit pas être possible d'avoir des interprétations différentes des besoins exprimés,
- *accessibilité* : les besoins doivent être exprimés sous une forme accessible et compréhensible à tout nouvel intervenant même non-spécialiste, afin de pouvoir, en particulier, être validés par les utilisateurs,
- *modifiabilité* : il faut pouvoir connaître et maîtriser les conséquences de toute modification,
- *traçabilité* : l'origine de chaque besoin doit être connue,
- *aspect commenté* : chaque besoin doit être accompagné d'un commentaire donnant en particulier son niveau d'importance, d'intangibilité, ...

- *vérifiabilité* : il faut pouvoir vérifier sur le produit final réalisé que les besoins exprimés ont été remplis,
- *réutilisabilité* : il faut que la forme et la structure des spécifications permettent la récupération et la réutilisation d'un maximum de travail d'un projet à un autre.

Les premières de ces propriétés (correction, utilité, complétude, cohérence) sont plus particulièrement relatives au fond des spécifications. Les suivantes (non—ambiguïté, accessibilité, ...) interviennent sous une forme qui doit permettre le dialogue et la compréhension entre les deux intervenants que sont le concepteur d'une part et les différents utilisateurs du produit, d'autre part. La modifiabilité, la traçabilité et l'aspect commenté prennent en compte le coté évolutif des informations et les modifications qui peuvent être apportées au cours des diverses entrevues visant à faire évoluer le cahier des charges vers sa version finale et donc vers le produit le plus en adéquation avec ce dernier.

Certaines de ces propriétés sont complémentaires : si l'origine de tous les besoins et leurs évolutions (traçabilité) est connue, il sera plus facile de déterminer les conséquences d'une modification des besoins (modifiabilité). D'autres propriétés, par contre, s'opposent : la propriété de non-ambiguïté sous-entend des formalismes stricts, ce qui s'oppose souvent à une bonne compréhension (accessibilité) par des non-spécialistes.

Des procédures de vérification existent pour certaines des propriétés du cahier des charges comme, par exemple, la cohérence. Par contre d'autres propriétés sont plus difficilement vérifiables, car faisant appel à des points de vue subjectifs, c'est le cas pour la complétude et l'utilité. L'utilisation de méthodes particulières avec des démarches rigoureuses est alors un moyen de pallier une partie de ces difficultés.

I.3.1.4- Démarche d'expression des besoins :

Dans un projet de développement, une première phase consiste à réunir l'ensemble des informations qui sont connues de tous les utilisateurs du futur produit sous une forme plus ou moins diffuse. Il s'agira alors de formaliser ces informations et à les confronter de nouveau pour en vérifier leur cohérence. Cette phase, appelée expression des besoins, est au sommet de tout cycle de développement. Elle permet aussi une récupération de différentes données et de contraintes. L'expression des besoins conduit donc à une première expression du cahier des charges. La qualité de ces informations est primordiale puisque c'est à partir de celles-ci que le concepteur va se baser pour

construire tout le reste du développement, même s'il y a toujours quelques incertitudes qui devront être résolues lors de la conception et après discussion avec les différents intervenants.

En ce sens, la norme NF-X 50-151 [AFN91] préconise pour les phases de récupération, de validation et d'affinement des besoins, la mise en place d'une organisation semblable à celle mise en œuvre lors d'une action d'analyse de la valeur. Elle demande donc la constitution d'un groupe spécial comprenant, entre autres, au moins un animateur et un décideur. Les membres du groupe doivent être choisis dans l'éventail le plus large possible afin de représenter l'ensemble des acteurs intervenant sur le produit et en particulier les futurs utilisateurs de ce dernier. Par contre, elle ne précise pas réellement de démarche pour guider les travaux du groupe.

I.3.2- Les cycles de développement :

Le cahier des charges est bien souvent à la base des cycles de développement. Ceci se reflète dans les différents cycles qui sont présentés ci-dessous par le fait que la phase de spécification-expression des besoins se trouve au sommet du cycle.

I.3.2.1- Le cycle de développement en cascade (ou Waterfall) :

Le cycle en cascade [ROY70] fait apparaître une succession d'activités allant de divers niveaux de description du problème jusqu'à la réalisation finale.

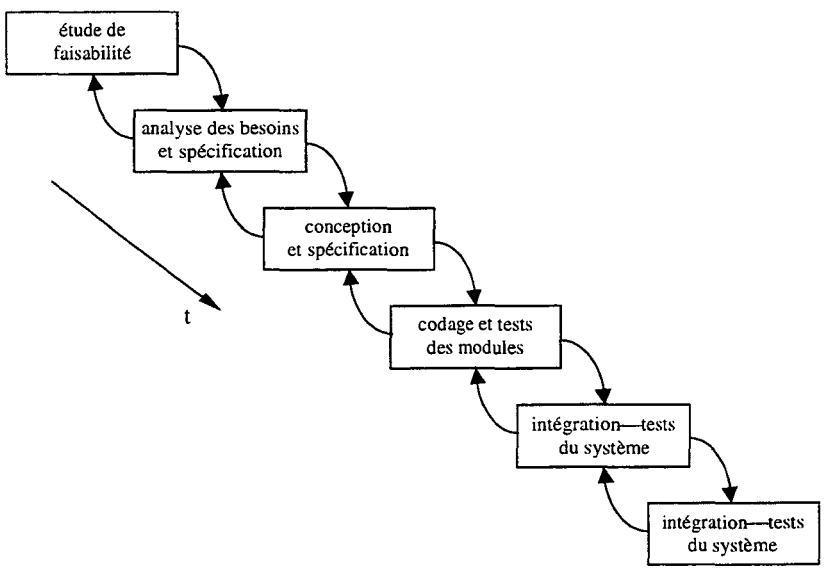


Figure 1.2 : Cycle de Vie en Cascade

Chaque étape du cycle est liée, d'une part, à l'étape suivante pour caractériser l'enchaînement des activités du cycle, et d'autre part, à la précédente pour représenter les corrections éventuelles qu'il faut apporter en cas de non-conformité. En effet, toute non—conformité nécessite la reprise de l'activité en cours voire de l'étape précédente.

Ce cycle montre qu'un développement ne se fait pas purement suivant une démarche descendante : les erreurs sont corrigées par des retours-arrières dès leurs observations. Cela suppose que l'on puisse, à chaque étape du cycle, avoir un résultat observable par rapport aux objectifs fixés dans l'activité.

I.3.2.2- Le cycle en V :

Ce type de modèle de développement a été proposé par le Mouvement Français pour la Qualité et a été repris dans [AFN96]. Par rapport au précédent, ce modèle considère en plus les phases de vérification pour chaque stade de la réalisation du produit.

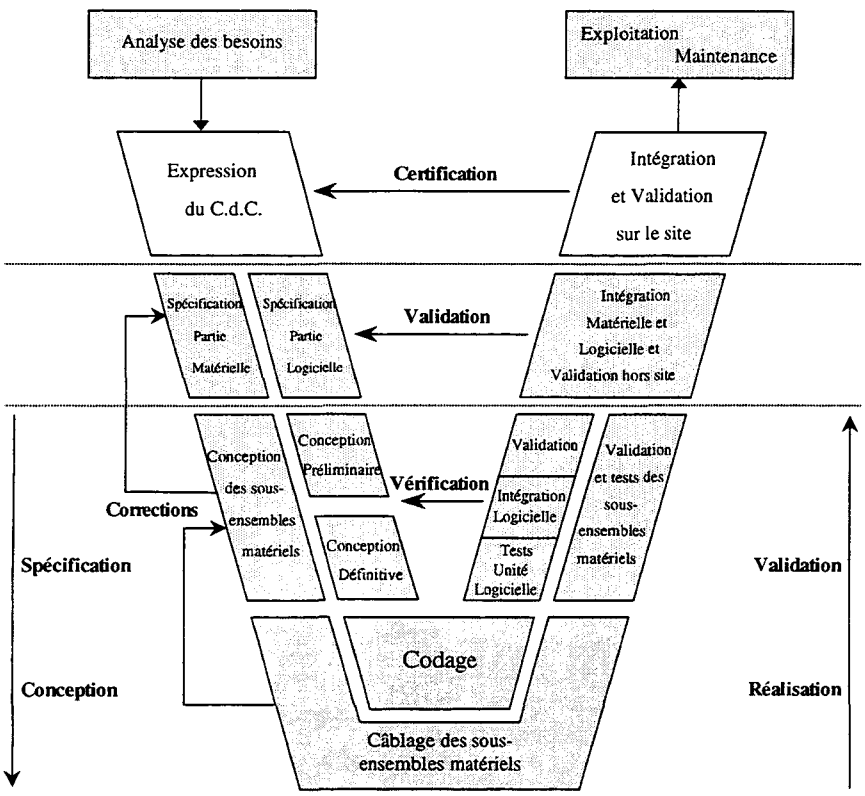


Figure 1.3 : Cycle de Vie en V

Ce cycle montre bien deux phases de la démarche : la phase de spécification - conception qui est globalement descendante et la phase de réalisation - tests qui est globalement ascendante. En outre, il montre aussi qu'il existe

deux parties possibles qu'on peut prendre en compte dans une démarche de conception : une partie logicielle et une partie plus matérielle qu'il ne faut pas non plus négliger.

Une extension du cycle en V à un cycle en X a aussi été proposée dans [HOD91] et [COU96]. Le cycle de développement en X ajoute au cycle en V une partie de réutilisation par l'acquisition et l'archivage dans des bibliothèques des composants, services et domaines acquis lors de la phase de réalisation du projet. Ces différentes bibliothèques servent alors dans d'autres projets ayant des similitudes avec celui qui a permis l'élaboration des éléments de la bibliothèque, réduisant ainsi les coûts d'études futurs.

I.3.2.3- Le cycle en spirale :

Les modèles précédents considèrent une progression linéaire du cycle de développement d'un produit. Ces derniers ne font pas apparaître le caractère itératif de la démarche du développement qui permet une amélioration successive du produit jusqu'à ce qu'il satisfasse au mieux ce qui est demandé dans le cahier des charges. En ce sens, les différents modèles vu précédemment se font en "boucle". Le modèle de cycle de développement en spirale [BOE88] nous montre le caractère itératif du développement d'un logiciel informatique :

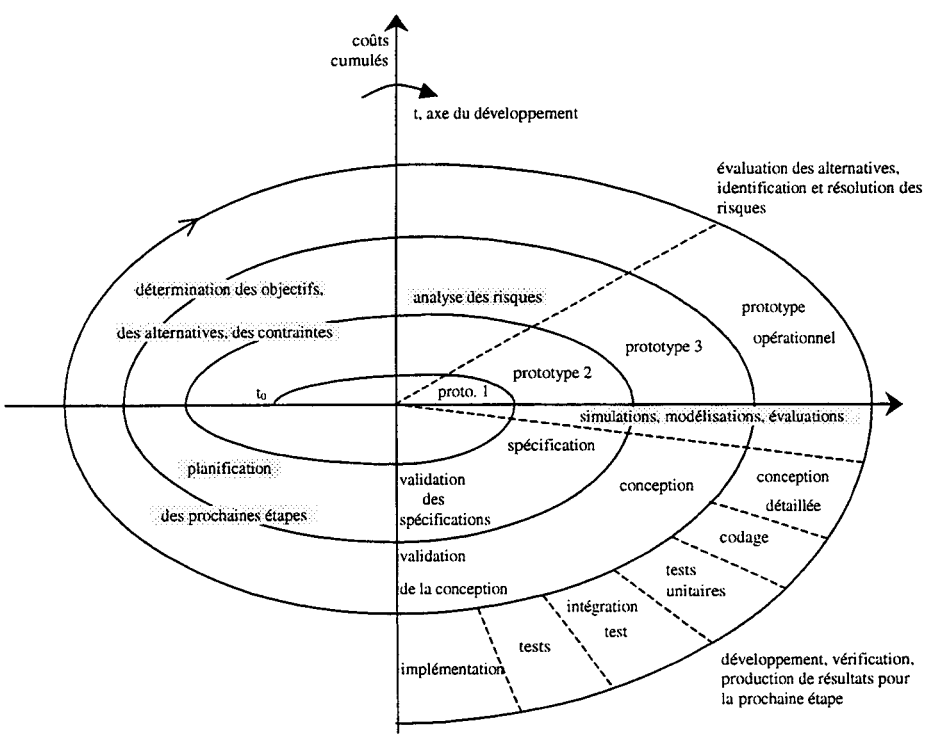


Figure 1.4 : Cycle de Vie en Spirale

On peut voir dans ce cycle quatre phases, une par cadran :

- la planification des phases suivantes,
- la détermination des objectifs, des alternatives et des contraintes,
- l'analyse des risques et une évaluation des solutions,
- le développement et la vérification du produit.

La prise en compte matérielle "hard" n'est malheureusement explicitée qu'à la fin du cycle, contrairement au cycle en V où l'aspect matériel est pris en compte dès la spécification du cahier des charges.

Les cycles précédents ne sont pas exhaustifs, ils reflètent cependant toutes les étapes d'un cycle de développement et mettent en évidence les deux aspects suivants :

- aspect séquentiel : la conception se fait par enchaînement d'étapes successives, certaines activités se faisant en parallèle (par exemple les cotés matériels et logiciels du produit),
- aspect cyclique : on peut remettre en question toute étape précédente à une étape donnée du cycle de développement.

I.3.3- Activités communes aux cycles de développement :

On constate que plusieurs activités sont communes à tous ces modèles, voyons-en les définitions générales.

I.3.3.1- Spécifier :

On désigne par-là l'activité de définition des fonctions que doit accomplir le système, abstraction faite de la manière dont elles seront réalisées ; on distingue habituellement :

- la spécification (ou analyse) des besoins, i.e. le cahier des charges fourni par le client ou celui que s'est fixé le développeur, qui donne une vue externe du produit et qui a une valeur contractuelle avec le client, elle définit les services rendus aux différents utilisateurs et les contraintes sous lesquelles le système devra fonctionner,
- la spécification du produit, qui donne une vue interne (logicielle et matérielle) et dont le résultat (fonctions, liens entre fonctions, contrôle des fonctions,...) sert de cahier des charges pour l'étape de conception suivante.

I.3.3.2- Concevoir :

Il s'agit, ici, de définir la manière dont le système spécifié sera réalisé, en le décomposant en sous–systèmes et en les structurant ; plusieurs approches sont utilisables pour supporter cette activité :

- conception fonctionnelle, par raffinements successifs,
- conception guidée par les données ; la conception du système est directement dérivée de l'analyse des données en entrées et en sorties.
- conception orientée objet ; le système est vu comme un ensemble d'objets auxquels on peut associer des opérations.

La première approche est associée à une approche descendante. Pour les deux autres, démarches descendantes et ascendantes coexistent.

I.3.3.3- Vérifier, tester, valider :

Vu la complexité croissante des systèmes, il est nécessaire de se doter de moyens de vérifications du modèle afin de s'assurer que le système réalisé correspond à ce que le client désire (conformité au cahier des charges fourni) et plus exactement à sa spécification. Il est donc indispensable de mettre en place des méthodes de vérification et de validation des différentes phases de spécification dans la conception. Par les deux termes vérification et validation, on entend [LAM97] :

- *vérification* : c'est la preuve de la cohérence intrinsèque d'un modèle indépendamment de son contexte d'utilisation et de son environnement,
- *validation* : c'est la preuve de l'adéquation du modèle à ce qu'il est censé représenter.

La vérification est une "étape" qui doit se faire à la fin de chaque activité de conception (de la spécification du cahier des charges jusqu'à l'intégration du système conçu). Pour démontrer la cohérence de chaque étape, on vérifie que le système spécifié au niveau de l'étape en cours possède les propriétés attendues, et, pour le faire, plusieurs moyens sont disponibles, entre autres :

- les tests : il s'agit de la mise en œuvre du système en utilisant des données similaires aux données réelles et en observant les résultats ; les méthodes de

tests peuvent être ascendantes ou descendantes. La génération des tests peut être formelle ou empirique, automatique ou manuelle.
- les vérifications formelles qui, par l'analyse de chaque résultat, à chaque étape, permettent de prouver formellement certaines propriétés du système.

I.3.3.4- Programmer, coder, intégrer :

Ces activités désignent les étapes de réalisation effective du produit ; les résultats de l'activité de conception forment le cahier des charges de cette étape : les méthodes mises en œuvre sont soit ascendantes soit descendantes.

Toutes les activités décrites ci-dessus sont communes à tous les cycles de développement. Elles sont donc nécessaires et devront être présentes dans toute méthode de conception afin de pouvoir finalement valider la méthode utilisée.

Les critères donnés pour le cahier des charges sont valables pour tout système informatisé. Nous ne nous intéressons, cependant, qu'à un type particulier de systèmes : les Systèmes Automatisés qui sont une classe plus "industrielle" des systèmes de production en comparaison à des systèmes plus tournés vers la gestion de données.

Nous allons nous attarder plus attentivement sur les équipements dits "intelligents" qui sont plus particulièrement utilisés dans les Systèmes Automatisés de Production sous la forme de capteurs et actionneurs intelligents [GEH94][STA94]. Les critères du cahier des charges sur lesquels il faut plus particulièrement porter notre attention lors de leur conception vont être explicités dans le paragraphe suivant.

I.4- Application aux équipements intelligents :

Dans le cadre des Systèmes Automatisés à Intelligence Distribuée (S.A.I.D.), les équipements intelligents ont pris leur place au niveau des équipements de terrain comme des substituts adéquats aux nouveaux moyens de distribuer localement les traitements consécutifs à ce qu'offrent, entre autres, les capteurs et les actionneurs. Cette distribution permet d'alléger les centres nerveux des applications de grande taille qui font encore bien souvent office de centralisateurs d'informations, tout en s'occupant de la gestion des objectifs majeurs du système dans son ensemble (i.e. en termes de production, objectifs à atteindre...).

I.4.1- Spécificités des équipements intelligents :

Les équipements intelligents sont des systèmes automatisés [BAY95] qui font eux-mêmes partie de l'ensemble des systèmes temps-réel réactifs. Après une définition des qualificatifs relatifs au système étudié (i.e. temps-réel et réactif), on s'attardera sur les domaines d'application plus particulier des équipements intelligents.

I.4.1.1- Temps-réel :

Nous n'aborderons pas ici les problèmes sur "le temps" dans sa généralité, le problème étant loin d'être réglé. Nous prendrons le temps sous la forme d'une succession d'événements ordonnés que nous plaçons sur une droite continue et infinie.

Cependant que faut-il pour qu'un système soit qualifié de "temps-réel" ? Plusieurs points de vue peuvent être proposés. On peut qualifier de système temps réel tout processus dont les données et les résultats sont relatifs à des événements "en cours". Mais, de manière générale, un système fonctionne en "temps-réel" lorsqu'il est question de contraintes de temps et que ces dernières doivent être respectées. Cette définition prend en compte l'adage suivant couramment énoncé dans les systèmes temps-réel :

"*Tout résultat juste mais hors délai est un résultat faux*"

Le fait qu'un système soit temps-réel fait apparaître que l'on a à traiter et à respecter des contraintes de temps différentes. En effet, les contraintes de temps auxquelles on a à faire sont relatives à la dynamique du processus piloté (géré, surveillé...) par le système temps-réel. Ainsi plus la dynamique du système sera rapide plus les contraintes "temps-réel" seront strictes et difficiles à respecter matériellement. On peut donner plusieurs ordres de grandeur de contraintes de temps par rapport au processus considéré :

- de l'ordre de la milliseconde pour les systèmes radar,
- de l'ordre de la seconde pour les systèmes de visualisation humaine,
- jusqu'à plusieurs heures pour certains réacteurs chimiques.

D'une manière générale, tout système automatisé, et a fortiori tout équipement intelligent, se doit de respecter des contraintes de temps plus ou moins sévères suivant le domaine de l'application considéré. Ces contraintes peuvent être imposées dans le cahier des charges (par exemple en terme d'échéance maximale) ou non. Cependant, dans tous les cas, les données temporelles apparaîtront inévitablement au cours du cycle

de développement ne serait-ce que pour ordonner les différentes actions du système automatisé.

I.4.1.2- Réactif :

De par le fait des différents intervenants sur le système d'automatisation, i.e. l'environnement physique et les différents opérateurs, le système acquiert le caractère de *réactif*. Le terme de système réactif a été introduit par [HAR85]. Un système est dit réactif lorsque son comportement dépend directement des informations qu'il échange avec son environnement. Ces informations arrivent au système d'automatisation sous forme de stimuli. Un stimulus est l'occurrence d'un événement en provenance de l'environnement de l'application vers le système. Cette occurrence est furtive, i.e. si elle n'est pas traitée ou mémorisée à l'instant de son occurrence, elle est perdue [DEL95]. Le système doit alors réagir de manière instantanée afin de pouvoir prendre en compte l'information contenue dans le stimulus. La rapidité de la réaction dans les systèmes réactifs est donc primordiale afin d'en assurer un fonctionnement conforme et sûr. De par ce fait, les systèmes réactifs sont souvent une classe particulière des systèmes temps-réel en opposition, mais non en contradiction, avec les systèmes dits "transformationnels" qui traitent des flux d'entrée, les flux de sortie étant directement liés aux flux d'entrée.

On peut caractériser le comportement réactif de tout équipement intelligent suivant que l'environnement est de nature continue, à événements discrets ou hybrides. Le plus souvent, il s'agit de systèmes automatisés hybrides. En effet, il est assez rare de trouver un système automatisé qui n'intègre qu'un seul des deux types de comportement précédents. Dans le cadre d'une boucle de régulation, par exemple, la commande se fait de manière continue alors que les changements de mode de fonctionnement se font de façon discrète.

I.4.1.3- Domaines d'application des équipements intelligents :

On peut différencier quatre grands domaines d'application dans lesquels les équipements intelligents sont plus particulièrement utilisés :

- l'industrie du manufacturier, qui possède généralement des contraintes temps-réel fortes et des contraintes environnementales variables suivant le type d'application manufacturière,
- l'industrie du process qui comporte des contraintes de temps-réel plutôt faibles mais qui ont des contraintes environnementales fortes (processus chimiques, thermiques, ...),
- les applications embarquées, qui ont aussi des contraintes temporelles variables mais des contraintes dues à l'environnement très fortes,
- le tertiaire, qui est le plus souvent le moins contraignant tant au niveau temps-réel qu'environnemental.

On retrouve dans les deux premiers domaines d'application tout ce qui concerne plus particulièrement les capteurs et actionneurs intelligents. Si les applications embarquées sont similaires aux deux précédentes, elles subissent généralement des contraintes d'encombrement très fortes (espace disponible, poids, ...). Le tertiaire correspond plus ou moins aux "autres domaines d'application". En effet, si les trois premiers sont très typés, le tertiaire est très varié. On peut y retrouver par exemple tout ce qui concerne la domotique ou ce qui concerne les problèmes de détection utilitaire (incendie, intrusion, ...).

Dans un cadre général, la réactivité de ces systèmes se trouve à la fois dans leur aspect temps-réel et dans leur relation avec l'environnement.

I.4.2- Cahier des Charges d'un Equipement Intelligent :

Le fait de distribuer tout ou partie des données, des traitements et le contrôle de ceux-ci rend l'expression des besoins dans le cahier des charges un peu plus délicate. En effet, il faut, dans un souci de complétude, prendre en compte tous les besoins exprimés par les différentes parties mises en jeu. L'ensemble des informations à exprimer peuvent se déduire de la structure d'un équipement intelligent [GEH94][STA94] :

- le système d'automatisation,
- l'environnement physique,
- les opérateurs humains et autres systèmes automatisés en communication avec l'équipement intelligent.

Le système d'automatisation rend des services aux deux autres parties en terme d'aide aux opérateurs (et autres systèmes automatisés) et en terme de mesure et de commande sur l'environnement physique. Tout ceci représente l'environnement de l'équipement qu'il nous faut d'abord bien cerner afin de pouvoir spécifier correctement le système d'automatisation.

I.4.2.1- Environnement du système d'automatisation :

L'environnement physique et les opérateurs forment le point de vue extérieur au système d'automatisation : ils représentent les utilisateurs du futur système d'automatisation. Dans un premier temps et afin de s'assurer de la complétude du cahier des charges, il nous faut regarder tous les utilisateurs du système d'automatisation et donc de l'équipement intelligent. Il sera alors plus aisé de pouvoir prouver la correction, l'utilité et la cohérence du cahier des charges fourni.

On peut dénombrer deux types d'utilisateurs : les consommateurs qui ne font qu'utiliser l'information, au sens large, produite par le système d'automatisation (par exemple, consommer une information élaborée ou non), et les utilisateurs actifs qui interviennent directement sur le comportement de l'équipement, i.e. sur ses conditions de fonctionnement et sur ses entrées. L'utilisateur passif attendra des performances particulières en terme de qualité (justesse et précision de l'information par exemple) et en terme temporel (périodiquement et/ou dans une fenêtre temporelle précise) sur le résultat qu'il attend alors que l'utilisateur actif imposera des contraintes sur l'équipement pouvant engendrer des conflits avec d'autres acteurs qu'il faudra alors pouvoir résoudre [ALA89].

On désignera donc simplement par **consommateur** un utilisateur qui consomme au moins une information et par **acteur** un utilisateur actif de l'équipement.

Un utilisateur peut, bien entendu, être à la fois un consommateur et un acteur d'un même équipement sans que cela n'engendre des incohérences dans le modèle.

Le cahier des charges ne doit pas seulement prendre en compte des problèmes fonctionnels, il doit aussi contenir des informations du point de vue matériel. Ainsi, il conviendra de bien différencier tous les utilisateurs "matériels" possibles et de préciser les moyens de communication sur l'équipement. On pourra ainsi éviter deux cas :

- Si deux utilisateurs physiquement différents (par exemple un opérateur de conduite et un opérateur de maintenance) ont accès à l'équipement via

exactement le même moyen de communication (une console, par exemple), il est nécessaire de les différencier d'une manière ou d'une autre sur ce moyen de communication, sinon ils seront vus de la même façon (i.e. un seul et unique utilisateur).

- Si un même utilisateur physiquement peut avoir accès à l'équipement par deux moyens de communication différents (par exemple par le réseau et par une console locale), cet utilisateur devra être vu comme deux utilisateurs différents. Cela permettra, dans la suite de la conception, d'une part de pouvoir gérer les problèmes de conflits d'accès aux données de l'équipement et d'autre part, de pouvoir prendre en compte le "service" de communication de l'équipement qui devra pouvoir gérer tous les types de communication différents possibles.

En regardant les problèmes matériels au niveau du cahier des charges, on évite des oublis et des incohérences dans l'étape suivante de spécification.

I.4.2.2- Besoins du Système d'Automatisation :

Les diverses activités du système d'automatisation fournissent des aides (au sens large) aux différents utilisateurs de l'équipement intelligent. Ces aides se matérialisent alors sous forme d'informations passives (une image de l'état du système par exemple) ou d'actions (la commande directe d'un matériel d'actionnement sur l'environnement physique).

Le cahier des charges se doit d'énumérer de la façon la plus exhaustive possible l'ensemble des aides requises que devra fournir l'équipement intelligent à l'ensemble de ses utilisateurs. Il faut ensuite vérifier l'utilité de ces aides et leur correction vis-à-vis de chaque utilisateur. Ceci ne peut se faire qu'à partir d'une concertation entre le concepteur-réalisateur, l'ensemble des utilisateurs et d'experts de l'application.

Voyons quelles sont les différentes aides possibles que peut fournir le système d'automatisation d'un équipement intelligent à ses interlocuteurs.

I.4.2.2.1- Les aides fournies par le système d'automatisation :

D'après ce que nous avons vu dans les cycles de développement lors du chapitre précédent, il semble plus judicieux de voir les aides fournies suivant les différentes phases d'activités que l'équipement intelligent offre tout au long de son cycle de vie et

qui doivent être prises en compte lors de sa conception et par rapport aux différents intervenants durant ces phases.

On peut répertorier trois activités principales dans le cycle de vie d'un équipement intelligent :

- l'activité de **Mise en Place**,
- l'activité d'**Exploitation**,
- l'activité de **Démantèlement**.

Les différents utilisateurs associés aux trois phases principales de développement d'un équipement intelligent sont donc respectivement :

- les **concepteurs-constructeurs-installateurs**,
- les **exploitants**,
- les **démantèleurs**.

I.4.2.2.1.1- Les aides fournies aux concepteurs-constructeurs-installateurs :

Ces intervenants ont pour but de fournir un équipement en état de fonctionner. On peut faire apparaître plusieurs phases successives d'**assemblage**, de **configuration**, de **mise en service** et d'**essais**. Pour les trois derniers services on peut définir les aides qui sont apportées :

- *Aides à la configuration, au paramétrage* : des divers constituants de l'équipement,
- *Aides à la mise en service* : par une mise en service progressive et structurée,
- *Aides à la validation* : grâce à des séquences de tests automatisés,
- *Aides au suivi* : par la mémorisation de certaines données facilitant le retour d'expérience.

Ces aides sont particulièrement utiles aux concepteurs-constructeurs-installateurs. En effet, ils permettent de limiter au maximum les erreurs inhérentes à l'activité de Mise en Place et sont donc susceptibles d'abaisser le coût du service après-vente en cas de problème.

I.4.2.2.1.2- Les aides fournies aux exploitants :

L'exploitation est la phase la plus importante au point de vue de la valeur ajoutée de l'équipement intelligent. Dans ce cadre, on doit recenser plusieurs utilisateurs possibles. Ces utilisateurs peuvent être répertoriés en trois groupes principaux :

- les opérateurs assurant le pilotage de l'application,
- les opérateurs assurant la maintenance de l'application,
- les opérateurs assurant la gestion de l'application.

On retrouve ici plus explicitement des aides communes destinées à la mesure de données environnementales, à l'élaboration de données non directement mesurables et à la surveillance. D'autres aides sont plus spécifiques à certains opérateurs comme les aides au pilotage, à la planification et à la prise de décision.

I.4.2.2.1.3- Les aides fournies aux démantèleurs :

Ces intervenants ont pour but d'enlever l'équipement dans un maximum de sécurité et dans un minimum de temps. On peut faire apparaître plusieurs phases successives de **mise hors service** et de **désassemblage.** De ce fait, ces aides ressemblent plus à celles proposées pour les installateurs mais en sens inverse. Les aides ici fournies sont plus d'ordre méthodologique :

- *Aide à la mise hors service* : par une mise hors service progressive et structurée,
- *Aide au désassemblage.*

Il faut cependant ajouter qu'il n'existe qu'assez peu de cas où ces aides au démantèlement sont explicitement prises en compte.

L'ensemble des aides fournies aux utilisateurs de l'équipement ne suffit pas en lui-même pour assurer la complétude du cahier des charges. Il faut pour chaque aide demandée fournir un ensemble de données offrant des informations plus précises sur l'aide aux utilisateurs concernés. Ces informations se rapportent plus particulièrement aux données relatives à la nature de l'aide et à son fonctionnement interne.

Pour chaque aide aux utilisateurs exprimée, il faut préciser plusieurs points essentiels à sa définition :

- sous quel format l'aide doit-elle être fournie aux consommateurs ?

- sous quelles contraintes et comment l'aide doit-elle fournie aux utilisateurs ?

L'intérêt d'une telle modularité dans le cahier des charges sous forme d'aide fournie aux utilisateurs permet une bonne traçabilité tout en permettant une modifiabilité plus aisée au cas où l'expression du besoin ne s'avèrerait pas tout à fait en adéquation avec ceux du client.

I.4.2.2.2- Format de l'aide :

Chaque aide demandée par au moins un utilisateur peut ensuite être assimilée à une sortie fournie soit à l'utilisateur en question soit à l'environnement physique en lui-même. Chacune des "sorties" peut prendre plusieurs formats. Il faut alors en déterminer un suivant ce que l'utilisateur voudra en faire. Il peut s'agir d'une sortie "simple" dans le sens où son format l'est aussi, ou il peut s'agir d'une sortie plus élaborée ou complexe.

I.4.2.2.2.1- Formats simples :

On peut distinguer deux grands types de format simple :

- le format analogique : ce format est surtout utilisé pour des sorties ayant directement à faire avec l'environnement physique. Ce type de signal est le plus souvent un signal électrique dont au moins une des caractéristiques sera prise en considération par le consommateur. Cela peut par exemple être la consigne d'un régulateur PID. Une version très simple de format analogique sont les signaux de type "Tout-Ou-Rien" encore très souvent utilisés dans les domaines du manufacturier et du process.
- le format "numérisé" : ce format est directement numériquement évaluable par un consommateur. Il faut ensuite déterminer le type précis du format numérique. Les formats les plus utilisés sont du type de ceux que fournissent les moyens de programmation classique genre C. On retrouve les formats suivants : booléen, chaîne de caractères, entier, flottant, double flottant, réel, word... A ces formats, on peut ajouter le flou qui est un cas particulier qu'une donnée peut prendre.

I.4.2.2.2.2- Formats complexes :

Il existe ici un très grand nombre de formats complexes. On peut en citer quelques-uns uns qui sont les plus souvent utilisés dans les équipements intelligents :

- les tableaux : ils contiennent une suite de données ordonnées d'un seul type. A ce titre, les tableaux sont couramment utilisés pour les historiques,
- les "objets" : ils sont plus structurés que les tableaux, ce qui leur permet de contenir plusieurs champs d'information (Nom.Prénom, par exemple). Les formats d'objet sont assez variés en fonction des champs qu'ils contiennent,
- les images : on entend par image une image vidéo. Les consommateurs de tel type de donnée s'intéressent au traitement d'image qui est un moyen de surveillance assez souvent utilisé dans ce domaine,
- ...

Les formats complexes sont assez variés. Cependant pour les deux premiers formats, tableau et objet, on constate qu'ils sont eux-mêmes composés de données de format simple. Il faut donc définir le format simple des données de chaque tableau et de tous les champs d'un objet. Les images sont quant à elles d'un format particulier qu'il faut bien définir.

Dans tous les cas, il faudra définir pour chaque sortie proposée par l'aide :

- un identificateur propre afin d'éviter toute ambiguïté,
- le format de la sortie pour des raisons d'interopérabilité et d'interfonctionnement vis à vis des consommateurs extérieurs.

Le choix du format sous lequel sera fourni l'aide est important. En effet, c'est une contrainte qui permettra au concepteur-réalisateur de choisir comment il va pouvoir proposer matériellement et opérationnellement l'aide, ce qui constitue une partie du paragraphe suivant.

I.4.2.2.3- Moyens de l'aide :

Ce que nous appelons "les moyens de l'aide" comporte plus les aspects matériels et opérationnels qui doivent aussi être présents dans le cahier des charges. Cela concerne donc plus un point de vue interne du fonctionnement de l'aide fournie (i.e. la sortie fournie par l'aide à ses consommateurs), les attentes externes de la part des utilisateurs

de l'aide et, enfin, les contraintes que l'environnement physique imposent sur le fonctionnement interne et externe de l'aide.

I.4.2.2.3.1- Contraintes environnementales :

Les contraintes environnementales sont des contraintes opérationnelles pures. En effet, les contraintes imposées par l'environnement physique ne peuvent généralement pas être changées de manière significative.

Ces contraintes concernent en particulier des phénomènes physiques. Les *conditions de fonctionnement* regroupent ici toutes les conditions particulières sous lesquelles l'équipement rend sa mission et qui risquent d'en altérer le bon fonctionnement si elles ne sont pas prises en compte dès la conception. On peut par exemple retrouver comme conditions de fonctionnement restrictives des atmosphères corrosives, des températures anormalement élevées... On peut aussi ajouter la configuration de l'encombrement maximal dont l'équipement disposera. Ce genre de conditions de fonctionnement peut restreindre tant le type de matériel utilisable que la méthodologie qui sera choisie afin de satisfaire les conditions de fonctionnement optimales.

Si les conditions de fonctionnement contraignent le matériel, le choix du matériel utilisable peut aussi être limité par le client :

- *technologie* : le client peut exiger qu'une certaine technologie soit utilisée pour des commodités d'intégration (dans un S.A.I.D. par exemple),
- *marque particulière* : de même on peut se voir restreindre les fournisseurs de composants utilisables afin de former l'équipement intelligent. Ces contraintes sont bien souvent liées à la maintenance future de l'équipement pour des raisons de disponibilité de pièces détachées et pour diminuer les coûts de formation du personnel à de nouveaux types de matériel.

A ce sujet, dans le cas où l'équipement serait relié à un réseau de terrain, le choix du réseau de terrain qui sera utilisé est crucial [THO93]. En effet, le réseau de terrain va imposer certaines choses sur l'équipement :

- Le service de communication de l'équipement subira des contraintes quant au protocole d'échange d'informations sur le réseau, tant au niveau technologique (analogique Tout Ou Rien pour ASI, par exemple) qu'au niveau performance.
- Le réseau va imposer que certaines aides soient impérativement fournies par l'équipement. Les companions standard de WorldFIP sont un exemple de ce qu'un réseau de terrain peut imposer aux équipements qui lui sont raccordés.

- Le plus généralement, certaines marques de matériels sont plus affiliées à un type de réseau particulier du fait de leurs collaborations passées ou par le fait que ces marques ont déjà prévu la passerelle vers le réseau choisi.

Enfin, on peut encore avoir certaines contraintes qu'on peut qualifier d' "environnementales" bien qu'elles n'en soient pas vraiment. Parmi celles-ci, nous mettrons le coût de l'équipement intelligent. Ce coût peut être restreint par le client. En outre, plus le coût pour l'équipement intelligent sera faible, plus les chances de prise de marché seront grandes. Le prix est donc aussi un critère non négligeable.

I.4.2.2.3.2- Attentes externes :

Les aspects externes attendus vont influencer sur les moyens fonctionnels matériels et opérationnels de résolution pratique de l'aide. Ils concernent les points de vue réactif, temps-réel et performances de l'équipement.

I.4.2.2.3.2.1- Aspect Réactif :

Le point de vue réactif fait plus particulièrement appel aux problèmes d'activation de l'aide, i.e. dans quelles conditions l'aide est-elle fournie à ces consommateurs et quels sont les acteurs qui interviennent dans son fonctionnement ? Cela peut se résumer aux différents points suivants :

- l'aide est-elle périodique ou apériodique ? Si elle est périodique, quelle doit être sa période d'activation ?
- sous quelles conditions matérielles informationnelles et environnementales l'aide peut-elle fournie ?
- quels acteurs demandent la réalisation de l'aide et par quels moyens ?

Tout ceci peut aussi conduire à la formation de modes de fonctionnement particuliers. Un mode de fonctionnement peut être qualifié comme une configuration particulière de l'équipement. Chaque configuration offre alors un certain nombre d'aides aux utilisateurs potentiels.

Des modes de fonctionnement peuvent ainsi être imposés par le client. Pour être cohérent, il faudra qu'il en définisse au moins deux. Le client devra alors définir pour chaque mode :

- quelles sont les aides fournies dans chaque mode de fonctionnement ?
- comment va-t-on d'un mode à un autre :
  - sous quelles conditions matérielles, informationnelles et environnementales change-t-on de mode de fonctionnement ?
  - quels acteurs peuvent demander un changement de mode et par quel moyen ?

Le point de vue réactif comporte donc le problème de la dynamique de l'aide et de son devenir.

I.4.2.2.3.2.2- Aspect temps-réel :

Au niveau temps-réel, on regarde les performances temporelles que devra respecter l'aide fournie aux utilisateurs :

- quelle est la limite de réaction au plus tôt après une demande ?
- quelle est la limite de réaction au plus tard après une demande ?

Ceci définit ainsi une fenêtre temporelle dans laquelle la sortie doit avoir été fournie. Il faut aussi déterminer durant combien de temps l'aide doit être fournie : fugitivement, pendant un laps de temps précis ou de manière durable par rapport à la dynamique de l'équipement.

I.4.2.2.3.2.3- Aspect performances :

Le point de vue temporel est un aspect des performances que doivent respecter les aides fournies par un équipement. Ce ne sont bien sûr pas les seules. Chaque aide doit posséder un certain nombre de performances quantitatives quant à la qualité de la prestation fournie en dehors des conditions de fonctionnement particulières comme nous l'avons précisé plus haut au paragraphe II.3.2.3.1. Pour un capteur cela concerne plus particulièrement des qualités au point de vue exactitude, de précision et de sensibilité des mesures faites [ASC87]. On peut retrouver des notions semblables pour un actionneur. Des performances en terme de sûreté de fonctionnement peuvent également être demandées, elles sont exprimées, par exemple, par un taux de disponibilité.

I.4.2.2.3.3- Méthodologie :

Ce dernier aspect entre en concomitance avec le point de vue interne de fonctionnement de l'aide fournie qui s'intéresse plus aux moyens de rendre le service suivant les exigences des utilisateurs et des contraintes qui ont été énoncées précédemment : i.e. les conditions de fonctionnement, les contraintes temporelles, les performances, ...

La méthode de réalisation de l'aide peut très bien être imposée par le client (régulation par un algorithme PID, par exemple), il faudra alors déterminer si cette méthode permet de répondre à toutes les contraintes. Sinon, il faudra préciser la méthode utilisée qui respectera toutes les contraintes précédentes.

Dans tous les cas, chaque aide devra être augmentée des informations suivantes qui permettent de déterminer quels sont les besoins nécessaires à cette méthode :

- au niveau des données consommées,
- au niveau des ressources nécessaires : matérielles et informationnelles (paramètres par exemple).

Les données consommées peuvent alors donner lieu à d'autres aides qui devront à leur tour être caractérisées ou ce seront des données qui devront être fournies par un acteur particulier.

Dans le cas où la méthode interne de réalisation de l'aide ne serait pas imposée, tant les données de l'environnement extractibles que les ressources disponibles peuvent aider à choisir le type de méthode qui sera utilisé.

I.4.2.2.4- Autres aides possibles :

Toutes les aides qui ont été mises en avant représentent le plus souvent le cadre normal de fonctionnement de l'équipement. On peut aussi, pour chaque aide, regarder deux cas significatifs qui sont susceptibles de se produire : le cas où on reconfigure l'aide et le cas où un des composants constituant l'aide devient défaillant.

I.4.2.2.4.1- Paramétrage d'une aide :

Des données semi-statiques décrivent le paramétrage en vigueur au moment du fonctionnement courant. Elles restent inchangées sur une fenêtre temporelle correspondant à des conditions d'exploitation constantes en opposition avec les données

statiques qui correspondent à des références de l'aide et avec les données dynamiques qui correspondent à l'évolution temporelle des données d'entrée et de sortie de l'aide.

Toutes les aides ne sont pas paramétrables, ce type de données pouvant être figées et donc confondues dans les données statiques de l'aide. Cependant, dans le cas où une aide pourrait être reparamétrée, il faut connaître comment elles sont configurées. Une solution est de prévoir une autre aide permettant à un acteur de paramétrer l'aide en question suivant un protocole qu'il faut établir.

I.4.2.2.4.2- Défaillance d'une aide :

De même, pour chaque aide, il faut pouvoir prévoir le cas d'une défaillance de la part de l'aide. Il peut s'agir d'une défaillance au niveau d'une entrée, d'une ressource ou d'un composant de l'aide. Si une telle défaillance peut survenir, il faut pouvoir la prendre en compte d'une part et pouvoir la traiter d'autre part.

Il faut donc prévoir une "autre" aide de surveillance permettant de détecter la défaillance en question. De plus, il faut traiter le cas lorsque la défaillance survient. On a alors plusieurs choix possibles :

- prévoir une solution de rechange à l'intérieur de l'aide, ce qui implique plusieurs manières de fournir l'aide avec un ordre dans les différentes versions possibles,
- mettre l'aide dans une configuration "par défaut", comme le sont les situations de repli. Il faut alors déterminer cette configuration "par défaut" au niveau des sorties de l'aide.

Cette défaillance au niveau d'une aide aux utilisateurs a aussi souvent des répercussions au niveau du fonctionnement général de l'équipement. Au niveau des modes de fonctionnement, il faudra déterminer les actions à exécuter : un arrêt d'urgence dans les cas extrêmes par exemple.

La difficulté principale de l'élaboration d'un tel cahier des charges est que toutes les informations ne sont pas forcément disponibles (par exemple, choix ouvert du réseau de communication qui devra être utilisé) ou strictement imposées. Ceci ne veut pas forcément dire que le cahier des charges est incomplet, mais que, soit il n'existe pas de connaissances suffisantes a priori pour permettre d'imposer ces contraintes sur les différentes aides, soit il est laissé au concepteur le libre choix des moyens qu'il pourra mettre en œuvre pour parvenir à réaliser l'aide demandée.

Toutes ces contraintes doivent évidemment être prises en compte par le concepteur afin de satisfaire au plus près les désirs du client. Enfin, il ne faut pas oublier que le cahier des charges est un document qui est la base du dialogue entre le concepteur et le client afin de construire un équipement intelligent. Son cahier des charges et toutes les étapes du cycle de développement doivent donc être "accessibles" aux diverses parties en présence, tant au niveau de l'expression des besoins que de la méthode de développement utilisée.

I.5- Conclusion :

Dans le cycle de vie de tout système informatisé, la première étape qui consiste en sa conception est primordiale. Cette première étape permet de construire un système cohérent et fiable pour la suite de son exploitation. La conception d'un système ayant des répercussions dans tout son cycle de vie, il est nécessaire d'y porter une attention toute particulière.

Dans la conception d'un système, la formation et/ou l'interprétation du cahier des charges est une des phases les plus délicates. Il convient alors d'utiliser des méthodes les plus strictes possibles afin d'éviter toute catastrophe ultérieure que ce soit au niveau du système en lui-même, qu'au niveau de son environnement (humain ou système global de l'entreprise) ou encore au niveau financier. C'est pourquoi le cahier des charges doit posséder les propriétés rapportées au paragraphe I.3.1.3. : contenir un ensemble d'informations tout en restant assez clair pour permettre une bonne interprétation dans les phases consécutives du cycle de développement du système.

Les critères donnés pour le cahier des charges sont valables pour tout système informatisé. Dans le cadre des S.A.I.D., les équipements de terrain intelligents ont pris une place importante car ils permettent une distribution des traitements localement sur la partie de l'environnement physique dont ils s'occupent et donc sur les données concernées par cette partie. Vu la particularité des équipements intelligents au sein des systèmes informatisés, le cahier des charges doit contenir un certain nombre d'informations plus spécifiques qui ont été énoncées au paragraphe I.4.2 afin d'obtenir un cahier des charges correct.

Cependant, toutes les données relatives au cahier des charges n'étant pas toujours disponibles, il convient d'avoir des méthodes de conception bien définies afin de pouvoir pallier le manque d'information et pour répondre aux problèmes qui en découlent. Il existe plusieurs méthodes de conception pouvant être utilisées pour les instruments intelligents comme nous allons le voir dans le chapitre suivant.

Ces méthodes de conception sont d'autant plus importantes que la complexité des systèmes tend à faire une distribution des traitements sur différents sites afin d'alléger les unités de gestion globale du système en son entier en laissant aux systèmes locaux une partie des

traitements "de base". Une erreur de conception au niveau de ces équipements rendrait alors l'ensemble des équipements distribués non-optimum en terme d'interopérabilité et/ou d'interfonctionnement. Il existe, à ce titre, plusieurs types de langage de conception qui peuvent être utilisées lors de la conception d'équipements intelligents. Ces types de langages et quelques-unes de leurs applications vont être exposés dans le chapitre suivant avec leurs avantages et inconvénients.

Références :

[AFN91] AFNOR NF-X 50-151 "Analyse de la valeur, analyse fonctionnelle. Expression fonctionnelle du besoin et cahier des charges fonctionnel", décembre 1991.

[AFN96] AFNOR NF-Z 68-901 "Génie automatique. Représentation des systèmes de contrôle et de commande des systèmes automatisés de production", novembre 1996

[ALA89] Alakl N., Lalanne J.-L., "Architecture technologique des systèmes d'information : la méthode TACT", les Editions d'Organisation, Paris, 1989.

[ASC87] Asch G., "Les capteurs en instrumentation industrielle", Dunod, 1987.

[BAY95] Bayart M., Simonot-Lion F., "Projet MESR2033 : Impact de l'émergence des réseaux de terrain et de l'instrumentation intelligente dans la conception des architectures des systèmes d'automatisation de processus", février 1995.

[BOE88] Boehm B.W., TRW defense systems group, "A Spiral Model of software development and enhancement", IEEE Computer, 21 (5), May 1988, pp.61-72.

[CAL90] Calvez J.-P., "Spécification et conception des systèmes : une méthodologie", Editions Masson, Paris, 1990.

[COU96] Coulange Bernard, "Réutilisabilité du logiciel", Editions Masson, Paris, 1996.

[DEL95] Delfieu D., "Expression et Validation de Contraintes Temporelles pour la Spécification des Systèmes Réactifs", Thèse de l'université Paul Sabatier, Toulouse, janvier 1995.

[GEH94] Gehin A.L., "Analyse Fonctionnelle et Modèle Générique des Capteurs Intelligents : Application à la Surveillance de l'Anesthésie", Thèse de l'université de Lille1, janvier 1994.

[GUE93] Guéguen H., "Spécification d'une Méthode d'Expression des Besoins de Commande pour les Systèmes de Production Continus et Semi-Continus", Thèse de l'Université de Rennes 1, décembre 1993.

[HAR85] Harel D., Pnueli A., "On the Development of Reactive Systems, in Logic and Models of Concurrent Systems", NATO Asi Series in Computer Science, pp. 447-498, 1985.

[HOD91] Hodgson R., "The X-Model : a process model for object-oriented software development", Proc. of Le génie logiciel & ses applications, Toulouse, 1991.

[LAM97] Lampierre-Couffin S., "De la vérification de cahiers des charges de systèmes à événements discrets à la validation des spécifications décrites en

Grafcet", thèse de l'Ecole Normale Supérieure de Cachan, Toulon, janvier 1997.

[ROY70] Royce W.W., "Managing the development of large software systems", Proc. WESTCON, Calif., USA, 1970.

[STA94] Staroswiecki M., Bayart M., "Actionneurs Intelligents", Editions Hermès, Paris, 1994.

[THO93] Thomesse J.P., "Time and industrial local area networks", COMPEURO'93, pp 365-374, Paris, mai 1993.

Chapitre II : Langages Utilisés dans la Conception

Table des Matières :

II.1- Introduction : **39**

II.2- Les différentes approches : **39**

II.2.1- Les points de vue : 39

II.2.2- Approches Orientées Données : 40

II.2.3- Approches Orientées Fonctions : 40

II.2.3.1- Décomposition fonctionnelle : 41

II.2.3.2- Décomposition par flots de données : 41

II.2.4- Approches Orientées Comportement : 42

II.2.4.1- Approches par systèmes de transitions : 43

II.2.4.2- Approches par Réseaux de Petri : 44

II.2.4.3- Approches par GRAFCET : 46

II.2.4.4- StateCharts : 47

II.2.4.5- Langages synchrones : 49

II.2.4.6- Langages asynchrones : ELECTRE 49

II.2.5- Approches Combinées : 50

II.2.6- Approches Orientées Objet : 51

II.2.6.1- Introduction : 51

II.2.6.2- Principes de l'approche orientée objet : 52

II.2.6.2.1- L'abstraction : 52

II.2.6.2.2- L'encapsulation : 53

II.2.6.2.3- L'héritage : 53

II.2.6.3- Langages orientée objet : 54

II.2.6.3.1- O.O.A. : 54

II.2.6.3.2- H.O.O.D. : 55

II.2.6.3.3- O.M.T. : 56

II.2.7- Tableau comparatif des différentes approches : 57

II.3- Conclusion : **58**

Références : **59**

Langages Utilisés dans la Conception

II.1- Introduction :

Dans ce chapitre, nous exposons plus particulièrement les différents langages pouvant être utilisés pour le **développement** de systèmes automatisés, et donc utilisables pour le développement d'équipements intelligents.

On présentera dans un premier temps les approches utilisées pour répondre correctement à tous les besoins exprimés dans le cahier des charges. On détaillera alors la ou les langages les plus couramment utilisées associées au type d'approche en question.

Il sera ensuite discuté pour chaque type de langage utilisé les avantages et inconvénients que possède le langage de développement par rapport aux propriétés que possède le cahier des charges. On s'intéressera plus particulièrement à la complétude du langage par rapport à son accessibilité vis à vis des différents interlocuteurs mis en jeu dans le cycle de développement de l'équipement intelligent.

II.2- Les différentes approches :

II.2.1- Les points de vue :

La modélisation d'un système peut être effectuée selon plusieurs points de vue. On peut alors classifier les différentes approches suivant l'aspect sous lequel elles abordent le problème. On peut dénombrer ainsi trois axes principaux :

- FAIRE : on modélise ici ce que le système FAIT, ce sont les approches orientées fonctions (ou traitements),
- ETRE : on modélise alors ce que le système EST, ce sont les approches orientées données,
- DEVENIR : on modélise ce que le système DEVIENT, ce sont les approches orientées comportement.

Ces différents points de vue pris séparément ne sont pas le seul point de vue par lequel on se doit d'aborder un Système Automatisé et donc tout équipement intelligent. Cependant nous allons les décrire un par un afin de pouvoir aborder plus clairement les approches dites "combinées" des précédentes afin de couvrir au maximum tous les aspects de l'équipement intelligent.

On supposera, dans la description de ces différentes approches, que les propriétés de correction et d'utilité de tous les besoins exprimés dans le cahier seront toujours vérifiées : leur vérification est une étape primordiale à toute approche objective d'une démarche de conception de tout équipement.

II.2.2- Approches Orientées Données :

On s'intéresse à ce que l'équipement est dans son ensemble, et, en particulier, on regarde les caractéristiques propres de l'équipement. L'équipement est alors décrit minutieusement. Cette "propriété" des approches orientées données implique qu'on s'intéresse au type d'équipement ainsi qu'à la technologie utilisée, i.e. à comment est réalisé ce que l'équipement fait.

On modélise par cette approche tous les aspects liés à l'information. On y retrouve par exemple les diagrammes entités-relations définies par Peter Chen [CHE76] : Objet + Attributs + Relations. Ceci correspond plus ou moins à ce que nous avons appelé Identificateur et Format dans le précédent chapitre, les relations donnent un niveau d'information supplémentaire entre les objets par l'introduction d'une certaine hiérarchie dans l'information. Ces types de diagramme sont particulièrement utilisés pour structurer les systèmes d'information de grande taille comme la méthode TACT [ALA89]. Ainsi, tous les langages de description formelle appartiennent à la catégorie des approches orientées données. Des langages tels que D.D.L. ont notamment été utilisés pour spécifier les protocoles des langages de communication [WES92][GEH94].

Les informations ainsi modélisées ne traitent cependant pas certains concepts comme le service, en termes de comportement et de structure, en termes de liens généralisation-spécialisation et de composés-composants. Cette méthode ne peut donc pas suffire en elle-même à l'entière conception d'un équipement intelligent mais son concept est malgré tout nécessaire pour prendre en compte l'aspect informationnel de tout équipement intelligent.

II.2.3- Approches Orientées Fonctions :

L'approche orientée fonctions s'appuie sur ce que fait l'équipement sans s'intéresser très finement à comment il le réalise exactement : on regarde le résultat qui sort d'une "boîte noire". Cela laisse au concepteur-réalisateur le choix de la technologie la mieux adaptée en termes de performances attendues et de coût de la technologie qui pourra être utilisée pour la dite "boîte noire".

II.2.3.1- Décomposition fonctionnelle :

Les approches orientées fonctions consistent généralement en la décomposition du problème de conception en plusieurs fonctions, sous-fonctions et d'interfaces entre ces fonctions. Pour ce dernier cas, on a affaire avec un genre particulier de fonctions.

A chacune des fonctions ainsi déterminées, on associe un ou plusieurs traitements particuliers. On se penche donc ici plutôt sur la spécification des différents traitements et interfaces nécessaires à la réalisation du cahier des charges de l'équipement [LUT97]. Cependant, de par le fait qu'il n'y ait dans cette approche, par décomposition fonctionnelle, que peu de correspondance entre la fonctionnalité et le domaine d'application concerné, il est difficile d'estimer si la spécification apportée recouvre tous les aspects de ce que la nouvelle application doit faire. La complétude du problème n'est donc pas forcément atteinte si on néglige certaines parties qui n'apparaissent pas explicitement dans l'énoncé du cahier des charges. Par exemple, la fonction "calcul de la vitesse" nécessite deux types de données : une distance et un temps. Si une de ces deux données n'apparaît pas explicitement dans le cahier des charges initial, la spécification est incomplète. Afin de s'assurer de la complétude du cahier des charges, il faut alors soit définir une nouvelle fonction permettant de fournir la ou les données manquantes, soit celles-ci devront être fournies par l'environnement de l'équipement intelligent : il faut alors l'expliciter dans le cahier des charges. On voit alors que la décomposition fonctionnelle seule ne couvre pas tous les aspects plus ou moins subtils du domaine du problème.

Il est donc nécessaire d'avoir une approche plus stricte et systématique. C'est ce qu'apportent les décompositions par flots de données.

II.2.3.2- Décomposition par flots de données :

Cette approche, en plus de la précédente, regroupe les flots de données et de contrôle en entrée et en sortie de l'équipement intelligent (processus, système d'automatisation, utilisateurs).

La décomposition se fait par raffinages successifs des différents traitements jusqu'à ce qu'on obtienne tous les traitements en termes de flots de données. Cette méthode implique le fait qu'on ait des notions sur les traitements qui sont utilisés pour obtenir les différentes données, ou tout au moins une idée globale des besoins en termes de données consommées par les traitements pour obtenir les différentes données de sorties

des traitements ainsi que les flots de contrôle générés. Ce point fait qu'on s'approche un peu du point de vue des approches orientées données en définissant assez précisément, mais pas dans le détail, les données qui circulent dans l'équipement.

Avec cette approche, le concepteur-réalisateur représente le monde réel directement à partir du cahier des charges. Ce dernier, comme l'indique la décomposition fonctionnelle, donne les différents flots de données demandées qui sont générés par la spécification en fonctions. Ces fonctions consistent en des traitements qui transforment et consomment des données, elles-mêmes issues d'autres fonctions. On voit ici que cette approche nécessite une bonne analyse des flots de données de la part des deux parties en présence : le concepteur et le demandeur du service. De plus, cette approche est plus structurée et complète que la seule décomposition fonctionnelle puisqu'elle s'appuie sur une démarche descendante itérative ne s'arrêtant que lorsque tous les flots de données (données et contrôles) ont été définis.

Plusieurs langages s'appuient sur la décomposition fonctionnelle par flots de données. On peut citer parmi eux les plus connus : S.A. (Structured Analysis) et son extension aux systèmes temps-réel incluant des problèmes temporels dans la spécification réalisée par Ward (S.A.R.T.) [WAR86][HAT90].

Par les approches orientées fonctions, on définit tous les flots de données nécessaires et tous les traitements associés à ceux-ci, tout en s'assurant, par une démarche rigoureuse, que rien n'a été oublié dans la spécification. Cependant, la cohérence n'est généralement pas explicitée tant au point de vue global de l'équipement que du comportement des différents traitements associés aux flots de données définis lors de la spécification fonctionnelle. De plus, dans les approches purement orientées fonctions (ou orientées flot de données), il n'est généralement fait que peu d'état de la structure des données comme le proposent les approches orientées données : la spécification de l'équipement dans son ensemble global n'est pas complète.

II.2.4- Approches Orientées Comportement :

L'aspect comportement est souvent une approche utilisée pour les systèmes d'automatisation simples. Il permet de prendre en compte le comportement des systèmes ayant une partie discrète, en opposition avec les systèmes purement continus.

Les approches orientées comportement s'intéressent donc plus à la dynamique du système et à son évolution dans le temps qu'à son état statique qui est plus représenté par les deux types d'approches précédentes. Il est ainsi courant d'adjoindre au modèle obtenu soit par une approche orientée données soit par une approche orientée fonctions,

un modèle de comportement "dynamique". En ce sens, elles s'appuient sur l'une ou l'autre des approches précédentes afin de déterminer les changements d'état du système dans le temps, car il n'est pas possible de déterminer le comportement du système sans savoir ce qu'il est et/ou ce qu'il fait.

Dans ces approches, il est donc fréquent de retrouver les méthodes liées à la théorie des graphes telles que, simplement, les graphes, les réseaux de Petri, le GRAFCET [DAV89], les StateCharts [HAR87]. Cependant ces méthodes sont loin d'être les seules permettant de vérifier le comportement dynamique d'un système, on prend alors les langages de programmation qui utilisent alors, généralement, soit le principe asynchrone, soit le principe synchrone. On se propose de donner, dans ce qui suit, les différents points de vue des approches comportementales.

II.2.4.1- Approches par systèmes de transitions [ARN92] :

Ces systèmes sont des systèmes assez simples. Ils sont constitués d'états dans lesquels le système peut se trouver et de l'ensemble des transitions (ou des changements d'états) que le système permet de par sa structure. Les changements d'états sont la conséquence d'événements extérieurs (stimuli) et/ou d'actions effectuées par le système en lui-même (on parle alors de réaction) et d'un ensemble de conditions dépendant de l'état d'origine et de l'état de destination. Les changements d'états sont donc la conjonction de l'état des variables internes du système, de leur éventuelle évolution et d'un stimulus.

On les représente graphiquement par des "ronds", correspondant aux états, et d'arcs orientés étiquetés, correspondant aux transitions. La figure 2.1, ci-dessous, est un exemple d'automate simple :

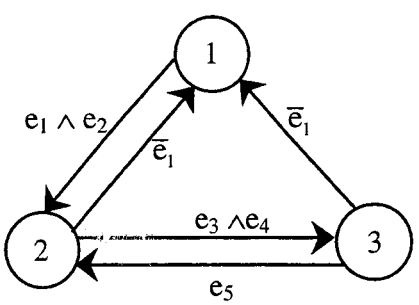


Figure 2.1 : Représentation d'un automate fini

L'avantage de cette représentation est qu'elle est simple et assez bien adaptée pour dialoguer avec le client : i.e. l'accessibilité est généralement acquise. Cependant, ce

genre de représentation se limite souvent à des systèmes assez simples de par les faits qu'ils ne supportent aucune représentation hiérarchique et qu'ils sont mal adaptés à la représentation d'évolutions parallèles. Tous les aspects comportementaux n'étant pas explicités pour des équipements relativement complexes, ces aspects ne peuvent pas être vérifiés s'ils apparaissent clairement dans le cahier des charges.

Les paragraphes suivants montrent des solutions à certains des points précités.

II.2.4.2- Approches par Réseaux de Petri :

Les réseaux de Petri sont aussi des systèmes à états finis comme le sont les automates finis. Cette approche a été établie premièrement par un allemand du nom de Carl Adam Petri [PET62] et ils ont énormément évolués depuis.

Les réseaux de Petri sont constitués de places dont les marquages représentent les états du système et de transitions dont le franchissement se fait sous l'occurrence d'événement entraînant au moins une modification de l'état du système [BRA83] (cf. figure 2.2). Chaque marque comporte un nombre entier de jetons. A chaque instant, le marquage (i.e. le nombre de jetons à chaque place) donne l'état du système. Une transition est franchissable ou validée si chacune des places situées en amont de la transition contient au moins un jeton ; le franchissement de la transition peut alors avoir lieu (une transition franchissable n'est pas nécessairement immédiatement franchie) [DAV89]. Le franchissement d'une transition entraîne une modification du marquage comme suit : les places en amont de la transition se voient chacune enlever un jeton et les places en aval de la transition se voient toutes ajouter un jeton.

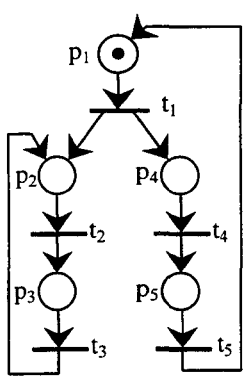


Figure 2.2 : Réseau de Petri marqué

Le principal intérêt des réseaux de Petri est leur capacité à traiter le parallélisme des systèmes. De plus, leur modélisation aussi bien graphique que matricielle permet

l'analyse des propriétés et la validation de la spécification engendrée. Ceci a permis la naissance de très nombreux travaux qui ont eux-mêmes permis d'avoir des cadres méthodologiques intéressants :

- On peut notamment citer le cadre C.I.M. (Computer Integrated Manufacturing) où les réseaux de Petri ont permis la prise en compte du comportement très réactif de ce type d'application. On peut citer les travaux autour du projet CASPAIM I [BOI91] et CASPAIM II [HUV94]...
- Les réseaux de Petri ont aussi été utilisés dans le cadre des systèmes hybrides [ALL94].
- ...

L'un des désavantages des réseaux de Petri réside dans sa capacité à modéliser le parallélisme : la modélisation de ce dernier entraîne bien souvent une grande complexité de construction et d'analyse du système de par le fait d'une explosion combinatoire des états possibles : il devient parfois impossible de surveiller tous ces états, ce qui nuit à la commandabilité-surveillabilité de l'équipement. L'un des autres problèmes des réseaux de Petri est que le déterminisme n'est pas un concept se trouvant au cœur de la méthode [DEL95], ce qui pose un grand problème dans le cadre des équipements intelligents : le déterminisme est une source de vérifiabilité. Afin de limiter la taille et la complexité de ces réseaux, il a été introduit des extensions aux réseaux de Petri assez différentes les unes des autres. On peut citer les plus courantes : réseaux de Petri colorés, à prédicats, à objets, ... Cela a aussi introduit un vocabulaire spécialisé, des méthodes et des représentations qui restent assez peu accessible à la majorité des industriels (i.e. les clients les plus fréquents des équipements intelligents). Ce problème de clarté entre les différents interlocuteurs de la conception peut, à terme, poser un défi insurmontable.

Les réseaux de Petri apportent donc un apport substantiel à l'étude comportementale d'un équipement en intégrant le parallélisme, ce qui tend vers la complétude du cahier des charges, mais cela se fait au détriment de la vérifiabilité, parfois, et surtout au détriment de l'accessibilité. Nous pouvons aussi rappeler que les réseaux de Petri ne sont pas normalisés, contrairement au GRAFCET que nous allons présenter ci-après.

II.2.4.3- Approches par GRAFCET :

Le GRAFCET est une simplification des réseaux de Petri (interprétés) [DAV89] qui a été établie afin d'harmoniser les différents outils de description des automatismes logiques et de fournir un outil commun. C'est un outil qui permet de modéliser le comportement attendu des systèmes de contrôle-commande. De plus, comme nous venons de le dire ci avant, le GRAFCET est normalisé : norme internationale [IEC91] et française [AFN95].

Le parallélisme est un élément fondamental du GRAFCET : il correspond à l'activation simultanée de plusieurs étapes d'un grafcet. Ce dernier peut comprendre plusieurs graphes qui évoluent en parallèle ou être constitué d'un seul graphe. Dans ce dernier cas, le parallélisme est obtenu grâce à une divergence ET (Figure 2.3).

L'autre intérêt de ce formalisme est la modularité : chaque étape peut elle-même constituer un grafcet à part entière, on a alors une macro-étape. Ceci permet d'introduire une certaine hiérarchie dans le GRAFCET et de diminuer ainsi la complexité de la représentation graphique. On obtient, grâce à la modularité, une accessibilité, une modifiabilité et une vérifiabilité fortement accrues.

L'évolution du grafcet se fait sous l'occurrence des événements externes en fonction des variables externes et internes. Ces événements sont définis par des fronts montants ou descendants des variables externes (dont les temporisations). A chaque événement est associé un instant d'occurrence. Le grafcet évolue suivant les hypothèses suivantes[LHO97] :

- deux événements indépendants ne se produisent jamais simultanément,
- tous les événements externes sont immédiatement pris en compte à leur occurrence,
- toutes les évolutions d'un grafcet consécutives à l'occurrence d'un événement sont faites avant l'occurrence de l'événement suivant (le grafcet évolue d'état stable en état stable).

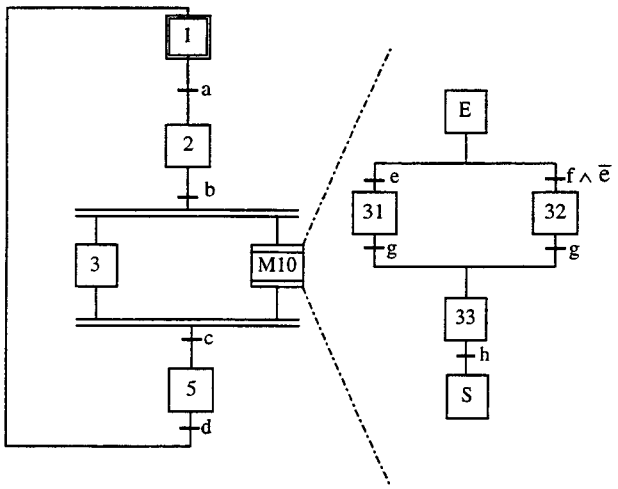


Figure 2.3 : Exemple de grafcet avec divergence en ET et macro-étape

Les problèmes que posent le GRAFCET sont souvent liés à la faiblesse des mécanismes de structuration des modèles et de préemption. Cette dernière notion est plus explicitée dans le modèle StateCharts. C'est malgré tout un outil encore fort utilisé dans le monde industriel et il est possible de faire des vérifications du comportement attendu comme il est proposé dans [LAM97].

II.2.4.4- StateCharts :

Les StateCharts ont été introduits par Harel [HAR87]. C'est un formalisme graphique de spécifications comportementales. C'est aussi un mécanisme basé sur les machines à états. On retrouve, à ce titre, les représentations des états et des transitions. Les transitions sont du type : [condition de franchissement]/action (i.e. la condition de franchissement entraîne aussi une action qui est simultanément lancée et traitée). On y retrouve des mécanismes qui permettent de prendre en compte la hiérarchie, le parallélisme et la préemption.

La hiérarchie est basée sur la capacité d'encapsulation (cf. III.2.6.2.2) des états du stateChart. Cette propriété permet d'aborder plus facilement les systèmes complexes de grande taille. Le parallélisme est obtenu, comme pour le GRAFCET, par une décomposition en ET d'un état : l'activation de l'état décomposé induit l'activation de plusieurs sous-états. Ces sous-états sont alors dits "orthogonaux".

La préemption est le troisième mécanisme qui est abordé par les StateCharts. La préemption est la capacité d'interrompre à tout moment un traitement en cours d'exécution. Cette notion sera vue plus explicitement dans le chapitre IV avec le SYNCCHARTS [AND96] qui sont une extension des StateCharts et de ARGOS [MAR90] et qui permettent plusieurs types de préemption. Les transitions d'un

stateCharts fonctionnent d'après la préemption forte : dès que la transition est validée, on désactive immédiatement l'état en cours ainsi que tous ses sous-états pour activer ceux de l'état de destination de la transition.

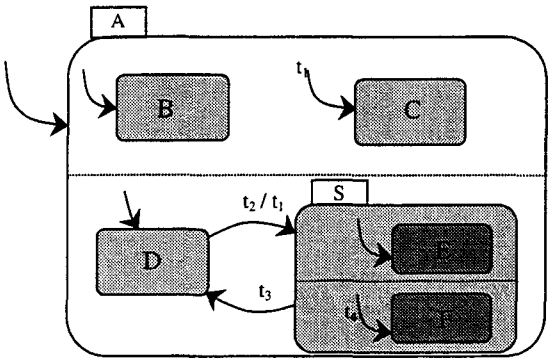


Figure 2.4 : exemple d'un stateChart.

Cette représentation graphique présente un intérêt visuel pratique immédiat permettant un dialogue concret avec les différents interlocuteurs du projet de conception. Cette représentation a été utilisée par H. Guéguen [GUE93] et par F. Maraninchi [MAR90] pour spécifier le comportement des systèmes continus et hybrides.

Les StateCharts semblent donc être un compromis intéressant à l'étude comportementale des équipements intelligents : ils possèdent une certaine modularité, une hiérarchie basée sur l'encapsulation et le parallélisme de sous-états orthogonaux, ils introduisent une notion de préemption des différents états du stateChart et possèdent un modèle graphique interprétable. Tout ceci permet d'obtenir un panorama complet de l'étude comportementale de tout équipement intelligent.

Les graphes sont une façon de représenter le comportement d'un système automatisé. Une autre manière de le représenter se situe dans les langages de programmation plus spécifiques aux systèmes informatiques. Il existe deux grandes catégories de langages permettant la modélisation du comportement de systèmes automatisés : les langages synchrones et les langages asynchrones.

II.2.4.5- Langages synchrones :

Les langages synchrones utilisent le paradigme synchrone qui peut se donner en cinq hypothèses de base [AND96] :

- **Signal** : le système interagit avec son environnement par l'intermédiaire de signaux : des signaux d'entrée et des signaux de sortie.
- **Perception globale** : on suppose que tous les signaux d'entrée et tous les signaux de sortie sont perçus simultanément et que cette perception est objective. On a alors affaire avec des tuples de signaux.
- **Temps logique** : il n'y a pas de temps physique, mais des instants logiques.
- **Hypothèse Zéro Délai :** les opérations internes se font, vu de l'extérieur, en un temps nul. Les signaux de sortie sont alors synchrones aux signaux d'entrée
- **Broadcasting** : tous les signaux sont instantanément émis.

On peut citer parmi les langages qui utilisent le paradigme synchrone, celui qui est certainement le plus connu : ESTEREL. ESTEREL est un langage qui permet de générer des automates à états finis à partir d'une description de haut niveau et modulaire. ESTEREL [BOU91] est un langage orienté vers la gestion des flux de contrôle, contrairement à d'autres comme SIGNAL [BEN90], plus orienté flux de données. ESTEREL permet une approche modulaire incluant le parallélisme et la préemption, cependant la forme textuelle du modèle rend sa compréhension difficile d'accès pour un non-spécialiste.

II.2.4.6- Langages asynchrones : ELECTRE [ROU92]

ELECTRE est un langage comportemental qui permet d'exprimer l'évolution des différentes tâches (ou modules) du système en fonction des événements internes ou externes susceptibles de les affecter. On y retrouve l'expression du séquencement, du parallélisme, de l'auto-exclusion et de la répétitivité des tâches. L'asynchronisme du langage est expliqué par le fait que l'apparition des événements sont asynchrones par rapport à l'évolution interne des modules et que les occurrences événements sont elles-mêmes asynchrones entre elles. Les événements sont, bien entendu, traités dès leur apparition.

On le voit, il existe un grand nombre de langages permettant d'étudier le comportement dynamique d'un système. Il n'en est pas moins exclu qu'il faille une

analyse "statique" complète préalable afin de pouvoir bien modéliser le comportement. Le nombre important d'approches s'explique par le fait qu'il existe plusieurs moyens d'aborder le comportement d'un système : les notions de séquencement de tâches, de parallélisme, de préemption, de synchronisation, ... ne sont pas toutes caractérisées par tous les modèles proposés. Seuls les StateCharts et leur extension aux SYNCCHARTS semblent aborder tous ces problèmes de l'aspect comportemental, tout en conservant une certaine accessibilité vis à vis du client.

II.2.5- Approches Combinées :

On l'a vu dans les paragraphes précédents, un seul type d'approche n'est pas suffisant pour prendre en compte tous les aspects d'un système : données, traitements et comportement.

Parmi ces approches combinées, on retrouve principalement des méthodes qui allient à la fois le coté Données et le coté Fonctions car elles permettent d'avoir une bonne vision du système tant au point de vue fonctionnel qu'au point de vue structurel. On peut citer par exemple la méthode Merise inspirée du modèle entité-association proposé par CHEN [CHE76] agrémenté d'un modèle de traitements. Une autre méthode couramment utilisée dans la modélisation de Systèmes Automatisés est la méthode S.A.D.T. (Structured Analysis and Design Technique) qui utilise des actigrammes inspirés d'une représentation par flots de données et de l'analyse structurée [IGL88][GEH94].

La figure ci-après représente les principaux langages utilisés pour le développement de Systèmes Automatisés et donc utilisables pour les équipements intelligents. Les pointes du triangle ainsi formé représentent les trois aspects de modélisation cités précédemment :

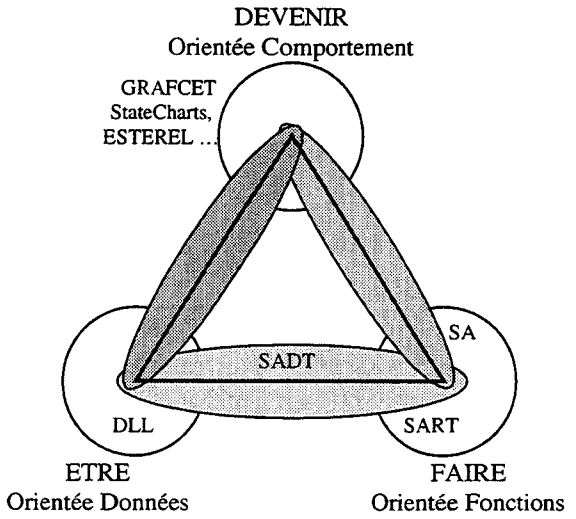


Figure 2.5 : Classification des langages de modélisation des Systèmes Automatisés.

Cette classification, quelque peu arbitraire, fait néanmoins ressortir qu'il n'existe malheureusement aucun langage qui se positionne au point idéal situé au centre du triangle et qui prendrait en compte les trois aspects Données, Fonctions et Comportement. Le fait est que, la plupart du temps, les études sur les systèmes se font suivant plusieurs langages dont les résultats sont ensuite comparés et réunis afin de former un résultat cohérent.

Afin d'apporter une vision incorporant l'aspect comportement aux aspects données et traitements, les approches orientées objet ont fait leur apparition dans la modélisation et sont de plus en plus utilisés, comme nous allons le voir ci-après.

II.2.6- Approches Orientées Objet : [GUE93] [LUT97]

II.2.6.1- Introduction :

Les approches orientées objet se basent sur la formation d'objets interagissant les uns avec les autres. On peut ainsi concevoir à partir d'objets tout type de système.

De nombreux langages utilisant le principe de l'objet, initialement liées au génie logiciel, sont apparus ces dernières années et sont désormais appliqués directement à la conception des systèmes industriels. L'émergence de ces langages les rend incontournable dans l'approche de la conception des équipements intelligents.

Nous allons nous attacher à décrire de manière formelle les différents apports de l'approche objet par rapport aux autres approches précédemment citées.

II.2.6.2- Principes de l'approche orientée objet :

C'est à la fin des années 1960 que le concept d'objet est apparu avec le langage Simula67. Les différentes versions de Smalltalk qui en ont résulté sont à l'origine de la popularité des langages à objets.

Les langages orientés objet se basent sur trois principes fondamentaux [AUB91] : l'abstraction, l'encapsulation et l'héritage. Nous allons exposer plus explicitement ces trois principes ci-dessous.

II.2.6.2.1- L'abstraction :

Comme pour la plupart des approches fonctionnelles, on s'intéresse tout d'abord au concept du domaine de l'application considérée. Cela permet de ne prendre des décisions concernant l'implémentation que quand le problème est bien cerné. L'objectif de l'abstraction est alors de faire ressortir du cahier des charges les points primordiaux de ceux qui ne le sont pas, afin de pouvoir mettre en place une solution en symbiose avec le problème posé. A ce titre, il n'existe pas un unique modèle représentant "La Solution Absolue" (i.e. unique), mais un ensemble de modèles adéquats ou inadéquats [RUM95]. Le but est bien entendu de prendre un modèle adéquat qui corresponde au problème posé par le cahier des charges.

L'abstraction choisie afin de modéliser le problème est l'objet. Un objet est défini comme la représentation d'une entité à laquelle sont associés un comportement et des informations. Parallèlement, seules les opérations que l'objet peut exécuter sont visibles de l'extérieur. Chaque objet dispose, premièrement, d'un état relatif à ses attributs (donc cachés aux autres objets) et, deuxièmement, d'un ensemble d'opérations qui fournissent des services aux autres objets qui sont les clients de l'objet. Les communications avec les autres objets sont considérées comme des opérations particulières réagissant à des stimuli associés (une interface par objet).

Les méthodes orientées objet ne sont pas les seules à proposer l'abstraction comme méthode d'investigation du cahier des charges. L'abstraction se retrouve aussi dans SADT lorsqu'on se trouve au plus haut de la description du modèle, l'affinement de ce dernier se faisant par encapsulations successives.

II.2.6.2.2- L'encapsulation :

L'encapsulation d'un objet consiste en la séparation de ses aspects externes, accessibles aux autres objets, de ses aspects internes concernant plus l'implémentation et donc généralement invisibles aux autres objets. Les aspects internes d'un objet peuvent alors être modifiés sans vergogne si cela n'affecte en rien les autres objets interagissant avec cet objet. De ce fait, l'encapsulation n'est pas un outil exclusivement réservé aux méthodes objets, mais c'est un outil puissant puisqu'il permet de combiner dans une même entité l'aspect données et l'aspect comportement.

II.2.6.2.3- L'héritage :

Le dernier principe du concept objet est l'héritage. Le but de ce principe est de minimiser les échanges entre les différents composants tout en réduisant les redondances qui sont souvent onéreuses. On obtient alors une "éco-organisation" par factorisation du système d'information à partir de la notion d'héritage.

"L'héritage est le mécanisme qui définit entre les classes une relation dans laquelle une classe partage la structure ou le comportement défini dans une ou plusieurs classes, i.e. la réutilisation d'association de propriétés entre classes. La relation d'héritage organise hiérarchiquement les classes en graphe d'héritage. L'héritage est dit simple si une classe ne peut hériter directement que d'une classe ancêtre. Il est dit multiple si une classe peut hériter directement de plusieurs classes ancêtres[LUT97]."

La notion d'héritage peut s'appliquer à bien d'autres domaines. On peut par exemple définir la classe "Fromage du monde". Cette classe a comme propriétés que tout fromage est fabriqué à partir de lait et qu'il est fabriqué quelque part sur Terre. On peut alors définir plusieurs sous-classes dont la classe "Fromage de France". Cette sous-classe possède les mêmes propriétés que son ancêtre (son héritage) : fabriqué à partir de lait et fabriqué quelque part sur Terre. On a, en plus, la propriété sous-jacente à cette dernière : en France. On peut ainsi définir un bon nombre de classes prenant en compte le type de lait utilisé (de brebis, de chèvre, de vache...), la région de fabrication, la marque, ... Ici, comme héritage multiple, on peut alors avoir le "Fromage Camembert" qui a comme ancêtres, par exemple : "Fromage de lait de vache", "Fromage de lait cru" et "Fromage de France".

Afin d'assouplir la notion d'héritage, il a été défini la notion de **polymorphisme**. On précise que lorsqu'on définit les relations entre les objets, l'objet n'a pas besoin de connaître exactement la classe de l'autre. On peut ainsi ajouter de nouvelles classes dans le modèle sans devoir modifier les autres.

Le polymorphisme permet donc de définir plusieurs propriétés ayant le même nom sur des classes différentes et d'obtenir ainsi des réactions différentes à un même stimulus selon la classe sollicitée.

II.2.6.3- Langages orientée objet :

Afin de passer des langages structurés de programmation telles que l'offrent les langages classiques comme Pascal ou C aux possibilités que permet le concept d'objet, des langages de programmation par objets ont été développés. On peut citer parmi eux les plus connus : SmallTalk, C++, CLOS ou Eiffel. Cependant, face à ce type de programmation un peu déroutante, force est de constater qu'il n'existe que peu de programmeurs utilisant pleinement la programmation objet : par exemple, bon nombre de programmeurs utilisent C++ comme un vulgaire compilateur C [PIG91].

Il existe un certain nombre de méthodes orientées objet utilisées pour la conception. Parmi elles, on peut citer les méthodes O.O.A. de Coad et Yourdon, la méthode H.O.O.D. [ESA89] et la méthode O.M.T. [RUM95].

II.2.6.3.1- O.O.A. [COA91] :

Cette méthode permet d'obtenir un modèle qui possède une représentation graphique. Le modèle est obtenu par la superposition de cinq couches qui correspondent aux activités principales de O.O.A. :

- couche des classe&objets : elles sont déterminées en fonction de leur pertinence par rapport au domaine d'application.
- couche des relations : il s'agit des relations entre les différents classe&objets. Il en existe de deux types : Généralisation-Spécialisation (une classe est une spécialisation d'une autre) et tout - partie (une partie d'un objet est une partie d'un objet d'une autre classe).
- couche des sujets : cela correspond aux regroupements de classe&objets en fonction des relations. Cela permet de guider la lecture dans le diagramme et ainsi d'en faciliter sa compréhension sans en diminuer le nombre de classe&objets.
- couche des attributs : pour chaque entité, on définit les différentes données représentatives de l'état de l'entité.

- couche des services : on définit enfin les aspects fonctionnels et comportementaux de toutes les entités définies ci-dessus, ainsi que l'aspect communication (messages échangés).

II.2.6.3.2- H.O.O.D. [ESA89] :

La méthode H.O.O.D. (Hierarchical Object Oriented Design) vient de la conception d'architecture logicielle et est surtout orientée vers le langage ADA.

On y retrouve un peu l'approche proposée par S.A.D.T. : on regroupe les traitements et les données dans des objets qui peuvent s'encapsuler. La méthode consiste à structurer ces objets, ce qui permet de modéliser des systèmes de grande taille. On a, ici aussi, deux types de relations pour structurer le modèle : un type de relation permet de définir des relations d'utilisation, et l'autre type définit des relations de composition, ce qui permet de hiérarchiser le modèle.

La structure est représentée sous forme graphique et à chaque objet graphique, on associe un nom et un ensemble d'opérations fournies à l'extérieur. Les synchronisations entre les objets se font à l'aide d'un objet spécifique de contrôle. Le comportement doit alors être spécifié à l'aide d'autres formalismes tels que les réseaux de Petri ou ESTEREL [GUE93].

Le modèle obtenu contient un objet père racine unique qui se décompose en un ensemble de fils (dont un objet contrôle), chacun de ceux-ci pouvant à son tour se décomposer... On obtient alors un modèle modulaire fortement structuré et hiérarchisé.

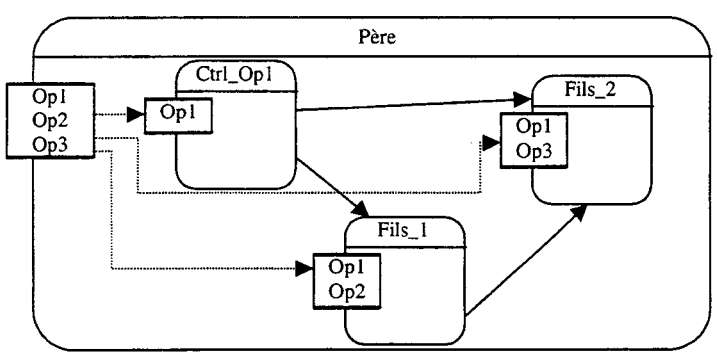


Figure 2.6 : Exemple d'un objet HOOD.

On obtient alors un développement complet au niveau fonctionnel et de données. Une partie du comportement entre les objets est obtenue via l'objet de contrôle présent dans chaque niveau graphique. L'aspect comportemental n'est cependant pas complètement abordé puisqu'il faut un autre formalisme pour le compléter. La vérifiabilité d'un tel modèle semble cependant difficile.

II.2.6.3.3- O.M.T. [RUM95][LUT97] :

La méthode O.M.T. (Object Modeling Technique) est une approche orientée objet de développement de système qui permet de supporter entièrement le cycle de vie d'un système : de la conception à l'implémentation. Elle considère les trois aspects du système : i.e. l'aspect informationnel qui donne la structure des données, l'aspect fonctionnel qui donne l'activité des différents objets et l'aspect comportemental pour l'évolution dynamique dans le temps du modèle.

Après une étude qui permet de définir les différents besoins de l'application par rapport aux utilisateurs potentiels, la méthodologie O.M.T. se décompose en trois phases : une étape d'analyse, une étape de conception et une dernière étape d'implémentation.

C'est ici le coté analyse de la méthode O.M.T. qui nous intéresse. L'analyse associe les trois aspects précédents :

- Un modèle objet qui permet de définir les propriétés statiques du système : i.e. les objets du système, les relations entre les objets, les attributs et opérations associées qui permettent alors de caractériser chaque classe du système. On obtient des diagrammes qui permettent de modéliser les objets, les classes et les relations. On a deux types de diagrammes :
  - les diagrammes de classes qui permettent de décrire les classes d'objets,
  - les diagrammes d'instances qui définissent les liaisons d'un objet par rapport à plusieurs classes d'objet.

- Un modèle dynamique permet de prendre en compte les aspects temporels, comportementaux et contrôle du système. On le représente, là aussi, par des diagrammes :
  - les diagrammes d'états pour mettre en évidence les séquences d'états et d'événements permis pour une classe d'objet,
  - les diagrammes de suivi des événements permettant de voir les échanges effectués entre les objets en tenant compte de l'aspect temporel de ces échanges.

- Un modèle fonctionnel prend en compte les aspects transformationnels en termes de données et de fonctions à réaliser sans s'occuper du moyen de les réaliser. Ils sont représentés par des diagrammes à flots de données.

Ces trois modèles échangent bien sûr des informations : le modèle dynamique a besoin des informations fonctionnelles et de celles rattachées aux objets. On obtient alors un assemblage cohérent de langages regroupant ainsi tous les aspects d'un équipement.

La technique O.M.T. a notamment été utilisée avec succès pour la modélisation de capteurs intelligents par D. Luttenbacher [LUT97]. Tous les aspects de l'équipement ont été abordés avec succès. Cependant, le passage du modèle de développement objets à son implémentation nécessite encore une "passerelle" pour traduire le modèle objet obtenu à un code directement exploitable. Malheureusement, on perd alors la modifiabilité du modèle objet, le concept d'objet se mettant difficilement en code directement dans un micro-contrôleur ou un FGPA.

Tous les langages orientés objet utilisent la notion d'héritage. Cette notion est un concept très puissant de modélisation, de hiérarchisation des objets dans l'architecture du système. L'intérêt d'un tel concept est qu'il facilite aussi grandement la réutilisation, comme le propose le cycle de vie en X [COU96][LUT97] et permet d'établir une base de données facilement consultable pour des applications similaires. Il n'en reste pas moins que les techniques de modélisation orientées objet, et notamment la méthode O.M.T., donnent une bonne base pour la réalisation de tout système en le structurant suivant les trois aspects essentiels d'un système.

II.2.7- Tableau comparatif des différentes approches :

Nous avons vu précédemment un certain nombre d'approches et de langages qui utilisés lors de la conception d'équipements intelligents. Nous proposons de faire une rapide comparaison sous la forme d'un tableau.

Nous prenons en compte les différents aspects des approches : données, fonction et comportement. Dans chacun de ces champs, nous regarderons plus particulièrement les points suivants :

- pour la part donnée : la prise en compte de l'information et sa hiérarchie
- pour le coté comportement : la Dynamique et le Parallélisme

On ajoutera à ces points deux autres aspects : l'accessibilité par rapport au client et la facilité d'intégration de la solution sur une architecture matérielle.

Tableau comparatif des différents langages utilisés dans la conception :

		Données		Fonction	Comportement		Accessibilité	Intégration
		Information	Hiérarchie		Dynamique	Parallélisme		
ESTEREL		+	+	-	++	+	-	+
GRAFCET		+	++	+	++	++	+++	+++
Réseaux de Petri		+	++	+	++	++	+	++
SADT		+	++	++	-	+	++	++
SART		++	+	++	+	-	+	+
StateCharts		+	+++	-	+	+++	++	+
OBJETS	HOOD	++	+++	++	+	++	+	-
	OMT	+++	+++	+++	++	++	+	-
	OOA	++	+++	++	+	++	+	-

Légende : - : Pas adapté au cas étudié,
+ : Peu adapté au cas étudié,
++ : Adapté au cas étudié,
+++ : Bien adapté au cas étudié.

II.3- Conclusion :

Les techniques de modélisation sont nombreuses. La plupart d'entre elles n'ont malheureusement qu'une vue partielle du domaine en n'envisageant qu'un, voire deux aspects du système. Ces langages ne permettent donc pas de prendre un système en son entier sans devoir en utiliser plusieurs qu'il faut adapter les unes aux autres dans un souci de cohérence.

A l'heure actuelle, seules certains langages orientés objet envisagent un système sous les trois aspects : données, fonctionnel et comportemental. A ce titre, ces techniques de modélisation remportent de plus en plus de succès et remplacent peu à peu les "anciennes" techniques comme S.A.D.T. pour le développement de Systèmes Automatisés et donc d'équipements intelligents.

Cependant, l'aspect d'héritage lié aux techniques orientées objet ne nous semble pas nécessaire à la bonne modélisation d'un système. Si ce concept est fort utile dans le cadre de la mise sous forme de base de données pour la réutilisation, ce concept est difficilement directement implantable sur les équipements intelligents. Nous allons donc proposer dans le chapitre suivant une méthode utilisant les services rendus aux utilisateurs conformément à ce qu'exprime le cahier des charges en termes de besoin (aides fournies aux utilisateurs).

Références :

[AFN95] NF C 03-190, "Diagramme fonctionnel GRAFCET pour la description des systèmes logiques de commande", septembre 1995.

[ALA89] Alakl N., Lalanne J.-L., "Architecture technologique des systèmes d'information : la méthode TACT", les Editions d'Organisation, Paris, 1989.

[ALL94] Alla H., "Les réseaux de Petri : un outil particulièrement bien adapté à la modélisation des systèmes hybrides", Symposium ADPM'94 : "Automation of Mixed Process : Dynamical Hybrid Systems", Bruxelles, novembre 1994.

[AND96] André C., "Representation and Analysis of Reactive Behaviors : a SyncChronous Approach", Papier invité, CESA'96, pp. 19-29, Lille, 1996.

[ARN92] Arnold A., "Systèmes de transitions finis et sémantique des processus communicants", Editions Masson, Paris, 1992.

[AUB91] Aubert J.P., Dixneuf P., "Conception et programmation par objets : techniques, outils et applications", Editions Masson, 1991.

[BEN90] Benveniste A., Le Guernic P., "Hybrid Dynamical Systems Theory and the SIGNAL Language", IEEE Transaction on Automatic Control, Vol 35, n°5, pp. 535-546, mai 1990.

[BOI91] Bois S., "Intégration de la gestion des modes de marche dans le pilotage d'un système automatisé de production", Thèse de l'Université de Lille1, novembre 1991.

[BOU91] Boussinot F., De Simone R., "The Esterel language", Proceeding of the IEEE, 79, pp. 1293-1304.

[BRA83] Brams G.W., "Réseaux de Petri : théorie et pratique", Tome 2, Ed. Masson, 1983.

[CHE76] Chen P.P.S., "The entity-relationship model : toward a unifiedview of data", ACM transaction on database systems, Vol. 1, n°1, mars 1976, pp. 9-36.

[COA91] Coad P., Yourdon E, "Object-Oriented Analysis", Prentice Hall, Eaglewood Cliffs, New Jersey, 1991.

[COU96] Coulange Bernard, "Réutilisabilité du logiciel", Editions Masson, Paris, 1996.

[DAV89] David R., Alla H., "Du GRAFCET aux réseaux de Petri", Editions Hermès, Paris, 1989.

[DEL95] Delfieu D., "Expression et Validation de Contraintes Temporelles pour la Spécification des Systèmes Réactifs", Thèse de l'université Paul Sabatier, Toulouse, janvier 1995.

[ESA89] European Spacial Agency, "HOOD Reference Manual-HOOD User Manual", septembre et décembre 1989.

[GEH94] Gehin A.L., "Analyse Fonctionnelle et Modèle Générique des Capteurs Intelligents : Application à la Surveillance de l'Anesthésie", Thèse de l'université de Lille1, janvier 1994.

[GUE93] Guéguen H., "Spécification d'une Méthode d'Expression des Besoins de Commande pour les Systèmes de Production Continus et Semi-Continus", Thèse de l'Université de Rennes 1, décembre 1993.

[HAR87] Harel D., "StateCharts : a visual formalism for complex system", Science Computer Program, vol. 8, n°3, juin 1987, pp. 514-530.

[HAT90] Hatley D.J., Pirbhai I.A., "Stratégies de spécification des systèmes temps-réel (SA-RT)", Editions Masson, 1990.

[HUV94] Huvenoit D., "De la conception à l'intégration de la commande modulaire et hiérarchisée des systèmes flexibles de production manufacturière", Thèse de l'Université de Lille1, octobre 1994.

[IEC91] IEC, "Preparation of Function Charts for Control Systems", IEC 848, Genève, Suisse, 1991.

[IGL88] I.G.L. Technology, "SADT un langage pour communiquer", Eyrolles, 1998.

[LAM97] Lampierre-Couffin S., "De la vérification de cahiers des charges de systèmes à événements discrets à la validation des spécifications décrites en Grafcet", thèse de l'Ecole Normale Supérieure de Cachan, Toulon, janvier 1997.

[LHO97] Lhoste P., Faure J.M., Lesage J.J., Zaytoon J., "Comportement temporel du GRAFCET", Journal Européen des Systèmes Automatisés, vol 34, n°4, pp. 695-711, 1997.

[LUT97] Luttenbacher D., "Modélisation du concept capteur intelligent par une approche orientée objet : application à un capteur intelligent de température", Thèse de l'Université Henri Poincarré, Nancy, février 1997.

[MAR90] Maraninchi F., "ARGOS : un langage pour la conception, la description et la validation des systèmes réactifs", Thèse de l'Université Joseph Fourier, Grenoble I, 1990.

[PET62] Petri C.A., "Kommunication mit automaten", Thèse de doctorat, Université de Bonn, 1962.

[PIG91] Pigot T., "Programmation Orientée Objet", INFOPC n°75, pp.213-232, octobre 1991.

[ROU92] Roux O., Creusot D., Cassez F., Elloy J.-P., "Le langage réactif asynchrone ELECTRE", Revue Technique et Science Informatique, AFCET/Hermès, vol 11, n°5, pp. 35-66, 1992.

[RUM95] Rumbaugh J., "O.M.T. : Modélisation et conception orientées objet", Edition Masson et Prentice Hall, Paris, 1995.

[WAR86] Ward P., Mellor S., "Structured Development of Real-Time Systems", Englewood Cliffs, Yourdon Press, 1986.

[WES92] Westbrock J., "Device Description Language Specification", Document COM-20, révision B, avril 1992.

Chapitre III : Le Modèle Externe

une méthode de spécification et de vérification statique
du cahier des charges d'un équipement intelligent

Table des Matières :

III.1- Introduction : 64

III.2- Présentation du Modèle Externe : 64
III.2.1- Utilisateurs : 65
III.2.2- Service Externe : 65
III.2.2.1- Conditions d'Activation : 66
III.2.2.2- Ressources : 67
III.2.2.3- Paramètres : 68
III.2.2.4- Versions de service : 69
III.2.3- Modes d'Utilisation : 70
III.2.3.1- Définition d'un Mode d'Utilisation : 70
III.2.3.2- Changement de Mode d'Utilisation : 71
III.2.4- Modes de Marche : 72

III.3- Une méthode de conception cohérente d'un équipement intelligent : 74
III.3.1- Spécification des besoins dans le cahier des charges : 74
III.3.1.1- Les consommateurs externes : 74
III.3.1.2- Les services externes : 75
III.3.1.2.1- La sortie principale : 75
III.3.1.2.2- Les entrées : 75
III.3.1.2.3- Les paramètres : 77
III.3.1.2.4- Les conditions d'activation : 78
III.3.1.2.4.1- Vérification des conditions d'utilisation : 78
III.3.1.2.4.2- Vérification des requêtes d'activation : 79
III.3.1.2.5- Les ressources : 80
III.3.1.2.5.1- La préemption : 81
III.3.1.2.5.2- Les différents cas de préemption : 82
III.3.1.2.6- La procédure : 82
III.3.2- Analyse de la spécification obtenue : 84
III.3.2.1- Analyse de l'ensemble des services : 84
III.3.2.1.1- Boucles dynamiques : 84
III.3.2.1.1.1- Cas d'un service en boucle sur lui-même : 85

III.3.2.1.1.2- Cas d'une boucle comprenant p services (p ≥ 2) : 86
III.3.2.1.2- Services de paramétrage : 86
III.3.2.2- Formation des Modes d'Utilisation : 88
III.3.2.3- Analyse des Modes d'Utilisation : 92
III.3.2.4- Les Changements de Mode d'Utilisation : 93
III.3.2.4.1- Vérifications préliminaires : 95
III.3.2.4.1.1- Vérification des conditions de transition : 95
III.3.2.4.1.2- Vérification des requêtes de changement de mode : 95
III.3.2.4.1.3- Vérification des commandes : 96
III.3.2.4.2- Vérification de l'automate des modes d'utilisation : 97
III.3.2.4.2.1- Connexité de l'automate des modes d'utilisation : 97
III.3.2.4.2.2- Déterminisme de l'automate des modes d'utilisation : 99
III.3.2.4.2.3- Cohérence des modes d'utilisation : 100

III.4- Résumé de la démarche : 101

III.5- Conclusion : 103

Références : 104

Le Modèle Externe

une méthode de spécification et de vérification statique
du cahier des charges d'un équipement intelligent

III.1- Introduction :

La mise en place du modèle externe d'une application suite à un cahier des charges bien défini s'inspire de la méthode S.A.D.T. (combinaison d'une approche orientée données et d'une approche orientée fonctions) à laquelle on ajoute la partie des problèmes temporels et comportementaux que l'application fait ressortir. De ce fait, le modèle externe fait appel à des techniques voisines de celles utilisées par les méthodes orientées objets. Nous ne gardons pas le principe d'héritage qui, bien que très puissant, ne nous semble pas entièrement approprié dans l'optique actuelle d'implémentation dans les équipements intelligents. La notion de "classe" d'objet sera par contre retenue afin de définir les différents "objets" que nous utiliserons afin de clarifier les termes utilisés lors de la démarche de développement d'un équipement intelligent.

Dans ce chapitre, nous allons premièrement donner les notions de base et le vocabulaire auxquels fait appel le modèle externe. Dans une deuxième partie, nous exposerons par étapes successives comment les différents éléments fournis dans le cahier des charges s'adaptent au modèle externe. Le modèle externe de l'équipement intelligent est alors utilisé à chaque étape afin de vérifier si la spécification est correcte.

III.2- Présentation du Modèle Externe :

Le modèle externe a été développé au L.A.I.L. au sein de l'équipe A.S.P.I.C. (Analyse et Surveillance des Processus Industriels Complexes) dans la branche Instrumentation Intelligente.

Comme le laisse supposer son nom, le modèle externe est un modèle qui s'appuie sur ce que l'équipement intelligent offre comme services vu de l'extérieur, i.e. d'un point de vue des "utilisateurs". On ne regarde donc pas, dans un premier temps, comment l'équipement fait pour offrir ces services aux utilisateurs. Dans le modèle externe, on se doit alors de définir les deux premiers termes que sont les utilisateurs et la notion de service externe.

III.2.1- Utilisateurs :

On définit, comme pour le cahier des charges d'un équipement intelligent, deux types d'utilisateurs. Le premier type d'utilisateur, le plus important, est le **consommateur** qui définit les besoins principaux en termes d'aide aux utilisateurs et de performances attendues. Le deuxième type d'utilisateur est l'**acteur** qui, soit fournit des informations à l'équipement intelligent (une donnée venant d'un autre équipement intelligent, par exemple), soit contraint le fonctionnement de l'équipement, ces contraintes pouvant engendrer des conflits avec d'autres acteurs qu'il faudra alors pouvoir résoudre [ALA89].

Ces deux définitions d'utilisateurs correspondent exactement à celles proposées dans le deuxième chapitre afin de coller au mieux aux besoins exprimés dans le cahier des charges. De même, un consommateur pourra être en même temps un acteur (et vice versa) sans que cela n'engendre d'incohérence dans les spécifications.

III.2.2- Service Externe :

La définition d'un service est donnée ci-dessous [STA94] :

> "En se plaçant d'un point de vue externe à l'équipement, un **service** est défini comme le résultat d'une procédure, à laquelle on peut associer une interprétation en termes fonctionnels."

D'après cette définition, on entend par interprétation en termes fonctionnels que l'exécution de la procédure résulte dans la modification d'au moins une donnée dans la base de données locale de l'équipement, et/ou au moins un signal sur son interface de sortie. La description d'un service consiste premièrement dans la description du résultat que produit son exécution, i.e. les sorties. De manière à définir ces valeurs obtenues, on doit décrire les traitements qui sont effectués (procédures algorithmiques et/ou séquentielles, inférences qualitatives ou floues...) selon d'éventuels paramètres internes, les variables sur lesquelles ils sont calculés (i.e. les entrées) et les ressources requises à l'exécution de la procédure. De plus, avant son exécution, un service doit vérifier des conditions d'activation. Un service peut donc se décrire premièrement par le quintuple :

(Service)::= (Entrées, Sorties, Procédure, Conditions d'Activation, Ressources)

Cette description qui vient d'être donnée peut être facilement résumée par la figure 3.1, ci-après :

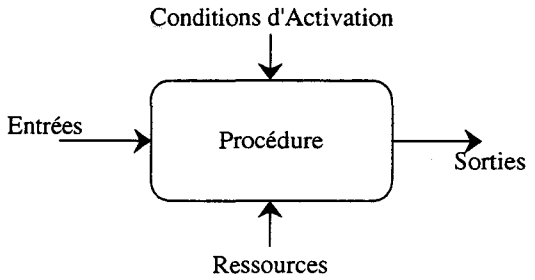


Figure 3.1 : Schéma d'un service.

L'ensemble des Entrées/Sorties de l'équipement intelligent est généralement fourni dans le cahier des charges donné par le client. Les sorties correspondent aux besoins exprimés par les consommateurs. Quant aux entrées, elles correspondent aux données dont la procédure a besoin pour fournir les sorties.

On voit donc malgré tout que le modèle externe nécessite une connaissance minimale du point de vue du fonctionnement interne de chaque service pour pouvoir décrire de manière efficace le point de vue externe. La procédure permettant l'exécution du service sera choisie en fonction des contraintes imposées dans le cahier des charges, i.e. performances statiques et temporelles, conditions de fonctionnement, ...

Deux autres points sont encore à préciser dans la définition d'un service : ses conditions d'activation et ses ressources.

III.2.2.1- Conditions d'Activation :

Les conditions d'activation peuvent encore se scinder en deux sous-groupes. En effet, pour qu'un service s'exécute, il faut qu'un certain nombre de conditions soient remplies et il faut que le service ait été demandé par une requête, qu'elle soit implicite ou explicite (en provenance d'un acteur). On peut donc définir les conditions d'utilisation suivant lesquelles le service peut être exécuté. Elles peuvent inclure des conditions sur les ressources du service. Une condition d'activation est donc constituée de deux éléments :

- une condition d'utilisation qui permet de définir les conditions sous-lesquelles le service peut fonctionner,
- une requête d'activation qui peut être soit implicite, soit explicite et qui spécifie le fait que le service ait été demandé.

Une condition d'utilisation est une expression logique dont le résultat autorise ou non le service à s'effectuer. On a alors deux cas simples :

- l'expression logique est VRAIE, le service est alors exécuté dès que la requête est envoyée par l'acteur adéquat,
- l'expression logique est FAUSSE, le service ne peut alors être exécuté et ce, même si la requête associée est validée, la procédure est alors à déterminer.

A chaque condition d'utilisation la requête associée possède, elle-même, les caractéristiques suivantes [BAY95b] :

- son nom qui permet de l'identifier,
- ses paramètres d'exécution qui permettent de moduler les résultats. L'ensemble de toutes les requêtes paramétrées que l'équipement intelligent reconnaît définit son langage de supervision,
- son origine, qui identifie l'entité qui la produit (opérateur de contrôle ou de maintenance, équipement de supervision, ordinateur de régulation...),
- le **lien de communication** au travers duquel elle est transmise. A cause du montage matériel de l'installation, les requêtes en provenance du système d'automatisation sont toujours transmises par le même médium (le réseau de terrain, par exemple, pour les requêtes faites par le PLC à une vanne intelligente). Par contre, les requêtes venant d'un acteur humain peuvent être transmises par n'importe quel moyen à leur disposition (terminal de poche, réseau de terrain, clavier de contrôle...).

De plus, toute requête est considérée comme n'importe quel événement, i.e. volatile. Afin que l'équipement garde son caractère réactif, chaque requête doit donc pouvoir être prise en compte au moment où cette dernière apparaît : si elle est valide, soit elle est traitée tout de suite, soit l'événement est enregistré afin d'être traité ultérieurement.

L'ensemble des conditions d'activation est donc constitué de l'ensemble des couples :
(Conditions d'Activation)::= ((Condition d'Utilisation, Requête d'Activation))

D'un point de représentation sur le schéma d'un service, on prendra la syntaxe suivante : [Conditions_Utilisation] $\wedge$ Requête.

III.2.2.2- Ressources :

Les ressources de l'équipement intelligent sont l'ensemble des composants matériels et non-matériels qui sont nécessaires à la bonne marche de l'équipement. Pour ce qui relève des composants matériels, on y retrouve tout ce qui a trait à l'énergie consommée

par l'équipement et les pièces vitales de ce dernier. L'ensemble des ressources non-matérielles est plus d'ordre informationnel comme l'absence de la valeur d'une donnée d'entrée par exemple. Tout ce qui relève du partage exclusif de composant, d'espace ou d'information est aussi un problème de ressource. A ce titre, le partage du moyen de calcul (comme le processeur) est un problème de ressource.

Les ressources sont une partie du service qu'il ne faut pas négliger. En effet, l'absence d'une ressource peut rendre le service indisponible temporairement ou de manière définitive en cas de défaillance grave d'une ressource majeure (exemple : coupure d'électricité).

III.2.2.3- Paramètres :

Nous introduisons, dans le service, un élément supplémentaire : l'ensemble des paramètres. Un paramètre est une donnée semi-statique du service qui sert généralement à le configurer. Dans ce cadre, les paramètres occupent une place à part dans le service. En effet, les paramètres sont le plus souvent nécessaires à la bonne marche du service : un paramètre est à la fois une entrée, dans la mesure où la valeur contenue dans le paramètre sert à la procédure fournissant une des sorties, et une ressource informationnelle, par le fait que l'absence du paramètre peut rendre le service indisponible. En ce sens, il existe le plus souvent une condition sur chaque paramètre du service dans les conditions d'utilisation du service.

L'existence de paramètre dans un service n'est, naturellement, pas nécessaire. Un paramètre est, par contre, toujours une des sorties d'un autre service qui permet de le modifier : la modification d'un paramètre se fait sur l'initiative d'un acteur intervenant par l'intermédiaire d'un service de modification de paramètre.

Un service externe s_i est donc caractérisé par :

- Entrées(s_i),
- Sortie(s_i),
- Procédure(s_i),
- Ressources(s_i),
- Paramètres(s_i),
- Conditions d'Activation(s_i)
  - Conditions d'Utilisation,
  - Requête d'Activation.

Le service peut maintenant être représenté par le schéma suivant :

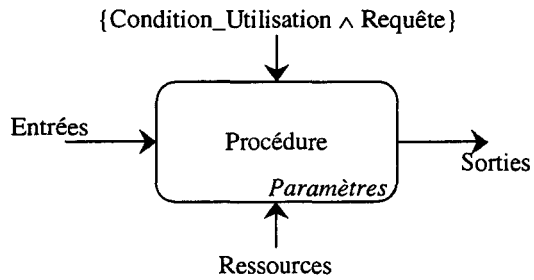


Figure 3.2 : Schéma complet d'un service

III.2.2.4- Versions de service : [STA95] [BOU97]

Ce que nous venons de décrire pour le service est valable pour le cas où les ressources du service sont toutes disponibles. Dans ce cas, on dit que le service est *nominal*. Cependant, il est possible qu'une ou plusieurs ressources soient défaillantes, ou qu'une des variables consommées soit incorrecte. On distingue alors deux situations possibles :

- Le cahier des charges n'indique pas qu'il faille un traitement de remplacement capable d'assurer le service. Le service devient alors indisponible en cas de défaillance d'une de ces ressources.
- Le cahier des charges stipule qu'il faut prévoir un traitement substitutif n'utilisant pas le même ensemble de ressources que celui de la version nominale. Dans le cas où ce traitement substitutif dispose effectivement des ressources qui lui sont nécessaires, il est exécuté en lieu et place du traitement nominal lorsque ces conditions d'activations propres sont réunies. On dit alors que le service est *dégradé*.

Il peut exister plusieurs versions dégradées correspondant chacune à un ensemble de ressources disponibles. Lorsque plus aucune version dégradée n'est disponible (i.e. il y a un trop grand nombre de défaillances), le service devient indisponible. On obtient alors pour chaque service un graphe de fonctionnement interne en fonction des ressources disponibles. De plus, cette possibilité de service dégradé laisse supposer qu'il existe un moyen de détection des défaillances des ressources du service.

Ainsi, l'introduction de versions dégradées augmente la disponibilité du service en le rendant plus robuste face à la défaillance d'une partie de ses ressources.

III.2.3- Modes d'Utilisation :

III.2.3.1- Définition d'un Mode d'Utilisation :

L'ensemble formé par tous les services caractérise, d'un point de vue externe, ce que fournit globalement l'équipement à son environnement. On obtient, comme en informatique, un ensemble de procédures (un ensemble d'instructions) qui seraient disponibles pour un utilisateur externe qui enclencherait les différentes opérations à effectuer. Cependant, on peut remarquer que tous les services ne sont ou ne peuvent pas être activés en même temps. En effet, on peut observer qu'il existe un certain nombre de services qui sont toujours activables ensemble, que d'autres sont incompatibles dans certaines situations du cycle d'exploitation de l'équipement, et qu'enfin, certains services ne peuvent être actifs que si d'autres l'ont été auparavant.

Si on prend le cas d'un actionneur, les services concernant la conduite ne peuvent être exécutés que si le service de paramétrage associé a été effectué avec succès au moins une fois auparavant. Parallèlement, il n'est pas toujours approprié qu'un service de maintenance soit actif en même temps qu'un service de conduite automatique.

On peut intégrer toutes ces conditions au niveau des conditions d'utilisation des services. Cependant, pour des équipements de taille importante au niveau des services fournis, cela peut aboutir à des formules assez complexes difficilement gérables en ligne. Une des solutions consiste alors à organiser l'ensemble des services en sous-ensembles de services cohérents entre eux. Nous appelons ces sous-ensembles les modes d'utilisation.

On donne au mode d'utilisation la définition suivante [GEH94] :

"Un Mode d'Utilisation (MU) est un sous-ensemble de l'ensemble des services de l'équipement intelligent. Un mode d'utilisation comprend au moins un service et chaque service appartient à au moins un mode d'utilisation."

La figure 3.3 montre l'ensemble des services et des modes d'utilisation.

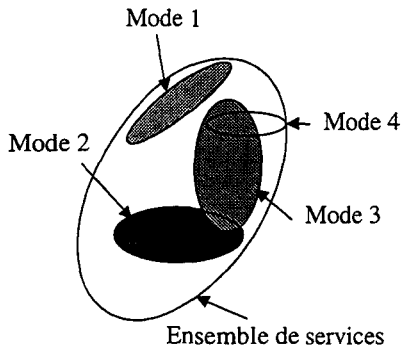


Figure 3.3 : Mode d'Utilisation et ensemble des services.

L'intérêt des modes d'utilisation est que l'équipement ne peut se trouver que dans un seul et unique mode d'utilisation à un moment donné ce qui réduit les ambiguïtés d'interprétation des situations possibles.

III.2.3.2- Changement de Mode d'Utilisation :

Une fois défini, chaque mode d'utilisation représente un contexte d'utilisation particulier de l'équipement : chaque contexte donne l'ensemble des services qu'il est possible d'exécuter, sous certaines conditions, à un moment donné.

Dans le cadre général, il est clair que l'équipement ne pourra pas se trouver, dans son cycle d'exploitation, dans un seul mode d'utilisation. Il faut prévoir l'ensemble des conditions sous lesquelles on change d'un mode d'utilisation, i.e. définir les différents changements de mode d'utilisation.

Comme le précise P. Biland [BIL94], on peut distinguer deux types de transition dans les changements de mode d'utilisation : les transitions forcées par les acteurs et les transitions spontanées qui sont dues à l'état du système. On retrouve aussi, ici, une analogie avec les Conditions d'Activation des services : une condition de changement de mode est alors constituée d'une condition de transition et d'une requête implicite ou explicite de changement de mode. Toutes les requêtes ont la même structure que celle qui a été définie précédemment pour les services externes (§III.2.2.1). On peut distinguer les deux cas précédents lorsque les conditions de transitions sont validées :

- si la requête de changement de mode est implicite, le changement de mode se fera automatiquement dès l'apparition de la condition de transition : on a un changement de mode spontané ;
- si la requête est explicite et provient d'un acteur extérieur à l'équipement, le changement de mode est forcé.

De plus, pour chaque transition existante entre deux modes, il faut définir les requêtes qui peuvent être lancées au moment du changement de mode. On peut par exemple avoir une remise à zéro d'une variable ou encore lancer une requête qui permet de demander un service particulier à l'entrée dans le mode de destination. On ajoute donc la ou les commandes qui sont lancées lors du changement de mode.

Un changement de mode, forcé ou spontané, est donc constitué de :

- une condition de transition qui est une condition logique à respecter pour permettre le changement de mode,
- une requête de changement de mode qui peut être soit implicite, soit explicite,
- un ensemble de commande à effectuer lors du changement de mode.

Une fois l'ensemble des changements de modes d'utilisation défini, on obtient directement l'automate fini déterministe du comportement statique de l'équipement intelligent, où les états de l'automate sont les modes d'utilisation et où les transitions entre les différents états sont les changements de mode d'utilisation précédemment définis.

On obtient ainsi un graphe des modes d'utilisation comme le montre la figure 3.4 :

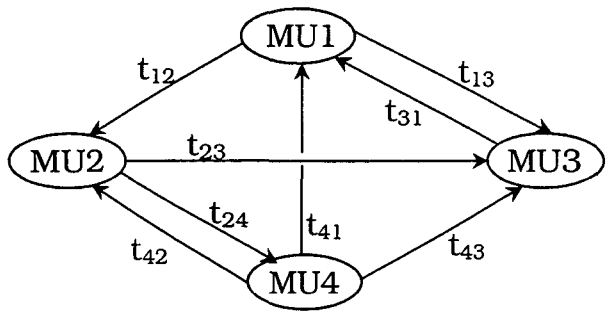


Figure 3.4 : Exemple d'un graphe des modes d'utilisation.

III.2.4- Modes de Marche :

Les modes de marche tel que le propose le modèle fourni par le GEMMA donné par l'ADEPA (Agence nationale pour le DEveloppement de la Production Automatisée) [ADE81] sont semblables à ce que nous avons défini comme mode d'utilisation. D'autres auteurs se sont inspirés cette même approche pour définir les modes de marche d'un système automatisé de production [BOI91][BIL94]. Nous voulons faire ici la différence entre les états possibles qui sont voulus (forcés) par les différents acteurs de

l'équipement et l'état subi [BIL94] par l'équipement en fonction de l'état des différentes ressources sur lesquelles il s'appuie pour parvenir à fournir ses services.

Nous l'avons déjà vu, un mode d'utilisation est un contexte possible d'utilisation de notre équipement. On définira donc un mode de marche comme suit :

> "Un Mode de Marche est la conjonction d'un Mode d'Utilisation d'un équipement et de l'ensemble des ressources disponibles pour chaque service du mode d'utilisation considéré."

Cette définition laisse supposer qu'il existe un moyen de détection des défaillances des ressources des services dans le mode d'utilisation. La "disparition" de certaines ressources peut entraîner l'indisponibilité d'un service. Ce dernier sera suivi d'un basculement du mode d'utilisation courant dans un autre mode d'utilisation qui pourra alors être qualifié de dégradé, comme proposé dans le GEMMA. Cependant la disparition de certaines ressources n'entraînant pas forcément l'indisponibilité d'un service (grâce aux versions dégradées de service) et un changement de mode d'utilisation, la définition des modes de marche pour chaque mode d'utilisation reste donc cohérente. La figure 3.5 ci-dessous donne un exemple de modes de marche dans un mode d'utilisation.

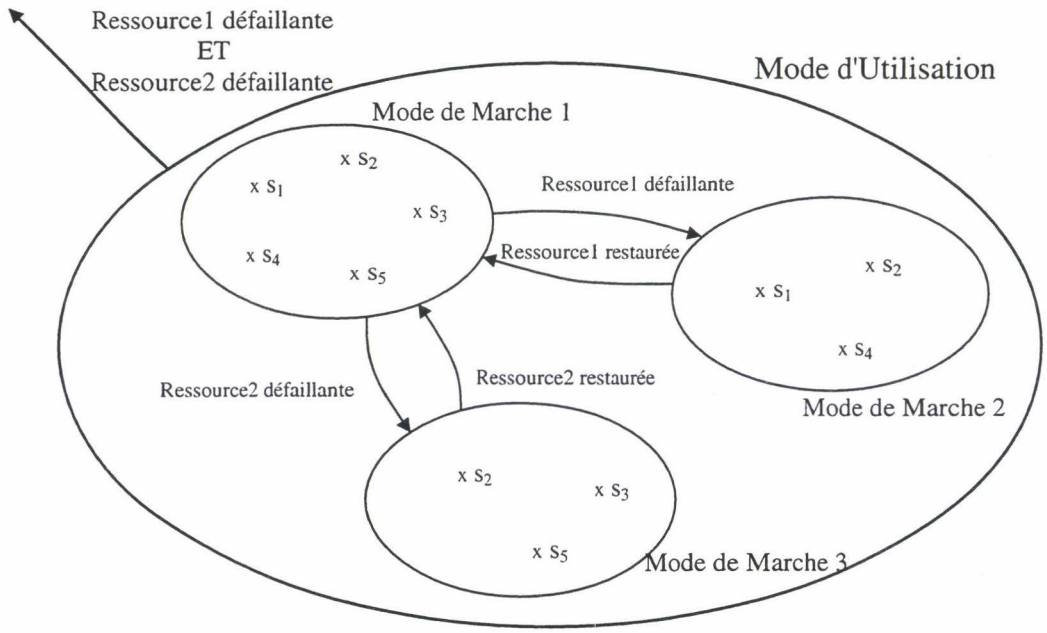


Figure 3.5 : Exemple de Modes de Marche dans un Mode d'Utilisation

III.3- Une méthode de conception cohérente d'un équipement intelligent :

Si l'erreur est humaine, la machine, elle, n'en commet jamais : elle fera fidèlement ce qu'on aura demandé de faire. Si ce qu'on a demandé de faire à la machine a été mal conçu, la faute n'en revient finalement qu'à l'homme. Or l'homme ne peut pas être parfait, il fait donc parfois des erreurs, le plus souvent inconsciemment. Pour pallier cet état de fait dans le cadre de la conception des systèmes automatisés, on se propose de donner une méthode permettant d'interpréter le cahier des charges donné avec le modèle externe et d'en vérifier la cohérence tout au long de son évolution. Dans chacune des étapes que nous allons développer nous introduirons un ensemble de notations qui seront utilisées tout au long de la démarche.

Notre démarche sera détaillée par étapes qui sont celles qui ont été données au niveau des cycles de développement dans le chapitre I.

III.3.1- Spécification des besoins dans le cahier des charges :

L'ensemble des besoins a été énoncé dans le cahier des charges fourni par le client. Il faut à présent faire ressortir toutes les données correspondantes au modèle externe afin de pouvoir passer à une phase d'analyse des spécifications qui est la phase suivante. La spécification des besoins est le fait de la définition de l'ensemble des services externes.

III.3.1.1- Les consommateurs externes :

L'ensemble des consommateurs extérieurs au futur équipement doit être fourni par le cahier des charges. Ces consommateurs sont les intervenants extérieurs qui sont les demandeurs des différentes aides que leur fournira l'équipement. On retrouve parmi eux plusieurs opérateurs et divers autres systèmes automatisés dont certains autres équipements intelligents. Soit Uc, l'ensemble des consommateurs des services de l'équipement :

$$Uc = \{c_k, k \in K\}$$

A chaque consommateur potentiel c_k, il faut associer l'ensemble des données qu'il consomme. L'ensemble de toutes les données consommées par tous les consommateurs de l'équipement fournira ce que nous appellerons les *Sorties Principales*. Soit D, l'ensemble de ces sorties :

$$D = \{d_h, h \in H, \exists!i \in I / d_h \in Sortie(s_i)\}$$

On définit la fonction fc qui associe à chaque consommateur c_k de Uc l'ensemble des données de l'ensemble D qu'il consomme :

$$fc : Uc \rightarrow D$$
$$c_k \rightarrow fc(c_k)$$

A chaque Sortie Principale, on associe un seul et unique service externe qui devra alors fournir cette sortie. Il faut donc définir pour chaque service externe l'ensemble des informations nécessaires à sa description. Ainsi, un service correspondra à une aide fournie aux différents utilisateurs que sont les consommateurs, conformément à ce que le cahier des charges exprime.

Comme à chaque sortie principale on associe un service, on définit S, l'ensemble des services que l'équipement est capable de rendre :

$$S = \{s_i, i \in I\}$$

III.3.1.2- Les services externes :

On reprend alors la description d'un service du modèle externe qui a été donnée au paragraphe III.2.2. pour caractériser chaque service différencié lors de l'étape précédente.

III.3.1.2.1- La sortie principale :

Il s'agit ici de donner le format de la sortie principale comme il a été précisé dans le deuxième chapitre au paragraphe I.4.2.2.2. ainsi que les performances tant temporelles que comportementales. Ces diverses informations permettent alors de déterminer quel type de procédure pourra être utilisé afin de fournir la sortie principale du service. Les contraintes environnementales, s'il y en a, restreindront alors le choix et il sera possible de fournir les autres informations relatives au service.

III.3.1.2.2- Les entrées :

La procédure du service a besoin d'un certain nombre d'entrées qui lui sont nécessaires afin de fournir la sortie principale. Ces entrées doivent être recensées afin de vérifier si elles sont produites par "quelqu'un".

On définit pour cela E, l'ensemble de toutes les entrées de tous les services :

$E = \{e_h, h \in H, \exists i \in I / e_h \in \text{Entrées}(s_i)\}$

Deux cas sont possibles pour chaque élément de E :

- L'entrée e_h en question est la sortie principale d'un service déjà défini, il n'y a alors pas de problème.
- L'entrée e_h n'est la sortie principale d'aucun service défini dans le cahier des charges. Il faut alors déterminer qui produit cette donnée :

• Le cahier des charges stipule que e_h est fournie par l'extérieur (l'environnement physique, un autre système automatisé, ...), on définit alors cet *acteur externe.*

Soit Ua, l'ensemble des acteurs externes qu'on définit ici :

$Ua = \{a_k, k \in K\}$

On définit ici aussi la fonction fa qui associe à chaque acteur externe a_k de Ua l'ensemble des données de l'ensemble E qu'il fournit :

$fa : Ua \rightarrow E$

$a_k \rightarrow fa(a_k)$

• Le cahier des charges ne stipule rien à propos de cette donnée, il faut alors déterminer si c'est un acteur externe qui fournit e_h qui devient alors un nouveau besoin de l'équipement (i.e. une donnée qui doit être fournie à l'équipement afin qu'il fonctionne correctement), ou si un nouveau service externe de l'équipement doit fournir cette donnée. Dans ce dernier cas, il faut définir un nouveau consommateur externe (ou compléter l'ensemble des données consommées par un consommateur existant) afin de prendre en compte cette nouvelle donnée.

On a alors pour l'ensemble des entrées l'expression suivante :

$\forall e_h, h \in H$, on a $\exists i \in I / e_h \in \text{Entrées}(s_i)$ OU $\exists k \in K / e_h = fa(a_k)$

Dans le cas particulier où le service consomme en entrée sa propre sortie principale, le service est paramétrable puisqu'il faudra au moins lui fournir ses valeurs par défaut au démarrage du service.

III.3.1.2.3- Les paramètres :

Si des données concernant le fonctionnement interne sont modifiables au cours du cycle de vie de l'équipement, le service est paramétrable. De même que pour les entrées, il faut expliciter l'ensemble des paramètres pour chaque service et pour l'ensemble de l'équipement.

Pour chaque paramètre explicité, il faut de même vérifier qu'il est produit par "quelqu'un" avec la restriction qu'un service ne peut pas être l'acteur d'un de ses propres paramètres de fonctionnement.

A ce titre, on définit l'ensemble P qui contient l'ensemble de tous les paramètres de l'équipement :

$P = \{p_h, h \in H, \exists i \in I / p_h \in \text{Paramètres}(s_i)\}$

Deux cas sont possibles pour chaque paramètre p_h :

- Le paramètre p_h en question est la sortie principale d'un service externe déjà défini et dont il n'est pas l'un des propres paramètres, il n'y a alors pas de problème.
- Le paramètre p_h n'est la sortie principale d'aucun service défini dans le cahier des charges. Il faut alors déterminer qui le produit : le cahier des charges ne stipule rien à propos de cette donnée, il faut définir un nouveau consommateur externe de "type" *paramétreur* (ou compléter l'ensemble des données consommées par un consommateur existant) afin de prendre en compte le paramètre.

On vérifie alors l'équation suivante :

$\forall h \in H, \exists i \in I / p_h = \text{Sortie}(s_i)$

Une fois encore, chaque service ne possède pas forcément de paramètre de fonctionnement. Ce type d'information n'est donc pas nécessaire à tous les services de l'équipement intelligent. Cependant, si un service est paramétrable, on peut souvent noter qu'un tel service sera indisponible tant que le service qui lui fournit ses paramètres n'aura pas été exécuté au moins une fois auparavant, à moins qu'à la conception on ait affecté aux paramètres des valeurs par défaut.

III.3.1.2.4- Les conditions d'activation :

Les conditions d'activation de chaque service doivent se traiter par couple (Condition d'Utilisation, Requête d'Activation). Il faut donc définir tous les couples qui entraînent le fonctionnement du service avec, cependant, les deux restrictions suivantes :

- Une condition d'utilisation est une conjonction de variables booléennes.
- Il existe deux types de requête :
  - *implicite*, i.e. dès que la condition d'utilisation est VRAIE le service s'exécute, une requête implicite équivaut alors à un "1" logique et sera noté tel quel,
  - *explicite*, dans les autres cas, il faut alors nommer la requête comme tout autre donnée de l'équipement.

Une fois que tous les couples (Conditions_Utilisation ; Requête d'Activation) de tous les services de l'équipement ont été déterminés, on peut recenser deux ensembles :

- l'ensemble de toutes les variables booléennes qui forment les diverses Conditions d'Utilisation des services de l'équipement,
- l'ensemble de toutes les requêtes explicites d'activation des services (i.e. différentes de "1").

Soit Ba, l'ensemble de toutes les variables booléennes intervenant dans les conditions d'activation de l'ensemble des services S :

$$Ba = \{b_h, h \in H / \exists i \in I, \exists q \in N, b_h \subset Conditions_Utilisation_q(s_i)\}$$

et soit Ra, l'ensemble des requêtes d'activation explicites intervenant dans les conditions d'activation de l'ensemble des services S :

$$Ra = \{r_h, r_h \neq 1, h \in H / \exists i \in I, \exists q \in N, r_h = RequêteA_q(s_i)\}$$

III.3.1.2.4.1- Vérification des conditions d'utilisation :

Pour chaque élément de Ba, il faut vérifier qu'il est produit par "quelqu'un". On a alors deux cas principaux qui peuvent survenir :

- L'élément b_h est la sortie principale d'un service externe déjà existant, on n'a alors aucun problème.
- L'élément b_h n'est la sortie principale d'aucun service défini dans le cahier des charges. Il faut alors déterminer qui produit cette donnée :

• Le cahier des charges stipule que la donnée b_h est fournie par l'extérieur, on définit alors cet *acteur externe* à Ua ou on complète les données fournies par un acteur a_k de Ua déjà défini.

• Le cahier des charges ne stipule rien à propos de b_h, il faut alors déterminer qui la fournit. On a alors trois solutions envisageables :

· un acteur externe qu'il faut définir à Ua ou on complète les données fournies par un acteur a_k de Ua déjà défini,

· la donnée b_h devient une des sorties secondaires d'un service externe existant (par exemple, un service calculant une valeur peut déterminer si cette valeur a dépassé un seuil),

· définir un nouveau consommateur externe (ou compléter l'ensemble des données consommées par un consommateur existant) afin de prendre en compte la donnée b_h qui deviendra une sortie principale.

III.3.1.2.4.2- Vérification des requêtes d'activation :

De même, pour chaque élément de Ra, il faut vérifier qu'il est produit par "quelqu'un". On a alors deux cas principaux qui peuvent survenir :

- L'élément r_h est la sortie principale d'un service externe déjà existant, on n'a alors aucun problème.

- L'élément r_h n'est la sortie principale d'aucun service défini dans le cahier des charges. Il faut alors déterminer qui produit cette donnée :

• Le cahier des charges stipule que la donnée r_h est fournie par l'extérieur, on définit alors cet *acteur externe* à Ua ou on complète les données fournies par un acteur a_k de Ua déjà défini.

• Le cahier des charges ne stipule rien à propos de r_h, il faut alors déterminer qui la fournit. On a alors encore une fois trois solutions envisageables :

· un acteur externe qu'il faut définir à Ua ou on complète les données fournies par un acteur a_k de Ua déjà défini,

· la donnée r_h est une des sorties secondaires d'un service externe existant (par exemple, un service calculant une valeur peut déterminer si cette valeur a dépassé un seuil),

· définir un nouveau consommateur externe (ou compléter l'ensemble des données consommées par un consommateur existant) afin de prendre en compte la donnée r_h qui deviendra une sortie principale.

On peut, à la suite de ces diverses attributions, réorganiser l'ensemble des sorties de chaque service s_i. Chaque Sortie(s_i) peut alors se diviser en deux sous-groupes : Sortie_P(s_i) et Sorties_S(s_i). Et nous modifions l'ensemble D comme il suit :

$$D = \bigcup_{i \in I} \{Sortie_P(s_i), Sorties_S(s_i)\}$$

La définition de l'ensemble des conditions d'activation, i.e. de l'ensemble des couples (Condition d'Utilisation, Requête d'Activation) est sans doute une des phases les plus délicates et souvent sujette à interprétation des données du cahier des charges de la part du concepteur.

III.3.1.2.5- Les ressources :

On doit aussi s'attacher aux ressources nécessaires à la bonne marche de chacun des services. On s'intéresse plus particulièrement à deux caractéristiques que peuvent posséder les ressources d'un service :

- les ressources *vitales* : ce sont les ressources qui sont absolument nécessaires à l'exécution du service. Leur absence rend l'exécution du service et/ou sa poursuite impossible.
- les ressources *partagées* : ce sont les ressources qui sont utilisables par plusieurs services. Un cas courant de ressource partagée est celui du processeur.

L'ensemble des ressources de chaque service s_i est donc "décomposé" en quatre groupes non forcément distincts comme l'explicite la figure 3.6 ci-dessous :

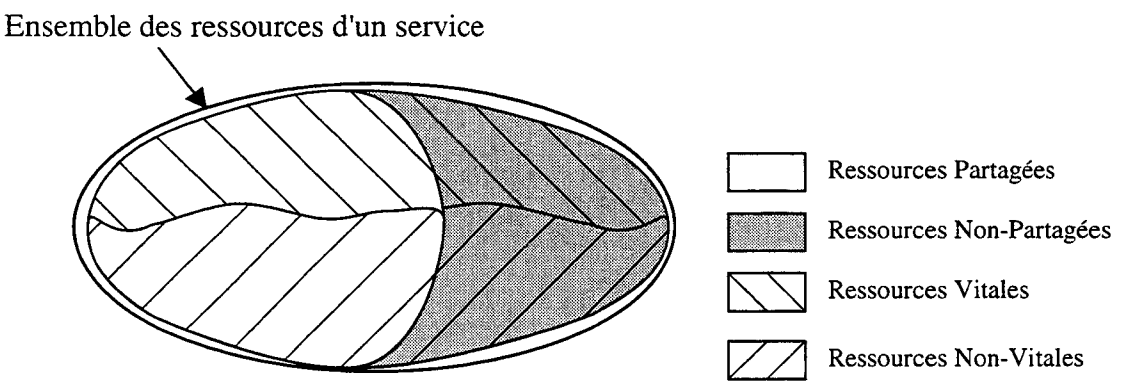


Figure 3.6 : Les ensembles des ressources

Ressources(s_i) = (Ressources_V(s_i) ; Ressources_P(s_i) ; Ressources_NV(s_i) ; Ressources_NP(s_i))

L'expression des ressources fait ressurgir un problème intéressant au niveau de l'exécution de chacun des services : comment traiter les cas de manquement d'une ressource au niveau d'un service en cours de fonctionnement ? On s'intéresse là au problème de la préemption.

III.3.1.2.5.1- La préemption :

La définition couramment donnée pour le droit de préemption est la suivante [LAR98] :

> "Faculté que détient une personne ou une administration, de préférence à tout autre, d'acquérir un bien qui a été mis en vente aux prix et conditions de la cession envisagée."

C'est une notion de droit juridique. Mais cette notion a été reprise en informatique surtout au niveau du processeur qui est une des pièces maîtresses de tout système informatique.

On peut recenser deux types de traitement de la préemption. Le premier est la suspension où l'on met en attente l'activité du processus en cours, celui-ci reprend son activité soit par la disparition du signal (requête) qui suspend le processus, soit suite à l'émission d'une requête spécifique. La deuxième forme de préemption est l' "avortement" où le processus en cours est arrêté de façon définitive (on le "tue"). Dans ce dernier cas, on a encore deux traitements de l'avortement : soit on laisse au processus exécuter l'instruction en cours, on a alors un "abandon faible", ou alors on ne lui laisse pas le temps de terminer l'instruction (ou le groupe d'instructions) en cours et on a alors affaire avec un "abandon fort" (exemple: ^C sous UNIX) [BER93].

La préemption en informatique est le plus souvent associée à l'accès de ce que nous appelons un service à la ressource partagée "processeur", mais elle peut être étendue à toute autre ressource partagée que pourrait posséder un service. On fera aussi la distinction entre la ressource partagée "accès au processeur" qui détermine qui, en fonction de sa *priorité*, a accès au processeur, et la ressource "fonctionnement du processeur" qui, elle, est généralement vitale puisqu'un processeur défaillant rend beaucoup de services indisponibles.

La *priorité* de chaque service est un nombre qui détermine quel service est prioritaire par rapport aux autres. Par convention, plus le chiffre est petit, plus le service est

prioritaire. Les priorités servent alors à déterminer quel ordonnancement pourra être pris pour l'équipement [JOS96].

III.3.1.2.5.2- Les différents cas de préemption :

Dans le cas qui nous intéresse, nous nous occupons des ressources partagées par service. Nous traitons donc en fonction de la propriété de partage :

- Si une ressource est non-partagée, elle appartient uniquement à ce seul service et ne pose alors aucun problème dans le cas de la préemption.
- Si une ressource est partagée, elle peut éventuellement être retirée au service qui la détient par un service de priorité supérieure. Il faut déterminer le type de préemption qu'on choisit par rapport à deux informations : la ressource en question et le service prioritaire qui demande la ressource. Suivant ces deux informations, soit on abandonne le service, soit on le suspend momentanément si son traitement le permet et si cela reste cohérent. Dans le cas d'un abandon, il faut encore déterminer si l'abandon est immédiat (abandon fort) ou si on laisse la dernière action s'effectuer (abandon faible).

III.3.1.2.6- La procédure :

Le cas de la procédure est omniprésent dans ce que nous avons vu précédemment. Même si elle n'est pas bien déterminée, le type de procédure définit bien des données précédentes : les entrées, les ressources et les conditions d'utilisation. Pour une description de chaque service, on peut ajouter les informations suivantes :

- la méthode utilisée,
- la priorité du service,
- le temps d'exécution au plus tôt.

Ces informations dépendent fortement de l'architecture matérielle choisie et ne sont donc pas toujours disponibles. Le temps d'exécution au plus tard, quant à lui, ne peut pas être déterminé à ce stade de la conception. On peut ajouter les informations suivantes :

- la périodicité du service et sa période s'il est périodique,
- si le service est préemptif.

On obtient ainsi pour chaque service s_i l'ensemble des informations suivantes :

- les données de sortie du service s_i : Sortie(s_i) = (Sortie_P(s_i) ; Sorties_S(s_i)),
- les données d'entrée du service s_i : Entrées(s_i),
- les paramètres du service s_i : Paramètres(s_i),
- les ressources du service s_i :

  Ressources(s_i) = (Ressources_V(s_i) ; Ressources_P(s_i) ; Ressources_NV(s_i) ; Ressources_NP(s_i)),
- les conditions d'activation du service s_i : Conditions_Activation(s_i) formées de couples {(Conditions_Utilisation$_q$(s_i) ; Requête$_q$(s_i))}, où q est un entier naturel permettant une numérotation croissante des différents couples.
- la procédure du service s_i contenant les champs suivants :
  - la méthode utilisée,
  - la priorité du service,
  - le temps d'exécution au plus tôt,
  - la périodicité éventuelle du service s_i
    - la période du service s_i le cas échéant,
  - la préemptabilité du service s_i.

A ces informations fournies sur l'ensemble des services, on peut ajouter les ensembles non encore exhaustifs Ua et Uc des acteurs et consommateurs externes à l'équipement. Ces ensembles sont enrichis au fur et à mesure de la spécification du cahier des charges à la suite de la spécification des ensembles des entrées, paramètres, conditions d'utilisation et requêtes d'activation de tous les services.

Les ensembles S et Ua des services et des acteurs de l'équipement ainsi obtenus sont, a priori, complets du point de vue des besoins de l'ensemble des consommateurs Uc défini jusqu'à présent. On a défini ces différents ensembles de manière descendante à partir des besoins des différents consommateurs, ainsi, de manière individuelle, chaque service est cohérent vis-à-vis de ses propres consommateurs et acteurs. Ainsi, les vérifications de la spécification des services peuvent être facilement informatisées. A ce titre, un outil de vérification et de validation de la spécification a été programmé sous Excel97 (Microsoft©) dont le détail est fourni en Annexe I.

Cependant, la cohérence de chacun des services par rapport à l'ensemble n'est pas encore prouvée. C'est ce que nous allons voir dans l'analyse de la spécification que nous venons de déterminer.

III.3.2- Analyse de la spécification obtenue :

La démarche précédente nous assure qu'il existe bien un consommateur pour chaque sortie de chaque service et que toutes les entrées de tous les services ont un seul et unique acteur, i.e. un des services de l'équipement ou acteur externe. De par cette démarche, la correction et l'utilité de la spécification sont recherchées au maximum. De plus, leur non-ambiguïté y est trouvée en évitant au maximum les problèmes de fautes de saisie.

L'analyse de la spécification des ensembles précédemment obtenus va permettre de résoudre une partie des problèmes qui se posent dans la cohérence de l'équipement associé. Dans un premier temps, on regardera la cohérence de chacun des services vis-à-vis de l'ensemble des autres puis on regardera la cohérence de sous-ensembles de services entre eux afin de former ce qui a été appelé les modes d'utilisation. Ces modes d'utilisation peuvent ou non avoir été proposés par le client. Une fois la cohérence prouvée, la complétude des spécifications fonctionnelles pourra alors être établie.

III.3.2.1- Analyse de l'ensemble des services :

On a obtenu un ensemble de services qui peuvent avoir, malgré tout, des incohérences entre eux. Il y a plusieurs choses à vérifier :

- la présence de boucles dynamiques entre plusieurs services de la spécification,
- le cas des services de paramétrage et les conditions d'activation des services utilisant ces paramètres.

III.3.2.1.1- Boucles dynamiques :

Une boucle dynamique s'exprime par le fait que des services fonctionnent en "boucle". Par exemple, un service s_1 produit une sortie d_1 qui est consommée par le service s_2 qui lui-même produit la sortie d_2 qui est à son tour utilisée en entrée du service s_1, on a alors une boucle dynamique entre les services s_1 et s_2 comme le montre la figure 3.7.

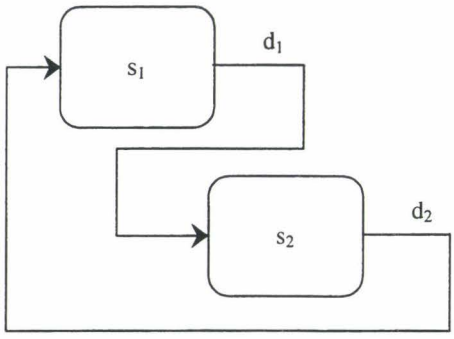


Figure 3.7 : deux services en boucle dynamique

Au niveau même de l'équipement, cette phase ne semble pas forcément nécessaire car il est assez rare que des boucles apparaissent. Cependant, pour des ensembles de services importants ou lors d'une agrégation de plusieurs équipements intelligents ayant été conçus séparément, des boucles dynamiques peuvent exister sans que le concepteur s'en soit rendu compte. C'est pourquoi on recherche leur existence éventuelle.

Pour pouvoir vérifier l'existence possible de telles boucles, nous formons un tableau avec les services en lignes et en colonnes que nous appellerons Boucle. Il est construit comme suit :

Boucle $^{1}_{ij}$ = 1, si le service s_i consomme une donnée produite par le service s_j :

$\exists i \in I, \exists j \in I$ / Sortie(s_j) $\cap$ Entrées(s_i) $\neq \phi$

Boucle $^{1}_{ij}$ = 0, sinon.

Tableau *Boucle* : tableau de précédence en termes de données consommées

Services \ Services	s_1	...	s_j	...
s_1				
⋮			.....	
s_i			0 ou 1	
⋮				

III.3.2.1.1.1- Cas d'un service en boucle sur lui-même :

Le fait qu'un service s_i soit lui-même dynamique se retrouve dans ce tableau. En effet, si Boucle $^{1}_{ii}$ = 1, alors le service s_i consomme une de ses propres sorties. Bien que ce problème ait déjà été traité précédemment (cf. § III.3.1.2.2 et III.3.1.2.3), il peut être

nécessaire de se soucier du fait que des paramètres par défaut doivent être définis dans le cas du démarrage du service. Ceci peut être fait de trois façons :

- figée matériellement, par exemple inscrit dans une EPROM,
- un acteur externe agit sur les paramètres, via un potentiomètre par exemple
- à l'aide d'un service de paramétrage qu'on doit définir.

III.3.2.1.1.2- Cas d'une boucle comprenant p services ($p \geq 2$) :

On a vu le cas d'une boucle dynamique très simple. Pour vérifier s'il existe d'autres boucles, on se propose de calculer progressivement la fermeture transitive idempotente du tableau Boucle. Les vérifications à faire deviennent alors les suivantes :

Pour chaque $p \in \aleph$, $p \geq 2$
si $\exists i \in I$ / $Boucle_{ii}^{(p)} = 1$

Alors il existe une boucle dynamique comprenant le service s_i et pouvant contenir jusqu'à p services différents (s_i compris).

Il faut alors déterminer tous les chemins hamiltoniens de longueur maximale p+1 partant du service s_i et y revenant en passant par le service s_j. Grâce à cet ensemble de chemins, on peut déterminer les ensembles de services liés entre eux par une ou plusieurs boucles dynamiques.

Pour chaque ensemble de services trouvé, on a la propriété que si l'un des services de l'ensemble appartient à un mode d'utilisation m_k, alors tous les autres doivent aussi y appartenir.

La vérification de l'existence de services liés par une boucle dynamique permet alors de compléter éventuellement les ensembles S et Ua dans le cas de la définition de paramètres d'exécution par défaut.

III.3.2.1.2- Services de paramétrage :

On vient de compléter l'ensemble des services qui modifient les paramètres de certains services. Ce qui caractérise les services de paramétrage est qu'il faut qu'ils aient été exécutés au moins une fois pour que les services dont ils modifient les paramètres puissent fonctionner.

Afin de s'affranchir d'une certaine cohérence, il faut éviter des services de paramétrage croisés. Il s'agit, en fait, d'un problème équivalent aux services en boucle dynamique mais au niveau des paramètres. Le problème, dans ce cas, est que chacun des services attend l'autre pour pouvoir s'exécuter et ne s'exécute donc jamais.

De la même manière, afin de vérifier l'existence possible de tels croisements de paramétrage, nous formons un tableau avec les services en lignes et en colonnes que nous appellerons Paramétrage. Il est construit comme suit :

Paramétrage $^{1}_{ij} = 1$, si le service s_i est paramétré par le service s_j :

$\exists i \in I, \exists j \in I$ / Sortie(s_j) $\cap$ Paramètres(s_i) $\neq \phi$

Paramétrage $^{1}_{ij} = 0$, sinon.

Tableau *Paramétrage* : tableau des services paramétrés par d'autres services.

Services \ Services	s_1	...	s_j	...
s_1				
⋮			.....	
s_i			0 ou 1	
⋮				

Bien que la démarche précédente doive nous en préserver, le fait qu'un service s_i se paramètre lui-même se retrouve dans ce tableau. En effet, si Paramétrage $^{1}_{ii} = 1$, alors le service s_i est paramétré par une de ses propres sorties. Il s'agit là d'une erreur de conception qui peut être néfaste dans le cadre de la sûreté de fonctionnement dans la phase d'exploitation de l'équipement.

Pour vérifier qu'il n'existe pas de paramétrage croisé entre les services, on se propose, de la même manière, de calculer progressivement la fermeture transitive idempotente du tableau Paramétrage. Les vérifications à faire deviennent alors les suivantes :

Pour chaque p,

si $\exists i \in I$ / Paramétrage $^{(p)}_{ii} = 1$

Alors il existe un paramétrage croisé comprenant le service s_i et pouvant contenir jusqu'à p services différents (s_i compris).

S'il existe de tels croisements au niveau du paramétrage, la spécification doit être revue afin de les éliminer. Une des solutions possibles consiste à créer un service s_k qui paramétrera les deux services s_i et s_j.

On vient de faire les vérifications possibles de l'aspect fonctionnel et de l'aspect données au niveau des services de l'équipement. Ceci nous assure que la spécification des services obtenue à partir du cahier des charges est complète, cohérente et non-ambiguë, ce qui nous permet de passer à la formation des Modes d'Utilisation et à leur analyse.

III.3.2.2- Formation des Modes d'Utilisation :

A ce stade, l'organisation cohérente des services dans des modes d'utilisation peut être abordée. On a généralement deux possibilités : soit le cahier des charges donne les modes de l'équipement, soit il faut déduire de la précédente spécification quels sont les modes d'utilisation existants. La première situation est la plus courante. En effet, le cahier des charges donne souvent des informations pouvant aider à discerner les modes d'utilisation possibles. Ceci est dû au fait que le client et/ou le concepteur ont une certaine expérience du produit et des possibilités offertes par la configuration demandée. Cependant, dans un cadre général, on se propose de donner une méthode afin de donner une base sur laquelle le concepteur pourra s'appuyer pour former les modes d'utilisation [LEM94].

L'objectif est de définir M l'ensemble des modes d'utilisation qui caractérisent les contextes d'utilisation dans lesquels peut se trouver l'équipement intelligent :

$$M = \{m_j, j \in J\}$$

Le fait le plus marquant qui indique que deux services peuvent être dans le même mode, c'est qu'ils possèdent des conditions d'utilisation commune. On ne s'intéressera pas, ici, aux requêtes d'activation des services : elles caractérisent simplement le fait qu'un service (ou un ensemble de services) ait été demandé. Les requêtes ne caractérisent donc pas le fait que plusieurs services puissent être dans le même mode.

Chaque Conditions_d'Utilisation de chaque service est sous la forme de conjonctions de variables booléennes appartenant à l'ensemble Ba. On définit alors pour chaque service l'ensemble des Conditions_d'Utilisation différentes permettant de définir qu'au moins une condition d'utilisation du service s_i est valide. Soit CU cet ensemble :

$$CU : \quad Ba^+ \rightarrow P(Ba^+)$$
$$s_i \rightarrow CU(s_i)$$

où Ba^+ est l'ensemble Ba enrichi des variables booléennes du type $\bar{a}$ (on différencie, dans Ba^+, les variables booléennes du type a et $\bar{a}$ comme étant deux variables différentes au point de vue symbolique) et où $P(Ba^+)$ est l'ensemble des partitions de l'ensemble Ba^+.

Chaque élément de $CU(s_i)$ différencie les variables booléennes du type a et $\bar{a}$ comme étant deux variables différentes au point de vue symbolique.

On vérifie premièrement qu'il n'existe aucun doublon dans chaque ensemble $CU(s_i)$:

$$\forall i \in I, \forall c_p \in CU(s_i), \forall c_q \in CU(s_i), c_p \neq c_q \text{ on a } c_p \wedge c_q \neq c_p \text{ ET } c_p \wedge c_q \neq c_q$$

Sinon, on élimine c_q de l'ensemble $CU(s_i)$ puisqu'il existe déjà sous la forme de c_p. On a alors des ensembles simplifiés de Conditions_d'Utilisation pour chaque service s_i.

Afin de pouvoir comparer efficacement chaque minterme d'un service aux mintermes des autres services, on se propose de former le tableau suivant, où on met les éléments de Ba^+ en ligne et chaque Conditions_d'Utilisation différente par service en colonne.

Tableau *Comp_CU* : tableau de comparaison des Conditions_Utilisation de chaque service

	...			service s_i			...		
	⋮	⋮	⋮	⋮	c_q	⋮	⋮	⋮	⋮
b_1									
$\overline{b_1}$									
⋮									
b_h					0/1				
$\overline{b_h}$					1/0				
⋮									

A chaque intersection de ce tableau, on regarde si la donnée symbolique b_h (ou $\overline{b_h}$) apparaît dans le minterme correspondant à la colonne. Dans ce cas,on met 1, sinon on met 0.

On simplifie ensuite le tableau précédent en éliminant les lignes qui ne contiennent que des zéros (i.e. dont la somme en ligne est nulle).

Le but est, ensuite, de calculer la distance entre les différents mintermes de services différents. On s'inspire pour cela de la distance de Hamming [CUL68][ROU70], mais au lieu de regarder la distance qui existe entre les deux vecteurs, on regarde ce qui les

rapproche : on s'intéresse uniquement au cas où la donnée de Ba^+ apparaît dans les deux mintermes. On calcule donc une distance de "proximité" entre les mintermes, soit $d_{ip\text{-}jq}$ cette distance, qui se calcule de la manière suivante :

pour $c_p \in CU(s_i)$, et $c_q \in CU(s_j)$
$d_{ip\text{-}jq} = 0$

POUR h=1 A Card(Ba^+)
SI (($Comp_CU(b_h, c_p) = 1$) ET ($Comp_CU(b_h, c_q) = 1$))
ALORS $d_{ip\text{-}jq} = d_{ip\text{-}jq} + 1$

On peut à partir du calcul de ces distances savoir si deux services sont disjoints ou simplement incompatibles :

si $\forall p \in N$, $c_p \in CU(s_i)$, $\forall q \in N$, $c_q \in CU(s_j)$, $d_{ip\text{-}jq} = °0$, on a deux possibilités :

- si $\forall p \in N$, $c_p \in CU(s_i)$, $\forall q \in N$, $c_q \in CU(s_j)$, il existe des valeurs des b_h telles que $c_p \wedge c_q \neq 0$, alors les services s_i et s_j sont dits disjoints.
- si $\forall p \in N$, $c_p \in CU(s_i)$, $\forall q \in N$, $c_q \in CU(s_j)$ et quelles que soient les valeurs des b_h $c_p \wedge c_q = 0$, alors les services s_i et s_j sont simplement incompatibles entre eux : ils ne peuvent être activés ensemble dans un même mode.

Soit Sd l'ensemble des paires (s_i, s_j) de services disjoints l'un de l'autre, et Sinc l'ensemble des paires (s_i, s_j) de services simplement incompatibles entre eux.

Nous proposons un taux de compatibilité entre deux services : plus $d_{ip\text{-}pq}$ sera important par rapport au nombre de données mises en jeu par c_p et c_q, plus les services s_i et s_j sont compatibles et donc plus ils auront de chances d'être dans le même mode à chaque fois que l'un ou l'autre apparaît dans un mode. Notre but est d'obtenir des classes de services regroupables deux à deux afin de donner au concepteur une base sur laquelle il pourra s'appuyer afin de pouvoir former les modes d'utilisation.

On se place dans une politique "du pire des cas" : on prend le plus mauvais pourcentage de compatibilité entre deux services :

$Comp_{ii} = 1$,
et si $i \neq j$, $Comp_{ij} = \underset{\substack{p \in N \\ q \in N}}{Min} \left(d_{ip-jq} / \left(Card(c_p) + Card(c_q) - d_{ip-jq} \right) \right)$

Nous mettons tous ces résultats dans le tableau Comp avec les services en lignes et en colonnes.

Tableau *Comp* : Tableau de compatibilité entre services

Services \ Services	s_1	...	s_j	...
s_1	1			
⋮			.....	
s_i			$Comp_{ij}$	
⋮				

Il nous faut alors des règles de composition entre les couples de services.

On commence donc par trouver le $Comp_{ij}$ maximum : $\underset{\substack{(i,j)\in I^2 \\ i\neq j}}{Max}(Comp_{ij})$.

Avec ce couple de services, on regarde ensuite quel est le service s_k qui est le plus compatible avec le service s_i et avec le service s_j. On choisit le service s_k qui répond à cette équation (on cherche le service le plus compatible aux deux services du couple) :

$$\exists k \in I / Comp_{ik} \neq 0, Comp_{kj} \neq 0, \underset{\substack{k\in I,\ i\neq k \\ j\neq k}}{Min}(Comp_{ij} - Min(Comp_{ik}, Comp_{kj}))$$

On obtient alors un ensemble de triplets de services (s_i, s_j, s_k) qui seront compatibles entre eux à $Min(Comp_{ik}, Comp_{kj})$. Nous complétons ensuite notre spécification de la même manière à partir de l'ensemble des triplets formés ci avant en formant les quadruplets de services compatibles entre eux. Et ainsi de suite, jusqu'à un maximum de (Card(S)-1) uplets de services, qui est un cas limite (le cas suivant étant S lui-même). On forme alors l'ensemble F des couples, triplets, quadruplets,... de services. A chaque élément de F, on lui associe la fonction cm(f) qui à tout élément de f de F donne le taux de compatibilité des services formant f.

Le concepteur peut choisir d'éliminer les éléments de F ayant un taux de compatibilité inférieur à un seuil. On élimine donc tout élément f de l'ensemble F réduit, dont cm(f) est inférieur ou égal à ce taux "minimum".

Ensuite, il faut enlever de F les éléments qui contiennent des services totalement incompatibles entre-eux. Pour deux services, une incompatibilité totale veut dire que les deux services ne peuvent jamais être ensemble dans un même mode, ce qui est une contrainte beaucoup plus forte que deux services simplement incompatibles. Par exemple, un service de configuration d'un PID ne doit pas être dans le même mode qu'un service de régulation PID associé. Par contre, le service d'ouverture d'une vanne et

le service de fermeture de la même vanne pourront être dans le même mode : ils ne pourront simplement pas être actifs en même temps. Ces paires de service (s_i, s_j) doivent être définis par le concepteur et/ou le cahier des charges [BIL94]. Soit l'ensemble Sti, l'ensemble des paires (s_i, s_j) de services totalement incompatibles. On modifie de manière à ce qu'il n'existe pas de services incompatibles dans chaque élément f de F. On a alors :

$\forall i \in I, \forall j \in I, s_i \in S, s_j \in S, (s_i, s_j) \in Sti,$

si $\exists f \in F / (s_i, s_j) \subset f$

alors $F = F \setminus \{f\}$

A la fin, on obtient l'ensemble F réduit à partir duquel le concepteur pourra décrire les modes d'utilisation qu'il compte introduire dans le modèle externe. Il faut cependant ajouter à cet ensemble F, l'ensemble des services s_i qui n'apparaissent dans aucun élément f de F (i.e. qui répond à l'équation suivante : $s_i \in S / f \in F, s_i \not\subset f$). On obtient alors un recouvrement de l'ensemble des services que l'on peut organiser de façon à former les modes d'utilisation.

Les éléments de l'ensemble F donnent une base des ensembles de services possibles pour les modes d'utilisation suivant la compatibilité maximale des services entre-eux. Cela ne peut constituer, dans tous les cas, l'ensemble des modes d'utilisation possibles d'un équipement mais une aide fournie au concepteur pour les former : le concepteur peut soit les utiliser tels quels, soit combiner différents éléments de F pour former un ou plusieurs modes. Une fois ces modes d'utilisation formés, leur cohérence doit être vérifiée comme nous allons le voir ci-dessous.

III.3.2.3- Analyse des Modes d'Utilisation :

Si une spécification des modes d'utilisation est déjà fournie dans le cahier des charges, et même pour vérifier la construction des modes que l'on vient de faire, on forme un autre tableau qui définit quels services appartiennent à quels modes. On met donc dans un tableau les services en ligne et les modes d'utilisation en colonne. Ce tableau que nous appelons *Appartenance* est construit comme suit :

Appartenance$_{ij}$ = 1, si le service s_i est présent dans le mode m_j

= 0, sinon.

Tableau *Appartenance* : tableau d'appartenance des services aux modes d'utilisation.

Modes \ Services	m_1	...	m_j	...
s_1				
⋮			...	
s_i			0 ou 1	
⋮				

D'après la définition d'un mode, il faut vérifier l'affirmation suivante :

"Un service appartient à au moins un mode", soit :

$$\forall i \in I, \sum_{i=1}^{n} \text{Appartenance}_{ij} \geq 1$$

Après avoir vérifié que cette propriété est vraie pour tous les services, on définit l'application Ls qui associe à tout mode d'utilisation m_j l'ensemble des services externes disponibles dans le mode m_j, P(S) est l'ensemble des sous-ensembles de S :

$$Ls : M \rightarrow P(S)$$
$$m_i \rightarrow Ls(m_i) = S_i$$

Une fois les modes définis par rapport aux services qu'ils proposent, il faut donner le tableau de transition entre les différents modes de l'équipement. On obtiendra alors le comportement de l'équipement par un automate fini déterministe. Il faudra, dans l'étape suivante, valider cette organisation des modes d'utilisation.

III.3.2.4- Les Changements de Mode d'Utilisation :

Les changements de mode d'utilisation permettent à l'équipement de modifier son fonctionnement. Les changements de mode se font par l'intermédiaire d'un service spécifique de changement de mode présent implicitement dans chaque mode [STA94]. Le service de changement de mode gère les conditions de changement de mode dans chaque mode. Ce qui nous permet de vérifier l'affirmation suivante :

"Chaque mode contient au moins un service", soit :

$$\forall j \in J, \sum_{j=1}^{M} \text{Appartenance}_{ij} \geq 1$$

Le service de changement de mode possède les mêmes entrées et conditions d'activation : les conditions de changement de mode. On considère, à ce titre, qu'un service de changement de mode possède toujours la priorité la plus haute. On définit T_{cm}, l'ensemble des changements de mode d'utilisation de l'équipement.

$T_{cm} = \{t_{qij}, q \in N, i \in J, j \in J, \}$

Chaque élément t de T_{cm} est lui-même un triplet :

- une condition de transition qui est une expression logique booléenne,
- une requête implicite ou explicite comme on l'a définie pour les requêtes d'activation,
- un ensemble de commandes à exécuter lors de la transition lorsqu'elle est validée.

On a donc :

$(Conditions_Changement_Mode_{qij})::=$

$\{(Conditions_Transition_{qij} ; RequêteM_{qij} ; Commandes_{qij})\}$

Le fait qu'on numérote les $Conditions_Changement_Mode_{qij}$ vient du fait qu'il peut exister plusieurs façons de quitter un mode vers un autre.

Une fois que tous les triplets (Conditions_Transition ; RequêteM ; Commandes) de tous les changements de mode d'utilisation de l'équipement ont été déterminés, on peut recenser deux ensembles :

- l'ensemble de toutes les variables booléennes qui forment les diverses Conditions de Transition des modes de l'équipement,
- l'ensemble de toutes les requêtes explicites de changement de mode (i.e. différentes de "1").

Soit Bm, l'ensemble de toutes les variables booléennes intervenant dans les conditions de transition de l'ensemble des modes M :

$Bm = \{b_h, h \in H / \exists q \in N, \exists i \in J, \exists j \in J, b_h \subset Conditions_Transition_{qij} \}$

et soit Rm, l'ensemble des requêtes explicites de changement de mode intervenant dans les conditions de changement de mode :

$Rm = \{r_h, r_h \neq 1, h \in H / \exists q \in N, \exists i \in J, \exists j \in J, r_h = RequêteM_{qij}\}$

III.3.2.4.1- Vérifications préliminaires :

De même que pour les Conditions d'Activation, il faut vérifier que chaque élément de Bm et de Rm existe bien dans la spécification.

III.3.2.4.1.1- Vérification des conditions de transition :

Pour chaque élément b_h de Bm, il faut vérifier qu'il est produit par "quelqu'un". On a alors deux cas principaux qui peuvent survenir :

- L'élément b_h est la sortie d'un service externe déjà existant, on n'a alors aucun problème.
- L'élément b_h n'est la sortie d'aucun service défini dans le cahier des charges. Il faut alors déterminer qui produit cette donnée :
  - Le cahier des charges stipule que la donnée b_h est fournie par l'extérieur, on définit alors cet *acteur externe* à Ua ou on complète les données fournies par un acteur a_k de Ua déjà défini.
  - Le cahier des charges ne stipule rien à propos de b_h, il faut alors déterminer qui l'a fournie. On a alors trois solutions envisageables :
    - un acteur externe qu'il faut définir à Ua ou on complète les données fournies par un acteur a_k de Ua déjà défini,
    - la donnée b_h est une des sorties secondaires d'un service externe existant (par exemple, un service calculant une valeur peut déterminer si cette valeur a dépassé un seuil),
    - définir un nouveau consommateur externe (ou compléter l'ensemble des données consommées par un consommateur existant) afin de prendre en compte la donnée b_h qui deviendra une sortie principale.

    La dernière solution oblige malheureusement à revenir sur la spécification qui n'était pas complète.

III.3.2.4.1.2- Vérification des requêtes de changement de mode :

De même, pour chaque élément r_h de Rm, il faut vérifier qu'il est produit par "quelqu'un". On a alors deux cas principaux qui peuvent survenir :

- L'élément r_h est la sortie principale d'un service externe déjà existant, on n'a alors aucun problème.

- L'élément r_h n'est la sortie principale d'aucun service défini dans le cahier des charges. Il faut alors déterminer qui produit cette donnée :
  - Le cahier des charges stipule que la donnée r_h est fourni par l'extérieur, on définit alors cet *acteur externe* à Ua ou on complète les données fournies par un acteur a_k de Ua déjà défini.
  - Le cahier des charges ne stipule rien à propos de r_h, il faut alors déterminer qui l'a fournie. On a alors encore une fois les trois solutions envisageables :
    - un acteur externe qu'il faut définir à Ua ou on complète les données fournies par un acteur a_k de Ua déjà défini,
    - la donnée r_h est une des sorties secondaires d'un service externe existant (par exemple, un service calculant une valeur peut déterminer si cette valeur a dépassé un seuil),
    - définir un nouveau consommateur externe (ou compléter l'ensemble des données consommées par un consommateur existant) afin de prendre en compte la donnée r_h qui deviendra une sortie principale.

  Encore une fois, la dernière solution oblige malheureusement à revenir sur la spécification qui n'était pas complète.

On doit, à la suite de ces diverses attributions, réorganiser l'ensemble des sorties Sortie(s_i) de chaque service s_i modifié.

III.3.2.4.1.3- Vérification des commandes :

Il s'agit ici d'une vérification formelle. En effet, il faut s'attacher à ne manipuler que des données de l'équipement déjà définies précédemment. On ne peut ainsi modifier que des données appartenant aux ensembles suivants : Ba, Bm, E, P, Ra ou Rm. Ces commandes peuvent s'avérer très utiles lors des changements de mode au point de vue de la cohérence de l'ensemble : par exemple remettre à "zéro" les paramètres d'un ensemble de service lorsqu'on entre dans un mode de configuration de l'équipement. A ce titre, l'attribution d'une commande à une transition reste le plus souvent au libre arbitre du concepteur : il existe plusieurs manières d'arriver à un même résultat.

Comme pour les conditions d'utilisation, la définition de l'ensemble des conditions de changement de mode est aussi une des phases les plus délicates et souvent sujette à

interprétation des données du cahier des charges de la part du concepteur, ce qui n'en facilite pas la tâche.

III.3.2.4.2- Vérification de l'automate des modes d'utilisation :

Une fois que nous avons vérifié que toutes les données régissant les changements de mode d'utilisation sont présentes dans la spécification, nous avons un automate des modes d'utilisation. Il faut alors déterminer si cet automate est connexe, s'il est déterministe et s'il est sûr au point de vue de la sûreté de fonctionnement des services en eux-mêmes.

III.3.2.4.2.1- Connexité de l'automate des modes d'utilisation :

On se propose de faire quelques vérifications au niveau de ce que les changements de modes entraînent sur l'accessibilité des modes d'utilisation : vérifier que le graphe des modes d'utilisation est bien connexe ou éventuellement partiellement connexe sous certaines conditions.

On construit alors le tableau des changements de mode d'utilisation, avec les modes d'utilisation en ligne et en colonne. Ce tableau que nous appelons *Transition_mode* est construit comme suit :

$$\text{Transition_mode}_{ij} = 1, \text{ si } \exists q \in N / t_{qij} \neq \phi$$
$$= 0, \text{ sinon.}$$

Tableau *Transition_mode* : tableau d'existence de changements de mode.

Modes \ Modes	m_1	...	m_j	...
m_1				
⋮			...	
m_i			1 ou 0	
⋮				

A partir de ce tableau, on doit vérifier que toutes les conditions de changement de mode sont correctement définies : il est toujours possible d'entrer dans un mode et il est toujours possible d'en sortir.

Nous devons vérifier les conditions suivantes sur le tableau Transition_mode :

- $\forall m_j \in M, \exists k \in J$ / $Transition_mode_{kj} \neq 0$,
- $\forall m_i \in M, m_i \neq m_{1final}, \exists j \in J$ / $Transition_mode_{ij} \neq 0$.

Dans certaines applications, deux modes particuliers $m_{initial}$ (par exemple mode "Installation") et m_{final} (par exemple mode "Démantèlement") peuvent exister.

$m_{initial}$ correspond à $\sum_{i=1}^{m} Transition_mode_{i,initial} = 0$ et m_{final} correspond à $\sum_{j=1}^{m} Transition_mode_{final,j} = 0$.

$m_{initial}$ et m_{final} doivent tous les deux être uniques pour un équipement et correspondent aux deux extrémités du cycle de vie de l'équipement. Dans le cas contraire, il existe un problème de conception. Ils peuvent alors ensuite être enlevés temporairement du tableau Transition_mode pour former le tableau $Transition_mode^{(2)}$ qui correspond au tableau Transition_mode sans les éventuels modes $m_{initial}$ et m_{final}.

On peut alors recommencer les vérifications pour s'assurer de l'existence de nouveaux modes $m_{initial}^{(2)}$ et $m_{final}^{(2)}$. On suit alors la même procédure jusqu'à obtenir le tableau $Transition_mode^{(p)}$ où $m_{initial}^{(p)}$ et $m_{final}^{(p)}$ n'existent pas ou lorsque $Transition_mode^{(p)}$ est vide. Dans ce dernier cas, le cycle de vie de l'équipement est linéaire.

Dans le cas général, on travaillera avec le tableau Transition_mode. Dans ce tableau, on doit vérifier que les modes forment entre eux un graphe fortement connexe : de tous les modes, on peut arriver à tous les autres ce qui inclut qu'il n'existe pas plusieurs graphes distincts. Pour cela, on calcule là aussi la fermeture transitive idempotente du tableau Transition_mode, i.e. on calcule $(Transition_mode)^q$ jusqu'à ce que $q = (Card(M) - 1)$ ou jusqu'à ce que $(Transition_mode)^q = (Transition_mode)^{q+1}$. On peut alors vérifier que :

$$\forall i \in J, \forall j \in J, (Transition_mode)^q = 1$$

Ceci prouve la connexité forte du graphe représenté par le tableau Transition_mode. Si ce n'est pas vérifié, il existe au moins deux graphes distincts dans la spécification des modes et elle doit être révisée. Les révisions à faire se situent principalement dans l'ensemble T_{cm} définissant l'existence et les conditions de changement de mode puisque l'erreur proviendra certainement de l'absence d'une ou plusieurs transitions.

Dans le cas particulier où il existe un mode initial et/ou un mode final, la vérification s'applique au dernier tableau $Transition_mode^{(p)}$ trouvé précédemment.

III.3.2.4.2.2- Déterminisme de l'automate des modes d'utilisation :

Une fois la connexité forte du graphe établie et vérifiée, il nous faut aussi prouver qu'il est déterministe et ceci pour des raisons de sécurité. Le déterminisme du graphe se vérifie facilement grâce au tableau Transition_mode et à l'ensemble S_{cm} des conditions de changement de mode :

$\forall i \in J, \forall j \in J, \forall p \in N, \forall q \in N, i \neq j$, $Transition_mode_{ij} = 1$,
alors $\forall k \in J, k \neq j, t_{pij} \wedge t_{qik} = 0$

Si toutes ces assertions sont vérifiées, le graphe est déterministe, sinon il y a plusieurs possibilités :

- si $\exists k \in J, k \neq j, t_{pij} \wedge t_{qik} = 1$, il y a une erreur de spécification puisque, à partir d'un mode et sous certaines conditions (i.e. $t_{pij} \wedge t_{qik}$), on peut aller dans au moins deux autres (les modes m_j et m_k). Le graphe est non-déterministe.

BU LILLE 1

- si $\exists k \in J$, $k \neq j$, $t_{pij} \wedge t_{qik}$ =Expression_Logique, il faut déterminer si cette Expression_Logique est :
  - soit toujours fausse, on a alors affaire à un invariant du système que l'on veut modéliser et il faudra en tenir compte dans la réalisation pour que cela soit toujours vérifié,
  - soit toujours vraie, là aussi on a un invariant du système mais dans le sens du terme et il faut revoir la spécification pour l'éliminer,
  - soit vraie soit fausse selon la valeur d'un ou plusieurs prédicats communs. Il faut là encore revoir la spécification. Plusieurs choix simples sont a priori possibles : ajouter une condition de choix supplémentaire dans les conditions de changement de mode (ex : $a \wedge \overline{b}$ pour t_{pij} et b pour t_{qik}) ou alors assigner une priorité fixe aux différentes transitions partant du mode m_i.

Le déterminisme dans les transitions entre modes étant vérifié, il nous faut regarder si nous n'accédons pas à un mode "incohérent" au point de vue sûreté de fonctionnement des services qui composent le mode.

III.3.2.4.2.3- Cohérence des modes d'utilisation :

Nous avons vu, entre autres, que certains services ne peuvent pas toujours s'effectuer si un autre ne l'a pas déjà été auparavant, comme c'est le cas de services qui ont besoin d'avoir été paramétrés. L'impossibilité d'exécuter un service ne pose cependant pas toujours de problème dans un mode, sauf dans le cas où le service serait indispensable à la bonne marche du mode d'utilisation.

On dira qu'un service est indispensable à un mode d'utilisation lorsque l'absence du service en question dans le mode ne permet plus au mode de remplir sa mission correctement. Afin d'approfondir l'étude, on monte le tableau *Services_indisp* avec les services en ligne et les modes en colonne et on le construit comme suit :

$Services_indisp_{ij}$ = 1, si le service s_i est indispensable au mode m_j,
= 0, sinon.

Tableau *Services_indisp* :

BU LILLE I

Modes \ Services	m_1	...	m_j	...
s_1	0			
⋮			.....	
s_i			0 ou 1	
⋮				

Remarque : Le fait d'assigner qu'un service est indispensable à un mode reste tout à fait arbitraire. En effet, il n'existe, a priori, aucune méthode permettant de dire que l'absence d'un service rend impossible la mission qui a été attribuée à un mode. Tout dépend donc de la connaissance du système modélisé que possède le concepteur et des connaissances (généralement empiriques) disponibles sur le problème.

La vérification que l'on peut faire est de regarder que lorsqu'on entre dans un mode tous ses services indispensables sont disponibles. En effet, si on entre dans un mode dont la fonction ne peut être remplie, on ne doit pas y rester, autant en restreindre l'accès sous certaines conditions.

On introduit pour ce faire une condition de disponibilité d'entrée dans le mode m_j qu'on nomme $Cond_Ent_j$. Cette $Cond_Ent_j$ se détermine à partir de l'ensemble des services indispensables au mode m_j. En effet, l'ensemble des conditions d'activation des

services indispensables donne une idée des conditions sous lesquelles le mode m_j pourra remplir sa mission. Chaque $Cond_Ent_j$ pourra alors se trouver en prenant "l'intersection" des conditions d'utilisation de chaque service indispensable ou encore en prenant la conjonction de certains des termes les composant. Ainsi, chaque $Cond_Ent_j$ est une conjonction de conditions à respecter que le concepteur devra déterminer. Après quoi, on ajoute à chaque condition de transition entrant dans le mode le fait que l'ensemble des services indispensables doivent être disponibles, puis on simplifie les expressions de chaque transition t_{pkj}.

On a donc :

$\forall i \in I, \forall j \in J$, si $Services_indisp_{ij} = 1$

alors

$\forall k \in J, \forall p \in N$, si $Transition_mode_{kj}=1$, alors $t_{pkj} = t_{pkj} \wedge (Cond_Ent_j)$

Une fois tous les modes d'utilisation et tous les changements de mode définis, on passe aisément à l'automate de fonctionnement équivalent, en définissant, comme on l'a déjà dit précédemment, les modes d'utilisation comme les états de l'automate et les changements de modes d'utilisation définissent les transitions entre les états de l'automate. On obtient alors une spécification fonctionnelle de l'instrument cohérente suivant ce que le cahier des charges demandait et contenait comme informations.

Comme pour la spécification des services, la procédure peut être automatisée. On a programmé sous Excel97 un fichier permettant de vérifier et de valider une grande partie de l'analyse de la spécification déjà obtenue. Le détail est donné en Annexe II.

III.4- Résumé de la démarche :

Nous donnons ici un résumé de la démarche que nous avons suivi pour la spécification du cahier des charges et pour son analyse :

- **Spécification des besoins dans le cahier des charges** :
  - déterminer les consommateurs externes et leurs besoins,
  - à chaque besoin exprimé associer une sortie principale,
  - à chaque sortie principale associer un service externe pour lequel on doit définir :
    - → les Entrées
    - *vérifier / définir qui les produit : service externe existant ou acteur externe*
    - → les Paramètres, s'ils existent

vérifier / définir qui les produit : service externe existant et différent, ou acteur externe

→ les Conditions d'Activation sous la forme d'ensembles de :

[Condition d'Utilisation] Requête,

pour chaque terme des Condition d'Utilisation et pour chaque Requête, vérifier qui les produit : service externe existant ou acteur externe

→ les Ressources vitales / non-vitales, partagées / non-partagées et leurs répercussions sur la Préemptabilité du service,

→ la Procédure avec les informations suivantes :

- ♦ la méthode utilisée si elle est précisée dans le cahier des charges,
- ♦ la priorité du service,
- ♦ le temps d'exécution au plus tôt,
- ♦ la périodicité du service et s'il est périodique sa période,
- ♦ la préemptabilité du service.

• donner les acteurs externes définis ci-dessus et l'ensemble de données qu'ils fournissent.

- **Analyse de la spécification des besoins dans le cahier des charges** :

• *vérifier l'existence possible d'ensemble de services en boucle dynamique*,

• *vérifier l'existence possible d'ensemble de services en paramétrage croisé*,

• former les Modes d'Utilisation,

vérifier que chaque service appartient à au moins un mode

• définir les conditions de changement de mode d'utilisation sous la forme de :

[Conditions de Transition] Requête / Commandes

pour chaque terme des Conditions de Transition et pour chaque Requête, vérifier qui les produit : service externe existant ou acteur externe

vérifier que les commandes modifient une donnée existante

• *vérifier sur l'automate des modes d'utilisation* :

→ *sa connexité*,

→ *son déterminisme*,

→ *la cohérence des modes d'utilisation.*

• compléter l'ensemble des acteurs externes avec ceux définis ci-dessus et pour chacun les ensembles de données qu'ils fournissent.

III.5- Conclusion :

Ce chapitre était consacré à la présentation du modèle proposé par le L.AI.L. : le modèle externe, modèle générique d'équipement "intelligents". Ce modèle propose une vision orientée Fonctions et orientées Données. Cette approche nous a permis de monter une méthode permettant, à partir du cahier des charges fourni par le client, d'une part d'obtenir le modèle externe accessible non-ambigu d'un équipement et, d'autre part, de faire un ensemble de vérifications afin de prouver l'utilité, la complétude, la cohérence et la traçabilité du modèle des points de vue fonctionnel et données. Ces vérifications prouvent que le modèle externe proposé de l'équipement intelligent répond bien aux critères énoncés au chapitre I.

Le modèle obtenu met en évidence les ressources matérielles nécessaires au fonctionnement de l'équipement. Il définit les interfaces avec l'environnement, i.e. les données en entrée et en sortie. Ces informations sont d'un grand intérêt au stade de la réalisation.

Le modèle ne prend que très peu en compte les problèmes liés au comportement dynamique de l'équipement tel que la préemption par exemple. De plus, ce modèle ne correspond généralement qu'au fonctionnement normal et optimal de l'équipement : sans aucune panne dans les composantes de l'équipement.

On se propose dans le chapitre suivant de remédier aux problèmes liés à ce qui est appelé la sûreté de fonctionnement et on proposera un modèle comportemental précis équivalent à l'automate obtenu ci-dessous. On obtiendra ce modèle à partir d'un progiciel graphique développé par le professeur C. André au sein de l'INRIA Sophia-Antipolis : SYNCCHARTS.

Références :

[ADE81] ADEPA, "Le GEMMA : Guide des Modes de Marche et d'Arrêt", 1981

[ALA89] Alakl N., Lalanne J.-L., "Architecture technologique des systèmes d'information : la méthode TACT", les Editions d'Organisation, Paris, 1989.

[BAY95] Bayart, M., A.L. Gehin and M. Staroswiecki (1996). "Smart Intelligent Design", MMAR'96 : Methods and Models in Automation and Robotics, pp. 727-732, Miedzyzdroje, Pologne.

[BEN95] Benkhellat M., " Formalisation et vérification de l'interopérabilité dans les systèmes de communication", Thèse de Doctorat de l'Institut National Polytechnique de Lorraine, janvier 1995.

[BER93] Berry G., "Preemption in Concurrent Systems", Proceedings FSTTCS 93, Sringer-Verlag Lecture Notes in Computer Science, Vol. 761, pp 72-93, 1993.

[BIL94] Biland P., "Modélisation des modes de marche d'un système automatisé de production", Thèse de doctorat, Ecole Centrale de Nantes, février 1994.

[BOI91] Bois S., "Intégration de la gestion des modes de marche dans le pilotage d'un système automatisé de production", Thèse de l'Université de Lille1, novembre 1993.

[BOU97] Bouras A., "Contribution à la Conception d'Architectures Réparties : Modèles Génériques et Interopérabilité d'Instruments Intelligents", Thèse de l'Université de Lille1, juillet 1997.

[CUL68] Cullmann G. "Codage et transmission de l'information", Editions Eyrolles, Paris, 1968.

[GEH94] Gehin A.L., "Analyse Fonctionnelle et Modèle Générique des Capteurs Intelligents : Application à la Surveillance de l'Anesthésie", Thèse de l'université de Lille1, janvier 1994.

[JOS96] Joseph M. :"Real-Time Systems : Specification, Verification and Analysis.", Prentice Hall International, 1996.

[LAR98] "Le Petit Larousse 1998", Larousse-Bordas, Paris, 1998.

[LEM94] Lemaire E., "Spécification des Modes d'Utilisation pour l'agrégation d'instruments intelligents", Mémoire de D.E.A., Université de Lille1, juillet 1994.

[ROU70] Roubine E., "Introduction à la théorie de la communication, Tome III : Théorie de l'information", Editions Masson, Paris, 1970

[STA94] Staroswiecki M., Bayart M., "Actionneurs intelligents", Editions Hermès, Paris, 1994.[STA95] Staroswiecki M., Bayart M., "Modes d'Utilisation et Modes de Marche des Instruments Intelligents", Journées d'Etudes SAPID

- Conception de Systèmes Automatisés de Production à Intelligence Distribuée, Paris, 30-31 mai 1995.

Chapitre IV :
Un Modèle Comportemental associé au Modèle Externe

Table des Matières :

IV.1- Introduction : **107**

IV.2- Le modèle SYNCCHARTS : **107**
IV.2.1- L'étoile : 108
IV.2.2- La constellation : 109
IV.2.3- Le macro-état : 110

IV.3- Passage du Modèle Externe à une description en SYNCCHARTS : **111**
IV.3.1- Méthode générale : 111
IV.3.2- Méthode d'encapsulation de macro-états : 112
IV.3.2.1- Détermination des macro-états encapsulables : 112
IV.3.2.2- Encapsulation de macro-modes d'utilisation : 114
IV.3.2.2.1- Paires indépendantes : 115
IV.3.2.2.2- Paires non-indépendantes : 115
IV.3.2.3- Modification de la structure du syncChart : 115
IV.3.2.3.1- Structure du syncChart global : 116
IV.3.2.3.2- Structure interne du macro-mode d'utilisation formé : 119
IV.3.2.4- syncChart final : 120
IV.3.3- Services dans le syncChart : 120
IV.3.3.1- Simplification des conditions d'activation des services dans le syncChart : 121
IV.3.3.1.1- Les conditions d'utilisation : 121
IV.3.3.1.2- Les requêtes d'activation : 123
IV.3.3.2- Fonctionnement interne de chaque service dans le syncChart : 124
IV.3.3.2.1- syncChart d'un service : 124
IV.3.3.2.2- Exemple de différentes versions d'un service : 125
IV.3.3.3- Les modes de marche : 127
IV.3.4- Vérification du syncChart obtenu : 129

IV.4- Conclusion : **130**

Références : **132**

Un Modèle Comportemental associé au Modèle Externe

IV.1- Introduction :

Le chapitre précédent présente la partie fonctionnelle d'un équipement : i.e. il donne le modèle abstrait de l'équipement du point de vue de ce qu'il offre comme possibilités au niveau services rendus et données échangées avec son environnement. La description fonctionnelle de chaque service et l'analyse de ses relations avec les autres services existants permettent de prendre en compte l'ensemble des services afin de pouvoir former un ensemble cohérent de modes d'utilisation. Un automate fini déterministe a alors été construit décrivant le fonctionnement global de l'équipement par rapport aux modes d'utilisation et aux services associés à chacun des modes.

Si l'automate propose une bonne vision globale du fonctionnement de l'équipement, il reste cependant un modèle plat, ne prenant en compte ni la préemption ni les problèmes de priorités. Ainsi, avec l'automate des modes d'utilisation, on ne prend pas en compte ce qui arrive au niveau des services lors des changements de modes d'utilisation : certains services sont désactivés alors que d'autres continuent à fonctionner normalement.

On se propose donc dans ce chapitre de se donner un modèle comportemental de fonctionnement de l'équipement à l'aide du logiciel SYNCCHARTS qui a été développé au sein de l'INRIA Sophia Antipolis [AND96][AND98] à Nice. Ce chapitre se découpera en plusieurs parties. La première consistera en une description plus approfondie de SYNCCHARTS et de ses apports par rapport aux StateCharts. La seconde partie développera une méthode qui permet à partir du modèle externe de l'équipement obtenu de former le syncChart équivalent du fonctionnement de l'équipement intelligent.

IV.2- Le modèle SYNCCHARTS : [AND96][AND98]

"SYNCCHARTS" est un acronyme pour "*Synchronous Charts*". SYNCCHARTS est une évolution du modèle StateCharts [HAR87] et de ARGOS [MAR90]. Ce sont des outils graphiques qui permettent de représenter le comportement réactif d'un système quelconque et sont basés sur le paradigme synchrone (cf. III.2.4.5).

Comme pour les StateCharts, SYNCCHARTS propose une structure hiérarchique par encapsulation d'états, le parallélisme et la préemption. Contrairement à StateCharts qui ne propose qu'un mécanisme de préemption sous forme d'avortement fort, toutes les formes de préemption (i.e. l'avortement faible, l'avortement fort et la suspension) sont proposées dans le modèle SYNCCHARTS. De plus, toute instance syncChart issue du modèle SYNCCHARTS est

directement traduisible en un programme ESTEREL équivalent, ce qui en fait un outil particulièrement intéressant de part les possibilités de vérifications comportementales offertes par le langage ESTEREL. De plus, la compatibilité complète de SYNCCHARTS avec ESTEREL le dote avec une sémantique mathématique bien définie.

Comme dans tout modèle synchrone, les SYNCCHARTS permettent d'échanger des informations sous forme de signaux qui sont instantanément diffusés. Toutes les composantes du modèle perçoivent ainsi les mêmes informations au même instant logique. Ces signaux qui sont les supports de la communication dans le modèle peuvent être (fortement) typés : i.e. les SYNCCHARTS peuvent manipuler des signaux valués qui contiennent une valeur en plus de leur statut de présence.

Le modèle SYNCCHARTS possède sa propre syntaxe graphique qui est articulée autour de l'étoile qui est la brique de base du modèle. Chaque étoile peut alors faire elle-même partie d'une constellation qui est à son tour une constituante d'un macro-état. Un macro-état n'est rien d'autre qu'une étoile de niveau supérieur d'où la hiérarchie d'état par encapsulation.

IV.2.1- L'étoile :

Une étoile est plus qu'un état : c'est un état et les moyens d'en sortir. Les arcs qui sortent de l'étoile de la figure 4.1 constituent ces différents moyens d'en sortir. Chaque arc, parfois appelé "rayon", est étiqueté de la manière générale suivante :

déclencheur [condition] / effet

où les différents éléments désignent :

- *déclencheur* est une expression booléenne portant sur la présence des signaux,
- *condition* est un prédicat qui peut faire intervenir la valeur des signaux valués,
- *effet* est une liste de signaux, éventuellement valués, à émettre lors de la transition

Chaque rayon est aussi étiqueté par une priorité ce qui induit le déterminisme du modèle au cas où plusieurs déclencheurs seraient valides au même instant. Lorsqu'il n'existe qu'un seul rayon sortant de l'étoile, sa priorité par défaut est de 1 et est omise.

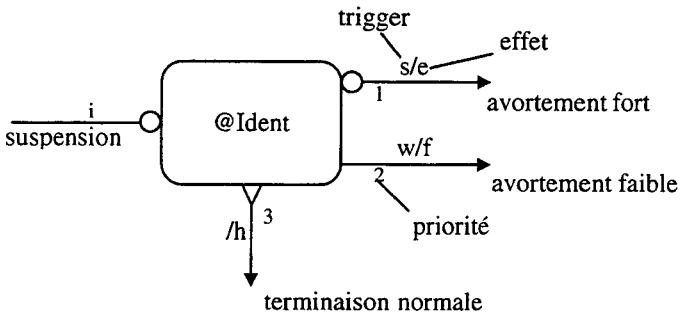


Figure 4.1 : Etoile d'un syncChart.

Le @Ident à l'intérieur de l'étoile indique tout simplement son nom et que l'étoile peut être développée à un niveau hiérarchique plus bas décrivant ainsi le fonctionnement interne de l'étoile où "Ident" est le nom associé à l'étoile.

Il existe trois façons de sortir de l'étoile :

- Sur une terminaison "normale", i.e. l'étoile a fini sa tâche et toutes ses constellations sont dans leurs états finaux respectifs (cf. IV.2.2). La terminaison normale se caractérise par le fait qu'il n'y a pas de déclencheur (il est implicite à l'état interne de l'étoile en question) et on le distingue par une flèche sortante ayant un triangle à sa base : ▷⟶.
- Par un avortement faible, on a alors un signal déclencheur qui déclenche la sortie de l'étoile, l'avortement faible se distingue par une flèche simple sortante : ⟶ .
- Par un avortement fort, qui se distingue de l'avortement faible par un rond à la base dune flèche simple sortante : ○⟶ .

A ces trois façons de sortir d'une étoile, il faut ajouter la dernière forme de préemption à laquelle SYNCCHARTS permet de faire appel : la suspension de l'étoile. Cette suspension est indiquée par une "flèche" terminée par un rond : ——○ . La suspension de tous les états de l'étoile est vraie tant que le déclencheur[condition] associé reste présent.

Les différentes manières de quitter une étoile permettent d'aller vers d'autres étoiles. On forme ainsi la structure suivante qui est la constellation.

IV.2.2- La constellation :

Une constellation est un ensemble d'étoiles reliées entre elles par des transitions de type terminaison normale, avortement faible ou avortement fort. De ce fait, une constellation ressemble beaucoup à un automate fini, cependant une constellation peut très bien se résumer à une seule et unique étoile.

Comme pour StateCharts, quand on entre dans une étoile (ou macro-état), il faut désigner quelle étoile de la constellation de niveau inférieur sera active. Cette étoile qui est unique est désignée par une flèche entrante sur une étoile de la constellation. Ensuite, dans chaque constellation, il n'existe qu'une seule et unique étoile active à un moment donné. Enfin, on peut définir, par constellation, un état final qui est, en termes de graphe d'automate, "un puits" et qui indique que la tâche imputée à la constellation est terminée. Cette étoile particulière est entourée une fois supplémentaire par rapport aux autres étoiles de la constellation.

Un exemple de constellation est représenté par la figure 4.2.

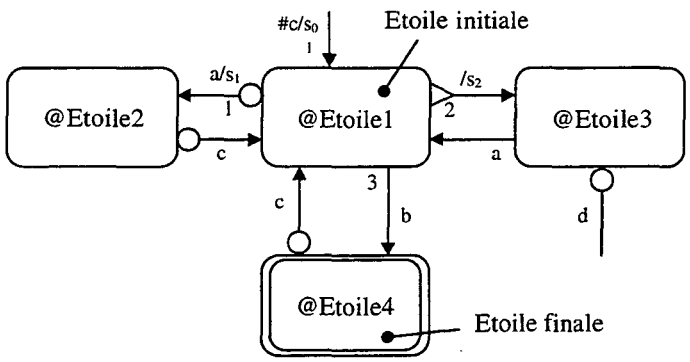


Figure 4.2 : Constellation d'un syncChart et transitions possibles entre étoiles.

Un signal précédé du signe # signifie que ce signal a un effet immédiat par rapport au "TICK" de l'horloge interne ESTEREL propre au modèle SYNCCHARTS.

Une constellation d'étoiles est la représentation particulière interne d'une étoile. Cependant, il peut exister plusieurs constellations concurrentes dans une même étoile, on nomme alors *macro-état* l'association en parallèle de plusieurs constellations.

IV.2.3- Le macro-état :

Un macro-état n'est rien d'autre qu'une étoile possédant plus d'une constellation en son sein. Par rapport à une étoile, le nom de l'étoile est indiqué en haut du cadre et séparé du reste du cadre par un trait horizontal. Chaque constellation est séparée des autres par une ligne en pointillée.

S'il existe des signaux qui sont communs à plusieurs constellations d'un même macro-état, on les déclare au bas du cadre en deçà du bas des lignes en pointillé séparant les différentes constellations en présence.

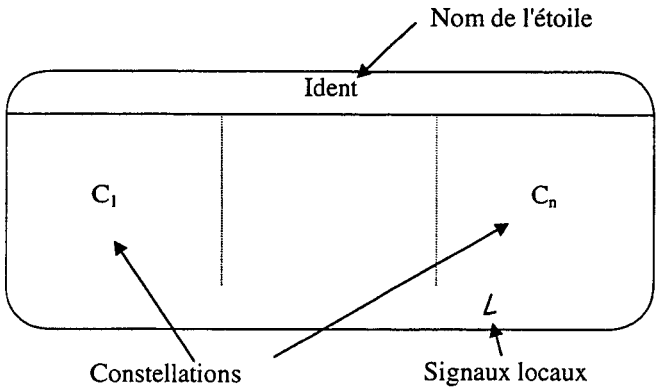


Figure 4.3 : Macro-état d'un syncChart.

Le modèle SYNCCHARTS ainsi posé, on se propose de l'utiliser afin de pouvoir faire une vérification comportementale d'une spécification qui serait obtenue à partir du modèle externe précédemment exposé et qui possède déjà des spécifications fonctionnelles correctes. La partie suivante propose une méthode qui permet de passer du modèle externe à son syncChart correspondant.

IV.3- Passage du Modèle Externe à une description en SYNCCHARTS : [BAY98][BAY99]

Par rapport au modèle SYNCCHARTS, l'automate fini déterministe que donne le modèle externe à la fin de la spécification fonctionnelle fournit un modèle "plat" sans hiérarchie d'état et sans vraiment de préemption entre les différents services de la spécification sauf au moment des changements de mode d'utilisation. Même dans ce dernier cas, on ne spécifie pas vraiment, dans le modèle externe, comment sont gérés les services qui sont communs à deux modes et ceux qui ne le sont pas. SYNCCHARTS propose ainsi une bonne alternative aux manques de la spécification fonctionnelle fournie par le modèle externe.

IV.3.1- Méthode générale :

La première idée qui vient à l'esprit pour passer du modèle externe à son syncChart équivalent est de transposer directement les modes d'utilisation de l'automate fini déterministe obtenu en macro-états. Dans chacun de ses macro-états, les différentes constellations sont formées par les services ou groupes de services en relation entre-eux présents dans le mode d'utilisation. Dans chaque mode d'utilisation, les services ou groupes de services sont concurrents. Les transitions de modes d'utilisation à modes d'utilisation restent les mêmes en termes de syntaxe, bien que certains "aménagements" soient parfois à prévoir dû à la nature de SYNCCHARTS qui perçoit plutôt les

changements d'état. Cependant le concepteur doit spécifier clairement dans le syncChart de base la nature de la transition : un avortement faible, un avortement fort ou une terminaison normale. Dans ce dernier cas, il faut prévoir tous les états finaux de chaque constellation du mode d'utilisation. Enfin, dans chaque mode d'utilisation, les cas de suspension de service doivent être établis en accord avec les spécifications obtenues.

La représentation SYNCCHARTS obtenue peut parfaitement bien être la représentation finale pour des équipements relativement simples en termes de fonctionnement. Cependant, à cause du nombre assez important de services dans les systèmes complexes, la représentation peut n'être ni très claire, ni très lisible. Dans ce cas, une méthode a été développée de manière à obtenir une représentation interne des macro-états beaucoup plus claire basée sur l'encapsulation productive de macro-états.

IV.3.2- Méthode d'encapsulation de macro-états :

La première étape de cette méthode est, bien entendu, de trouver quels sont les modes d'utilisation qui peuvent être encapsulés dans un même macro-état en fonction des informations fournies par le modèle externe de l'équipement intelligent. Les modes d'utilisation ainsi "regroupés" permettront d'utiliser au mieux la propriété d'encapsulation des SYNCCHARTS. La deuxième étape consistera, elle, à décider parmi les modes d'utilisation encapsulables quels sont ceux qui seront encapsulés et quels sont les aménagements à faire au niveau du syncChart afin de conserver la cohérence du modèle. Cette dernière étape aboutira à un syncChart composé de macro-modes d'utilisation, de modes d'utilisation et de services hiérarchisés entre-eux ce que ne propose pas le modèle externe.

IV.3.2.1- Détermination des macro-états encapsulables :

Afin de repérer quels sont les modes d'utilisation encapsulables de l'ensemble M, on définit l'ensemble Ms qui est, à ce niveau, identique à M. On part ensuite du tableau Transition_mode qu'on applique à Ms. Ce tableau a été défini au chapitre III (cf. III.3.2.4.2.1) et on en rappelle ici la définition :

Tableau *Transition_mode* : tableau d'existence de changements de mode.

Modes \ Modes	m_1	...	m_j	...
m_1				
⋮			...	
m_i			1 ou 0	
⋮				

où $Transition_mode_{ij} = 1$, si $\exists q \in N / t_{qij} \neq \phi$

$= 0$, sinon.

Ce tableau donne l'existence de conditions de changement de mode entre deux modes d'utilisation, il nous permet de déterminer si deux modes peuvent être encapsulés en fonction des ensembles de leurs conditions de changements de mode respectifs. En effet, même en encapsulant deux modes, leur comportement commun doit rester déterministe et cohérent par rapport au reste des modes d'utilisation.

La condition qui permet d'encapsuler deux modes d'utilisation correspond donc aux restrictions précédentes. Pour chaque entrée du tableau Transition_mode, on doit vérifier ce qui suit :

$\forall m_i \in Ms$, $\forall m_j \in Ms$, le macro-mode m_{ij} peut être formé si et seulement si les conditions suivantes sont respectées :

- $\forall k \in J / Transition_mode_{ik} \neq 0$, $Transition_mode_{jk} \neq 0$

alors $\forall p \in N, \exists q \in N / t_{pik} = t_{qjk}$,

- $\forall k \in J, \forall l \in J / k \neq l$, si $Transition_mode_{ik} = 0$

alors $\forall p \in N, \forall q \in N, t_{pik} \wedge t_{qjl} = 0$,

- $\forall k \in J, \forall l \in J / k \neq l$, si $Transition_mode_{jk} = 0$

alors $\forall p \in N, \forall q \in N, t_{pjk} \wedge t_{qil} = 0$,

C'est surtout ces deux dernières conditions qui permettent d'éviter d'introduire un indéterminisme dans le modèle en interdisant à partir du macro-mode m_{ij} d'aller vers les modes m_k et m_l. comme le montre l'exemple de la figure 4.4.

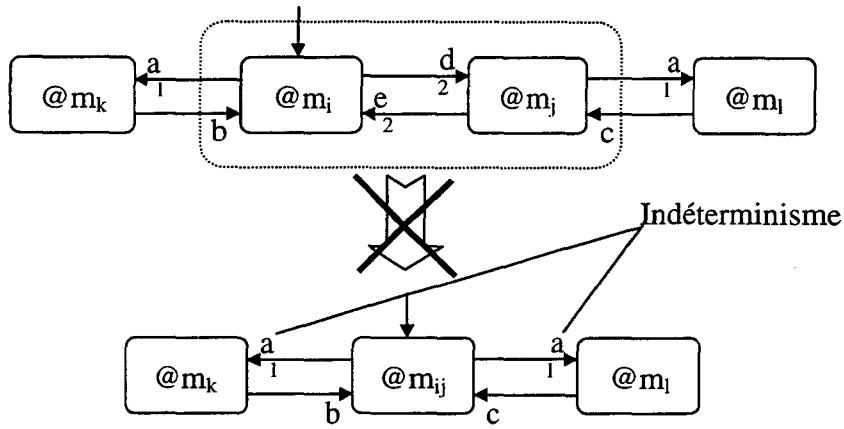


Figure 4.4 : Cas d'indéterminisme à éviter.

L'ensemble de toutes les paires de modes d'utilisation (m_i, m_j) respectant les conditions précédentes est nommé Mp. Il faut alors choisir parmi les paires de l'ensemble Mp lesquelles seront choisies afin de rendre le syncChart le plus clair et lisible possible.

IV.3.2.2- Encapsulation de macro-modes d'utilisation :

La philosophie d'encapsulation de macro-modes d'utilisation est assez simple : "plus deux modes d'utilisation ont de services externes en commun, plus il est intéressant de les encapsuler". En effet, la lisibilité du syncChart sera d'autant plus grande que les services externes de l'équipement n'apparaissent qu'une seule fois. On lie donc le choix des paires de l'ensemble Mp au nombre de services en commun qu'ils montrent.

On fait alors appel à la fonction Ls définie en III.3.2.3. qui associe à chaque mode m_i les services externes disponibles dans le mode m_i. Avec la fonction Ls, il est alors facile de déterminer le meilleur macro-mode qui correspond à la paire (m_i, m_j) de l'ensemble Mp montrera le plus de services en commun. Soit Mp_1, un sous-ensemble de Mp défini par :

$$Mp_1 = \{(m_i, m_j) / \text{Card}(Ls(m_i) \cap Ls(m_j)) \geq \text{Card}(Ls(m_k) \cap Ls(m_l)), \forall (m_k, m_l) \in Mp\}.$$

En fonction de Mp_1, plusieurs cas doivent alors être étudiés afin de déterminer la paire ou les paires qui seront choisies afin de former un macro-mode. On détermine ici plus exactement deux cas : le cas où les paires de Mp_1 sont ou ne sont pas indépendantes.

IV.3.2.2.1- Paires indépendantes :

Au moins une partie des paires (m_i, m_j) de Mp_1 sont indépendantes, i.e. telles que :

$\forall(m_i, m_j) \in Mp_1, \forall m_k \in Ms,$

soit $(m_i, m_k) \in Mp_1 \Rightarrow m_k = m_j,$

soit $(m_k, m_j) \in Mp_1 \Rightarrow m_k = m_i.$

Dans ce cas, le concepteur peut former tous les macro-modes correspondant aux paires indépendantes de Mp_1.

IV.3.2.2.2- Paires non-indépendantes :

Il correspond bien entendu au cas où il existerait des paires qui ne sont pas indépendantes entre elles et où plusieurs paires peuvent être regroupées ensemble comme, par exemple, les paires (m_i, m_j), (m_i, m_k) et (m_j, m_k).

On y voit ici encore deux sous-cas en fonction de la proposition :

$Ls(m_i) \cap Ls(m_j) = Ls(m_i) \cap Ls(m_k) = Ls(m_j) \cap Ls(m_k)$

Si cette proposition est vérifiée, le macro-mode m_{ijk} est formé puisque le nombre de services en commun est optimum et que ce sont les mêmes pour les trois modes m_i, m_j et m_k.

Sinon, les trois modes d'utilisation ne peuvent être encapsulés ensemble et une seule des trois paires doit être considérée. La paire qui sera choisie parmi les trois devra prendre en compte un autre critère qui peut dépendre de l'équipement en lui-même : les entrées, les ressources, les paramètres ou d'autres caractéristiques de l'équipement qui sont communes aux différents services en commun dans les modes d'utilisation pris deux à deux.

IV.3.2.3- Modification de la structure du syncChart :

La formation d'un macro-mode d'utilisation m_{ij} implique forcément des changements dans la structure du syncChart existant jusqu'à présent. Les changements à réaliser concernent plus particulièrement deux aspects du syncChart : les aspects concernant les relations du macro-mode d'utilisation créé avec les autres modes d'utilisation et les aspects plus internes au macro-mode d'utilisation généré. Ces derniers changements sont

plus principalement dus à la hiérarchisation des différents services qui sont proposés aux utilisateurs dans chaque mode d'utilisation.

IV.3.2.3.1- Structure du syncChart global :

La première chose à faire est de reformer l'ensemble Ms qui correspond à l'ensemble Ms auquel on a fait les modifications suivantes :

- ajouter le macro-mode m_{ij},
- les modes d'utilisation m_i et m_j (et voire d'autres modes m_k) correspondant y sont eux enlevés.

$$Ms = Ms \cup \{m_{ij}\} \setminus \{m_i, m_j\}$$

La deuxième chose à faire est de compléter l'ensemble M avec le macro-mode m_{ij} formé. En complétant l'ensemble M, on garde une trace de tous les modes d'utilisation.

$$M = M \cup \{m_{ij}\}$$

Il faut ensuite modifier les transitions existantes entre le mode d'utilisation m_{ij} vers les autres modes d'utilisation par rapport aux "anciennes" transitions en provenance de m_i et de m_j. On peut diviser en deux étapes les différentes opérations à faire sur le syncChart en ce qui concerne les transitions.

La première phase consiste à isoler du reste de la spécification l'ensemble des transitions entre les modes m_i et m_j. En effet, si ces transitions sont toujours présentes à l'intérieur du maco-mode m_{ij}, celles-ci deviennent tout à fait transparentes du point de vue du reste des modes. A ce titre, on "supprime" temporairement l'ensemble des transitions du mode m_i vers le mode m_j et du mode m_j vers le mode m_i pour former le nouvel ensemble Ts qui est au début identique à l'ensemble T_{cm} et qui correspond ensuite à :

$$Ts = Ts \setminus \{\forall q \in N / t_{qij} \in Ts\}$$

L'ensemble Ts doit être redéfini pour l'ensemble des macro-modes d'utilisation que l'on forme à cette itération de la procédure de simplification du syncChart. L'ensemble Ts ainsi formé permettra ensuite de renouveler la procédure afin de trouver de nouveaux macro-modes d'utilisation ayant un nombre de services en commun moindre.

L'étape suivante consiste à s'occuper des transitions partant du macro-mode m_{ij}, d'une part, et des transitions entrant dans le mode m_{ij}, d'autre part.

En ce qui concerne les transitions sortantes du macro-mode m_{ij}, pour chaque k, il se présente trois cas qui sont les suivants :

- Si $Transition_mode_{ik} \neq 0$ et $Transition_mode_{jk} \neq 0$, alors de par le moyen de construction des macro-modes d'utilisation employé les transitions $t_{q(ij)k}$ sont identiques aux transitions t_{qik} (ou t_{qjk}) en structure et syntaxe.
- Si $Transition_mode_{ik} \neq 0$ et $Transition_mode_{jk} = 0$, alors le mode m_k n'est accessible que lorsque le macro-mode m_i est actif. Les transitions $t_{q(ij)k}$ sont formées des t_{qik} et une condition "$In(m_i)$" doit être ajoutée à chaque $Conditions_Transition_{qij}$. Le signal $In(m_i)$ certifie qu'on est dans le mode d'utilisation m_i avant d'autoriser la transition du macro-mode m_{ij} vers le mode m_k. Il faudra éventuellement le programmer à l'intérieur du syncChart.
- De même, si $Transition_mode_{ik} = 0$ et $Transition_mode_{jk} \neq 0$, alors le mode m_k n'est accessible que lorsque le macro-mode m_j est actif. Les transitions $t_{q(ij)k}$ sont formées des t_{qjk} et on ajoute ici une condition "$In(m_j)$" à chaque $Conditions_Transition_{qij}$.

Le problème des transitions sortantes traité, on modifie l'ensemble T_{cm} afin de prendre en compte les modifications intervenues. On y ajoute toutes les transitions $t_{q(ij)k}$, puis on enlève les transitions des modes m_i et m_j vers les autres modes d'utilisation restant. L'ensemble T_{cm} est alors défini par :

$$T_{cm} = T_{cm} \cup \{t_{q(ij)k},\ q \in N, k \in J\} \setminus \{\{\forall a \in N / t_{aik} \in T_{cm}\} \cup \{\forall b \in N / t_{bjk} \in T_{cm}\}\}$$

Enfin, pour les transitions entrantes dans le macro-mode m_{ij}, le problème se situe dans la constellation gérant les activités des modes m_i et m_j. En effet, comme on l'a vu, il faut, dans chacune des constellations d'un macro-état, déterminer quelle étoile est activée à l'entrée dans ce macro-état. On doit donc, de même, déterminer quel mode d'utilisation m_i ou m_j est activé à l'entrée dans le macro-mode m_{ij}. Une fois encore, on peut distinguer trois scenarii pour chaque k :

- Si $Transition_mode_{ki} \neq 0$ et $Transition_mode_{kj} = 0$, quand on vient dans le macro-mode m_{ij} en provenance du mode m_k, on va toujours dans le mode m_i. Les transitions $t_{qk(ij)}$ sont les transitions t_{qki} auxquelles on ajoute à chacune d'elle une commande supplémentaire : e_i.
- Si $Transition_mode_{ki} = 0$ et $Transition_mode_{kj} \neq 0$, quand on vient dans le macro-mode m_{ij} en provenance du mode m_k, on va toujours dans le mode m_j. Les transitions $t_{qk(ij)}$ sont les transitions t_{qkj} auxquelles on ajoute, ici, à chacune d'elle la commande supplémentaire : e_j.
- Si $Transition_mode_{ki} \neq 0$ et $Transition_mode_{kj} \neq 0$, on a les deux cas précédents réunis en un seul. Dans ce cas, pour garder le même comportement,

il faut différencier les deux possibilités. Les transitions $t_{qk(ij)}$ sont, elles, la réunion de l'ensemble des transitions t_{aki} et t_{bkj} auxquelles on a ajouté les signaux respectifs e_i et e_j au niveau des commandes effectuées, donnant ainsi le mode d'utilisation engagé suite à la transition en provenance du mode m_k.

La figure 4.5 donne un exemple de signaux d'entrée dans un macro-mode :

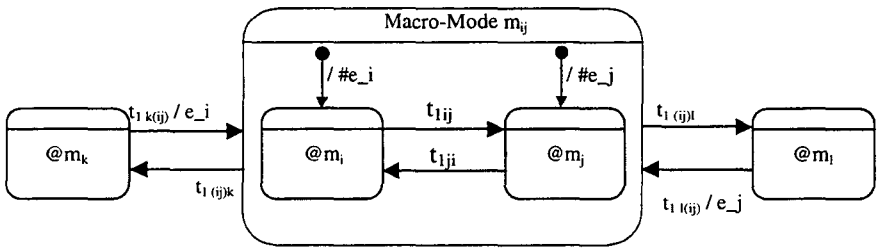


Figure 4.5 : Signaux d'entrée dans un macro-mode d'utilisation.

Lorsque toutes les transitions entrantes sont traitées, on s'occupe des signaux e_i et e_j qu'on vient d'introduire. En effet, e_i et e_j indiquent dans quel mode d'utilisation m_i ou m_j on est, à l'entrée dans le macro-mode m_{ij}. Cela se reflète dans le syncChart du macro-mode m_{ij} par une flèche entrante sur le mode d'utilisation étiqueté du signal #e_i pour le mode d'utilisation m_i et du signal #e_j dans le cas du mode d'utilisation m_j. Le "#" indique que l'effet est immédiat et qu'il est pris en compte dès l'entrée dans le macro-mode m_{ij}.

Après cette étape, des simplifications peuvent encore avoir lieu en relation avec les signaux e_i et e_j en eux-mêmes. En effet, si un et un seul de ces signaux (e_i ou e_j) existe pour le macro-mode m_{ij}, il n'est pas nécessaire de l'expliciter : le mode d'utilisation par défaut du macro-mode m_{ij} est alors simplement désigné par une flèche entrante dans le mode d'utilisation correspondant au signal généré.

Pour finir, on modifie encore une fois l'ensemble T_{cm} pour prendre en compte les transitions entrantes dans le macro-mode m_{ij} : on y ajoute les $t_{qk(ij)}$ puis on y enlève les transitions de l'ensemble du reste des modes vers les modes m_i et m_j. Cela se traduit alors comme suit :

$$T_{cm} = T_{cm} \cup \{t_{qk(ij)}, q \in N, k \in J\} \setminus \{\{\forall a \in N / t_{aki} \in T_{cm}\} \cup \{\forall b \in N / t_{bkj} \in T_{cm}\}\}$$

La figure 4.6 ci dessous illustre toutes les modifications apportées à l'ensemble T_{cm} par rapport aux transitions entrantes et sortantes du macro-mode m_{ij} créé :

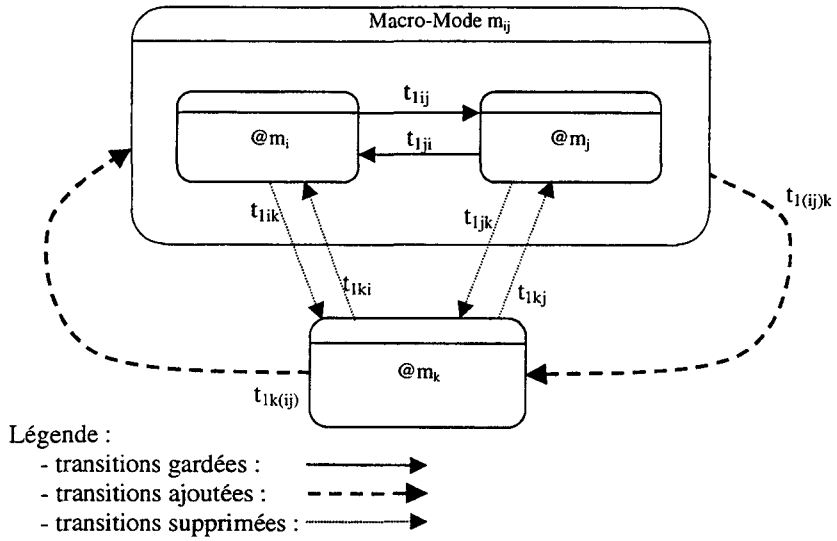


Figure 4.6 : Modification des transitions entrantes et sortantes.

IV.3.2.3.2- Structure interne du macro-mode d'utilisation formé :

Comme nous l'avons dit, la méthode s'appuie sur le fait que plus deux modes d'utilisation ont de services en commun, plus il est intéressant de les encapsuler afin d'augmenter la lisibilité du syncChart. En effet, si un service externe n'apparaît qu'une seule fois, le syncChart sera plus clair aux yeux du concepteur-réalisateur. Il convient donc d'organiser l'ensemble des services au sein de chaque macro-mode d'utilisation formé ou existant.

Pour ce faire, il faut distinguer les services communs des autres pour chacun des (macro-) modes d'utilisation situés un même niveau hiérarchique de description : i.e. au sein d'un même macro-état du syncChart. On met chaque service (ou graphe de services) commun aux modes m_i et m_j de m_{ij} dans des constellations orthogonales dans m_{ij}. Une constellation est ensuite réservée au graphe d'état régissant le comportement des modes d'utilisation m_i et m_j encapsulés : il est une copie du syncChart précédent restreint au graphe des modes d'utilisation agrégés. Le comportement interne m_i-m_j n'est ainsi pas altéré.

En ce qui concerne les services (ou graphes de services) qui sont propres à chaque macro-mode d'utilisation (i.e. m_i et m_j), ils restent dans leur mode d'utilisation où ils sont considérés en constellations distinctes et orthogonales, comme précédemment. Ces services ne restent disponibles que si on entre dans le mode d'utilisation considéré ou dans un autre mode d'utilisation où l'un de ces services apparaît.

IV.3.2.4- syncChart final :

Le syncChart final est obtenu en réitérant l'algorithme précédent (de IV.3.2.1 à IV.3.2.3) qui modifie les ensembles Ms et Ts tout au long de son déroulement. De cette façon, l'algorithme de simplification du syncChart global est itératif. La condition d'arrêt de l'algorithme se fait dans deux cas :

- il n'y a plus qu'un seul macro-mode dans Ms, ce qui se traduit par Card(Ms) = 1,
- il n'existe plus de paire de modes (ou macro-mode) d'utilisation de Ms encapsulables, ce qui se traduit par Card(Mp) = 0.

Une fois les macro-modes établis, on peut regarder les transitions entre ceux-ci. En effet, il faut encore regarder comment on sort d'un macro-mode pour chacune des transitions sortantes : suite à une terminaison normale ou un avortement :

- On peut sortir d'un macro-mode par une terminaison normale si et seulement si chacun des services spécifiques du macro-mode possède un état final.
- On sort généralement d'un macro-mode par un avortement. Le choix entre un avortement faible ou fort se fait suivant un critère : la préemptabilité de chacun des services spécifiques du macro-mode. L'avortement ne pourra être fort que si tous les services spécifiques sont préemptifs, sinon l'avortement sera faible, donnant ainsi la possibilité de terminer la dernière action de chacun des services non-préemptifs.

Après la définition des différentes possibilités de changement de mode et avec l'algorithme précédent de simplification-clarification par hiérarchisation du syncChart de base, on obtient le comportement global du syncChart. Il est maintenant possible de regarder au niveau de chacun des services ce qu'offre la hiérarchisation du graphe des modes d'utilisation.

IV.3.3- Services dans le syncChart :

La hiérarchisation des services en modes d'utilisation d'une part, puis, ensuite en macro-mode d'utilisation d'autre part offrent plusieurs possibilités de simplifications au niveau des services. En effet, les conditions d'activation peuvent se voir réduites de par l'introduction d'état de variables induites dans les macro-modes d'utilisation offerts par le modèle des SYNCCHARTS. De plus, la description de fonctionnement interne de chaque service est alors possible en introduisant les différentes versions dans l'étoile du

service (cf. III.2.2.4). Ces cas peuvent alors induire à leur tour les modes de marche, comme on l'a introduit sommairement au chapitre III (cf. III.2.4) en fonction de l'état des ressources de chaque service.

IV.3.3.1- Simplification des conditions d'activation des services dans le syncChart :

A ce niveau, les simplifications seront plus efficaces que dans le modèle externe car la structure globale des macro-modes et des services dans le syncChart est très bien hiérarchisée, ce que ne propose pas le modèle externe.

De part la structure des conditions d'activation des services, il y a deux possibilités de simplification : au niveau des conditions d'utilisation et au niveau des requêtes d'activation. Ces simplifications sont d'autant plus efficaces que le service n'apparaît qu'une seule fois dans le syncChart. Cependant, dans le cas où le service apparaîtrait plusieurs fois dans le syncChart global, les simplifications se doivent d'être faites dans chaque macro-mode d'utilisation où le service est présent.

IV.3.3.1.1- Les conditions d'utilisation :

Un macro-mode d'utilisation peut induire une condition d'utilisation. En effet, l'entrée dans un macro-état se fait suivant des conditions de changement de mode. Certaines de ces conditions de changement de mode d'utilisation sont parfois vraies tant qu'on reste dans le macro-état, elles deviennent alors implicites au macro-état. En ce sens, au niveau des conditions d'utilisation de chaque service du macro-mode, si la condition en question apparaît dans toutes les conditions d'utilisation du service, on peut en quelque sorte les "supprimer", puisque la condition est implicite.

Un exemple typique de ce genre de cas est celui du "Marche-Arrêt" d'un équipement. En effet, la plupart des services requièrent que l'équipement soit sous tension, i.e. en marche, afin qu'ils puissent fonctionner. Au niveau des modes d'utilisation, il est courant de définir au moins deux modes : le mode qu'on peut qualifier de "Hors Tension" et le mode "En Tension". De ce fait, tous les services appartenant au mode "En Tension" n'ont plus besoin de la condition indiquant que l'équipement est sous tension. La figure 4.7 montre ce cas.

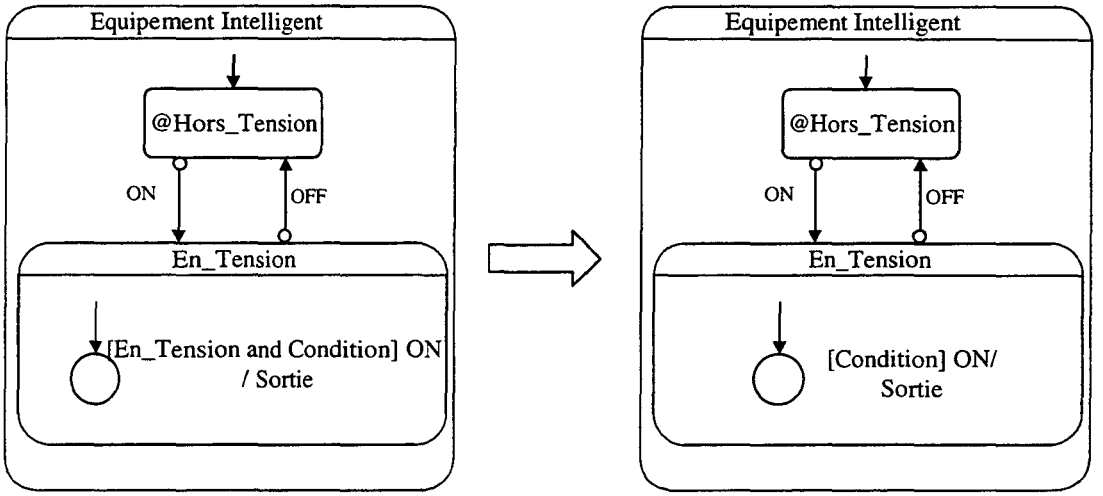


Figure 4.7 : Cas de condition d'utilisation induite.

Ce cas est assez simple mais il met bien en évidence l'apport qu'offrent les SYNCCHARTS. Ces états implicites sont cependant difficiles à expliciter de manière automatique car ils sont assez subjectifs et doivent donc être l'objet de toutes les attentions de la part du concepteur.

Il existe une deuxième possibilité de simplification plus facilement identifiable. Il s'agit du cas où l'entrée dans le macro-mode est sujette à une condition de transition particulière. Cette condition doit alors être présente dans toutes les conditions de transition vers ce macro-mode et reste vraie dans le mode. Cette dernière condition implique qu'aucun service du macro-mode ou acteur ne change cette condition sans entraîner un changement de mode. Si m_j est le macro-mode étudié, cela se traduit par :

$\text{Si } \exists b_h \in Ba \cap Bm /$

$\forall k \in J, \forall q \in N, b_h \subset t_{qij}$

$\text{ET } (\forall i \in I, \text{Appartenance}_{ij} = 1, b_h \notin \text{Sorties}(s_i))$

Alors, il est possible de remplacer b_h dans toutes les Conditions_Utilisationq(si) par un "1" logique où b_h apparaît, i.e. où :

$\forall q \in N, b_h \subset \text{Conditions_Utilisation}_q(s_i)$

Ainsi, si un service n'apparaît qu'une seule fois dans le syncChart, la condition d'utilisation pourrait alors être simplifiée au maximum de par son appartenance à un macro-mode d'utilisation. Si la condition d'utilisation du service n'est qu'une simple question de contexte (i.e. de mode d'utilisation), elle devient un "1" logique.

De plus, on peut remarquer que ces simplifications peuvent aussi être appliquées aux conditions de transitions entre modes d'utilisation appartenant au macro-mode d'utilisation suivant des critères semblables.

IV.3.3.1.2- Les requêtes d'activation :

De même, l'entrée dans un macro-mode peut entraîner l'exécution automatique d'un service. Par rapport aux conditions d'utilisation qu'on a abordé ci avant, la simplification ne peut se faire que sous une condition bien précise. En effet, pour que l'exécution d'un service devienne implicite suite à l'entrée dans un macro-mode d'utilisation, il faut que toutes les requêtes de changement de mode vers ce macro-mode soient identiques à au moins une des requêtes d'activation du service. Si m_j est le macro-mode d'utilisation, on obtient les conditions suivantes :

Si $\exists i \in I$, $Appartenance_{ij} = 1$ ET $\exists r_h \in Ra \cap Rm$ /

$\forall k \in J$, $\forall p \in N$, $r_h = RequêteM_{pkj}$ ET $\forall p \in N$, $r_h = RequêteA_q(s_i)$

Alors, r_h devient implicite pour le service s_i dans le macro-mode m_j et est alors remplacée par un "1" logique. La figure 4.8 montre un exemple de requête induite :

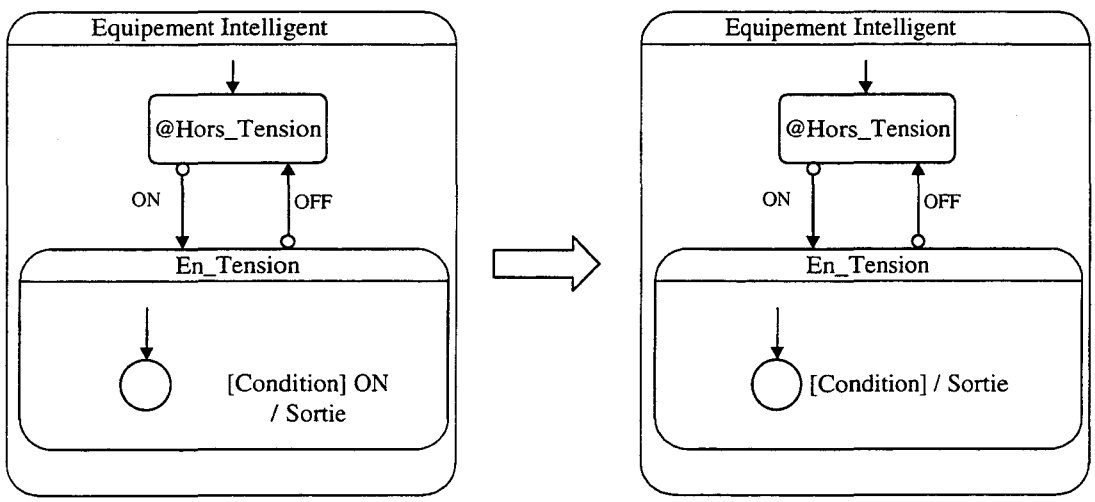


Figure 4.8 : Cas de requête induite.

Par rapport au paragraphe précédent, ces simplifications au niveau des requêtes d'activation ne sont valables que pour les services du macro-mode et ne peuvent, en aucun cas s'appliquer aux requêtes de changement entre modes d'utilisation appartenant au macro-mode m_j.

Ces simplifications faites au niveau des conditions d'activation des services du syncChart peuvent amener à alléger l'ensemble des données nécessaires à la description du comportement de l'équipement intelligent. Les requis au niveau de l'architecture opérationnelle pour la réalisation future de ce dernier sont alors diminués d'autant.

Cela apporte aussi un éclaircissement au niveau des conditions de fonctionnement d'un service. Pour qu'un service puisse fonctionner, on a les conditions préliminaires suivantes :

- le service doit appartenir au macro-mode d'utilisation courant,
- les conditions d'utilisation du service doivent être remplies,
- le service doit avoir été demandé par une requête explicite ou implicite.

Il existe encore une autre condition qui implique le fonctionnement du service : il doit être disponible. Ce fonctionnement interne est plus exactement décrit au paragraphe suivant.

IV.3.3.2- Fonctionnement interne de chaque service dans le syncChart :

Chaque service est dans une constellation d'un macro-mode. A ce titre, chaque service peut être mis dans un état SYNCCHARTS. On peut alors, dans cet état, décrire plus précisément le fonctionnement interne du service.

IV.3.3.2.1- syncChart d'un service :

Le syncChart du service est souvent la dernière étape à décrire. Dans le cas le plus général, on obtient un automate plus ou moins complexe à l'intérieur de l'état pour chaque sortie du service : un pour la sortie principale et un pour chaque sortie secondaire. Ces automates sont alors disposés dans des constellations orthogonales.

Un cas le simple de description d'un service périodique peut se résumer par la figure 4.9. La sortie est alors générée à chaque fois que le signal périodique TOP apparaît.

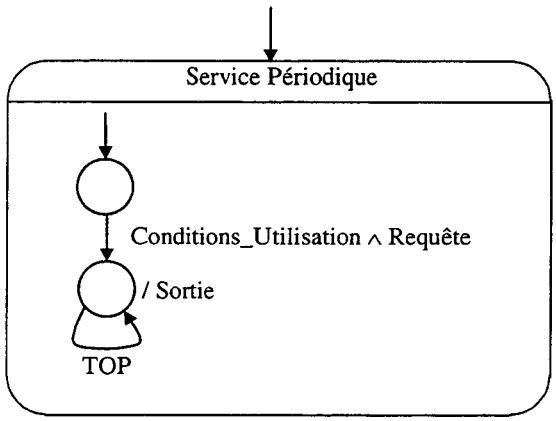


Figure 4.9 : Exemple de syncChart d'un service périodique.

La sortie doit alors être déclarée comme un signal. Ce signal peut même être "évolué" dans le sens où il peut être une fonction. Les conditions d'activation sont uniques et composées d'une Conditions_Utilisation et de sa requête associé.

On décrit, ici, la version nominale d'un service, i.e. sans défaillance. Il est cependant possible de prendre en compte plusieurs cas de fonctionnement du service en fonction de l'état de ses ressources propres.

IV.3.3.2.2- Exemple de différentes versions d'un service :

La structure fortement modulaire et hiérarchique du modèle SYNCCHARTS permet de fournir un modèle interne de chaque service. Si après avoir vérifié le fonctionnement normal du syncChart (i.e. sans défaillance) et s'il a été prévu des versions dégradées d'un service, on peut les expliciter dans le syncChart de ce service.

Les cas possibles de versions dégradées d'un service sont multiples. Il n'est pas ici le sujet de développer les différentes possibilités palliatives à la suite d'une défaillance. On donne simplement un exemple illustrant les possibilités offertes par le modèle SYNCCHARTS afin de prendre en compte les défaillances. On se donne le cas d'un service de mesure de température à partir de la mesure de deux capteurs indépendants C_1 et C_2. La version nominale du service qu'on nomme T_{12} donne la moyenne des deux mesures fournies respectivement par C_1 et C_2. La spécification prévoit cependant que les capteurs peuvent devenir défaillants. On prévoit alors deux versions dégradées :

- dans le cas de la défaillance du capteur C_2, on définit la version dégradée T_1 qui donne la mesure du capteur C_1,
- dans le cas de la défaillance du capteur C_1, on définit la version dégradée T_2 qui donne la mesure du capteur C_2.

Lorsque les deux capteurs deviennent l'un à la suite de l'autre défaillants, le service devient indisponible.

Dans le syncChart de ce service, on obtient alors le schéma donné par la figure 4.10. Ce syncChart est une solution parmi d'autres pour prendre en compte les différentes versions du service de mesure de température.

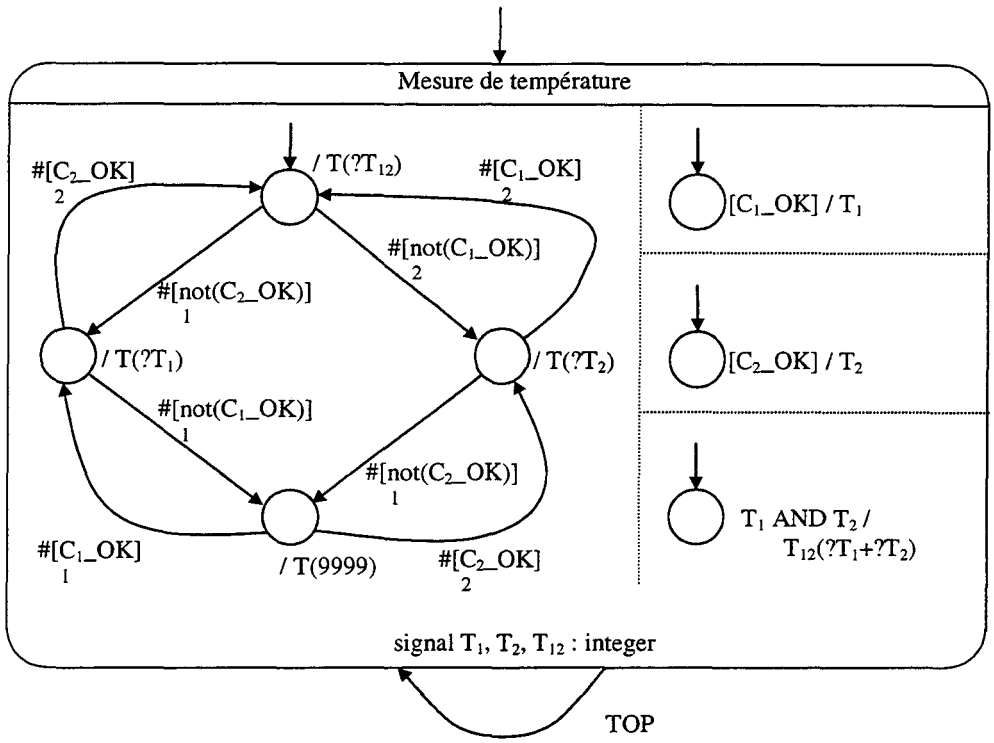


Figure 4.10 : syncChart évolué d'un service de mesure de température.

Cet exemple illustre bien les possibilités et la complexité de prise en compte des différentes versions d'un service. Par ailleurs, on peut noter dans ce syncChart que les différents traitements sont internes au service et ne transparaissent pas à l'extérieur de l'état. La seule information qui ressort est la température T qui est, ici, la seule donnée qui nous intéresse pour les versions possibles. En ce qui concerne l'état des capteurs, ces informations doivent être traitées séparément du service de mesure de la température à un niveau hiérarchique supérieur, c'est pourquoi ils n'apparaissent pas dans la déclaration des signaux locaux du syncChart. Le cas défaillance du service de mesure est caractérisé par la mise à une valeur extrême (9999) de la sortie T. Cette information pourra alors être utilisée dans le syncChart global par ses consommateurs afin de prendre les mesures adéquates.

On peut, ici, compléter les dernières conditions qui permettent l'exécution du service dont on a donné les conditions préliminaires un peu plus haut. Pour qu'un service puisse fonctionner, on doit réunir les conditions suivantes :

- le service doit appartenir au macro-mode d'utilisation courant,
- les conditions d'utilisation du service doivent être remplies,
- le service doit avoir été demandé par une requête explicite ou implicite,
- il doit exister au moins une version, nominale ou dégradée, disponible,
- le service doit avoir été choisi par l'ordonnanceur pour s'exécuter au moins partiellement.

IV.3.3.3- Les modes de marche :

Le problème des modes de marche fait immédiatement suite au cas défaillant d'un ou plusieurs services. Il concerne plus particulièrement la sûreté de fonctionnement des modes d'utilisation comme on l'a déjà vu au paragraphe III.3.2.4.2.3. avec les services indispensables qui peuvent devenir indisponibles suite à une défaillance. On ne revient pas ici sur le moyen de résoudre les transitions spontanées [BIL94] vers un autre macro-mode puisqu'il a déjà été abordé et que ces transitions sont déjà prises en compte de fait dans le syncChart. Ces transitions sont cependant directement en relation avec ce qui a été appellé les modes de marche, qui sont, on le rappelle, la conjonction d'un mode d'utilisation voulu par les utilisateurs et de l'état des ressources de l'ensemble des services offerts dans le macro-mode d'utilisation.

L'introduction dans le paragraphe précédent de versions dégradées des services peut induire de nouvelles transitions spontanées. En effet, un service qui fournit une version dégradée d'une de ses sorties rend malgré tout, par rapport à sa version nominale, un service de moindre qualité. Cela n'a généralement pas d'action sensible sur le macro-mode d'utilisation courant puisque le service est globalement rendu. Cependant, pour des raisons de sécurité, un macro-mode d'utilisation peut nécessiter qu'un service soit rendu dans sa version nominale ou au pire uniquement dans certaines versions dégradées. De la même manière, plusieurs défaillances successives peuvent amener plus d'un service à rendre une version dégradée. Dans ce cas, il se peut que dans l'objectif global du macro-mode d'utilisation ne puisse plus être rendu dans des conditions suffisantes au point de vue sécurité, en termes de qualité de service rendu ou suivant d'autres critères. Il est alors intéressant de donner des profils de modes d'utilisation en deçà desquels il devient nécessaire de changer de mode d'utilisation comme on l'a vu au paragraphe III.3.2.4.2.3.

Les profils d'utilisation minimum se définissent pour chaque macro-mode d'utilisation en fonction des services qu'ils offrent et des versions que chacun d'entre-eux permette. Pour chaque macro-mode d'utilisation on forme alors une sorte de tableau qui comporte en colonne tous les services du macro-mode. On définit alors en ligne toutes les combinaisons possibles de profils d'utilisation en fonction des versions de chaque service.

Par exemple, si on prend trois services s_1, s_2 et s_3 d'un mode possédant chacun une version nominale, une version dégradée et la "version indisponible". Ce qui nous donne ici vingt sept profils différents. Pour chacun de ces profils d'utilisation, il faut alors définir quelle marche à suivre : mode correct ou mode incorrect. On obtient le tableau suivant :

s_1	s_2	s_3	Mode
n_1	n_2	n_3	correct
n_1	n_2	d_3	correct
n_1	n_2	i_3	incorrect
n_1	d_2	n_3	correct
n_1	d_2	d_3	correct
n_1	d_2	i_3	incorrect
n_1	i_2	n_3	correct
n_1	i_2	d_3	correct
n_1	i_2	i_3	incorrect
d_1	n_2	n_3	correct
d_1	n_2	d_3	correct
d_1	n_2	i_3	incorrect
d_1	d_2	n_3	correct
d_1	d_2	d_3	incorrect
d_1	d_2	i_3	incorrect
d_1	i_2	n_3	correct
d_1	i_2	d_3	correct
d_1	i_2	i_3	incorrect
i_1	n_2	n_3	incorrect
i_1	n_2	d_3	incorrect
i_1	n_2	i_3	incorrect
i_1	d_2	n_3	incorrect
i_1	d_2	d_3	incorrect
i_1	d_2	i_3	incorrect
i_1	i_2	n_3	incorrect
i_1	i_2	d_3	incorrect
i_1	i_2	i_3	incorrect

Lorsque le profil du mode par rapport à la disponibilité de ses services est incorrect, il faut en interdire l'entrée (i.e. autoriser uniquement des transitions vers des modes dont

le profil actuel est correct) et qu'il faut forcer un changement de mode lorsque le profil du mode courant devient incorrect.

Pour l'entrée possible dans un mode correct, on pratique comme au paragraphe III.3.2.4.2.3. où on ajoute une condition d'entrée dans le macro-mode m_j à toutes les t_{qij}. Cette condition s'obtient en utilisant les méthodes classiques de l'automatique (tableau de Karnaugh, par exemple) qui permettent de diminuer au maximum la condition d'entrée dans le mode suivant le profil d'utilisation du mode m_j.

La difficulté pour les changements de mode suite à un profil insuffisant réside dans le fait que le changement de mode d'utilisation ne peut se faire que vers un mode d'utilisation acceptant le dit profil incorrect. Cela implique de connaître tous les profils de tous les macro-modes d'utilisation.

La prise en compte éventuelle des modes de marche termine alors la description de l'équipement suivant les SYNCCHARTS. Dans le syncChart de l'équipement, il faut alors définir correctement tous les signaux locaux, entrées, sorties, commentaires ... etc. Il peut enfin parfois être nécessaire d'introduire une ou plusieurs constellations dans le firmament d'un macro-mode afin de pouvoir prendre en compte l'évolution de variables pilotées par les acteurs externes.

IV.3.4- Vérification du syncChart obtenu :

La première vérification du syncChart de l'équipement intelligent se fait par l'intermédiaire de compilateurs associé aux SYNCCHARTS. Les compilateurs C et ESTEREL permettent entre autres de vérifier la cohérence comportementale du modèle sur plusieurs points :

- les erreurs de déclarations de variables, de signaux et de constantes,
- les problèmes de priorités au niveau des transitions entre modes,
- les problèmes de boucles induites dues au caractère synchrone des SYNCCHARTS.

Ces vérifications préliminaires permettent alors de valider le modèle obtenu au point de vue de la complétude de la spécification SYNCCHARTS. Une fois les diverses compilations réussies et validées, le syncChart peut ensuite être simulé avec le logiciel XES au niveau du comportement. Il est alors possible de vérifier, à partir de tests qu'il faut définir, la cohérence du syncChart obtenu. On vérifie alors plusieurs points :

- tous les macro-modes d'utilisation sont accessibles au moins une fois,
- le comportement des macro-modes est bien celui spécifié,

- tous les services peuvent être exécutés suivant leurs critères fonctionnels relatifs à leurs conditions d'activation et leur périodicité propre,
- tous les macro-modes d'utilisation et leurs services associés rendent bien leur rôle comme il a été prévu dans la spécification fonctionnelle.

Les SYNCCHARTS ne permettent par contre pas directement de vérifier le comportement temporel de par leur nature synchrone. L'introduction de compteurs et/ou de chien de garde peut remédier au moins partiellement à cela.

Les SYNCCHARTS ne permettent pas non plus d'aborder directement l'ordonnancement des différents services. Cependant, de part la structure graphique hiérarchique des SYNCCHARTS, il est possible de discerner quels services seront susceptibles d'être activés durant la même période et de pouvoir ainsi choisir au mieux l'algorithme et la politique d'ordonnancement qui répondront aux contraintes temporelles énoncées dans le cahier des charges.

IV.4- Conclusion :

Nous avons proposé une méthode permettant de passer du modèle fonctionnel proposé par le modèle externe au modèle comportemental SYNCCHARTS. En s'appuyant sur le modèle externe, on a pu encapsuler et ainsi hiérarchiser les modes d'utilisation afin de proposer une vue claire et simplifiée du comportement global des différents services proposés par un équipement intelligent. De plus, les SYNCCHARTS permettent de compléter le modèle en proposant une description interne du fonctionnement avec les versions de chaque service offrant ainsi la possibilité d'accroître la robustesse du service et d'en vérifier localement le fonctionnement. Les différents cas de modes de marche de l'équipement en sont ainsi accrus.

Les SYNCCHARTS permettent de plus de valider la spécification obtenue du syncChart d'un équipement intelligent par l'intermédiaire des différents compilateurs nécessaires à l'obtention d'un programme simulable sous XES. Ce simulateur donne alors l'occasion de vérifier le comportement de l'équipement en regardant la cohérence au niveau des besoins exprimés dans le cahier des charges par rapport au modèle SYNCCHARTS obtenu. Le syncChart permet ainsi de valider le comportement global de l'équipement.

Les SYNCCHARTS ne permettent par contre pas de vérifier directement la cohérence temporelle des services de l'équipement. Cependant, de par la structure graphique hiérarchisée, ils guident le concepteur-réalisateur dans son choix de la politique d'ordonnancement optimale pour l'ensemble des services de l'équipement en relation avec les spécifications temporelles données par le modèle externe.

Après cette étape de validation, nous disposons d'un cahier des charges complet et cohérent du système à concevoir. C'est un point d'entrée de l'étape de spécification matérielle et logicielle. Les outils de conception de logiciels développés au LAMI [TAI99] ou dans le cadre du projet LARII (Logiciel d'Aide à la réalisation d'Instruments Intelligents) [ARE99] peuvent être utilisés lorsque le choix de l'architecture matérielle a été effectué. Différentes validations restent à faire, en particulier, celles qui concernent l'architecture opérationnelle : temps de réponse, encombrement mémoire ...

Afin d'illustrer les méthodes de spécifications fonctionnelles du modèle externe et de passage au modèle SYNCCHARTS, on se propose de les appliquer à un exemple de régulateur de vitesse.

Références :

[AND96] André C., "Representation and Analysis of Reactive Behaviors : a SyncChronous Approach", Papier invité, CESA'96, pp. 19-29, Lille, 1996.

[AND98] André C., Boufaïed H., Dissoubray S., "SyncCharts : un modèle graphique synchrone pour systèmes réactifs complexes", RTS'98 (Real-Time Systems 1998), pp. 175-193, Paris, 1998.

[ARE99] AREMI, "LARII : Guide de l'utilisateur", rapport interne, 1999.

[BAY98] Bayart M., Lemaire E., Péraldi M.A., André C., "External model and SyncCharts description of an automobile cruise control", Proc. IFAC Intelligent Components for Vehicle ICV'98, pp. 135-140, Séville, Espagne, 1998.

[BAY99] Bayart M., Lemaire E., Péraldi M.A., André C., "External model and SyncCharts description of an automobile cruise control", article étendu, Control Engineering Practice, Elsevier, octobre 1999.

[HAR87] Harel D., "StateCharts : a visual formalism for complex system", Science Computer Program, vol. 8, n°3, juin 1987, pp. 514-530.

[MAR90] Maraninchi F., "ARGOS : un langage pour la conception, la description et la validation des systèmes réactifs", Thèse de l'Université Joseph Fourier, Grenoble I, 1990.

[TAI99] Tailland J., Foulloy E., Benoît E., "Automatic Code Generation from Internal Intelligent Instrument Mode", Proc. INNOCAP'99 : Réseaux de Capteurs et Communications, Grenoble, 28-29 avril 1999.

Chapitre V :
Application à un Régulateur de Vitesse de Croisière

Table des Matières :

V.1- Introduction : 135

V.2- Présentation du régulateur de vitesse d'un véhicule : 135
V.2.1- Le problème de la régulation de vitesse : 136
V.2.2- Autres besoins : 136

V.3- Spécifications des besoins du régulateur de vitesse : 137
V.3.1- Les consommateurs externes : 137
V.3.2- Les services externes : 138
V.3.2.1- Service Commande_Manuelle : 138
V.3.2.2- Service Calibrage : 139
V.3.2.3- Services Elab_Distance : 139
V.3.2.4- Service Elab_Vitesse : 140
V.3.2.5- Service Commande_Régulée : 140
V.3.2.6- Service Saisie_VR : 141
V.3.2.7- Service Elab_Consommation : 142
V.3.2.8- Service Elab_Vitesse_Moyenne : 142
V.3.2.9- Service Saisie_Trajet : 143
V.3.2.10- Récapitulatif des données du régulateur de vitesse : 143
V.3.2.10.1- L'ensemble des consommateurs externes : 144
V.3.2.10.2- L'ensemble des services externes : 144
V.3.2.10.3- L'ensemble des acteurs externes : 150
V.3.2.10.4- Ensembles de la spécification : 151

V.4- Analyse de la spécification obtenue du régulateur de vitesse : 152
V.4.1- Analyse de l'ensemble des services externes du régulateur : 152
V.4.2- Graphe des modes d'utilisation du régulateur : 152
V.4.2.1- Formation des modes d'utilisation : 152
V.4.2.2- Transitions entre les modes d'utilisation : 155
V.4.2.2.1- Mode Stop : 155
V.4.2.2.2- Mode Conduite : 155
V.4.2.2.3- Mode Configuration : 156

V.4.2.2.4- Mode Régulation : 156
V.4.2.2.5- Mode Freinage_en_Régulation : 157
V.4.3- Vérification du graphe des modes d'utilisation du régulateur : 158
V.4.3.1- Vérifications préliminaires : 158
V.4.3.2- Vérifications sur l'automate du régulateur : 159
V.4.3.2.1- Déterminisme du graphe du régulateur : 159
V.4.3.2.2- Cohérence du graphe du régulateur : 160

V.5- syncChart du régulateur de vitesse : 162
V.5.1- Obtention du syncChart équivalent au graphe des modes d'utilisation : 162
V.5.2- Simplifications et fonctionnement interne des services : 167
V.5.2.1- Mode Stop : 168
V.5.2.2- Macro-mode En_Fonctionnement : 168
V.5.2.3- Mode Configuration : 169
V.5.2.4- Macro-mode En_Circulation : 170
V.5.2.5- Macro-mode Régulation_Engagée : 171
V.5.2.6- Mode Régulation : 173

V.6- Validation du modèle SYNCCHARTS : 174

V.7- Conclusion : 176

Références : 177

Application à un Régulateur de Vitesse de Croisière

V.1- Introduction :

Dans les chapitres précédents, nous avons donné une méthode permettant, dans un premier temps de vérifier que la spécification d'un cahier des charges est correcte et d'en vérifier l'aspect comportemental à l'aide du logiciel SYNCCHARTS.

On se propose, dans ce chapitre, d'utiliser cette méthode afin de spécifier l'exemple d'un régulateur de vitesse de croisière (Cruise Speed Controller ou Cruise Control en anglais). Cet exemple possède plusieurs avantages :

- Il comporte plusieurs types de comportements : une partie continue avec la régulation de la vitesse et une partie discrète pour la gestion des événements par lesquels le conducteur agit sur son environnement, ce qui en fait un exemple de système réactif hybride très intéressant.
- C'est un exemple bien connu de la communauté scientifique : il a été décrit en détails par Hatley [HAT90] qui l'a utilisé pour illustrer la méthode S.A.RT. D'autres auteurs tels P. Ward [WAR88], J.P. Calvez [CAL90], J.M. Atlee [ATL93], ont repris cet exemple pour illustrer leurs méthodes de modélisation ou de validation de leur modèle.

L'exemple du régulateur de la vitesse de croisière d'un véhicule nous semble donc bien indiqué pour illustrer notre méthode à partir du modèle externe et du modèle SYNCCHARTS que nous y avons associé.

V.2- Présentation du régulateur de vitesse d'un véhicule :

Le cahier des charges de cet exemple est tiré de celui proposé par J.P. Calvez [CAL90]. Nous avons repris le problème de la régulation en elle-même ainsi que toutes les fonctions annexes qui n'ont pas de rapport direct avec la régulation de la vitesse de croisière afin d'illustrer au mieux tous les cas pouvant se produire lors de la conception.

V.2.1- Le problème de la régulation de vitesse :

Le régulateur de vitesse est un équipement supplémentaire qui permet au conducteur de régler une consigne de vitesse pour les longs trajets. Quand la vitesse appropriée est atteinte, le conducteur peut engager la régulation. Le cahier des charges décrivant le problème du régulateur en lui-même se partage en plusieurs axes :

- La vitesse du véhicule est contrôlée par une action sur une vanne électrique qui commande l'injection du carburant dans les chambres de combustion du moteur.
- La consigne de vitesse du régulateur est fixée comme suit : lorsque la vitesse de régulation désirée est atteinte et si elle est supérieure à 50km/h, le régulateur est engagé.
- Quand le régulateur est engagé, le conducteur doit rester, à tout instant, maître de son véhicule et, par conséquent, il peut engager des actions qui modifierait la vitesse de son véhicule. Il peut le faire de deux manières différentes en ce qui concerne la vitesse : en agissant soit sur l'accélérateur, soit sur le frein. On a alors les actions suivantes :
  - en appuyant sur l'accélérateur, la vitesse du véhicule augmente et, à la fin de l'accélération, la vitesse est à nouveau commandée par le régulateur à la consigne précédemment fixée.
  - en appuyant sur le frein, le régulateur est désactivé, le conducteur peut réactiver le régulateur, à la consigne précédemment fixée, à tout moment en appuyant sur un bouton "Réengager".

Le problème des besoins de la régulation de la vitesse du véhicule est ici posé en quelques lignes. Il s'agit essentiellement d'un besoin du conducteur du véhicule. D'autres besoins sont aussi exprimés ci-après.

V.2.2- Autres besoins :

Le conducteur possède plusieurs autres besoins que nous allons définir ci-dessous. Ces besoins ne sont pas directement en rapport avec le problème du régulateur de vitesse mais peuvent éventuellement interférer avec lui. Ils sont au nombre de trois :

- A la première mise en marche et à la demande ultérieure du conducteur, une procédure de calibrage détermine le facteur de conversion entre un codeur impulsionnel situé au niveau de la roue et un calculateur. Cette procédure

permet au régulateur de s'adapter à n'importe quel type de roue dès la mise en service du régulateur de vitesse et tout au long du cycle de vie de l'équipement.
- Le conducteur peut suivre la vitesse moyenne sur un trajet donné : le conducteur enregistre le départ du trajet et peut ensuite demander la vitesse moyenne par rapport au début du trajet enregistré.
- De la même manière, le conducteur peut suivre la consommation moyenne de son véhicule : elle est fournie à chaque plein d'essence effectué, le conducteur obtient alors la consommation moyenne par rapport au dernier plein effectué.

Ces trois fonctionnalités supplémentaires demandées par le conducteur sont aussi à prendre en compte lors de la spécification des besoins qui va suivre.

Cette présentation "en langage naturel" du cahier des charges n'est évidemment pas complète. D'une part, les informations ne sont pas formalisées ce qui les rend inexploitables pour l'instant. D'autre part, il manque les informations relatives aux aspects comportementaux de l'exemple. Ces informations n'étant pas connues à priori, elles seront à définir au cours de l'étude.

V.3- Spécifications des besoins du régulateur de vitesse :

Comme on l'a décrit dans le chapitre III, il faut expliciter clairement les différents besoins des utilisateurs selon ce qui a été exprimé dans le cahier des charges. On commence donc par déterminer les différents consommateurs externes du régulateur de vitesse.

V.3.1- Les consommateurs externes :

On recense deux grands consommateurs : le *conducteur* et ce que nous appellerons le *moteur* qui représente l'environnement physique de l'ensemble régulateur de vitesse.

Les besoins exprimés par ces deux consommateurs sont :

- le calibre Cal,
- la commande manuelle de la vanne électrique CMV,
- la commande régulée de la vanne électrique CRV,
- la consommation moyenne CM,
- la distance parcourue D,
- la vitesse moyenne sur un parcours VM,
- la vitesse de régulation VR.

Les fonctions de consommations associées à chacun des consommateurs sont alors les suivantes :

- fc(Conducteur) = {CM, D, V, VM, VR}
- fc(Moteur) = {Cal, CMV, CRV}

V.3.2- Les services externes :

On a recensé huit besoins en termes de services. On les nomme ainsi :

- Calibrage,
- Commande_Manuelle,
- Commande_Régulée,
- Elab_Consommation,
- Elab_Distance,
- Elab_Vitesse,
- Elab_Vitesse_Moyenne,
- Saisie_VR.

La description de chaque service externe exprimé, est ensuite décrite plus en détail, service par service. On n'indiquera pas, dans cette description, les ressources nécessaires à la bonne marche du véhicule (batterie, moteur, roues, ...), on n'y indiquera simplement les ressources impliquées directement dans le régulateur de vitesse.

V.3.2.1- Service Commande_Manuelle :

Ce service est le service primordial du véhicule. On supposera d'ailleurs qu'il tombe que très rarement en panne. Ce service permet de donner la commande manuelle CMV en appuyant sur l'accélérateur et/ou sur le frein. Bien que le levier de vitesse influe sur les possibilités de vitesses que l'on peut atteindre, il n'agit cependant pas directement sur la commande manuelle de la vanne.

Ce service possède évidemment deux entrées ACC et FREIN qui indiquent respectivement la position relative de l'accélérateur et du frein. Ces deux données sont fournies par le même acteur : le conducteur. Ce service n'est pas paramétrable en cours de fonctionnement. Au niveau des conditions d'activation, ce service est "actif" dès que le moteur est en marche : on a donc une seule condition d'activation caractérisée par une condition d'utilisation unique. Cette donnée booléenne que nous nommons M_E_M sera

bien entendu fournie par le moteur. Les deux ressources essentielles et vitales à ce service sont les pédales d'accélérateur et de frein : Ped_ACC et Ped_FREIN. D'un point de vue comportemental, ce service répond à la position relative "physique" de l'accélérateur et/ou du frein. Il est donc toujours actif tant que le moteur est en marche et ne dépend donc pas du processeur. De ce fait, il n'est ni périodique ni préemptif.

V.3.2.2- Service Calibrage :

Ce service exprimé dans le cahier des charges permet de paramétrer les services Elab_Distance et Elab_Vitesse à partir des impulsions fournies par le capteur situé au niveau d'une roue. Il est actif si le moteur est en marche et dès qu'on commence la séquence de calibrage qui s'effectue sur un kilomètre. Ce service s'arrête lorsqu'on lui signale que le kilomètre a été parcouru. Dès que le kilomètre a été parcouru, on met à jour le paramètre Cal.

Ce service a donc besoin en entrée d'impulsions fournies par un capteur qu'il faudra situer au niveau d'une roue et d'une donnée Stop_KM qui devra être fournie par le conducteur qui indique que le kilomètre "étalon" a été effectué. Calibrage n'est, par contre, pas en lui-même paramétrable. Au niveau, des conditions d'activation, il faut que le moteur soit en marche, cette donnée sera fournie par le moteur via la variable booléenne M_E_M et le service est activé sur la requête Start_KM au libre choix du conducteur.

D'un point de vue comportemental, l'absence du capteur de roue implique l'abandon immédiat du service. Le processeur est ici une ressource partagée non-vitale, le service est interrompu par avortement fort.

V.3.2.3- Services Elab_Distance :

Ce service permet d'élaborer la distance D que le véhicule a parcourue depuis sa mise en service. Il n'a besoin pour son fonctionnement que d'une seule entrée : les impulsions en provenance du capteur de roue. Son paramètre est, comme on l'a dit ci-dessus, le calibre Cal mis à jour par Calibrage. Ces conditions d'activation sont qu'il faille que le service Calibrage ait été effectué avec succès au moins une fois auparavant. Cette condition sera symbolisée par la variable booléenne Cal_OK qu'on attribue tout simplement au service Calibrage déjà défini.

Ce service a un comportement non périodique : la mise à jour de la variable D dépend uniquement du nombre d'impulsions prises en compte lorsque le service

ci—dessus est actif. L'absence du capteur de roue rend, ici aussi, le service indisponible et le processeur est la seule ressource partagée qui ne sert qu'à mettre le compteur d'impulsions du service à jour et permet la mise à jour de D. Ce service est préemptif par rapport au processeur par avortement fort.

V.3.2.4- Service Elab_Vitesse :

Ce service est aussi un service primordial de la spécification puisque c'est sur lui que repose tout le problème de la régulation de vitesse. De plus, la vitesse V est un indicateur pour le conducteur ce qui lui permet de mettre le véhicule à une vitesse adaptée à la situation et en accord avec la législation en vigueur.

D'un point de vue fonctionnel, le service Elab_Vitesse ressemble beaucoup au service Elab_Distance : ils ont les mêmes ressources, les mêmes conditions d'activation et le même paramètre. Ils diffèrent au point de vue des entrées et du comportement.

En effet, il nous faut regarder comment se calcule une vitesse. La vitesse est le rapport d'une distance parcourue sur le temps mis pour parcourir cette distance. La distance parcourue est fournie par le capteur de vitesse à l'aide des impulsions qu'il envoie. En comparant le nombre d'impulsions au calibre on obtient la distance parcourue. Il faut cependant une base de temps qui sera caractérisée par un "TOP" d'horloge toutes les 500 ms. Ce signal devra être fourni par l'environnement du régulateur de vitesse.

D'un point de vue comportemental, la procédure permettant d'élaborer la vitesse se fait périodiquement toutes les 500 ms. L'absence de la ressource "Capteur_Roue" rend, comme pour Elab_Distance, le service indisponible et le processeur est la seule ressource partagée qui sert à mettre le compteur d'impulsions du service à jour et permet la mise à jour de V toutes les 500 ms.

V.3.2.5- Service Commande_Régulée :

Le service de régulation de la vitesse de croisière possède des similarités avec le service Commande_Manuelle. Ce service est actif lorsqu'une demande de régulation a été faite par la requête du conducteur Reg_ON et ceci, tant que la régulation n'est pas désactivée.

Pour donner la consigne de régulation automatique à la vanne, ce service a donc besoin de deux choses : la vitesse actuelle V du véhicule et la vitesse de régulation VR qu'a réglée le conducteur en appuyant sur le bouton associé à Reg_ON. La vitesse de

régulation VR peut ici être considérée comme un paramètre du service Commande_Régulée : VR ne change pas significativement toutes les périodes de régulation. VR est fournie par le service Saisie_VR décrit ci-dessous. De plus, on ajoute aux entrées la position du frein qui désactive temporairement le service si le frein est utilisé par le conducteur. Au niveau des conditions d'activation, il existe des conditions d'utilisation nécessaires à l'activation du service : il faut que le moteur soit en marche (M_E_M), que la pédale de frein ne soit pas pressée ($\overline{\text{FREIN}}$) et que le service Calibrage ait été effectué correctement au moins une fois auparavant (Cal_OK). Il y a ensuite deux manières d'activer le service : à partir de la requête Reg_ON ou à partir de la requête ReEngage avec la condition supplémentaire que le service Saisie_VR soit actif, ce que ce dernier signale par la variable booléenne VR_OK. Les deux requêtes Reg_ON et ReEngage sont toutes deux fournies par le conducteur.

D'un point de vue comportemental, le service Commande_Régulée est périodique de période 500 ms pour des raisons de stabilité de la régulation PID choisie [CAL90]. De plus, il existe un événement qui force l'abandon immédiat par avortement fort du service : Reg_OFF.

V.3.2.6- Service Saisie_VR :

Ce service est associé à la prise en compte de la vitesse de régulation dès l'apparition de la requête Reg_ON. Ce service a pour unique but de mémoriser la vitesse de régulation VR et de la fournir à ses utilisateurs : le conducteur et le service Commande_Régulée.

Le service Saisie_VR n'a besoin que de la seule entrée V pour obtenir ses sorties VR et VR_OK et il n'est pas paramétrable. Ce service ne marche que lorsque le véhicule a son moteur en marche, que le calibre existe et il n'enregistre la vitesse de régulation que si la vitesse était supérieure à 50 km/h lorsque le conducteur lance la requête Reg_ON. Le fait que la vitesse soit supérieure à 50 km/h est explicité par la variable booléenne V_OK qui est naturellement produite par le service Elab_Vitesse en tant que sortie secondaire.

D'un point de vue comportement, ce service est présent dès et tant qu'une régulation a été engagée et il modifie la donnée VR à l'apparition de Reg_ON. De ce fait, il n'est pas périodique et préemptif par rapport au processeur. Il n'est interrompu par avortement fort que si on désengage la régulation : i.e. à l'apparition de Reg_OFF.

V.3.2.7- Service Elab_Consommation :

Elab_Consommation calcule la consommation du véhicule entre deux pleins effectués et signalés par le conducteur et met à jour cette information. Cette fonction n'est accessible, pour des raisons de sécurité, qu'à l'arrêt du véhicule : i.e. quand le moteur ne fonctionne pas.

Pour calculer la consommation, le service a besoin de plusieurs données en entrée : le kilométrage actuel et la quantité de carburant qui a été ajoutée pour faire le plein. Le kilométrage est donné par le service Elab_Distance avec D, quant à la quantité de carburant QCarb, c'est une information qui devra être fournie par le conducteur via une interface appropriée. La seule condition pour que le service fonctionne est que le moteur ne soit pas en marche ($\overline{M_E_M}$), et que la requête de demande de consommation DCons ait été faite par le conducteur. Dans ce service, il existe un "paramètre" qui n'en ait pas vraiment un : le kilométrage au précédent plein. Ce dernier est supposé être mis à zéro physiquement à la première mise en service du véhicule, il est ensuite modifié de façon interne à chaque plein : i.e. le calcul de la consommation est exécuté puis le kilométrage courant remplace celui qui était en mémoire.

D'un point de vue comportemental, ce service est totalement apériodique, sans compter que le conducteur peut "oublier" de vérifier sa consommation à chaque fois qu'il ajoute du carburant. Les informations ne sont donc pas fiabilisées à cent pour cent : elles dépendent beaucoup du conducteur. Le service ne possède qu'une seule ressource partagée : le processeur. Il est préemptif par avortement faible puisqu'il faut que le service, s'il est actif, ait fini sa dernière action, ce qui ne valide pas forcément la consommation.

V.3.2.8- Service Elab_Vitesse_Moyenne :

Ce dernier service demandé par le conducteur propose à celui-ci la vitesse moyenne sur un trajet qu'il a préalablement défini. De ce fait, ce service a besoin de plusieurs informations : les distances actuelles au moment du début du trajet et le temps écoulé depuis le début du trajet. Pour obtenir le temps écoulé depuis le début du trajet, au lieu de prendre une horloge qui s'écoule dès que le trajet a commencé et qui risque de prendre beaucoup de temps au niveau du processeur pendant de très longues périodes, on choisit de prendre l'heure et la date courante ainsi que celle au moment du début du trajet. Ces deux données sont des fonctions couramment fournies dans un véhicule moderne. Cela permet de plus de pallier le problème d'un trajet qui se fait sur plusieurs

jours. L'objet Date_Heure doit donc être fourni par le "moteur" à l'application régulateur de vitesse.

Les entrées du service sont alors les données D et Date_Heure. En ce qui concerne ces mêmes données correspondantes au début du trajet, il s'agit ici d'un paramètre du service qu'on nomme Trajet qui contient les champs D et Date_Heure à l'instant du début du trajet. Trajet est le paramètre du service Elab_Vitesse_Moyenne et n'est la sortie d'aucun autre service défini dans le cahier des charges. On définit donc que le moteur consomme la donnée Trajet à qui on associe alors le nouveau service externe *Saisie_Trajet*. On définira Saisie_Trajet ci-après.

Les conditions d'activation du service, sont assez simples : il faut simplement que le paramètre Trajet soit défini, que le service Elab_Distance soit opérationnel (i.e. que Cal_OK soit VRAI) et que le conducteur ait demandé la vitesse moyenne. Le fait que Trajet soit défini se caractérise par la variable booléenne Trajet_OK qui sera elle-aussi fournie par Saisie_Trajet. En ce qui concerne la requête de demande de vitesse moyenne, on la nomme DVM dont l'acteur associé est le conducteur.

Le comportement du service est apériodique à la demande du conducteur. Il possède plusieurs ressources : le processeur qui est une ressource partagée et la ressource vitale qu'est la montre qui fournit la date et l'heure. Ce service est préemptif par rapport au processeur par avortement faible : on laisse là aussi exécuter la dernière action en cours.

V.3.2.9- Service Saisie_Trajet :

Ce service "configure" le service Elab_Vitesse_Moyenne en fournissant la date du début du trajet. Il ressemble à ce titre beaucoup au service Elab_Vitesse_Moyenne : ils ont les mêmes entrées, les mêmes ressources et le même comportement. Il n'est par contre pas paramétrable et il n'est activé que sur une requête du conducteur que nous nommerons STrajet (Set Trajet).

V.3.2.10- Récapitulatif des données du régulateur de vitesse :

L'ensemble de tous les services ici définis peuvent ensuite être mis en forme d'après le format fonctionnel récapitulatif qui a été développé au chapitre III. On donne ainsi tous les services de la spécification, on rectifie l'ensemble des consommateurs et on donne l'ensemble des acteurs externes du régulateur de vitesse.

V.3.2.10.1- L'ensemble des consommateurs externes :

Les consommateurs externes que nous avons vus au paragraphe V.3.1 sont toujours les mêmes : le conducteur et le moteur qui correspond au véhicule.

Les fonctions de consommations associées à chacun des consommateurs sont les suivantes :

- fc(Conducteur) = {CM, D, V, VM, VR}
- fc(Moteur) = {Cal, CMV, CRV, Trajet}

Ces données ont été modifiées au cours de la spécification pour prendre en compte la donnée Trajet.

V.3.2.10.2- L'ensemble des services externes :

Ces services sont énoncés ci-après dans l'ordre alphabétique par rapport aux besoins exprimés dans les fonctions de consommation données ci-dessus :

- Service Calibrage :
  Sorties : ((Cal) ; (Cal_OK))
  Entrées : (Impulsions, Stop_KM)
  Paramètres : ϕ
  Ressources : ((Capteur_Roue) ; (Processeur))
  Conditions Activation : {(M_E_M ; Start_KM)}
  Méthode : "Comptage d'impulsions"
  Priorité : 2
  Temps d'exécution au plus tôt : 28 s (1 kilomètre à 130 km/h)
  Périodicité : NON
  Préemptabilité : OUI

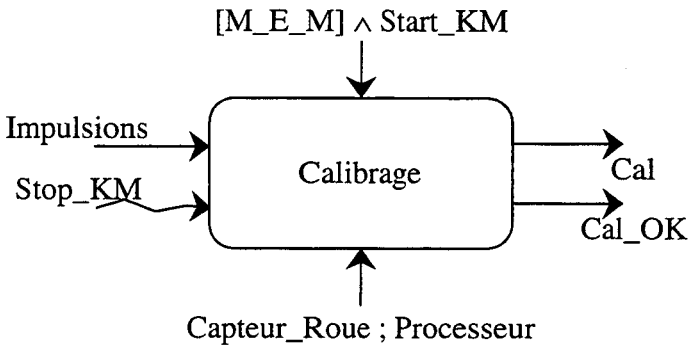


Figure 5.1 : service Calibrage

- Service Commande_Manuelle :

Sorties : ((CMV) ; ϕ)

Entrées : (ACC, FREIN)

Paramètres : ϕ

Ressources : ((Ped_ACC, Ped_FREIN) ; ϕ)

Conditions Activation : {(M_E_M ; 1)}

Méthode : "commande directe par rapport à la position des pédales"

Priorité : 1

Temps d'exécution au plus tôt : 0

Périodicité : NON

Préemptabilité : NON

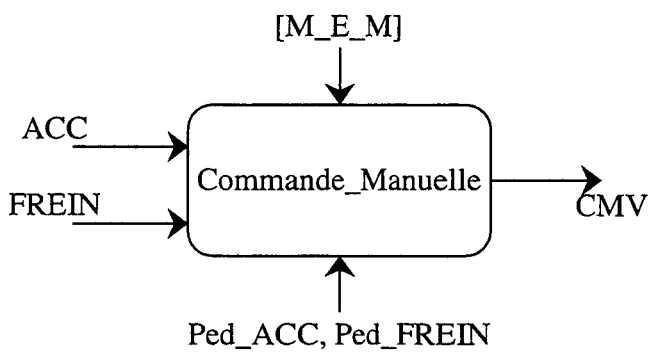


Figure 5.2 : service Commande_Manuelle

- Commande_Régulée :

Sorties : ((CRV) ; ϕ)

Entrées : (V)

Paramètres : (VR)

Ressources : ((Ped_ACC, Ped_FREIN, Capteur_Roue) ; (Processeur))

Conditions Activation :

{ $(M_E_M \wedge \overline{FREIN} \wedge Cal_OK$; Reg_ON) ;

$(M_E_M \wedge \overline{FREIN} \wedge Cal_OK \wedge VR_OK$; ReEngage)}

Méthode : "Régulation type PID"

Priorité : 4

Temps d'exécution au plus tôt : 0

Périodicité : OUI

Période : TOP = 500 ms

Préemptabilité : OUI

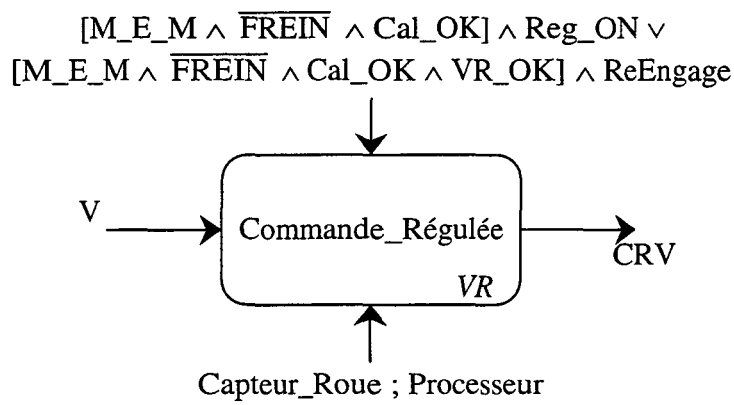


Figure 5.3 : service Commande_Régulée

Service Elab_Consommation

Sorties : ((CM) ; ϕ)

Entrées : (D, QCarb)

Paramètres : ϕ

Ressources : (ϕ ; (Processeur))

Conditions Activation : {($\overline{M_E_M}$; DCons)}

Méthode : "Calcul - Mise à jour"

Priorité : 5

Temps d'exécution au plus tôt : 0

Périodicité : NON

Préemptabilité : OUI

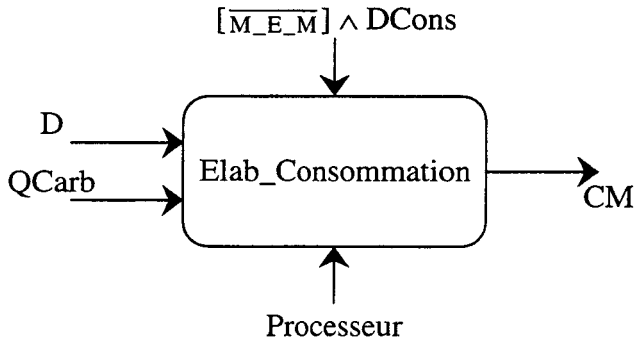


Figure 5.4 : service Elab_Consommation

Service Elab_Distance

Sorties : ((D) ; ϕ)

Entrées : (Impulsions)

Paramètres : (Cal)

Ressources : ((Capteur_Roue) ; (Processeur))

Conditions Activation : {(M_E_M ∧ Cal_OK ; 1)}

Méthode : "Comptage - Mise à jour "

Priorité : 4

Temps d'exécution au plus tôt : 0

Périodicité : NON

Préemptabilité : OUI

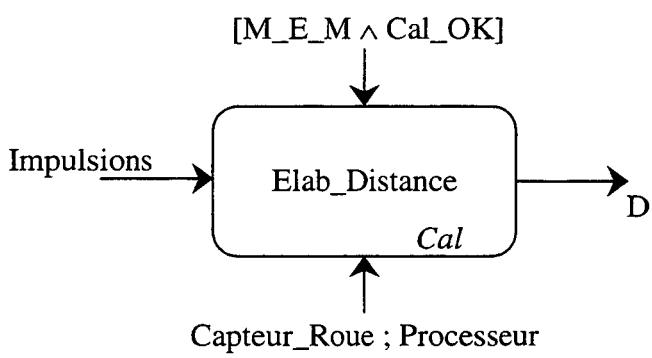


Figure 5.5 : service Elab_Distance

Service Elab_Vitesse

Sorties : ((V) ; (V_OK))

Entrées : (Impulsions)

Paramètres : (Cal)

Ressources : ((Capteur_Roue) ; (Processeur))

Conditions Activation : {(M_E_M ∧ Cal_OK ; 1)}

Méthode : "Comptage - Mise à jour périodique"

Priorité : 3

Temps d'exécution au plus tôt : 0

Périodicité : OUI

Période : TOP = 500 ms

Préemptabilité : OUI

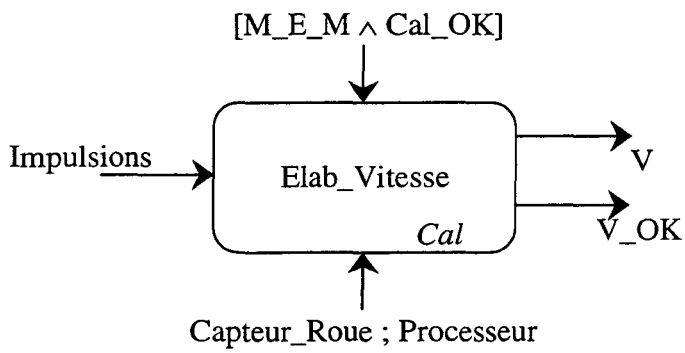


Figure 5.6 : service Elab_Vitesse

Elab_Vitesse_Moyenne :

Sorties : ((VM) ; ϕ)

Entrées : (D, Date_Heure)

Paramètres : (Trajet)

Ressources : ((Montre) ; (Processeur))

Conditions Activation : {(M_E_M ∧ Cal_OK ∧ Trajet_OK ; DVM)}

Méthode : "Calcul - Mise à jour"

Priorité : 5

Temps d'exécution au plus tôt : 0

Périodicité : NON

Préemptabilité : OUI

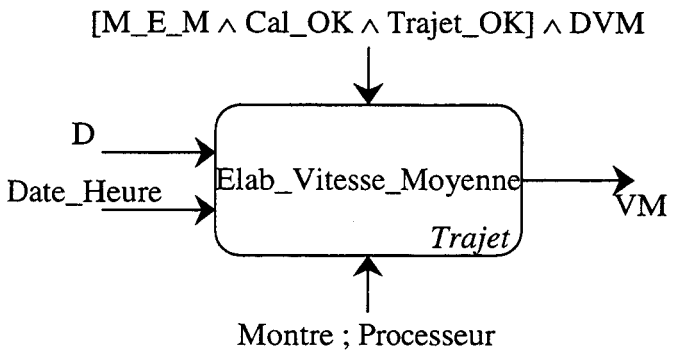


Figure 5.7 : service Elab_Vitesse_Moyenne

Service Saisie_VR :

Sorties : ((VR) ; (VR_OK))

Entrées : (V)

Paramètres : ϕ

Ressources : (ϕ ; (Processeur))

Conditions Activation : {(M_E_M ∧ $\overline{\text{FREIN}}$ ∧ Cal_OK ∧ V_OK ; Reg_ON)}

Méthode : "Enregistrement"

Priorité : 2

Temps d'exécution au plus tôt : 0

Périodicité : NON

Préemptabilité : OUI

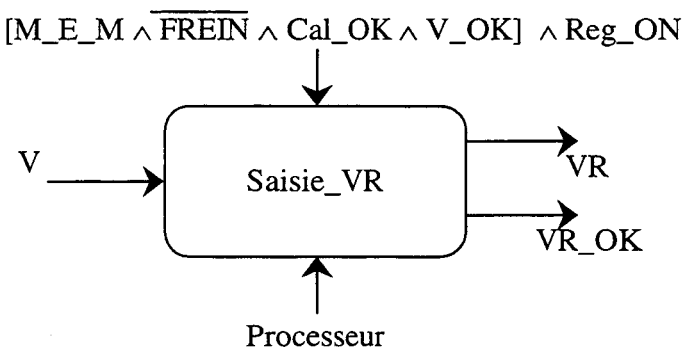


Figure 5.8 : service Saisie_VR

Service Saisie_Trajet :

Sorties : ((Trajet) ; (Trajet_OK))

Entrées : (D, Date_Heure)

Paramètres : ϕ

Ressources : (Montre ; (Processeur))

Conditions Activation : {(M_E_M $\wedge$ Cal_OK ; STrajet)}

Méthode : "Enregistrement"

Priorité : 3

Temps d'exécution au plus tôt : 0

Périodicité : NON

Préemptabilité : OUI

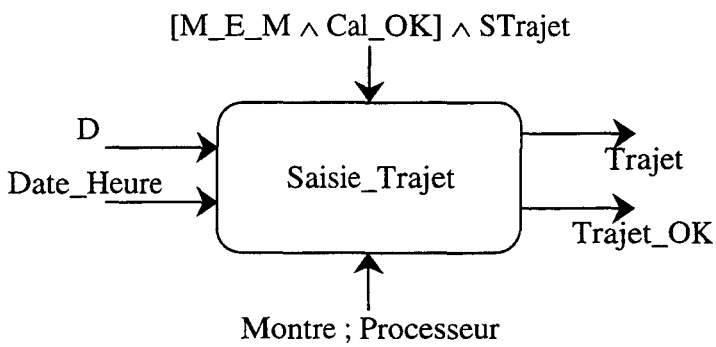


Figure 5.9 : service Saisie_Trajet

Toutes les données relatives aux services en fonction des besoins exprimés par les consommateurs. On peut alors donner tous les acteurs externes : i.e. les données fournies et/ou modifiées par des acteurs autres que les services externes.

V.3.2.10.3- L'ensemble des acteurs externes :

Les acteurs externes sont exactement les mêmes que les consommateurs externes : le conducteur et l'ensemble moteur. On leur associe donc leurs fonctions fa associées donnant quel acteur produit quelle donnée :

- fa(Conducteur) = {ACC, DCons, FREIN, QCarb, ReEngage, Reg_ON, Start_KM, Stop_KM, STrajet}
- fa(Moteur) = {Date_Heure, Impulsions, M_E_M, TOP}

Les données FREIN et M_E_M sont des booléens, c'est pourquoi leurs valeurs $\overline{\text{FREIN}}$ et $\overline{\text{M_E_M}}$ ne sont pas explicitement mises, ici, dans les différents ensembles d'acteurs, puisqu'elles sont implicites à leur format.

V.3.2.10.4- Ensembles de la spécification :

On résume ici les différents ensembles qui ont été formés au cours de la spécification :

- l'ensemble des acteurs :

Ua = {Moteur, Conducteur},

- l'ensemble des consommateurs :

Uc = {Moteur, Conducteur},

- l'ensemble des sorties de l'équipement :

D = {Cal, Cal_OK, CM, CMV, CRV, D, Trajet, Trajet_OK, V, V_OK, VR, VR_OK},

- l'ensemble des services offerts par l'équipement :

S = {Calibrage, Commande_Manuelle, Commande_Régulée, Elab_Consommation, Elab_Distance, Elab_Vitesse, Elab_Vitesse_Moyenne, Saisie_VR, Saisie_Trajet}

- l'ensemble des entrées de l'équipement :

E = {ACC, D, Date_Heure, FREIN, Impulsions, QCarb, Stop_KM, V},

- l'ensemble des paramètres :

P = {Cal, Trajet, VR},

- l'ensemble des conditions d'utilisation :

Ba = {Cal_OK, FREIN, M_E_M, Trajet_OK, V_OK, VR_OK}

- l'ensemble des requêtes d'activation des services :

Ra = {DCons, DVM, ReEngage, Reg_ON, Start_KM, STrajet, TOP},

- l'ensemble des ressources :

Ressources = {Capteur_Roue, Montre, Ped_ACC, Ped_FREIN,Processeur}

On obtient ici la spécification de tous les besoins du régulateur de vitesse qui donne les consommateurs, les acteurs et les services externes de cet équipement intelligent. Il nous faut maintenant analyser cette spécification pour en déterminer la cohérence, puis en déterminer l'organisation cohérente en modes d'utilisation.

V.4- Analyse de la spécification obtenue du régulateur de vitesse :

V.4.1- Analyse de l'ensemble des services externes du régulateur :

Après vérification du tableau Boucle pour le régulateur de vitesse, on ne repère aucune boucle dynamique dans la spécification des services. De même, pour le tableau Paramétrage, on vérifie facilement qu'aucun service ne modifie lui-même un de ses propres paramètres et qu'il n'existe pas de paramétrage croisé dans la spécification. On peut donc passer à la formation des modes d'utilisation.

V.4.2- Graphe des modes d'utilisation du régulateur :

Comme on l'a vu au chapitre III, il est possible de former les modes d'utilisation à partir des différentes conditions d'activation des services. On se propose donc, dans un premier temps, d'utiliser cette méthode pour donner la base des ensembles de services pouvant donner les modes d'utilisation du régulateur de vitesse. On s'occupera ensuite de la définition des différentes transitions entre les modes choisis pour le graphe des modes d'utilisation.

V.4.2.1- Formation des modes d'utilisation :

L'ensemble Ba^+ pour le régulateur de vitesse comprend les données suivantes : Cal_OK, $\overline{\text{FREIN}}$, M_E_M, $\overline{\text{M_E_M}}$, Trajet_OK, V_OK et VR_OK. Après avoir formé le tableau Comp_CU à partir des éléments de Ba^+, il est aisé de calculer les différentes distances entre services qui permettent de remplir le tableau Comp qui est donné ci-après :

Tableau Comp du régulateur de vitesse :

	Calibrage	Commande_Manuelle	Commande_Regulee	Elab_Consommation	Elab_Distance	Elab_Vitesse	Elab_Vitesse_Moyenne	Saisie_VR	Saisie_Trajet
Calibrage	1	1	0,25	0	0,5	0,5	0,33	0,25	0,5
Commande_Manuelle	1	1	0,25	0	0,5	0,5	0,33	0,5	0,5
Commande_Régulée	0,25	0,25	1	0	0,5	0,5	0,4	0,6	0,5
Elab_Consommation	0	0	0	1	0	0	0	0	0
Elab_Distance	0,5	0,5	0,5	0	1	1	0,66	0,5	1
Elab_Vitesse	0,5	0,5	0,5	0	1	1	0,66	0,5	1
Elab_Vitesse_Moyenne	0,33	0,33	0,4	0	0,66	0,66	1	0,33	0,66
Saisie_VR	0,25	0,5	0,6	0	0,5	0,5	0,33	1	0,5
Saisie_Trajet	0,5	0,5	0,5	0	1	1	0,66	0,5	1

On constate, tout d'abord que le service Elab_Consommation est simplement incompatible avec tous les autres services. L'ensemble Sinc est donc composé des huit paires correspondantes.

Ensuite, on peut former les deux premiers couples principaux de l'ensemble F qui sont le couple (Calibrage, Commande_Manuelle) et le triplet (Elab_Distance, Elab_Vitesse, Saisie_Trajet) dont le taux de compatibilité est de 1. Les services de chacun de ces couples seront donc toujours ensemble dans un mode, là où l'un d'entre-eux apparaît. On forme ensuite les autres éléments de F qui sont énoncés ci-dessous avec leur taux de compatibilité maximum cm associé qui devra être supérieur à zéro.

On définit ensuite les paires de services totalement incompatibles entre-eux. On en définit plusieurs par rapport à la spécification du régulateur de vitesse. Cela réside dans le fait que le service Calibrage ne peut être dans le même mode que les services Elab_Distance, Elab_Vitesse, Elab_Vitesse_Moyenne, Saisie_VR et Saisie_Trajet. En effet, tous ces services nécessitent la condition d'avoir été paramétrés plus ou moins directement. On prend aussi en compte les besoins des différents services en termes de consommation de données produites par d'autres services. De plus, hormis le singleton Elab_Consommation, tous les autres services impliquent le fait que le service Commande_Manuelle soit actif : la condition M_E_M est présente dans toutes les

conditions d'utilisation de ces services. Ceci réduit la taille de l'ensemble F qui est alors donné ici avec les taux de compatibilité maximum associés à chaque élément :

F = {((Elab_Consommation), 1),
((Calibrage, Commande_Manuelle), 1),
((Commande_Manuelle, Elab_Distance, Elab_Vitesse, Saisie_Trajet), 0,5),
((Commande_Manuelle, Elab_Distance, Elab_Vitesse, Saisie_VR, Saisie_Trajet), 0,5),
((Commande_Manuelle, Elab_Distance, Elab_Vitesse, Elab_Vitesse_Moyenne, Saisie_Trajet), 0,33),
((Commande_Manuelle, Elab_Distance, Elab_Vitesse, Elab_Vitesse_Moyenne, Saisie_VR, Saisie_Trajet), 0,33),
((Commande_Manuelle, Commande_Régulée, Elab_Distance, Elab_Vitesse, Elab_Vitesse_Moyenne, Saisie_VR, Saisie_Trajet), 0,25),
}

On obtient, ici, sept possibilités de modes de base pour le graphe des modes d'utilisation dont au moins trois sont nécessaires. Les deux premiers modes sont formés par les deux premiers éléments de F puisque Elab_Consommation et Calibrage n'apparaissent que dans ces deux premiers éléments distincts et exclusifs des autres éléments. Il faut ensuite déterminer au moins le troisième mode. On établit que les services Elab_Vitesse_Moyenne et Saisie_Trajet se trouvent toujours dans le même mode d'utilisation : on permet ainsi la reconfiguration du début du trajet en cours de voyage. On obtient alors plus que les cinq modes d'utilisation suivants :

- (Elab_Consommation),
- (Calibrage, Commande_Manuelle),
- (Commande_Manuelle, Elab_Distance, Elab_Vitesse, Elab_Vitesse_Moyenne, Saisie_Trajet)
- (Commande_Manuelle, Elab_Distance, Elab_Vitesse, Elab_Vitesse_Moyenne, Saisie_VR, Saisie_Trajet),
- (Commande_Manuelle, Commande_Régulée, Elab_Distance, Elab_Vitesse, Elab_Vitesse_Moyenne, Saisie_VR, Saisie_Trajet).

Par rapport au cahier des charges fourni, ces cinq modes correspondent respectivement aux contextes d'utilisation suivants : Stop, Configuration, Conduite, Freinage_en_Régulation et Régulation.

L'utilisation du Tableau Appartenance permet aisément de vérifier que l'ensemble des modes est un recouvrement de l'ensemble des services.

Il faut alors déterminer clairement les transitions entre ces différents modes d'utilisation formés.

V.4.2.2- Transitions entre les modes d'utilisation :

Pour définir les différentes transitions, on se propose de définir toutes les transitions sortantes de chaque mode d'utilisation. Ainsi, on est sûr qu'il existe au moins une manière de sortir de chaque mode d'utilisation s'assurant ainsi d'une certaine façon la connexité du graphe des modes d'utilisation. Toutes les transitions non définies n'existeront pas dans le graphe.

V.4.2.2.1- Mode Stop :

Il n'existe qu'une seule façon de sortir du mode d'utilisation : vers le mode Conduite. Cette transition notifie le fait que le conducteur mette en marche le moteur en tournant la clé de contact de manière efficace qui est traduite par la requête du conducteur Moteur_ON. La première transition est donc la transition $t_{1\ Stop\text{-}Conduite}$.

$t_{1\ Stop\text{-}Conduite} = \{1 ; Moteur_ON ; \phi\}$

V.4.2.2.2- Mode Conduite :

Une fois arrivé dans le mode Conduite, il existe plusieurs manières d'en sortir. On peut retourner dans le mode Stop ou aller dans le mode Configuration ou aller dans le mode Régulation.

Pour retourner dans le mode Stop, il suffit que le conducteur coupe le contact pour modifier la donnée M_E_M ce qui se traduit simplement par la requête conducteur Moteur_OFF. On a donc la transition suivante :

$t_{1\ Conduite\text{-}Stop} = \{1 ; Moteur_OFF ; \phi\}$

La transition vers le mode Configuration permet de calibrer les services Elab_Distance et Elab_Vitesse. Cette demande de changement de mode se fait par la requête conducteur déjà définie pour les conditions d'activation Start_KM. Il est ici intéressant d'envoyer une commande permettant d'initialiser la donnée Cal. Ainsi, même si le service Calibrage ne s'exécute pas entièrement, il n'y a pas d'incohérence : Cal n'est pas validé (Cal_OK = FAUX) et n'a alors pas de conséquence fâcheuse sur les services

Elab_Distance, Elab_Vitesse, Elab_Vitesse_Moyenne, Saisie_VR, Saisie_Trajet qui nécessitent dans leur condition d'utilisation que Cal_OK soit à VRAI. On a alors la transition suivante :

$t_{l\ \text{Conduite-Configuration}} = \{1 ; \text{Start_KM} ; \text{Cal_OK} = \text{FAUX}\}$

Enfin, la transition vers le mode Régulation se fait là aussi sur la demande du conducteur à l'apparition de la requête Reg_ON. Il faut cependant que certaines conditions soient remplies. En effet, d'après le cahier des charges et la spécification des services du mode, la pédale de frein ne doit pas être appuyée ($\overline{\text{FREIN}}$) et la vitesse doit être supérieure à 50km/h (V_OK). On a donc la transition ci-dessous :

$t_{l\ \text{Conduite-Régulation}} = \{ \overline{\text{FREIN}} \wedge \text{V_OK} ; \text{Reg_ON} ; \phi\}$

V.4.2.2.3- Mode Configuration :

Il existe deux destinations possibles à la sortie du mode Configuration : vers le mode Stop ou vers le mode Conduite. La sortie vers le mode Stop peut être considérée comme une défaillance à la bonne exécution du service Calibrage. Le changement de mode vers Stop est identique à $t_{l\ \text{Conduite-Stop}}$. On a donc :

$t_{l\ \text{Configuration-Stop}} = \{1 ; \text{Moteur_OFF} ; \phi\}$

La sortie vers le mode Conduite constitue, en fait, la fin du service Calibrage et se caractérise donc par la requête conducteur Stop_KM. La transition se définit alors comme suit :

$t_{l\ \text{Configuration-Conduite}} = \{1 ; \text{Stop_KM} ; \phi\}$

V.4.2.2.4- Mode Régulation :

Le mode Régulation a accès à tous les autres modes d'utilisation de la spécification, cela en fait un des modes charnières.

Tout d'abord, même si cela semble assez dangereux de couper le contact alors que le véhicule se trouve en régulation, à priori rien n'empêche le conducteur de le faire. On peut donc accéder au mode Stop à partir de Régulation. Cela se fait de la même manière que précédemment :

$t_{l\ \text{Régulation-Stop}} = \{1 ; \text{Moteur_OFF} ; \phi\}$

On peut de même accéder au mode configuration si le conducteur envoie la requête Start_KM. On a donc, comme pour le mode conduite, la transition suivante :

$t_{l\ \text{Régulation-Configuration}} = \{1 ; \text{Start_KM} ; \text{Cal_OK} = \text{FAUX}\}$

Le conducteur peut décider de sortir du mode régulation afin de reprendre le contrôle du véhicule. On peut distinguer deux cas. Le premier cas est celui où le conducteur désactive volontairement la régulation pour retourner dans le mode Conduite. Cela se traduit par la simple requête Conducteur Reg_OFF et la transition suivante :

$t_{l\ \text{Régulation-Conduite}} = \{1 ; \text{Reg_OFF} ; \phi\}$

Le deuxième cas est celui où le conducteur reprend "brusquement" le contrôle de la vitesse du véhicule en appuyant sur la pédale de frein. On a alors une transition vers le mode Freinage_en_Régulation qui permet de garder la vitesse de régulation enregistrée lors de l'entrée dans le mode de régulation de la vitesse. La transition vers le mode Freinage_en_Régulation se décline alors ci-dessous :

$t_{l\ \text{Régulation-Freinage_en_Régulation}} = \{\text{FREIN} ; 1 ; \phi\}$

V.4.2.2.5- Mode Freinage_en_Régulation :

Ce mode a, lui aussi, accès à tous les autres modes d'utilisation du graphe. Les transitions vers les modes Stop, Conduite et Configuration possèdent la même syntaxe que celle vue juste auparavant pour le mode Régulation. On les donne pour ce mode :

$t_{l\ \text{Freinage_en_Régulation-Stop}} = \{1 ; \text{Moteur_OFF} ; \phi\}$

$t_{l\ \text{Freinage_en_Régulation-Conduite}} = \{1 ; \text{Reg_OFF} ; \phi\}$

$t_{l\ \text{Freinage_en_Régulation-Configuration}} = \{1 ; \text{Start_KM} ; \text{Cal_OK} = \text{FAUX}\}$

A la suite d'un freinage fortuit, le conducteur peut vouloir revenir dans le mode Régulation sans avoir, pour cela, à redéfinir la vitesse de régulation si celle-ci lui convenait. Afin de permettre le retour dans le mode Régulation, à la consigne de vitesse précédente, il faut que la pédale de frein ne soit pas enfoncée ($\overline{\text{FREIN}}$) et que le conducteur envoie la requête déjà définie ReEngage. Sinon, le conducteur peut décider de reconfigurer la vitesse de régulation par une transition équivalente à la transition $t_{l\ \text{Conduite-Régulation}}$:

$t_{l\ \text{Freinage_en_Régulation-Régulation}} = \{\overline{\text{FREIN}} \wedge \text{V_OK} ; \text{Reg_ON} ; \phi\}$

$t_{2\ \text{Freinage_en_Régulation-Régulation}} = \{\overline{\text{FREIN}} ; \text{ReEngage} ; \phi\}$

Avec les transitions sortantes du mode Freinage_en_Régulation, on finit la définition de l'ensemble T_{cm} des transitions et du graphe des modes d'utilisation. On peut alors passer à la phase de vérification du graphe.

V.4.3- Vérification du graphe des modes d'utilisation du régulateur :

On doit vérifier deux choses : les vérifications préliminaires sur la complétude du modèle au niveau des conditions et les requêtes de changement de mode, et la vérification de l'automate en lui-même.

V.4.3.1- Vérifications préliminaires :

On obtient les ensembles suivants, en plus de ceux définit au paragraphe V.3.2.10.4 sur lesquelles on doit faire des vérifications préliminaires :

- Bm = {FREIN, V_OK}
- Rm = {Moteur_ON, Moteur_OFF, ReEngage, Reg_ON, Reg_OFF, Start_KM, Stop_KM}
- T_{cm} = {$t_{1\ \text{Conduite-Configuration}}$, $t_{1\ \text{Conduite-Régulation}}$, $t_{1\ \text{Conduite-Stop}}$, $t_{1\ \text{Configuration-Conduite}}$, $t_{1\ \text{Configuration-Stop}}$, $t_{1\ \text{Freinage_en_Régulation-Conduite}}$, $t_{1\ \text{Freinage_en_Régulation-Configuration}}$, $t_{1\ \text{Freinage_en_Régulation-Régulation}}$, $t_{2\ \text{Freinage_en_Régulation-Régulation}}$, $t_{1\ \text{Freinage_en_Régulation-Stop}}$, $t_{1\ \text{Régulation-Conduite}}$, $t_{1\ \text{Régulation-Configuration}}$, $t_{1\ \text{Régulation-Freinage_en_Régulation}}$, $t_{1\ \text{Régulation-Stop}}$, $t_{1\ \text{Stop-Conduite}}$}

Toutes les données de Bm sont déjà définies. Par contre les requêtes Moteur_ON, Moteur_OFF et Reg_OFF n'ont pas été définies. Ces dernières étant données par l'acteur externe Conducteur, on redéfinit la fonction fa pour l'acteur Conducteur :

fa(Conducteur) = {ACC, DCons, FREIN, Moteur_ON, Moteur_OFF, QCarb, ReEngage, Reg_ON, Reg_OFF Start_KM, Stop_KM, STrajet}

En ce qui concerne les commandes, il n'y en a qu'une seule qui est effectuée dans le graphe pour l'entrée dans le mode Configuration. On manipule alors à chaque fois la donnée Cal qui est déjà définie dans la spécification des services.

V.4.3.2- Vérifications sur l'automate du régulateur :

On vérifie facilement à partir du tableau Transition_mode que le graphe est fortement connexe. On ne repère ni mode initial, ni mode final pour le régulateur. On s'attaque alors au déterminisme du graphe et à sa cohérence.

V.4.3.2.1- Déterminisme du graphe du régulateur :

Le déterminisme du graphe doit se vérifier, quant à lui, plus attentivement. On peut déjà expliciter que deux requêtes du conducteur (i.e. $r_h \in f_a(\text{Conducteur}) \cap \text{Rm}$) ne peuvent, ici, être simultanément activées. Ainsi, on obtient que deux requêtes de l'ensemble {Moteur_ON, Moteur_OFF, ReEngage, Reg_ON, Reg_OFF Start_KM, Stop_KM} ne sont jamais présentes au même instant. Ce postulat permet de régler la plupart des cas d'indéterminisme pouvant se présenter dans le graphe. Il existe cependant un petit indéterminisme au niveau du mode Régulation. En effet, la transition $t_{l\ \text{Régulation-Freinage_en_Régulation}}$ est en concurrence avec toutes les autres puisque cette transition est "spontanée" (sans requête). Il faut alors remarquer que les couples de transitions $(t_{l\ \text{Régulation-Conduite}}, t_{l\ \text{Freinage_en_Régulation-Conduite}})$, $(t_{l\ \text{Régulation-Configuration}}, t_{l\ \text{Freinage_en_Régulation-Configuration}})$ et $(t_{l\ \text{Régulation-Stop}}, t_{l\ \text{Freinage_en_Régulation-Stop}})$ possèdent la même syntaxe. Ainsi, si la transition vers le mode Freinage_en_Régulation se fait avant, on obtient quand même un changement de mode vers l'autre mode dont la transition était validée : on a alors une petite instabilité dans le graphe des modes. Pour pallier cela, il suffit d'assigner une priorité plus forte aux transitions de Régulation possédant une requête de changement de mode. Cette capacité étant intégrée dans le modèle SYNCCHARTS, on l'intégrera alors très facilement à ce moment.

V.4.3.2.2- Cohérence du graphe du régulateur :

Elle s'étudie en formant le tableau des services indispensables à chaque mode :

Tableau Services_indisp du régulateur de vitesse :

	Stop	Conduite	Configuration	Régulation	Freinage_en_Régulation
Calibrage	0	0	1	0	0
Commande_Manuelle	0	1	1	1	1
Commande_Régulée	0	0	0	1	0
Elab_Consommation	0	0	0	0	0
Elab_Distance	0	0	0	0	0
Elab_Vitesse	0	0	0	1	1
Elab_Vitesse_Moyenne	0	0	0	0	0
Saisie_VR	0	0	0	1	1
Saisie_Trajet	0	0	0	0	0

Hormis pour le mode Stop, le service Commande_Manuelle est supposé être disponible ou la requête Moteur_ON n'est pas validée de manière efficace puisque le moteur ne fonctionne pas : on reste alors dans le mode Stop.

Le service Calibrage est lui-aussi supposé toujours disponible, il l'est donc aussi dans le mode Configuration.

Dans le mode Régulation, trois autres services, hormis Commande_Manuelle, sont indispensables : Commande_Régulée, Elab_Vitesse et Saisie_VR. En regardant dans leurs conditions d'utilisation respectives on constate que la donnée Cal_OK est la seule condition de disponibilité commune à ces trois services. On ajoute donc à toutes les transitions ayant pour destination le mode Régulation la condition Cal_OK, ce qui donne :

- $t_{1\ \text{Conduite-Régulation}} = \{ \overline{\text{FREIN}} \wedge \text{Cal_OK} \wedge \text{V_OK} ; \text{Reg_ON} ; \phi \}$
- $t_{1\ \text{Freinage_en_Régulation-Régulation}} = \{ \overline{\text{FREIN}} \wedge \text{Cal_OK} \wedge \text{V_OK} ; \text{Reg_ON} ; \phi \}$
- $t_{2\ \text{Freinage_en_Régulation-Régulation}} = \{ \overline{\text{FREIN}} \wedge \text{Cal_OK} ; \text{ReEngage} ; \phi \}$

Les mêmes modifications sont à faire pour le mode Freinage_en_Régulation qui nécessite aussi que les services Elab_Vitesse et Saisie_VR soient présents et disponibles.

La condition Cal_OK est fournie/modifiée par le service Calibrage. Il n'est donc point besoin de modifier la spécification existante pour la prendre en compte.

La figure 5.10 ci-dessous donne les interfaces matérielles nécessaires pour la conception d'un régulateur de vitesse tout en donnant un résumé des ensembles.

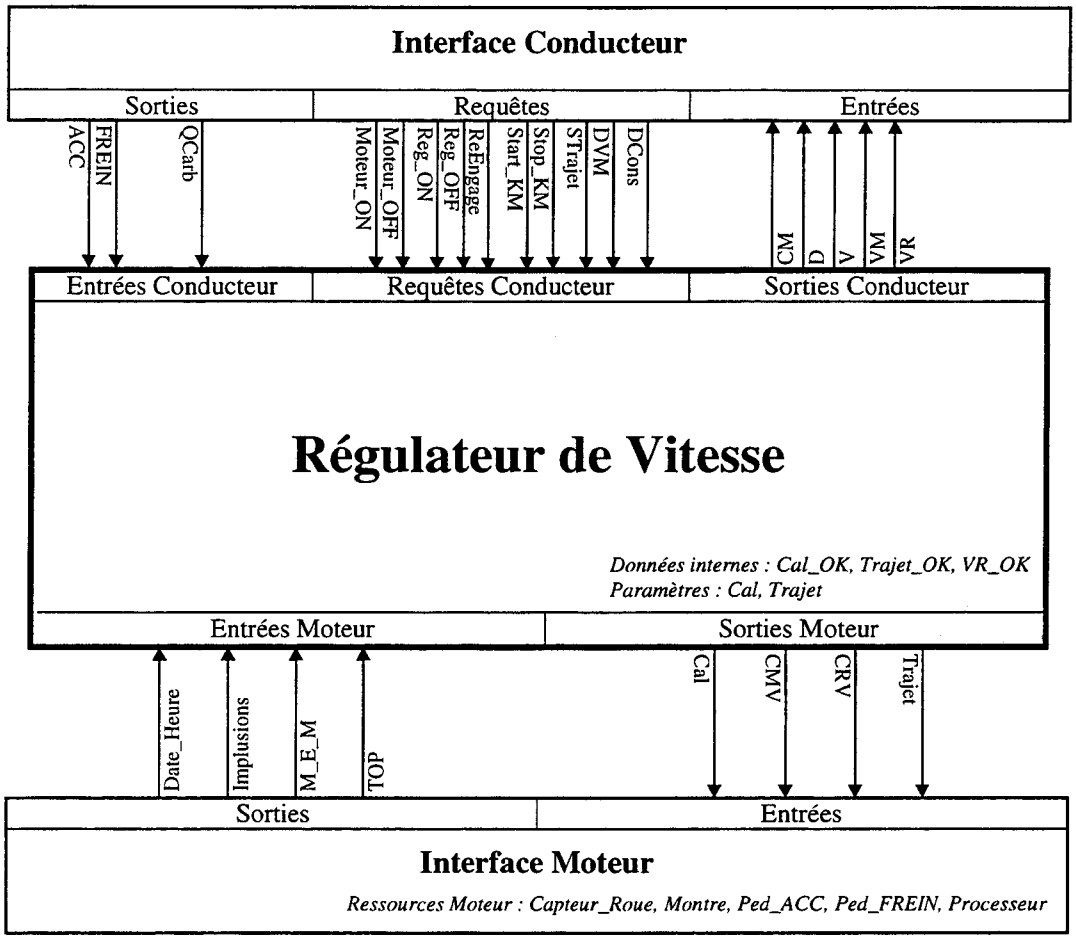


Figure 5.10 : Interfaces matérielles du Régulateur de Vitesse.

Une fois les vérifications de la connexité et du déterminisme effectués, il n'y a, pour l'instant, aucune prise en compte de défaillance quelconque dans le cahier des charges, on construit le graphe constitué de l'automate fini déterministe du régulateur de vitesse. Cet automate est donné à la figure 5.11.

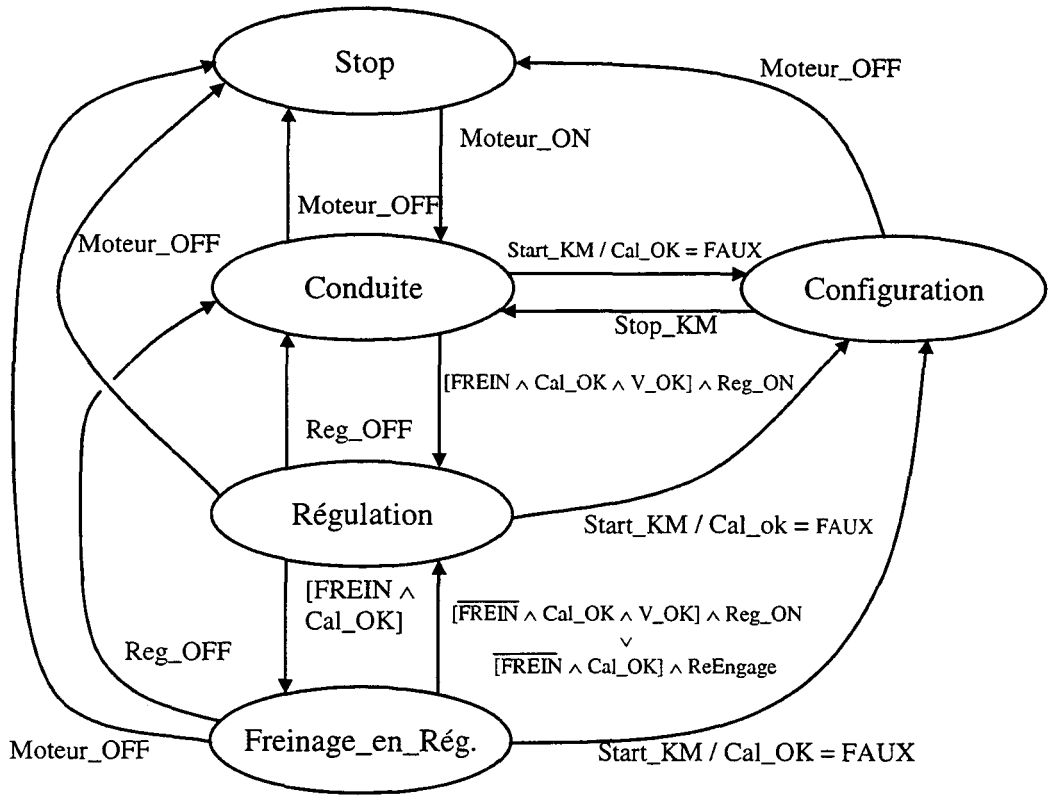


Figure 5.11 : Graphe des modes d'utilisation du régulateur de vitesse

La formation de ce graphe conclut momentanément le modèle externe du régulateur de vitesse tel qu'il a été décrit dans le cahier des charges exposé au paragraphe V.1. La description fonctionnelle du régulateur de vitesse ayant été faite de manière cohérente, il est désormais possible de transformer cet automate fini déterministe en son syncChart équivalent afin d'en étudier le comportement.

V.5- syncChart du régulateur de vitesse :

Le syncChart "brut" du régulateur de vitesse ne permet pas de voir l'organisation des services. On utilise donc la méthode proposée afin de réduire le syncChart brut et de le hiérarchiser.

V.5.1- Obtention du syncChart équivalent au graphe des modes d'utilisation :

En regardant les transitions entre les différents modes d'utilisation, on forme l'ensemble Mp des paires de modes d'utilisation regroupables. On obtient cinq paires de modes possibles :

Mp = {(Conduite, Configuration) ; (Conduite, Freinage_en_Régulation) ; (Conduite, Régulation) ; (Conduite, Stop); (Régulation, Freinage_en_Régulation)}

La formation du nouveau syncChart commence alors. De Mp, la paire (Régulation, Freinage_en_Régulation) possède le plus grand nombre de services en commun : {Commande_Manuelle, Elab_Distance, Elab_Vitesse, Elab_Vitesse_Moyenne, Saisie_VR, Saisie_Trajet}. Un macro-mode est alors formé des modes Régulation et Freinage_en_Régulation, on l'appelle "Régulation_Engagée". Ce macro-mode d'utilisation symbolise en effet que la régulation a été engagée et est toujours non désactivée.

On modifie ensuite la structure du syncChart :

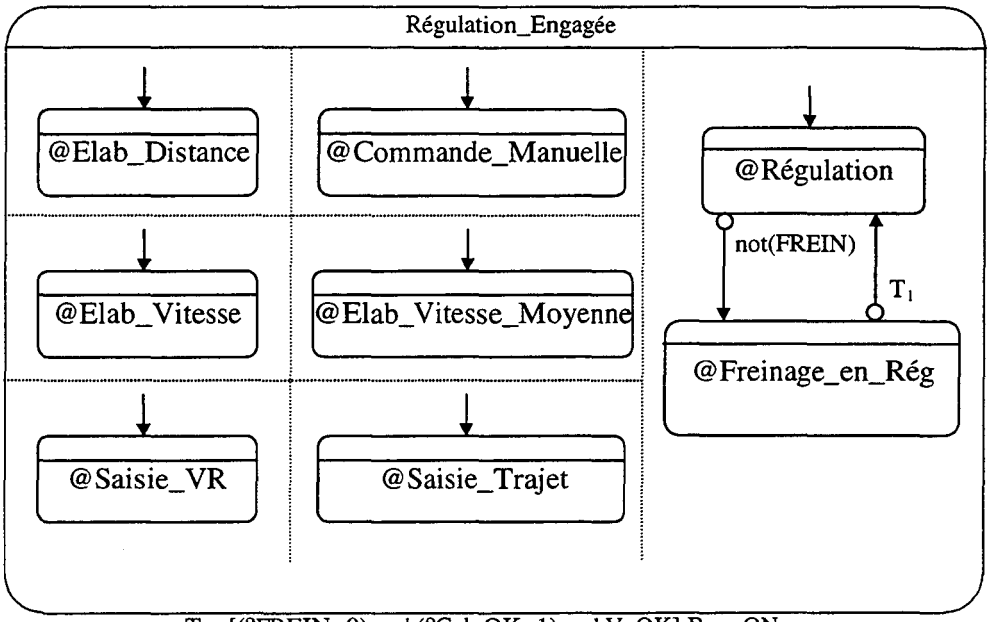
- Ms = {Stop, Conduite, Configuration, Régulation_Engagée}
- M = {Stop, Conduite, Configuration, Freinage_en_Régulation, Régulation, Régulation_Engagée}
- Pour les transitions entrantes et sortantes de Régulation_Engagée, on obtient pour nouvel ensemble des transitions l'ensemble T_{cm} suivant :

$T_{cm} = \{t_{1\ \text{Conduite-Configuration}}, t_{1\ \text{Conduite-Régulation_Engagée}}, t_{1\ \text{Conduite-Stop}}, t_{1\ \text{Configuration-Conduite}}, t_{1\ \text{Configuration-Stop}}, t_{1\ \text{Freinage_en_Régulation-Régulation}}, t_{2\ \text{Freinage_en_Régulation-Régulation}}, t_{1\ \text{Régulation-Freinage_en_Régulation}}, t_{1\ \text{Régulation_Engagée-Conduite}}, t_{1\ \text{Régulation_Engagée-Configuration}}, t_{1\ \text{Régulation_Engagée-Stop}}, t_{1\ \text{Stop-Conduite}}\}$

avec :

- $t_{1\ \text{Conduite-Régulation_Engagée}} = \{\ \overline{\text{FREIN}} \wedge \text{Cal_OK} \wedge \text{V_OK}\ ;\ \text{Reg_ON}\ ;\ \phi\}$
- $t_{1\ \text{Régulation_Engagée-Conduite}} = \{1\ ;\ \text{Reg_OFF}\ ;\ \phi\}$
- $t_{1\ \text{Régulation_Engagée-Configuration}} = \{1\ ;\ \text{Start_KM}\ ;\ \text{Cal} = 0\}$
- $t_{1\ \text{Régulation_Engagée-Stop}} = \{1\ ;\ \text{Moteur_OFF}\ ;\ \phi\}$

- A l'entrée dans le macro-mode Régulation_Engagée, on entre toujours dans le mode Régulation, il n'y a donc aucun signal supplémentaire généré.
- D'un point de vue interne au macro-mode Régulation_Engagée, on met les services Commande_Manuelle, Elab_Distance, Elab_Vitesse, Elab_Vitesse_Moyenne, Saisie_VR et Saisie_Trajet en constellations orthogonales. Le graphe constitué par les modes Régulation et Freinage_en_Régulation est, lui, intégré dans une constellation propre. Le syncChart du macro-mode Régulation Engagée est alors celui de la figure 5.12 :



T_1 : [(?FREIN=0) and (?Cal_OK=1) and V_OK] Reg_ON or
[[(?FREIN=0) and (?Cal_OK=1)] ReEngage

Figure 5.12 : syncChart du macro-mode Régulation_Engagée

A la seconde itération, on reforme Mp à partir de Ms, on obtient alors l'ensemble des nouvelles paires de modes regroupables :

Mp = {(Conduite, Configuration) ; (Conduite, Régulation_Engagée) ; (Conduite, Stop)}

C'est ici la deuxième paire de mode qui l'emporte au point de vue du nombre de services en commun. On forme alors le macro-mode d'utilisation "En_Circulation". Ce macro-mode désigne le fait que le véhicule fonctionne et n'est pas en configuration.

On modifie ensuite la structure du syncChart :

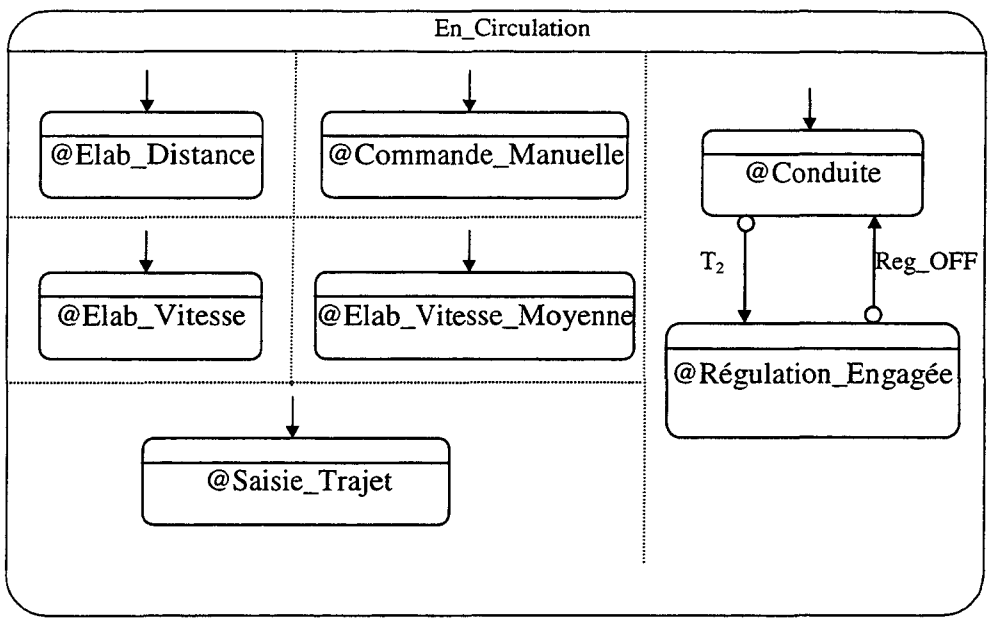
- Ms = {Stop, En_Circulation, Configuration }
- M = {Stop, Conduite, Configuration, En_Circulation, Freinage_en_Régulation, Régulation, Régulation_Engagée}
- Pour les transitions entrantes et sortantes de En_Circulation, on obtient pour nouvel ensemble des transitions l'ensemble T_{cm} suivant :

T_{cm} = {$t_{1\ \text{Conduite-Régulation_Engagée}}$, $t_{1\ \text{Configuration-Stop}}$, $t_{1\ \text{En_Circulation-Configuration}}$, $t_{1\ \text{En_Circulation-Stop}}$, $t_{1\ \text{Freinage_en_Régulation-Régulation}}$, $t_{2\ \text{Freinage_en_Régulation-Régulation}}$, $t_{1\ \text{Régulation-Freinage_en_Régulation}}$, $t_{1\ \text{Régulation_Engagée-Conduite}}$, $t_{1\ \text{Stop-En_Circulation}}$}

avec :

- $t_{1\ \text{En_Circulation-Configuration}}$ = {1 ; Start_KM ; Cal = 0}
- $t_{1\ \text{En_Circulation-Stop}}$ = {1 ; Moteur_OFF ; ϕ}
- $t_{1\ \text{Stop-En_Circulation}}$ = {1 ; Moteur_ON ; ϕ}

- A l'entrée dans le macro-mode En_Circulation, on entre toujours dans le mode Conduite, il n'y a donc aucun signal supplémentaire généré.
- D'un point de vue interne au macro-mode En_Circulation, on met les services Commande_Manuelle, Elab_Distance, Elab_Vitesse, Elab_Vitesse_Moyenne, et Saisie_Trajet en constellations orthogonales. Le graphe constitué par les modes Conduite et Régulation_Engagée est, lui, intégré dans une constellation propre. Le syncChart du macro-mode En_Circulation est alors celui de la figure 5.13 :



T_2 : [(?FREIN=0) and (?Cal_OK=1) and V_OK] Reg_ON

Figure 5.13 : syncChart du macro-mode En_Circulation

On entame alors la troisième itération, Mp est alors l'ensemble suivant :

Mp = {(En_Circulation, Configuration) ; (En_Circulation, Stop)}

La première paire de Mp présente le plus grand nombre de services en commun avec le seul service Commande_Manuelle. Le macro-mode "En_Fonctionnement" est donc formé.

On modifie ensuite la structure du syncChart :

- Ms = {Stop, En_Fonctionnement}
- M = {Stop, Conduite, Configuration, En_Circulation, En_Fonctionnement, Freinage_en_Régulation, Régulation, Régulation_Engagée}
- Pour les transitions entrantes et sortantes de En_Fonctionnement, on obtient pour nouvel ensemble des transitions l'ensemble T_{cm} suivant :

T_{cm} = {$t_{1\ Conduite\text{-}Régulation_Engagée}$, $t_{1\ En_Circulation\text{-}Configuration}$, $t_{1\ En_Fonctionnement\text{-}Stop}$,
$t_{1\ Freinage_en_Régulation\text{-}Régulation}$, $t_{2\ Freinage_en_Régulation\text{-}Régulation}$,
$t_{1\ Régulation\text{-}Freinage_en_Régulation}$, $t_{1\ Régulation_Engagée\text{-}Conduite}$, $t_{1\ Stop\text{-}En_Fonctionnement}$}

avec :

- $t_{1\ En_Fonctionnement\text{-}Stop}$ = {1 ; Moteur_OFF ; ϕ}
- $t_{1\ Stop\text{-}En_Fonctionnement}$ = {1 ; Moteur_ON ; ϕ}

- A l'entrée dans le macro-mode En_Fonctionnement, on entre toujours dans le mode En_Circulation, il n'y a donc aucun signal supplémentaire généré.
- D'un point de vue interne au macro-mode En_Fonctionnement, on met le service Commande_Manuelle dans une constellation et dans une autre, on met le graphe constitué par les modes En_Circulation et Configuration.

A l'itération suivante, il n'y a plus qu'une seule paire dans Mp : (Stop, En_Fonctionnement). il n'y a plus de modifications possibles du syncChart, on s'arrête donc là pour le syncChart global du régulateur de vitesse en précisant que le mode Stop est le mode initial.

Les différents ensembles pour le syncChart sont alors les suivants :

- M = {Stop, Conduite, Configuration, En_Circulation, En_Fonctionnement, Freinage_en_Régulation, Régulation, Régulation_Engagée}
- T_{cm} = {$t_{1\ Conduite\text{-}Régulation_Engagée}$, $t_{1\ En_Circulation\text{-}Configuration}$, $t_{1\ En_Fonctionnement\text{-}Stop}$,
$t_{1\ Freinage_en_Régulation\text{-}Régulation}$, $t_{2\ Freinage_en_Régulation\text{-}Régulation}$,
$t_{1\ Régulation\text{-}Freinage_en_Régulation}$, $t_{1\ Régulation_Engagée\text{-}Conduite}$, $t_{1\ Stop\text{-}En_Fonctionnement}$}

On constate que si la taille de M a augmenté pour prendre en compte les macro-modes d'utilisation formés, la taille de T_{cm} a, elle, fortement diminuée en passant d'un cardinal égal à quinze à un cardinal de huit, simplifiant d'autant la spécification au niveau des transitions. De plus, on remarque que chacun des services n'apparaît qu'une seule et unique fois dans la spécification SYNCCHARTS. La simplification du syncChart du régulateur de vitesse est donc optimale en ce qui concerne la représentation des modes et de l'organisation des services. Le syncChart intermédiaire (i.e. sans déclaration de tous les signaux) est alors représenté figure 5.14. Après cela, on s'intéresse aux simplifications qu'il est possible de faire au niveau des services et des transitions qu'on a définis.

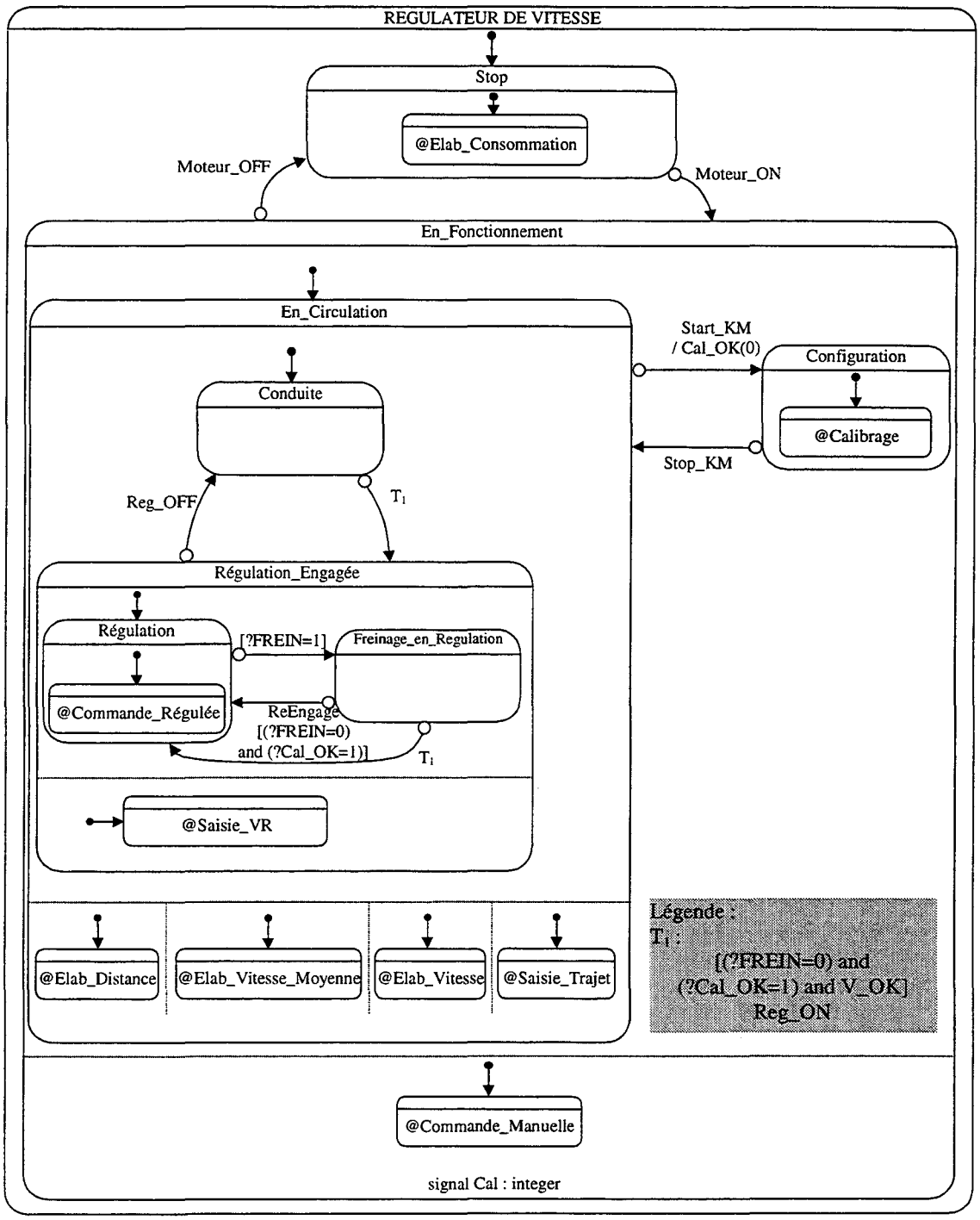


Figure 5.14 : syncChart intermédiaire du régulateur de vitesse

V.5.2- Simplifications et fonctionnement interne des services :

La simplification des conditions d'utilisation des services se fait suivant chaque macro-mode d'utilisation. Dans cet exemple du régulateur de vitesse, chaque service n'apparaissant qu'une seule fois, il sera plus facile d'évaluer les simplifications possibles pour chacun d'entre eux.

On regarde alors les états induits dans chaque macro-mode, on s'attachera ensuite à s'occuper de chacun des services communs à l'intérieur du macro-mode d'utilisation afin de voir quelles sont les simplifications possibles. On regardera aussi celles qui peuvent être effectuées sur les transitions dans le graphe de mode d'utilisation réduit du macro-mode étudié. Comme il n'existe aucun service commun du syncChart global, on commence par les deux modes Stop et En_Fonctionnement.

V.5.2.1- Mode Stop :

Le mode Stop n'induit qu'une seule chose : le fait que le moteur ne soit pas en marche, i.e. M_E_M = FAUX. Cette variable booléenne est d'ailleurs mise à FAUX à chaque fois qu'on quitte le macro-mode En_Fonctionnement par la requête Moteur_OFF. La condition $\overline{M_E_M}$ est donc implicite dans ce mode.

Le seul service appartenant au mode est le service Elab_Consommation. Ces conditions d'activation sont : {($\overline{M_E_M}$; DCons)}. Il n'en existe qu'une seule et dans les Conditions_Utilisation de Elab_Consommation, $\overline{M_E_M}$ est l'unique condition préalable. On choisit donc de la supprimer dans le syncChart de Elab_Consommation qui est alors celui de la figure 5.15 :

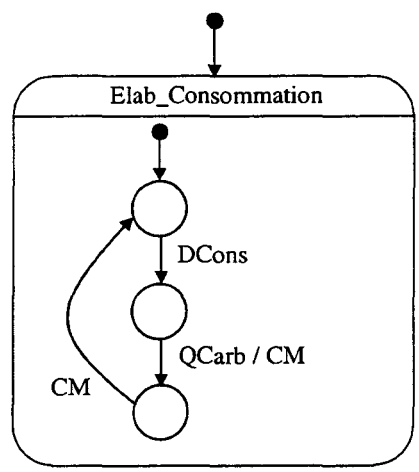


Figure 5.15 : syncChart du service Elab_Consommation.

V.5.2.2- Macro-mode En_Fonctionnement :

Contrairement au mode Stop, M_E_M est mise à VRAI à chaque fois qu'on quitte ce mode pour aller dans le macro-mode En_Fonctionnement par la requête Moteur_ON. La condition M_E_M est donc implicite dans ce macro-mode.

Le seul service commun au mode est le service Commande_Manuelle. Ces conditions d'activation sont : {(M_E_M ; 1}. De même, on la supprime dans le syncChart de Commande_Manuelle qui est alors le syncChart très simple de la figure 5.16 :

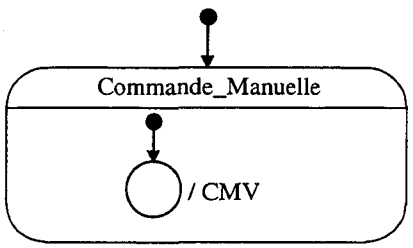


Figure 5.16 : syncChart du service Commande_Manuelle.

L'autre constellation du macro-mode En_Fonctionnement contient le graphe réduit du macro-mode En_Circulation et du mode Configuration. Il n'existe malheureusement aucune simplification possible sur les transitions entre ces étoiles du syncChart. En effet, les transitions se font suivant les requêtes respectives Start_KM et Stop_KM.

V.5.2.3- Mode Configuration :

Il n'y a aucune condition qui soit induite dans cet état à part la condition précédente M_E_M. Le service "privé" du mode est le service Calibrage. Ces conditions d'activation sont : {(M_E_M ; Start_KM)}. On supprime là aussi M_E_M comme pour Commande_Manuelle.

On constate, de plus, que la requête d'entrée dans le mode Configuration est la même que la requête d'activation du service. Cette dernière peut donc être supprimée dans le mode, elle devient implicite. La figure 5.17 permet de donner une version possible du service Calibrage tel qu'il a été spécifié dans le modèle externe.

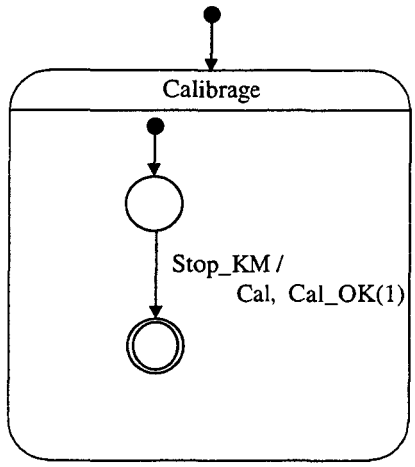


Figure 5.17 : syncChart du service Calibrage.

Une autre simplification apparaît : le signal Stop_KM met le service Calibrage dans un état final. On peut alors utiliser cet état final pour modifier le syncChart global du régulateur afin de prendre en compte la terminaison normale du service Calibrage et donc du mode Configuration. La transition Configuration vers En_Circulation devient alors une terminaison normale.

V.5.2.4- Macro-mode En_Circulation :

De la même manière que pour Configuration, il n'y a que M_E_M qui soit induit de par l'appartenance de En_Circulation au macro-mode En_Fonctionnement. Ces simplifications peuvent se faire sur les quatre services en commun de ce macro-mode : Elab_Distance, Elab_Vitesse, Elab_Vitesse_Moyenne et Saisie_Trajet. Leurs syncCharts respectifs sont résumés dans la figure 5.18. On choisit de rassembler les services Elab_Vitesse_Moyenne et Saisie_Trajet dans un même service Elab_Vitesse_Moyenne pour des raisons de lisibilité et de simplification du syncChart. Cette simplification reste possible grâce au fait que Saisie_Trajet est consommée uniquement par le service Elab_Vitesse_Moyenne.

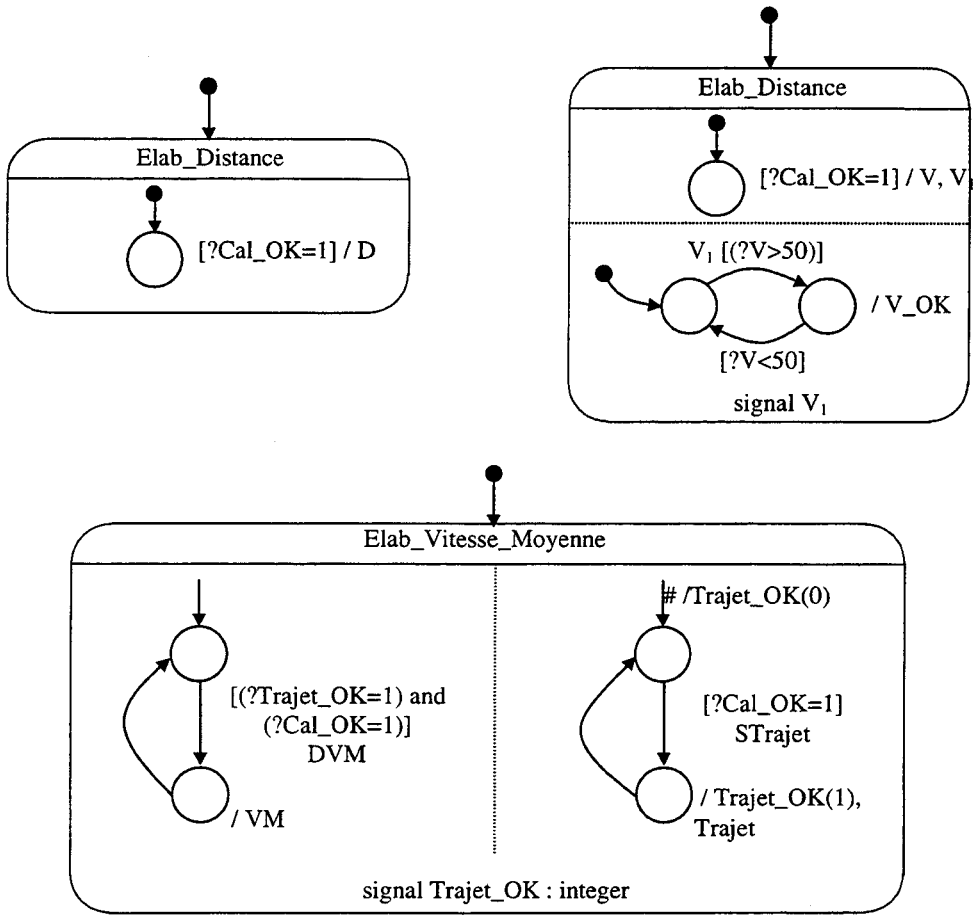


Figure 5.18 : syncChart des services propres au mode En_Circulation.

La dernière constellation du macro-mode En_Circulation contient le graphe réduit du mode Conduite et du macro-mode Régulation_Engagée. Il n'existe, là non plus, aucune simplification possible sur les transitions entre ces étoiles du syncChart.

Le mode Conduite ne possédant pas de service propre, on passe de suite au macro-mode Régulation_Engagée.

V.5.2.5- Macro-mode Régulation_Engagée :

La requête d'entrée dans le mode Régulation_Engagée implique un état particulier : le fait qu'une fois dans le mode, la vitesse de régulation est validée, i.e. que VR_OK est VRAI. Cette donnée n'est utilisée que par le Service Commande_Régulée dans une de ses conditions d'utilisation et elle est produite par le service Saisie_VR. Cette donnée devient ici inutile. On se permet donc de la supprimer de la spécification fonctionnelle et du syncChart. C'est une autre simplification qui est permise par les SYNCCHARTS.

De plus, pour entrer dans le mode, on doit avoir les conditions Cal_OK et V_OK. La donnée Cal_OK n'étant pas produite dans ce macro-mode, elle est toujours vraie dans le

mode Régulation_Engagée si on y est entré. En ce qui concerne V_OK, elle reste une condition nécessaire à toute modification de VR, on la garde donc.

Au niveau du service commun au macro-mode Saisie_VR, on peut alors supprimer M_E_M et Cal_OK des conditions d'activation. On constate aussi que Reg_ON est à la fois la requête d'entrée dans le macro-mode Régulation_Engagée et la requête d'activation du service Saisie_VR. On pourrait donc la supprimer. Cependant, Reg_ON peut aussi être utilisé par le conducteur afin de régler une autre vitesse de régulation, on choisit donc de laisser Reg_On dans les conditions d'activation du service Saisie_VR. Le syncChart de ce dernier service est donné figure 5.19.

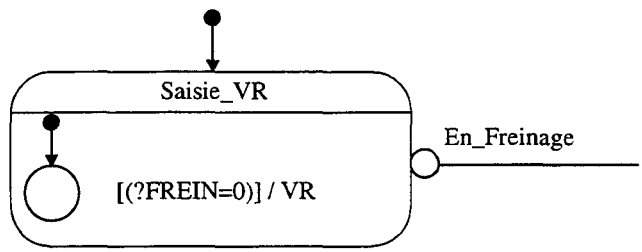


Figure 5.19 : syncChart du service Saisie_VR.

On a ajouté au syncChart du service le fait que celui-ci peut être suspendu. En effet, si on regarde le cahier des charges, on constate que lorsque le conducteur freine, cela suspend la régulation : i.e. Commande_Régulée et Saisie_VR. Le premier de ces services est "suspendu" en le tuant : on se retrouve dans le mode Freinage_en_Régulation. Cependant, on ne peut tuer le service Saisie_VR sans perdre la valeur de la vitesse régulée VR. Il faut donc le suspendre. Le service est alors suspendu tant que le signal En_Freinage est présent. Ce signal est produit dans Freinage_en_Régulation et indique tout simplement qu'on se trouve dans ce mode.

On simplifie ensuite les transitions qui contiennent toutes deux la condition Cal_OK qui est induite dans le macro-mode Régulation_Engagée. Elles deviennent alors :

- $t_{1\ \text{Freinage_en_Régulation-Régulation}} = \{ \overline{\text{FREIN}} \wedge \text{V_OK} ; \text{Reg_ON} ; \phi \}$
- $t_{2\ \text{Freinage_en_Régulation-Régulation}} = \{ \overline{\text{FREIN}} ; \text{ReEngage} ; \phi \}$

Le mode Freinage_en_Régulation ne contient pas de service propre, on regarde donc le dernier mode qu'est le mode Régulation.

V.5.2.6- Mode Régulation :

Ce dernier mode permet de simplifier le service Commande_Régulée. On peut simplifier ici les conditions d'utilisation suivantes : M_E_M, Cal_OK et V_OK. De plus, le mode Régulation implique forcément le fait que FREIN soit à FAUX. En ce qui concerne les requêtes d'activation, on constate que soit Reg_ON soit ReEngage impliquent le fait qu'on active le service Commande_Régulée. On obtient alors le syncChart très simple pour le service Commande_Régulée de la figure 5.20.

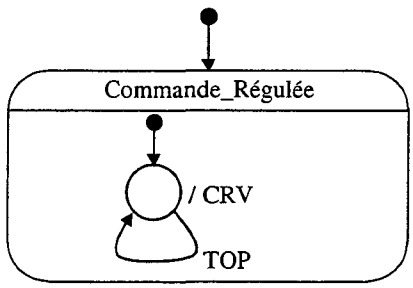


Figure 5.20 : syncChart du service Commande_Régulée.

En ce qui concerne les transitions dans ce mode qui est le dernier étudié, il n'en existe pas. On arrête donc là les différentes simplifications au niveau des services et des transitions du syncChart global.

On obtient ainsi le syncChart du régulateur de vitesse complet en remplaçant les différentes étoiles du syncChart par leur contenu exposé ci-dessus et en remplaçant les transitions modifiées par leur nouvelle syntaxe respective. Enfin, on regarde ici la syntaxe des transitions au niveau de leur nature : i.e. on regarde si la transition est un avortement faible ou fort, les cas de terminaisons normales ont été vus juste auparavant. On constate que dans la spécification le seul service qui soit non-préemptif est le service Commande_Manuelle. Le mode qui le "possède" est le macro-mode En_Fonctionnement. On remplace donc la nature de toutes les transitions sortantes du macro-mode En_Fonctionnement par des avortements faibles, i.e. la transition $t_{|\ En_Fonctionnement\text{-}Stop}$. On obtient alors le schéma global du syncChart du régulateur de vitesse de la figure 5.21 auquel il faut ajouter les syncChart de chaque service qui n'ont pas été représentés entièrement sur cette figure pour des raisons de clarté.

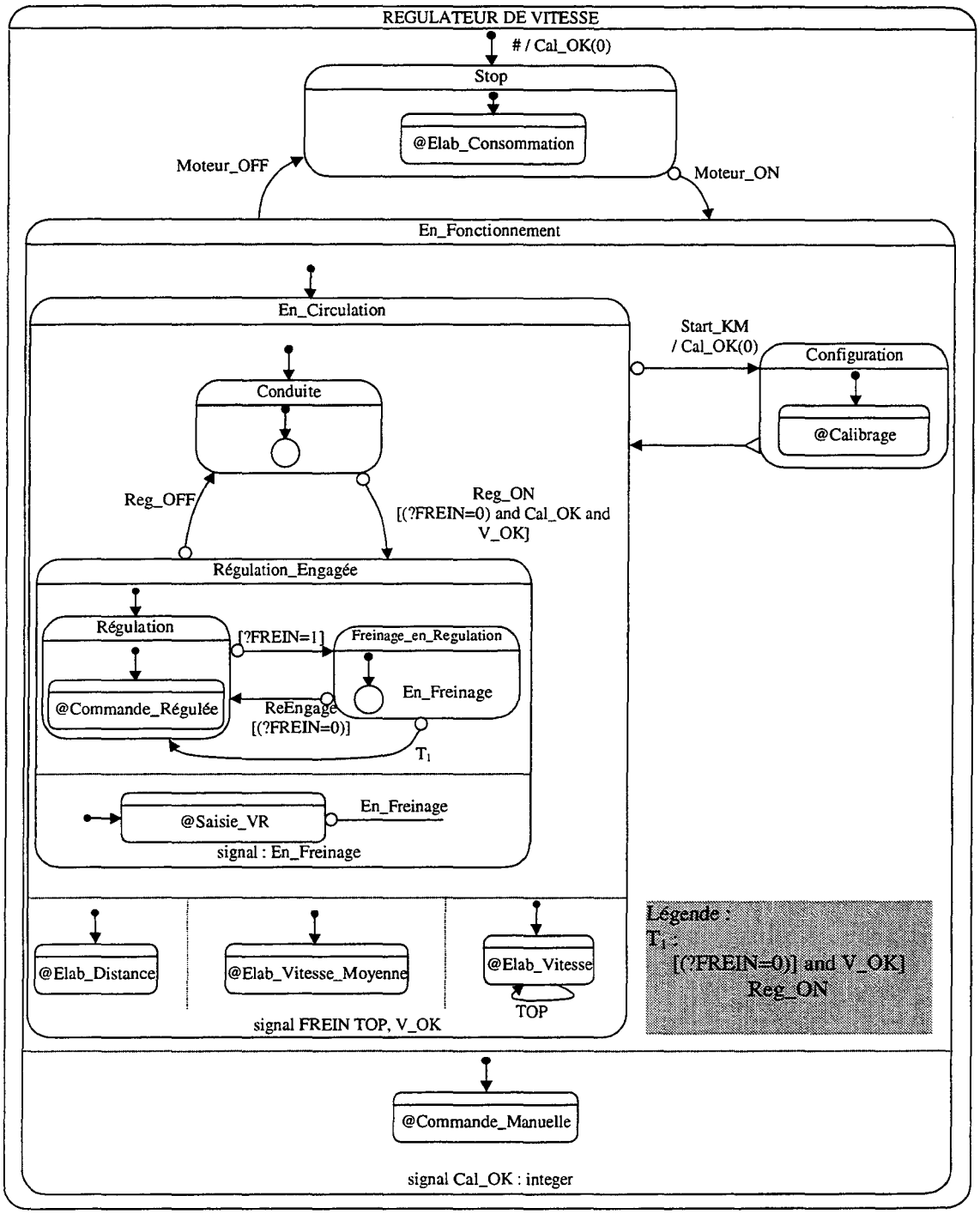


Figure 5.21 : syncChart final du régulateur de vitesse

V.6- Validation du modèle SYNCCHARTS :

Les premières vérifications se font par les compilations successives du modèle. Ainsi, grâce aux compilations C et ESTEREL du syncChart on vérifie que :

- tous les signaux qu'on a introduit dans le modèle ont correctement été déclarés,

- grâce à la hiérarchisation qui a été faite, il n'existe plus aucun problème de priorité dans les transitions entre modes et macro-modes d'utilisation,
- il n'existe pas non plus de boucles dans le modèle.

Les compilations effectuées, la simulation par le logiciel XES permet de vérifier le comportement proprement dit du syncChart du régulateur de vitesse. Ainsi, on vérifie plusieurs aspects :

- Tous les modes et macro-modes d'utilisation sont infiniment accessibles. On vérifie bien en cela la connexité forte déjà prouvée au paragraphe V.4.3.2.
- On vérifie bien que chaque mode et macro-mode est bien accessible suivant ces antécédents.
- Tous les services sont activables dans leurs conditions d'activation respectives.

On vérifie bien que le comportement du syncChart est bien celui attendu :

- Tous les services ont un comportement prévu. Il n'est pas possible d'avoir la consommation moyenne sans avoir rentré la quantité de carburant, tout comme il est impossible d'avoir la vitesse moyenne sur un trajet sans avoir au préalablement défini un début de trajet.
- Les services Elab_Distance, Elab_Vitesse, Commande_Régulée et Saisie_VR ne peuvent être activés que si on est entré dans le mode Configuration et qu'on en est ressorti en ayant terminé le service Calibrage de manière correcte.
- De même, le macro-mode Régulation n'est accessible que si le calibre a été défini correctement, s'il a été établie que la vitesse est supérieure à 50km/h.
- Lorsque le régulateur de vitesse est dans le macro-mode Régulation_Engagée, le comportement est correct par rapport à la position de la pédale de frein (i.e. FREIN à 0 ou à 1). On ne peut retourner dans le mode Régulation que lorsque la requête ReEngage est activée ou si on appuie à nouveau sur Reg_ON dans des conditions d'entrées adéquates dans le macro-mode Régulation_Engagée.

On valide ainsi le comportement du modèle SYNCCHARTS du régulateur de vitesse par rapport à ce qui était précisé dans le cahier des charges et dans la spécification à partir du modèle externe, chose qu'il n'était pas possible de faire directement uniquement à partir du modèle externe.

V.7- Conclusion :

Dans ce chapitre, nous avons illustré par un exemple les méthodes de conception, de simulation et de validation exposées aux chapitres précédents.

L' exemple du Régulateur de Vitesse de Croisière d'un Véhicule à partir du modèle externe que nous avons introduit au chapitre III nous a permis de vérifier le cahier des charges fourni au paragraphe V.2.

Une fois le modèle externe établi, le passage au modèle SYNCCHARTS associé a fortement hiérarchisé les modes d'utilisation et les services du régulateur de vitesse. Cette hiérarchisation, tout en clarifiant le comportement, a permis de modéliser le fonctionnement interne de chaque service dans chacun des macro-modes d'utilisation ainsi obtenus. Il a alors été possible de simuler le syncChart et d'en vérifier le comportement.

Références :

[ATL93] Atlee, J.M., Gannon J., "State-Based Model Checking of Event-Driven System Requirements.", IEEE Transactions on Software Engineering, Vol. 19 n°1, pp. 24-40, 1993.

[CAL90] Calvez J.P., "Spécification et Conception des Systèmes", pp. 61-98, Masson, Paris, 1990.

[HAT90] Hatley D.J., Pirbhai I.A., "Stratégies de spécification des systèmes temps-réel (SA-RT)", Editions Masson, 1990.

[WAR88] Ward P., Jensen and Al "ESML : An Extended Systems Modeling Language Based on the Data Flow Diagram", ACM Software Engineering Notes, Vol. 13, pp. 1-19, 1988.

Conclusion Générale

Dans ce mémoire, nous avons abordé le problème de la conception appliqué aux instruments intelligents. Dans la formation du cahier des charges, nous avons précisé les éléments nécessaires à une bonne interprétation fonctionnelle et ceux susceptibles de nous aider pour la conception.

Nous avons ensuite donné une démarche permettant de modéliser un équipement intelligent à partir des informations contenues dans le cahier des charges. Cette démarche comporte deux grandes étapes :

- la modélisation fonctionnelle des services rendus par l'équipement à partir du Modèle Externe,
- la modélisation comportementale de l'équipement à partir du modèle SYNCCHARTS.

La méthode exposée dans le chapitre III s'appuyant sur les besoins exprimés par les consommateurs ressortant du cahier des charges donne une démarche descendante de spécification du cahier des charges et d'analyse de la spécification obtenue. Cette démarche structurée a pu être informatisée. Ainsi, à partir de fichiers modèle programmés sous le logiciel Excel de Microsoft©, il nous a été permis de spécifier et d'apporter les éléments nécessaires afin d'assurer la complétude de la spécification. Une étape d'analyse permet d'assurer sa cohérence. De même que pour la spécification, un fichier modèle sous Excel97 a été programmé afin de faciliter la démarche. Les deux fichiers modèles pourront être appliqués facilement pour d'autres applications.

Dans le chapitre IV, nous avons introduit le modèle SYNCCHARTS. Après avoir donné une méthode permettant de passer du Modèle Externe à son syncChart équivalent, nous introduisons une étape de simplification compte tenu du fait que la structure du syncChart obtenu est très hiérarchisée. Les macro-modes sont utilisés pour clarifier la description graphique : le syncChart simplifié nous permet alors de simuler le modèle, de le vérifier et de valider le comportement par rapport aux indications fournies dans le cahier des charges.

L'exemple du régulateur de vitesse de croisière d'un véhicule a été utilisé pour valider les méthodes présentées. L'application de ces dernières nous a permis de spécifier facilement les besoins exprimés dans le cahier des charges et de les analyser afin d'en former un modèle cohérent. Les fichiers Excel ont alors été appliqués de manière efficace. Le syncChart équivalent a ensuite été construit, hiérarchisé, simplifié et simulé. Cette simulation nous a permis de valider le modèle.

Les résultats de cette étude serviront de point d'entrée des outils d'aide à la conception.

L'étude proposée s'est focalisée sur un équipement intelligent. Les perspectives de développement concernent pour une part l'outil Excel développé pour lequel on pourrait :

- créer et introduire dans les feuilles de calculs Excel des macro-fonctions facilitant l'écriture des vérifications et diminuant la taille des fichiers modèle correspondants,
- intégrer les étapes de vérifications manquantes comme l'est l'étape de test du déterminisme du graphe des modes d'utilisation.

Il serait aussi intéressant de développer un méta langage spécifique au Modèle Externe. Ce méta langage permettrait alors, à terme, un passage plus aisé et automatisé vers son modèle SYNCCHARTS associé.

L'étude comportementale sous SYNCCHARTS est actuellement poursuivie afin d'intégrer la notion de versions de services. A terme, nous disposerons donc d'un outil de validation du fonctionnement normal et dégradé d'un équipement.

Nous avons vu par ailleurs que le modèle proposé spécifiait les ressources et les interfaces avec l'environnement. Il peut donc être utilisé pour le choix d'une architecture matérielle, en particulier des moyens de communication (point à point, réseau de terrain, ...) avec l'environnement. L'étape suivante permet alors de spécifier les services qui pourront être développés par un logiciel approprié. A ce titre, le logiciel LARII (Logiciel d'Aide à la Réalisation d'Instruments Intelligents) développé par le LAIL est un des débouchés direct du travail de spécification qui a été fait dans ce mémoire. LARII sera utilisé alors dans une démarche de conception-réalisation complète d'instruments intelligents.

Enfin, l'étude pourra être étendue à une architecture de Système Automatisé intégrant plusieurs équipements intelligents.

Annexe I :

Présentation de la spécification sous Excel97

1- Introduction :

Le logiciel Excel97 de Microsoft© a été choisi pour plusieurs raisons. La première de celle-ci est que ce logiciel est largement répandu et donc facilement accessible. Ensuite, Excel est un logiciel facilement utilisable qui offre les possibilités suivantes :

- chaque étape de la spécification peut se faire sur une feuille Excel ce qui facilite la démarche,
- les options de présentation, la couleur entre autres, peuvent être utilisées afin de signifier des erreurs et/ou des manques dans la spécification,
- la saisie dans chaque cellule peut être programmée afin d'éviter des doublons ou des blancs,
- les fonctions existantes dans Excel facilitent les vérifications,
- enfin, un fichier modèle peut être créer pour la réutilisabilité du fichier à d'autres applications autres que le régulateur de vitesse.

On présente ici l'outil de spécification général qui est sous la forme du fichier **Specification.xlt**. Ce fichier modèle est à mettre dans le répertoire C:\Program Files\Micosoft Office\Modèles. Pour utiliser le fichier de spécification, on doit faire les opérations suivantes :

- lancer Excel97,
- dans la barre d'outil standard, sélectionner Fichier - Nouveau,
- sélectionner l'icône Specification.xlt.

Une fois sélectionnée, on obtient un fichier de spécification Specification1.xls d'une application vierge qu'on doit compléter. Ce fichier est constitué des neuf feuilles numérotées suivantes :

1- Consommateurs
2- Services-Entrées-Paramètres
3- Acteurs-Entrées
4-Vérification des Paramètres
5-Services-Cond. d'Activation
6-Acteurs-CU
7-Acteurs-Requêtes

8-Résumé Services
9- Résumé Utilisateurs

L'explication de ces neuf feuilles est donnée au paragraphe suivant.

2- Démarche de spécification feuille par feuille :

2.1- Feuille Consommateur :

La feuille se présente comme le montre la figure 1 ci-dessous :

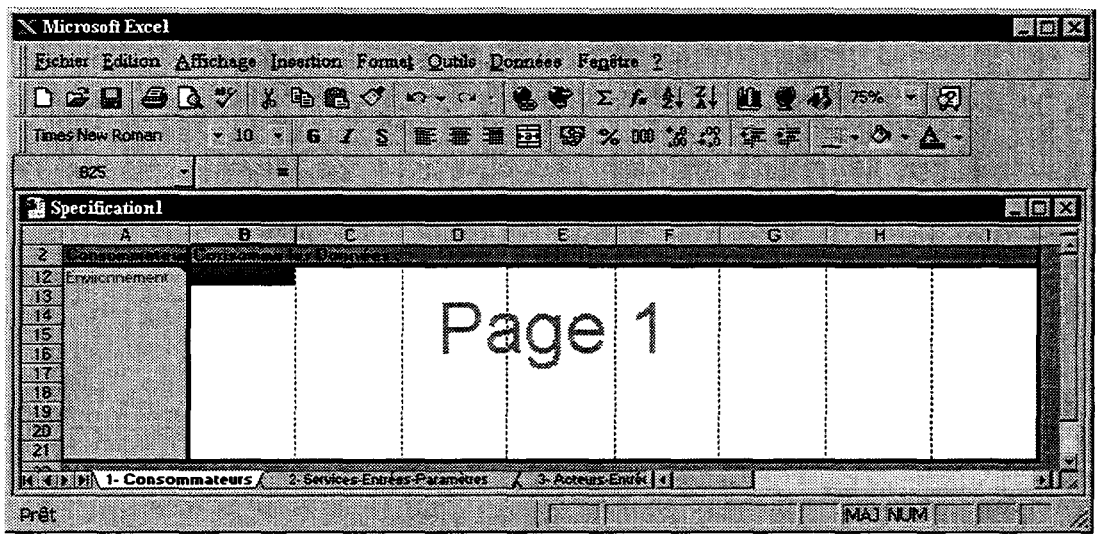


Figure 1 : Feuille consommateur.

Dans cette feuille, on doit entrer les consommateurs externes de l'équipement intelligent dans la première colonne. A droite de chaque consommateur, on peut mettre jusqu'à huit données consommées qui forment les sorties principales de l'équipement intelligent.

2.2- Feuille Services-Entrées-Paramètres :

Dans cette feuille, chaque sortie principale apparaît : vingt sorties principales différentes peuvent être prises en compte. Le concepteur doit alors donner pour chaque sortie principale un nom qualifiant le service externe la produisant. Ensuite, pour chaque service, le concepteur peut donner jusqu'à quatre entrées et quatre paramètres. Cette feuille est représentée à la figure 2.

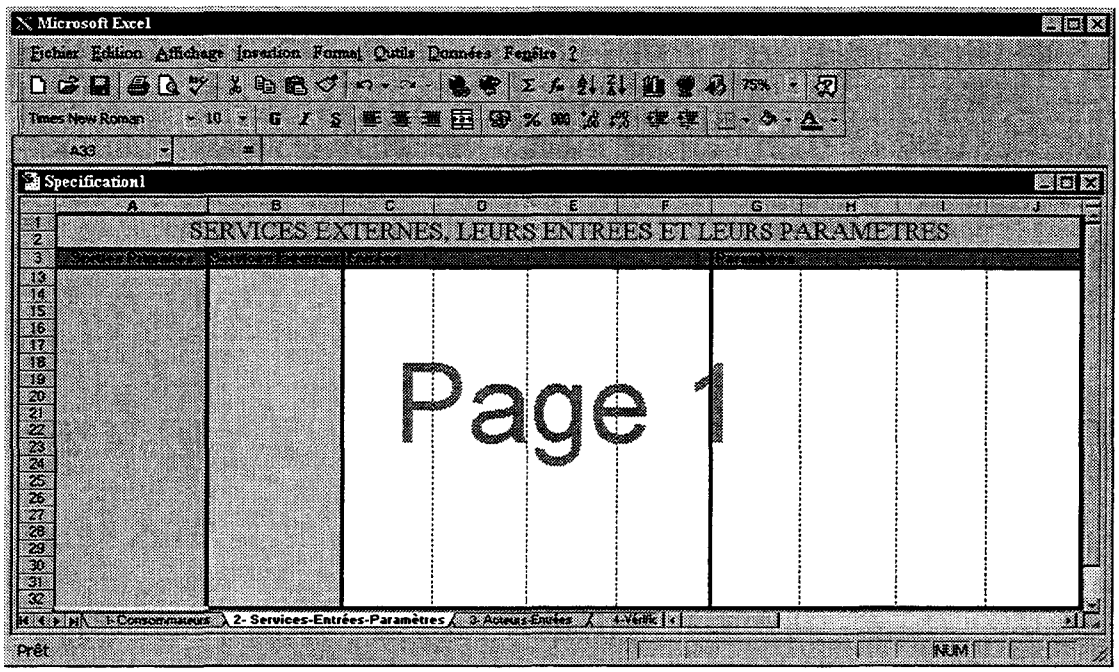


Figure 2 : Feuille Services-Entrées-Paramètres

2.3- Feuille Acteurs-Entrées :

Sur cette feuille, on vérifie qui produit les Entrées des différents services. Si l'entrée est produite par un autre service, le service producteur est signalé, dans le cas contraire un message apparaît pour signaler qu'il faut entrer un Acteur Extérieur la produisant dans la colonne correspondante. Si un acteur est entré alors qu'un service la produit, un message d'erreur apparaît sur la ligne correspondante. La troisième feuille est représentée par la figure 3 ci-après.

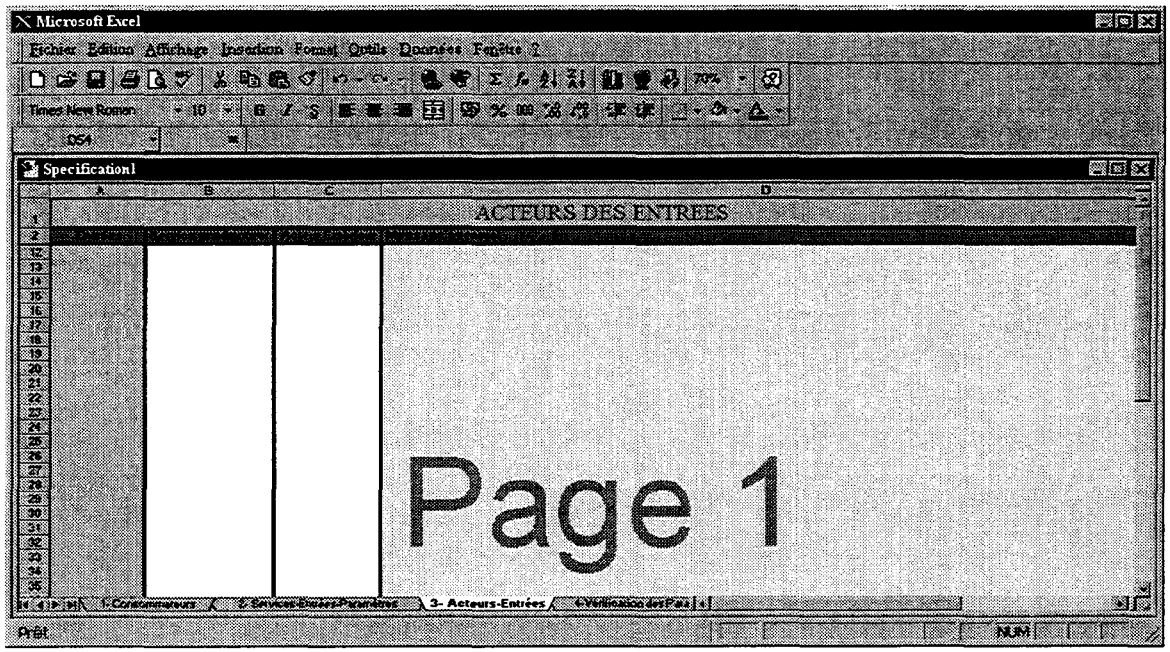


Figure 3 : Feuille Acteurs-Entrées.

2.4- Feuille Vérification des Paramètres :

Cette feuille est donnée ci dessous figure 4. Rien n'est à entrer ici. On y indique simplement si un service produit effectivement le paramètre indiqué dans la première colonne. Si ce n'est pas le cas, un message indique qu'il faut compléter l'ensemble des données consommées (Feuille Consommateurs) pour prendre en compte le paramètre en question. Cela permet, en fait, de définir un nouveau service de paramétrage.

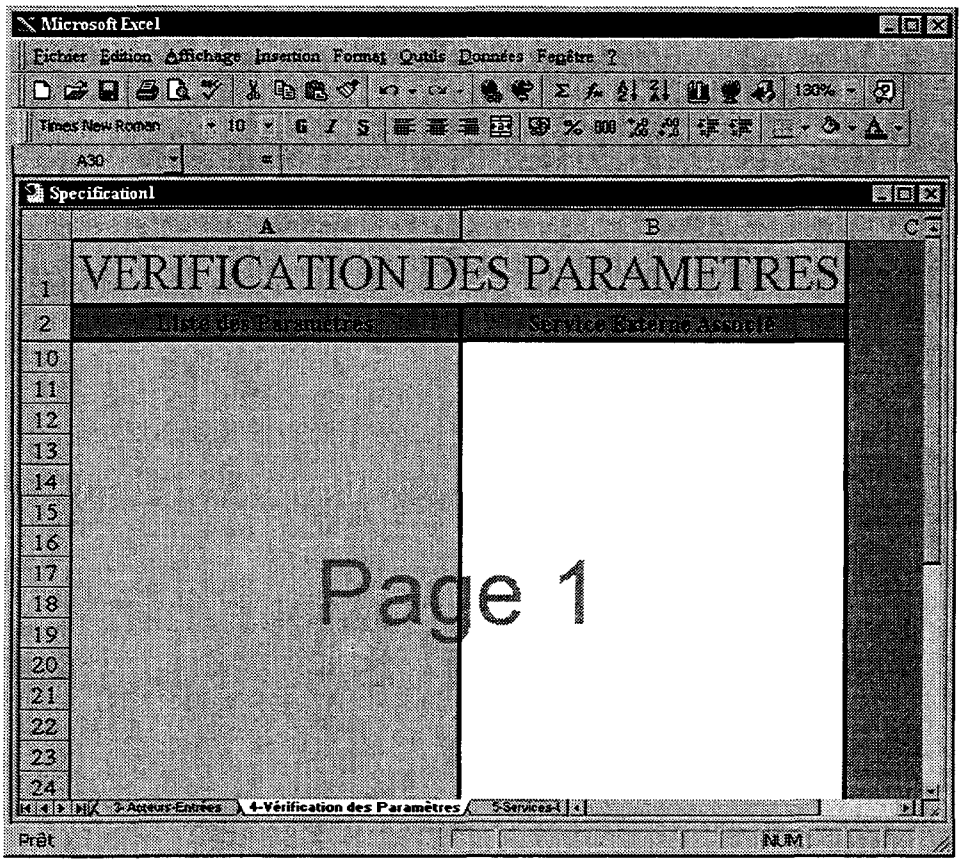


Figure 4 : Feuille Vérification des Paramètres.

2.5- Feuille Services-Cond. d'Activation :

Chaque service apparaît dans la première colonne. Pour chacun d'eux, on peut entrer jusqu'à quatre conditions d'activation différentes. Celles-ci sont sous la forme de [Conditions d'Utilisation] Requête. Chacune d'elle s'entre ligne par ligne.

Pour les Conditions d'Utilisation, on peut entrer jusqu'à huit termes différents. Les conditions d'utilisation sont des variables booléennes. Afin de ne prendre en compte un minimum de variables, on notera pour la variable X les valeurs correspondantes suivantes :

- X = VRAI : X
- X = FAUX : nonX ou NONX

Cependant, on impose pour les conditions d'utilisation, que "non" et "NON" soient des mots réservés et ne doivent pas non plus apparaître à l'intérieur des noms des variables ! Par exemple, ne pas prendre de variable ayant pour nom "CANON" qui contient "NON".

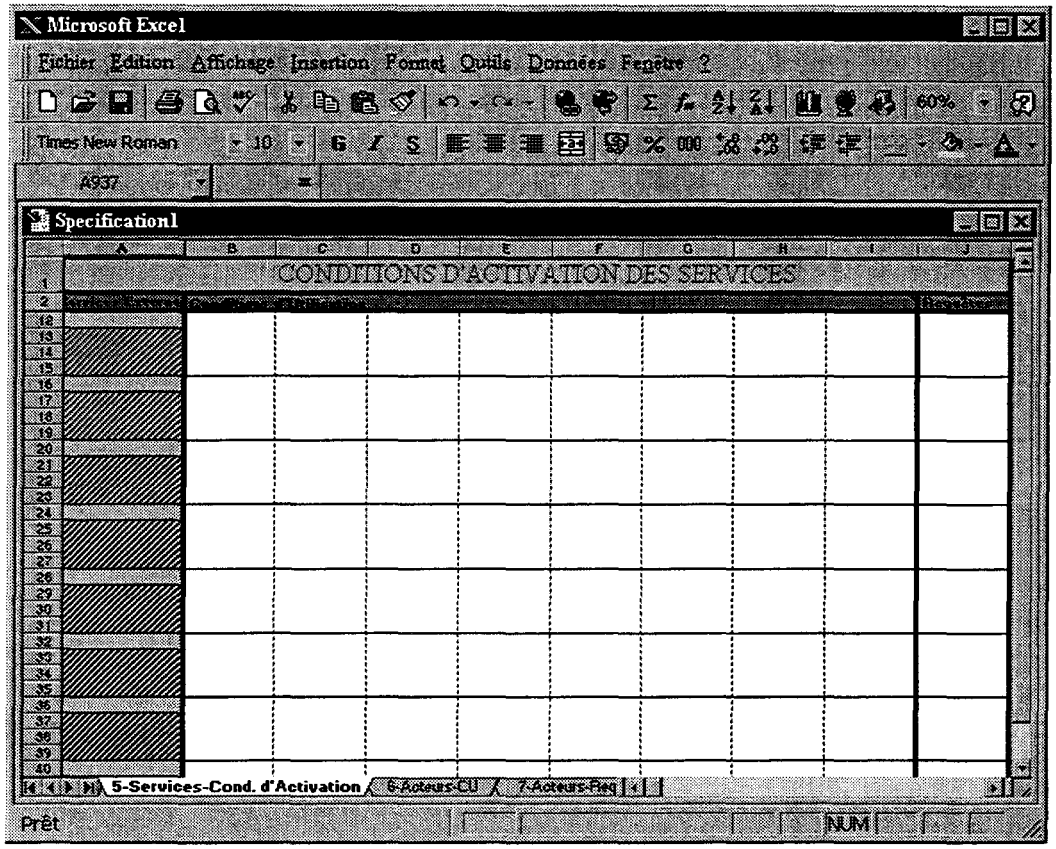


Figure 5 : Feuille Services-Cond. d'Activation.

2.6- Feuilles Acteurs-CU et Acteurs-Requêtes :

Ces deux feuilles sont similaires dans leur rôle : on vérifie qui produit les termes des Conditions d'Utilisation et les requêtes.

Cette feuille ressemble à celle des Entrées : chaque terme des Conditions d'Utilisation ou Requête est affiché dans la première colonne. On a alors plusieurs cas :

- si la donnée est la sortie principale d'un service, celui-ci est indiqué dans la colonne "Service Producteur",
- si la donnée n'est pas la sortie principale d'un service, on a trois choix :
  - la définir comme une sortie secondaire d'un service existant, il faut alors entrer dans la colonne "Sortie secondaire" le service concerné,
  - indiquer l'acteur extérieur qui la fournit dans la colonne correspondante,
  - la définir une nouvelle donnée consommée dans la première feuille et reprendre la démarche.

Une colonne est réservée aux messages afin d'indiquer au concepteur ce qu'il doit faire et/ou lui indiquer des erreurs de saisie comme l'éventuel cas où la donnée serait fournie par plusieurs moyens.

Les figures 6 et 7 donnent les feuilles correspondantes :

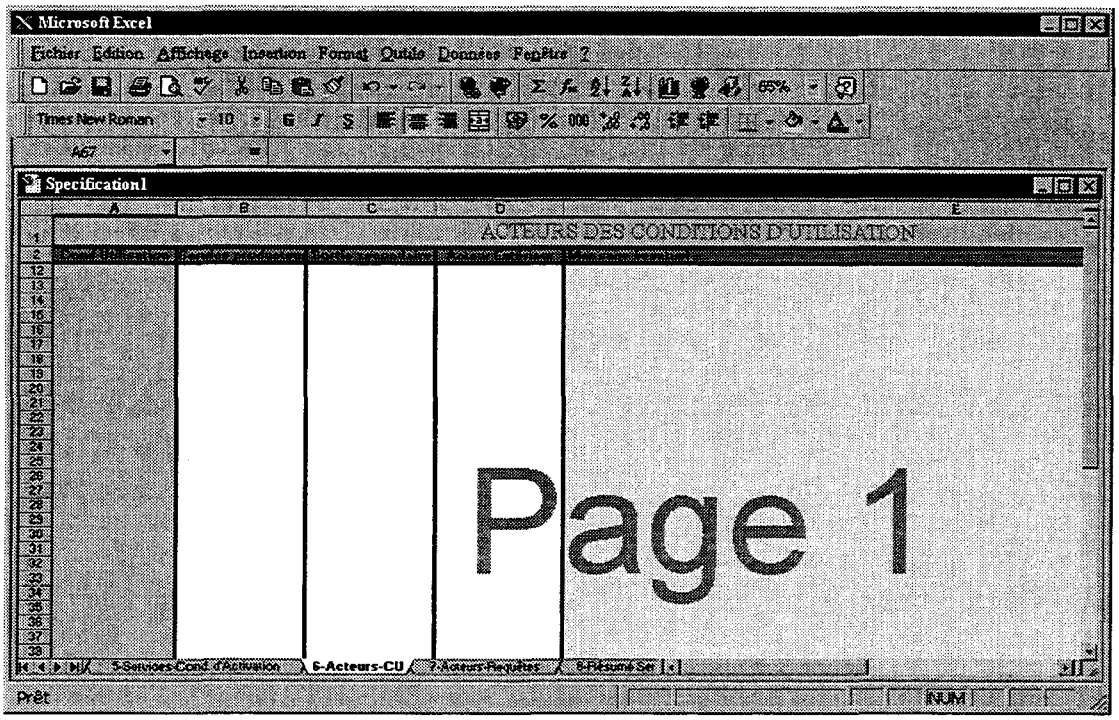


Figure 6 : Feuille Acteurs-CU.

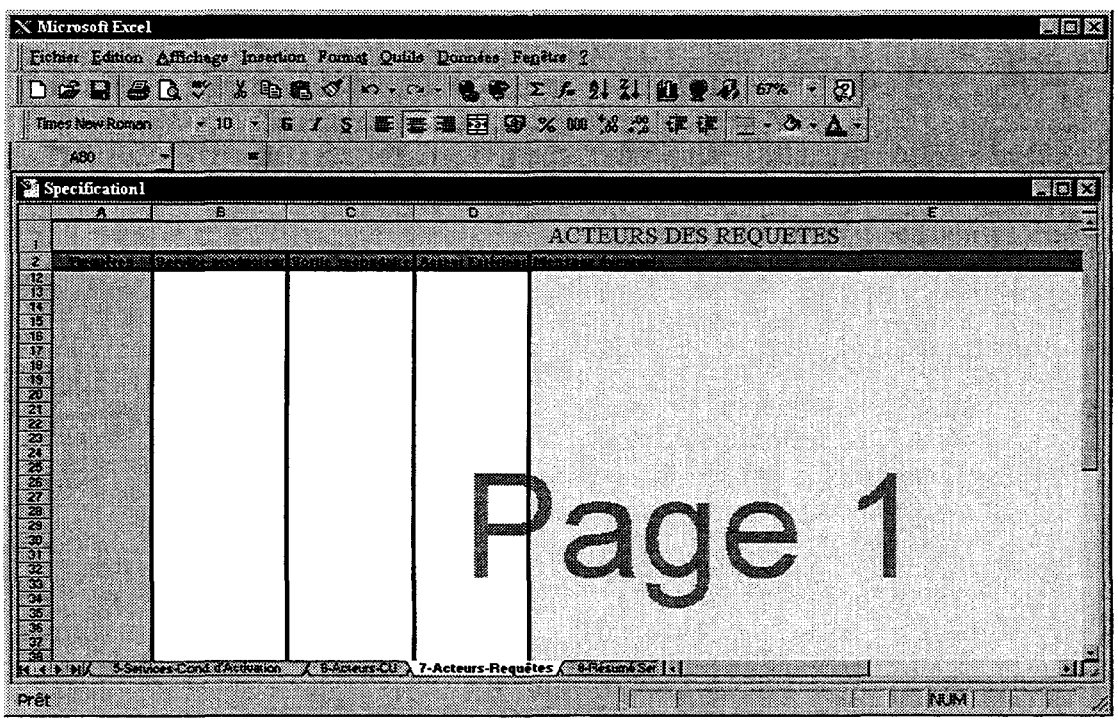


Figure 7 : Feuille Acteurs-Requêtes.

2.7- Feuilles Résumé Services et Résumé Utilisateur :

Ces deux feuilles résument l'ensemble des données qu'on a entré dans les feuilles précédentes. Il n'y a donc rien à entrer dans ces feuilles qui sont données à titre indicatif et pour faciliter le passage au fichier Analyse_Spécification qui suit dans l'annexe II. Ces deux dernières feuilles sont données ci-après aux figures 8 et 9.

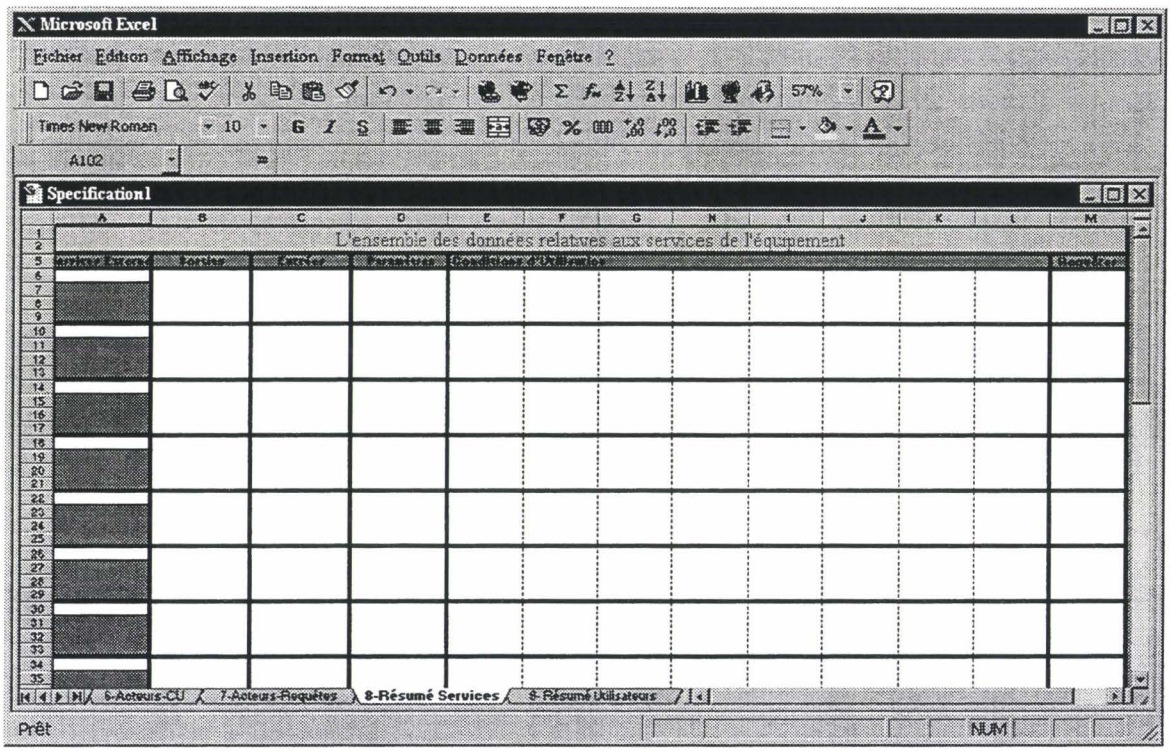


Figure 8 : Feuille Résumé Services.

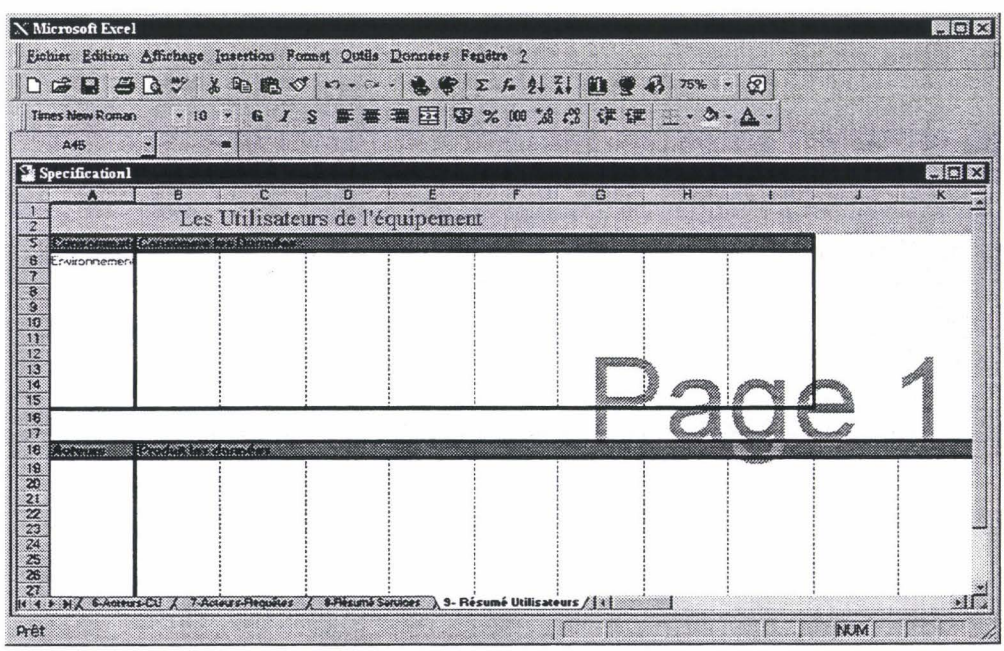


Figure 9 : Résumé Utilisateurs.

Annexe II :

Présentation l'analyse de la spécification sous Excel97

1- Introduction :

On utilise un autre fichier afin d'analyser la spécification obtenue à partir d'un fichier Specification. L'outil d'analyse de la spécification générale qui est sous la forme du fichier **Analyse_Specification.xlt**. Ce fichier modèle est aussi à mettre dans le répertoire C:\Program Files\Micosoft Office\Modèles. Pour utiliser le fichier d'analyse de la spécification, on doit faire les opérations suivantes :

- lancer Excel97,
- dans la barre d'outil standard, sélectionner Fichier - Nouveau,
- sélectionner l'icône Analyse_Specification.xlt.

Une fois sélectionnée, on obtient un fichier de spécification d'une application vierge qu'on doit compléter. Ce fichier est constitué des plusieurs feuilles suivantes :

SERVICES
UTILISATEURS
1-Analyse services, Boucles Dyn
2-Analyse services, Paramétrage
MODES D'UTILISATION
3-Analyse Modes-Services
4-Changements de Mode
5-Acteurs CCM
6-Acteurs Requêtes CM
7-Connexité du graphe des modes
Résumé Utilisateurs
Résumé Services
Résumé Modes d'Utilisation

Ces treize feuilles sont décrites ci après.

1- Feuilles SERVICES et UTILISATEURS :

Ces deux feuilles sont identiques dans leur forme et présentation aux feuilles correspondantes du fichier Specification. On peut alors faire un "Copier-Coller" tout

simple ou encore faire une liaison entre le fichier Specification1.xls et son fichier analyse Analyse_Specification1.xls correspondant. Cette manœuvre se fait comme suit :

- ouvrir le fichier Specification1.xls correspondant à la spécification de l'équipement intelligent,
- sélectionner la feuille 8-Résumé Services,
- sélectionner dans cette feuille la totalité du tableau des services,
- retourner au fichier Analyse_Specification1.xls, sélectionner la même plage de cellules,
- dans la barre d'outil standard, sélectionner Edition - Collage Spécial,
- sélectionner l'option Valeurs dans les options et cliquer sur Coller avec liaison,
- recommencer l'opération pour les utilisateurs.

Les deux fichiers sont alors en liaison, chaque modification de Specification1.xls affectera Analyse_Specification1.xls s'ils sont ouverts au même moment ou à chaque ouverture de Analyse_Specification1.xls au choix du concepteur.

2- Feuilles Analyse services, Boucles Dyn et Analyse services, Paramétrage :

Ces deux feuilles calculent les tableaux successifs Boucles et Paramétrage jusqu'au rang 4, il n'y a donc rien à entrer dans ces deux feuilles. Jusqu'au rang 3, ils signalent les ensembles de services en boucles dynamiques et les ensembles de services en paramétrage croisé. Au rang 4, on y indique simplement l'existence possible de quatre services en boucles dynamiques et de quatre services en paramétrage croisé. Un exemple de ces feuilles est donné en Annexe III.

3- Feuille MODES D'UTILISATION :

On entre sur cette feuille, figure 1, jusqu'à huit modes d'utilisation qui sont précisés dans le cahier des charges ou qui ont été déterminés par la méthode établie au chapitre III.

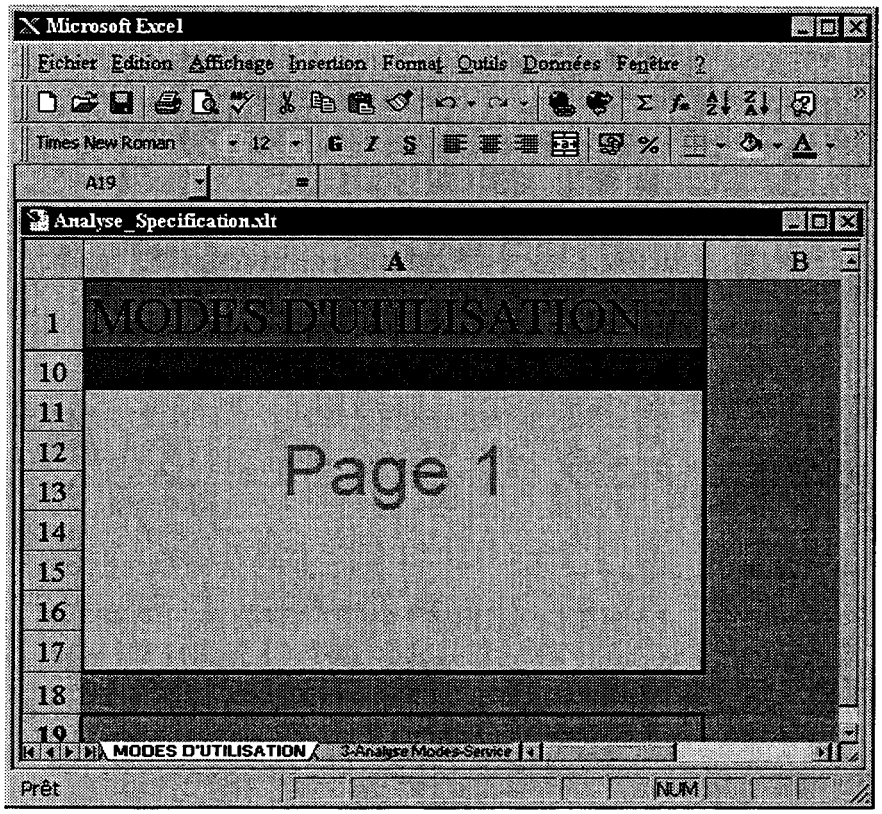


Figure 1 : Feuille MODES D'UTILISATION

4- Feuille Analyse Modes-Services :

Sur cette feuille, le concepteur doit entrer dans chaque case du tableau si le service en ligne appartient au mode d'utilisation en colonne. Il faut mettre un 1 si c'est le cas, sinon on met un 0 ou on ne met rien.

On vérifie aussi que chaque service appartient à au moins un mode d'utilisation ce qui apparaît dans les commentaires. La figure 2 représente la feuille Analyse Modes-Services.

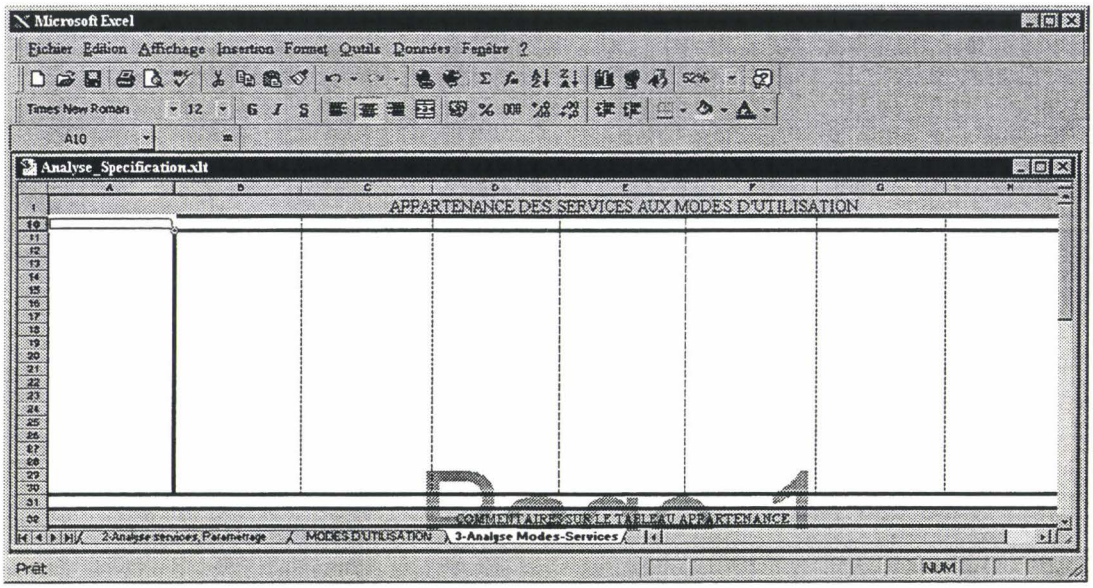


Figure 2 : Feuille Analyse Modes-Services.

5- Feuille Changements de Mode :

Cette feuille est l'équivalent de la feuille Services-Cond. d'Activation pour les changements de mode d'utilisation à laquelle on a ajouté les commandes. On a aussi limité les possibilités de changement d'un mode à un autre à deux.

Comme pour les Conditions d'utilisation, on peut entrer jusqu'à huit termes différents pour les Conditions de Changement de Mode. Ces conditions sont des variables booléennes. Afin de ne prendre en compte qu'un minimum de variables, on notera pour la variable X les valeurs correspondantes suivantes :

- X = VRAI : X
- X = FAUX : nonX ou NONX

Cependant, on impose pour les conditions, que "non" et "NON" soient des mots réservés et ne doivent pas non plus apparaître à l'intérieur des noms des variables ! Par exemple, ne pas prendre de variable ayant pour nom "CANON" qui contient "NON".

La figure 3 représente une partie de la feuille Changements de Mode.

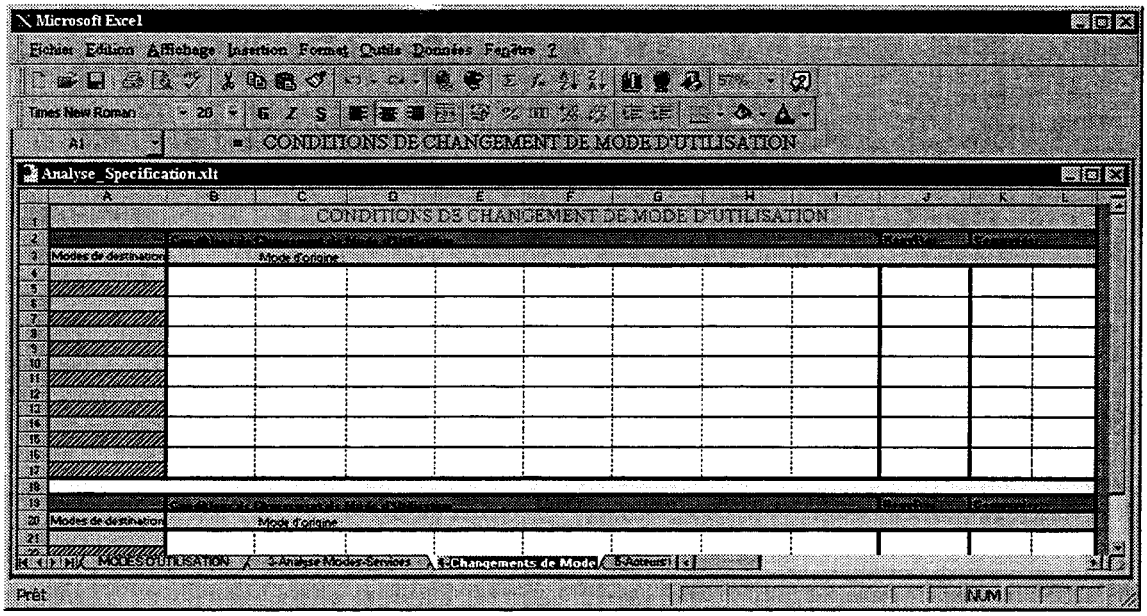


Figure 3 : Feuille Changements de Mode.

6- Feuilles Acteurs CCM et Acteurs RequêtesCM :

On a là encore l'équivalent des feuilles Acteurs-CU et Acteurs-Requêtes vues dans l'annexe I, le principe y est le même. Un exemple est présenté dans l'annexe III.

7- Feuille Connexité du graphe des modes :

Cette feuille est utilisée pour vérifier la connexité du graphe des modes d'utilisation. Il n'y a donc rien à y entrer. On y vérifie l'existence d'un seul mode initial et/ou final, et si le graphe est simplement connexe ou fortement connexe en fonction de l'existence des modes précédents. Un exemple est présenté dans l'annexe III.

8- Feuilles Résumé Utilisateurs, Résumé Services et Résumé Modes d'Utilisation :

Les deux premières feuilles sont identiques à celles présentées dans l'annexe I, elles prennent en compte les ajouts faits dans l'analyse de la spécification, notamment au niveau des changements de modes d'utilisation. La dernière feuille donne un résumé des Modes d'Utilisation et des transitions entre ceux-ci et est présenté figure 4.

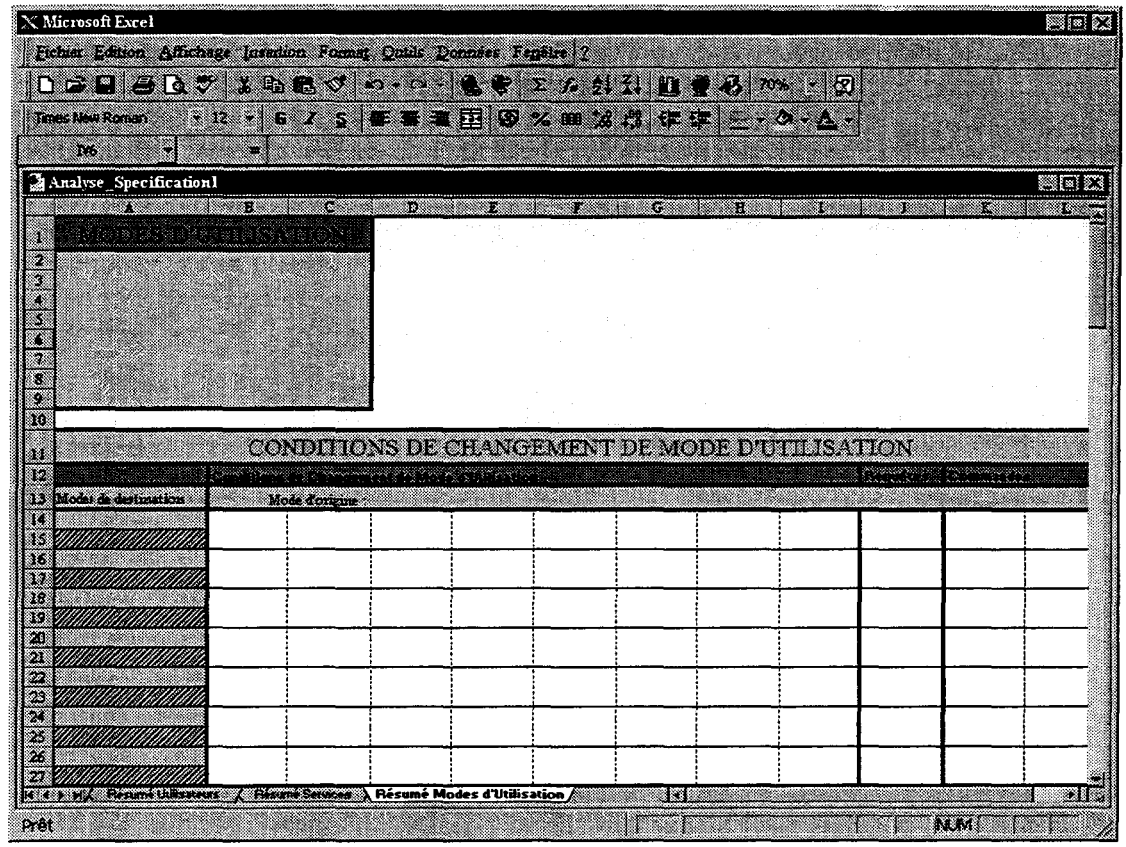
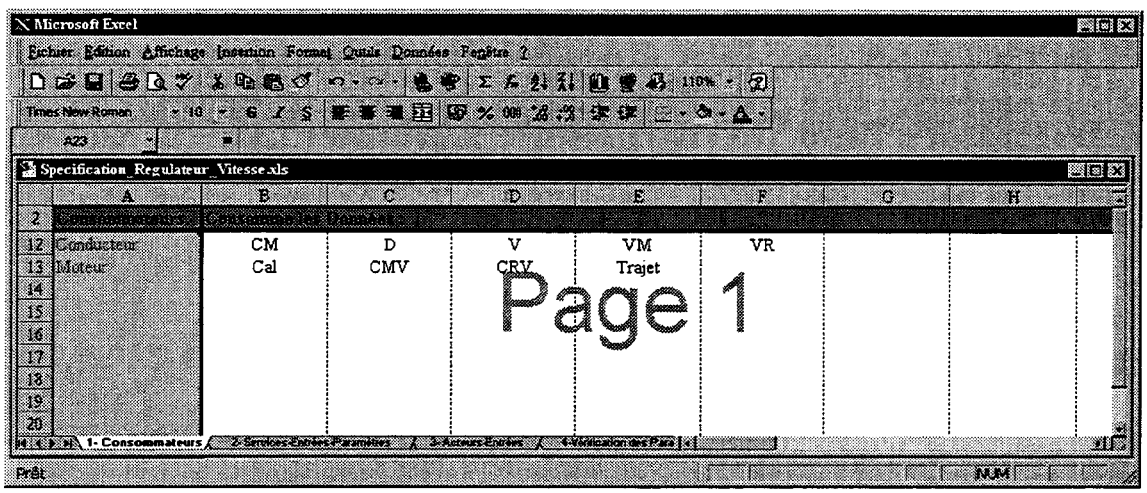


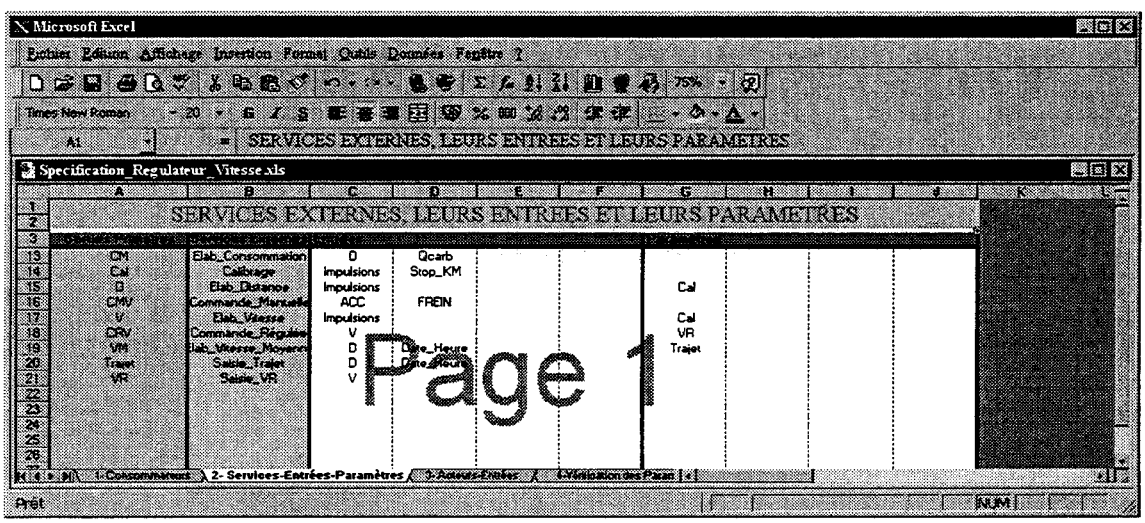
Figure 4 : Feuille Résumé Modes d'Utilisation.

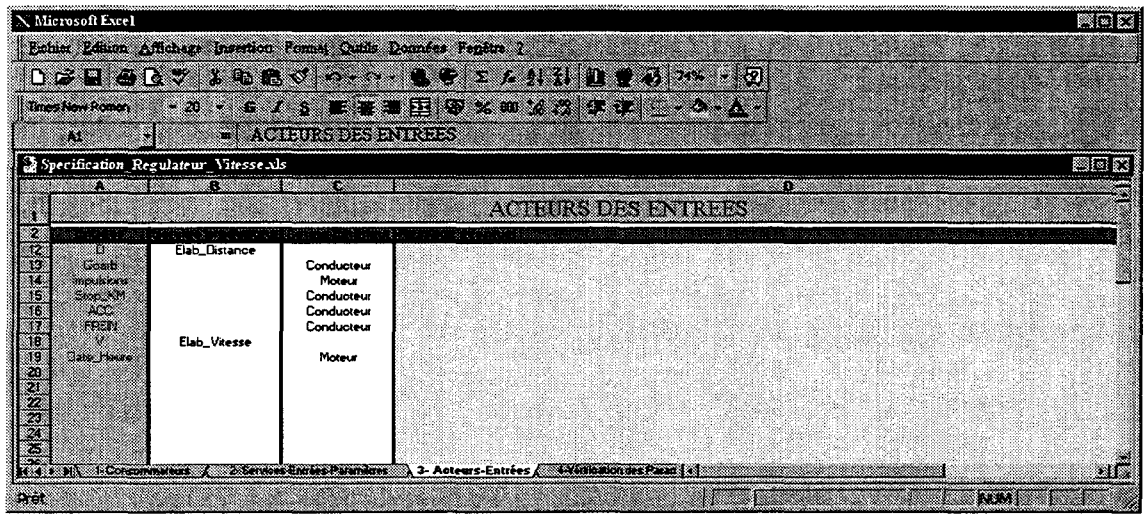
Annexe III :

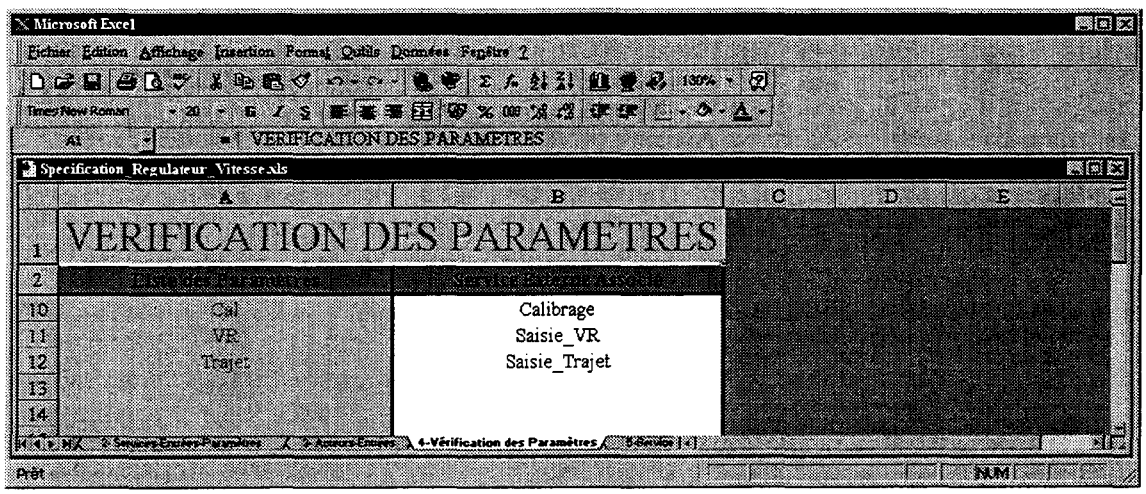
Feuilles de la spécification et de l'analyse de la spécification du Régulateur de Vitesse

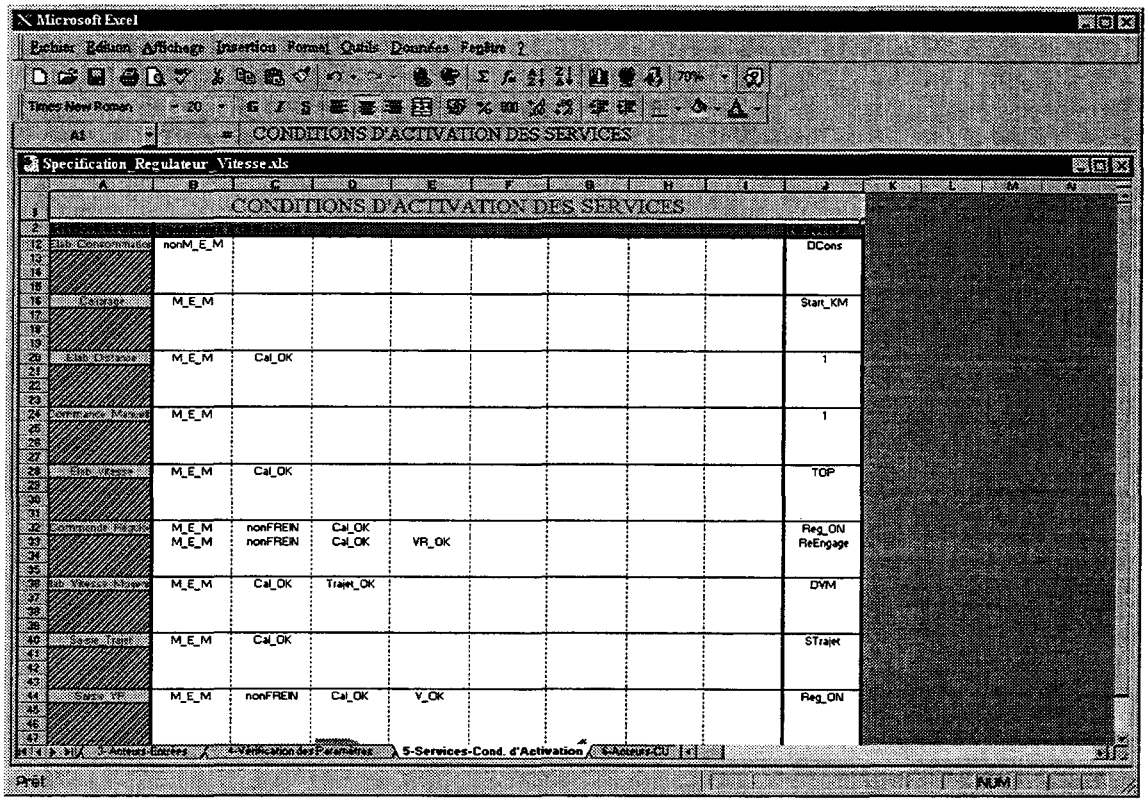
1- Spécification du régulateur de vitesse :

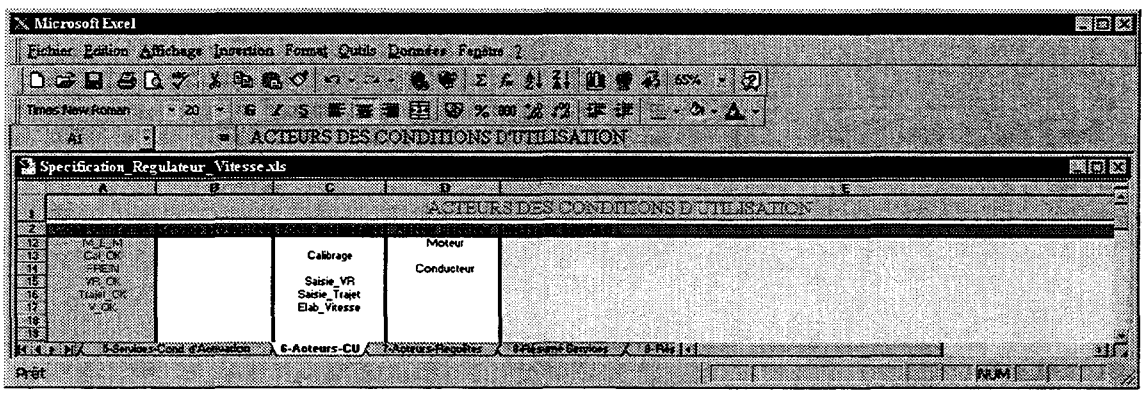


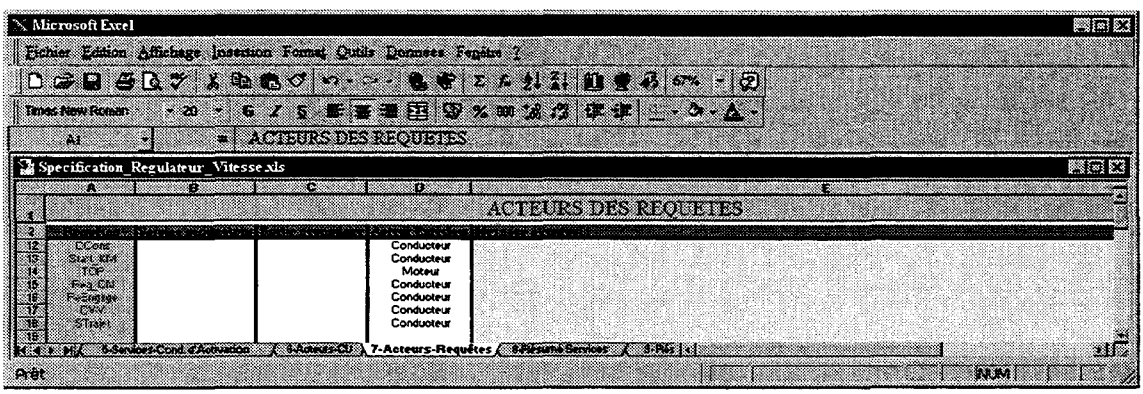


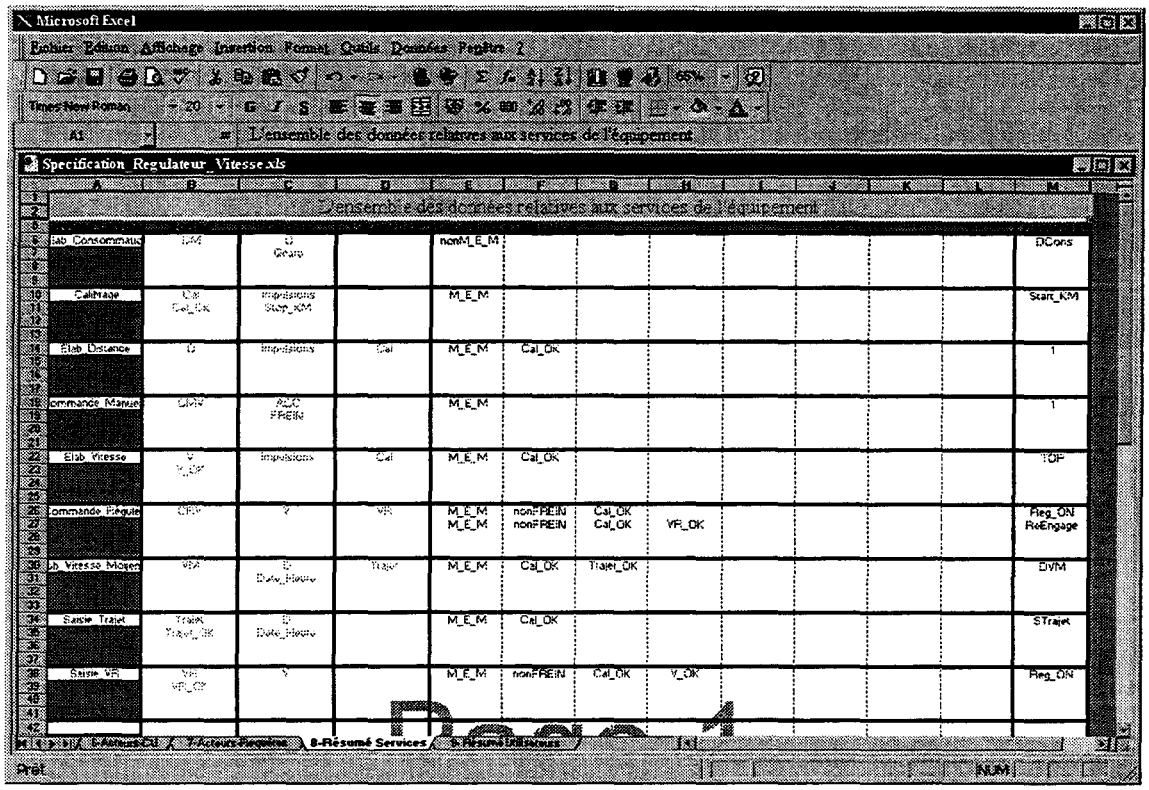

Microsoft Excel
Fichier Edition Affichage Insertion Format Outils Données Fenêtre ?
Times New Roman
A1
ACTEURS DES ENTREES
Specification_Regulateur_Vitesse.xls
ACTEURS DES ENTREES
Elab_Distance
Conducteur
Moteur
Conducteur
Conducteur
Conducteur
Elab_Vitesse
Moteur
2- Acteurs-Entrées
Prêt
NUM

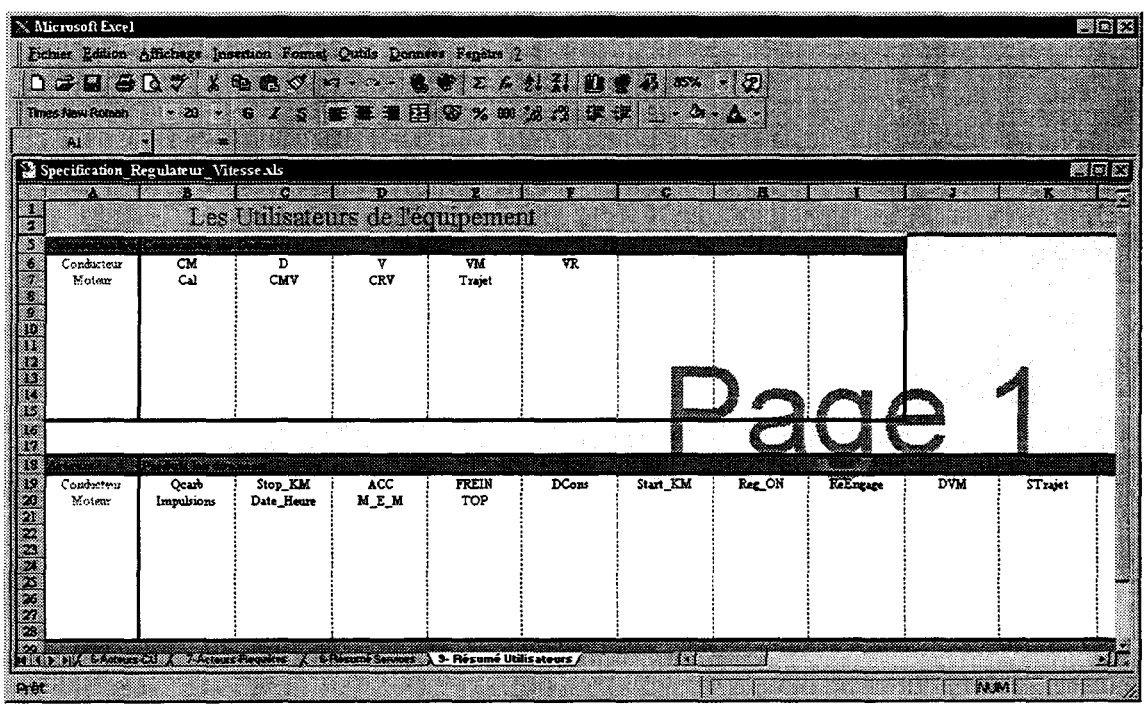

Microsoft Excel
Fichier Edition Affichage Insertion Format Outils Données Fenêtre ?
Times New Roman
A1
VERIFICATION DES PARAMETRES
Specification_Regulateur_Vitesse.xls
VERIFICATION DES PARAMETRES
Calibrage
Saisie_VR
Saisie_Trajet
4-Vérification des Paramètres
Prêt
NUM


Microsoft Excel
CONDITIONS D'ACTIVATION DES SERVICES
Specification_Regulateur_Vitesse.xls
CONDITIONS D'ACTIVATION DES SERVICES
5-Services-Cond. d'Activation
6-Acteurs-CU

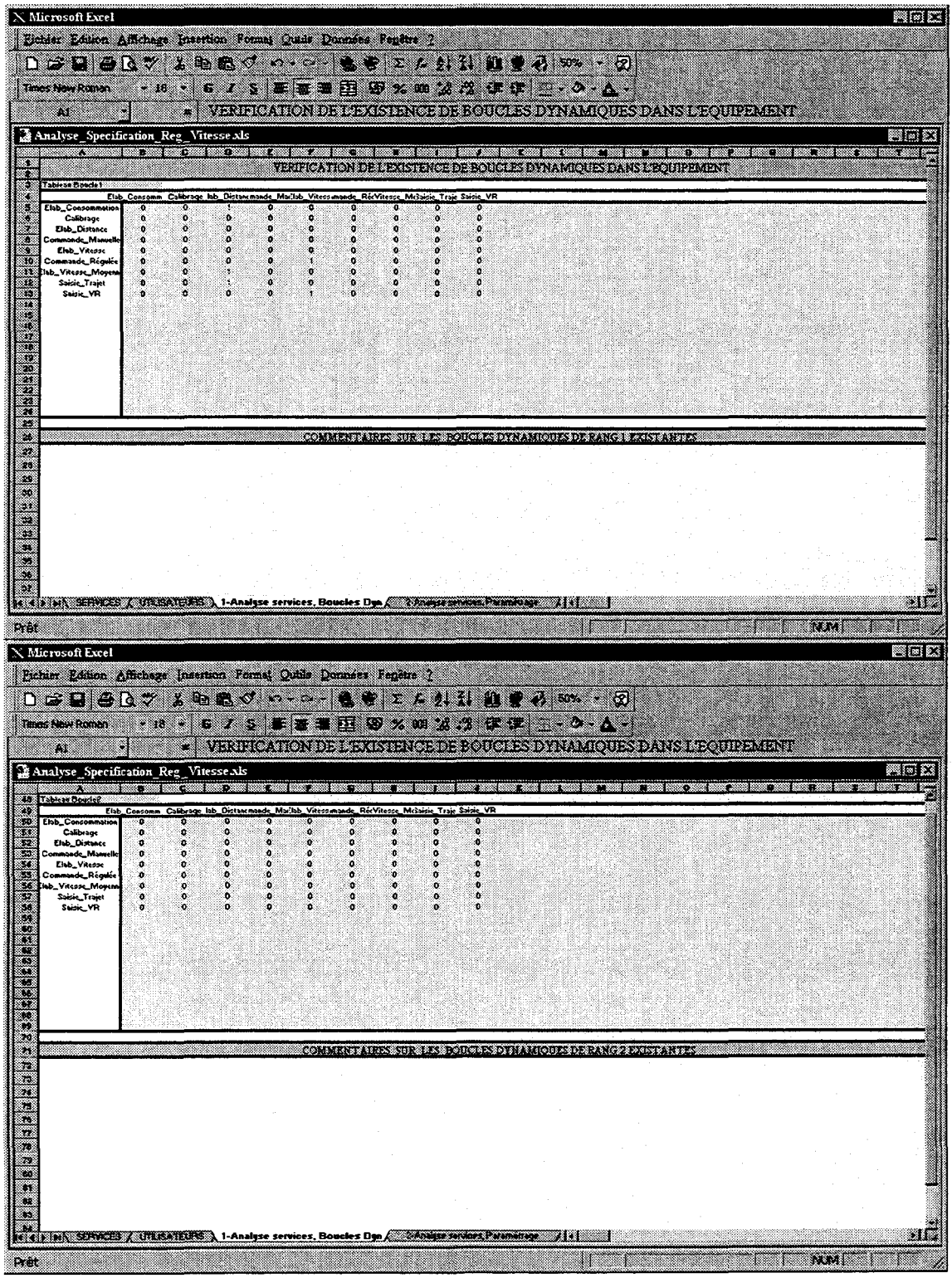

Microsoft Excel
ACTEURS DES CONDITIONS D'UTILISATION
Specification_Regulateur_Vitesse.xls
ACTEURS DES CONDITIONS D'UTILISATION
Calibrage
Saisie_VR
Saisie_Trajet
Elab_Vitesse
Moteur
Conducteur
5-Services-Cond. d'Activation
6-Acteurs-CU
7-Acteurs-Requêtes

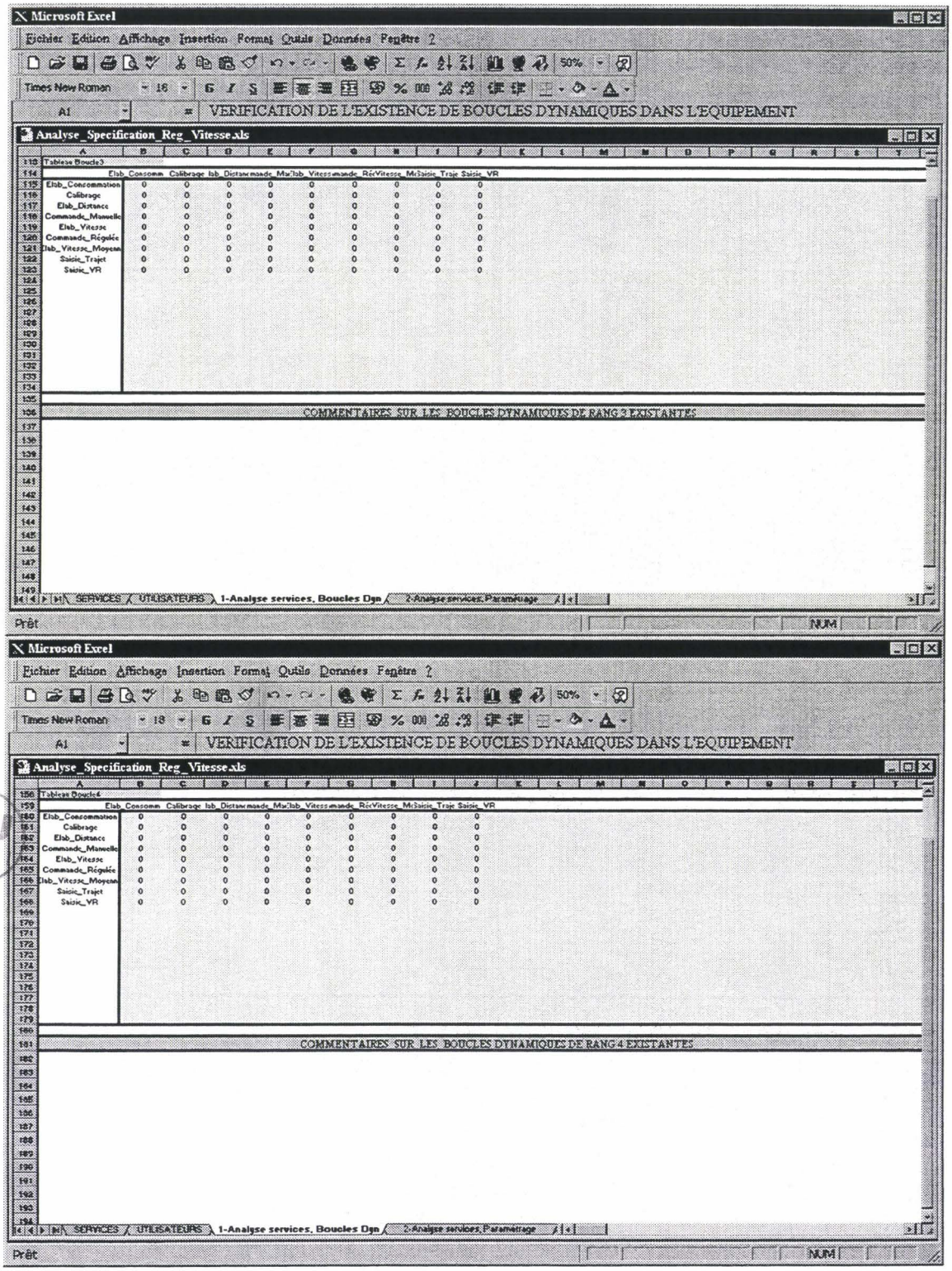

Microsoft Excel
ACTEURS DES REQUETES
Specification_Regulateur_Vitesse.xls
ACTEURS DES REQUETES
Conducteur
Moteur
5-Services-Cond. d'Activation
6-Acteurs-CU
7-Acteurs-Requêtes


Microsoft Excel
Specification_Regulateur_Vitesse.xls
L'ensemble des données relatives aux services de l'équipement
Page 4

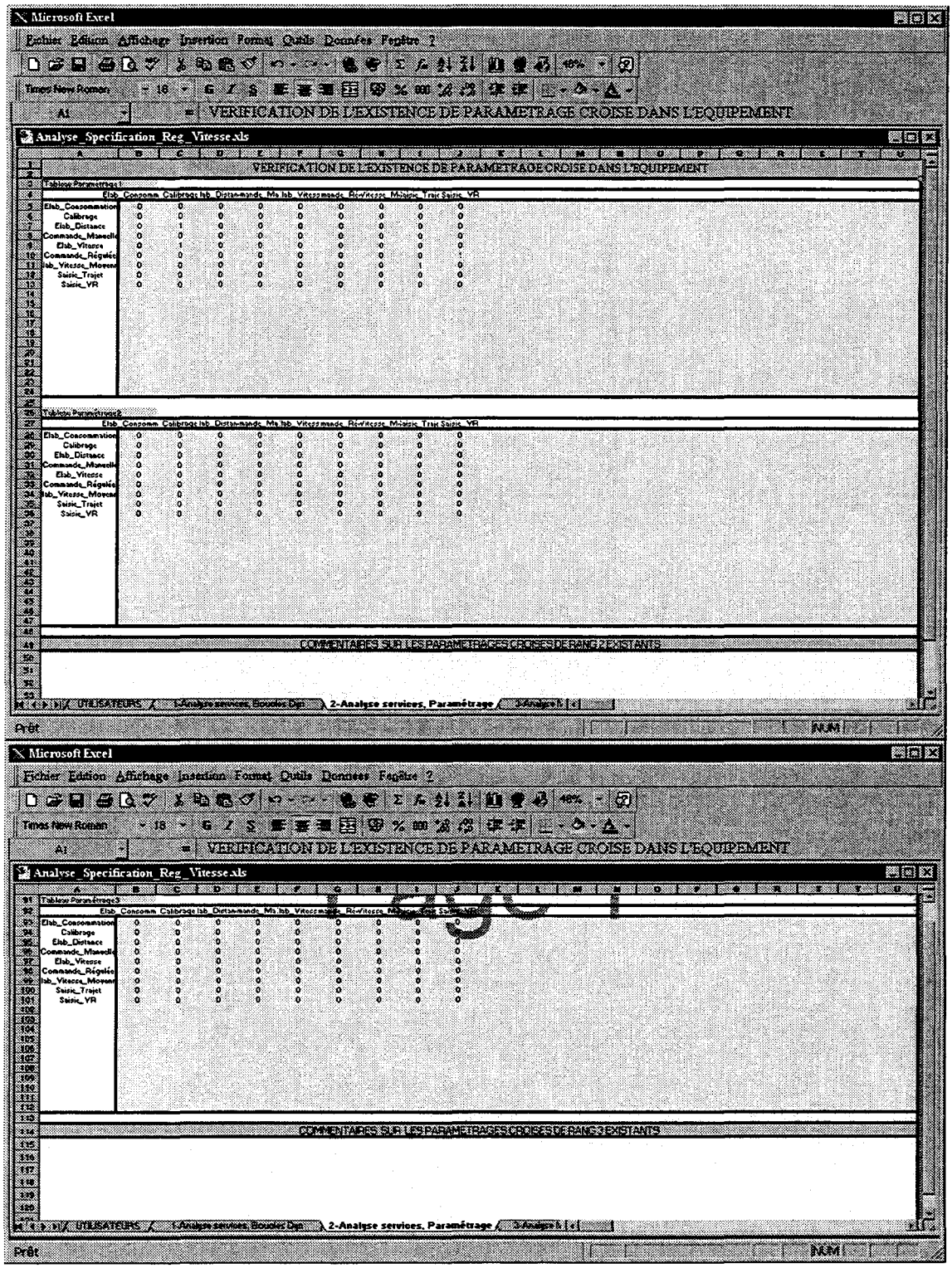

Microsoft Excel
Specification_Regulateur_Vitesse.xls
Les Utilisateurs de l'équipement
Conducteur
Moteur
CM
Cal
D
CMV
V
CRV
VM
Trajet
VR
Page 1
Conducteur
Moteur
Qcarb
Impulsions
Stop_KM
Date_Heure
ACC
M_E_M
FREIN
TOP
DCons
Start_KM
Reg_ON
ReEngage
DVM
STrajet

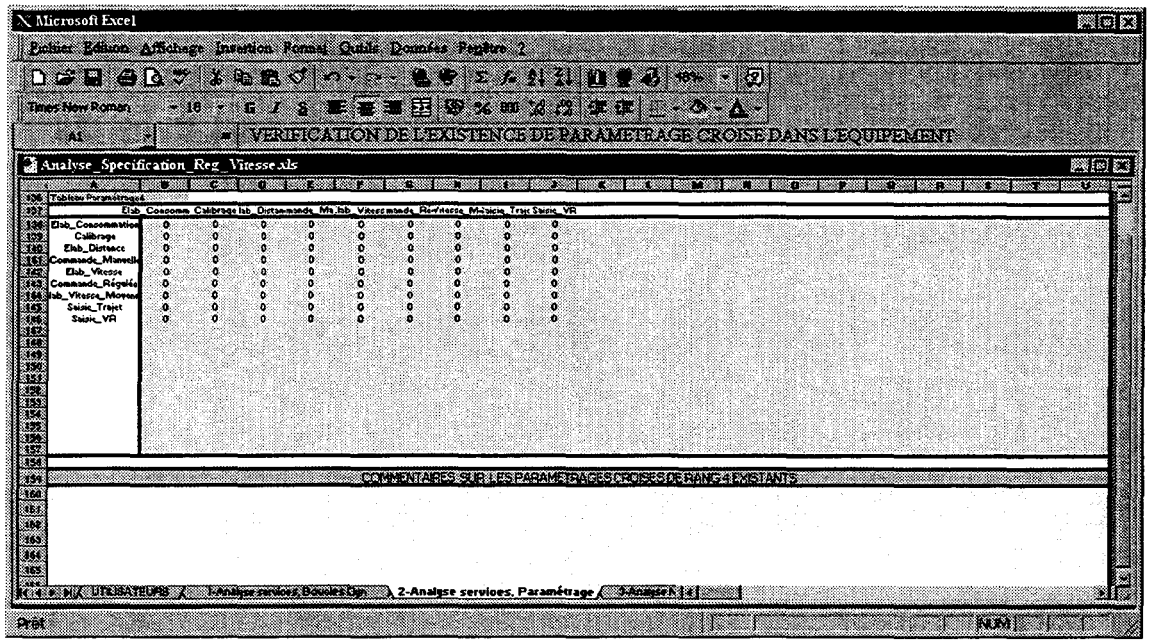
2- Analyse de la spécification du régulateur de vitesse :

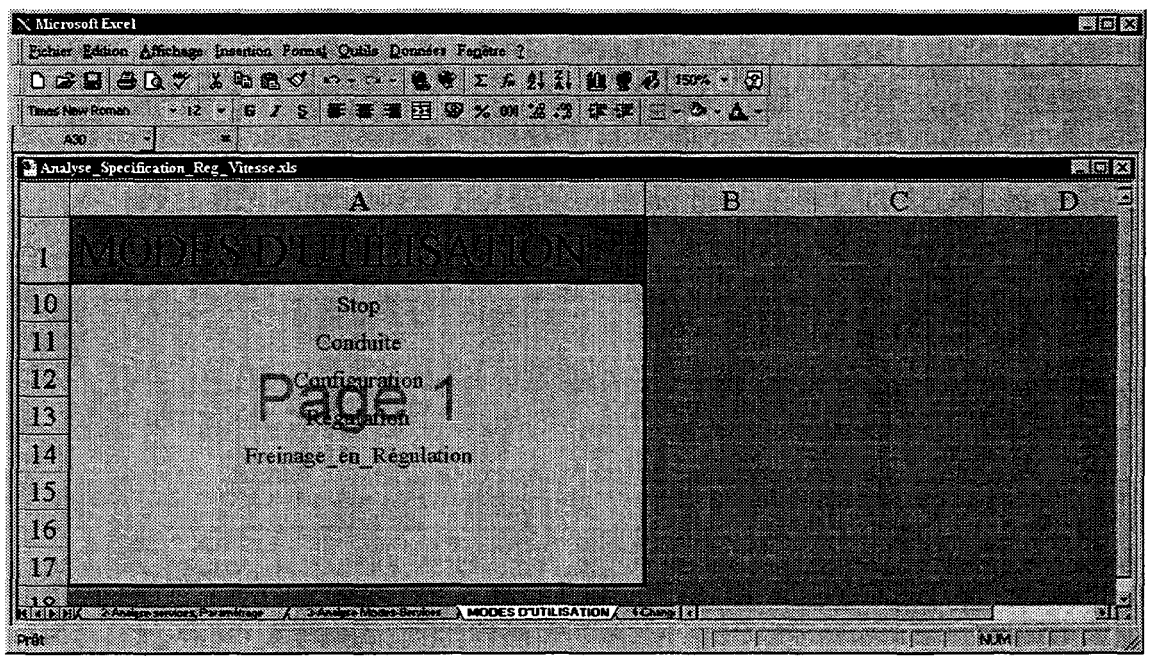


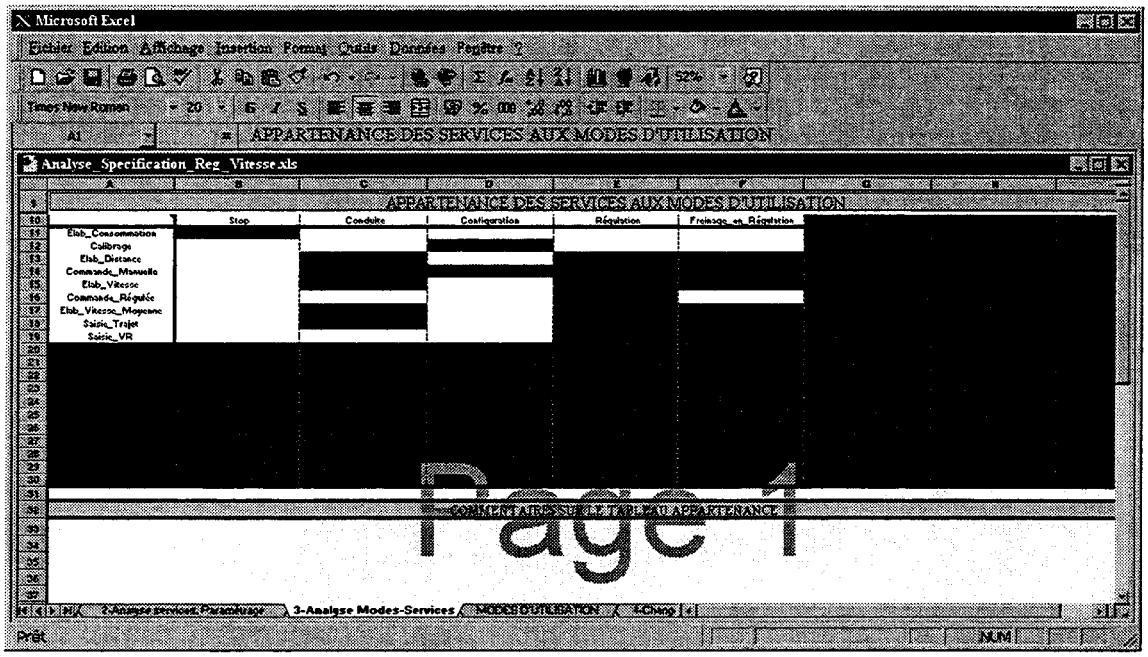

Microsoft Excel
Fichier Edition Affichage Insertion Format Outils Données Fenêtre ?
Times New Roman
A1
VERIFICATION DE L'EXISTENCE DE BOUCLES DYNAMIQUES DANS L'EQUIPEMENT
Analyse_Specification_Reg_Vitesse.xls
Tableau Boucle3
Elsb_Consomm Calibrage Isb_Distance Commande_Manuelle Elsb_Vitesse Commande_Régulée Elsb_Vitesse_Moyenne Saisie_Traje Saisie_VR
Elsb_Consommation
Calibrage
Elsb_Distance
Commande_Manuelle
Elsb_Vitesse
Commande_Régulée
Elsb_Vitesse_Moyenne
Saisie_Traje
Saisie_VR
COMMENTAIRES SUR LES BOUCLES DYNAMIQUES DE RANG 3 EXISTANTES
SERVICES
UTILISATEURS
1-Analyse services, Boucles Dyn
2-Analyse services, Paramétrage
Prêt
NUM
Microsoft Excel
Fichier Edition Affichage Insertion Format Outils Données Fenêtre ?
Times New Roman
A1
VERIFICATION DE L'EXISTENCE DE BOUCLES DYNAMIQUES DANS L'EQUIPEMENT
Analyse_Specification_Reg_Vitesse.xls
Tableau Boucle4
Elsb_Consomm Calibrage Isb_Distance Commande_Manuelle Elsb_Vitesse Commande_Régulée Elsb_Vitesse_Moyenne Saisie_Traje Saisie_VR
Elsb_Consommation
Calibrage
Elsb_Distance
Commande_Manuelle
Elsb_Vitesse
Commande_Régulée
Elsb_Vitesse_Moyenne
Saisie_Traje
Saisie_VR
COMMENTAIRES SUR LES BOUCLES DYNAMIQUES DE RANG 4 EXISTANTES
SERVICES
UTILISATEURS
1-Analyse services, Boucles Dyn
2-Analyse services, Paramétrage
Prêt
NUM

BU LILLE

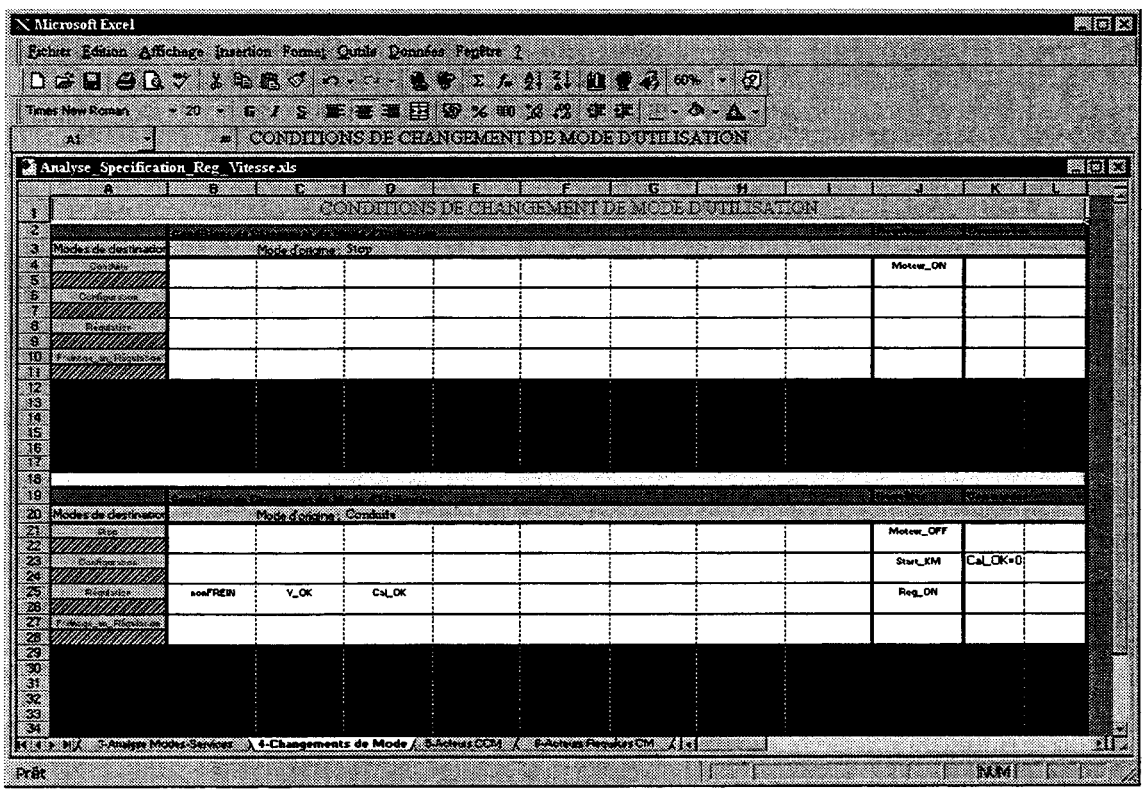

Microsoft Excel
VERIFICATION DE L'EXISTENCE DE PARAMETRAGE CROISE DANS L'EQUIPEMENT
Analyse_Specification_Reg_Vitesse.xls
COMMENTAIRES SUR LES PARAMETRAGES CROISES DE RANG 2 EXISTANTS
COMMENTAIRES SUR LES PARAMETRAGES CROISES DE RANG 3 EXISTANTS
2-Analyse services, Paramétrage
Prêt

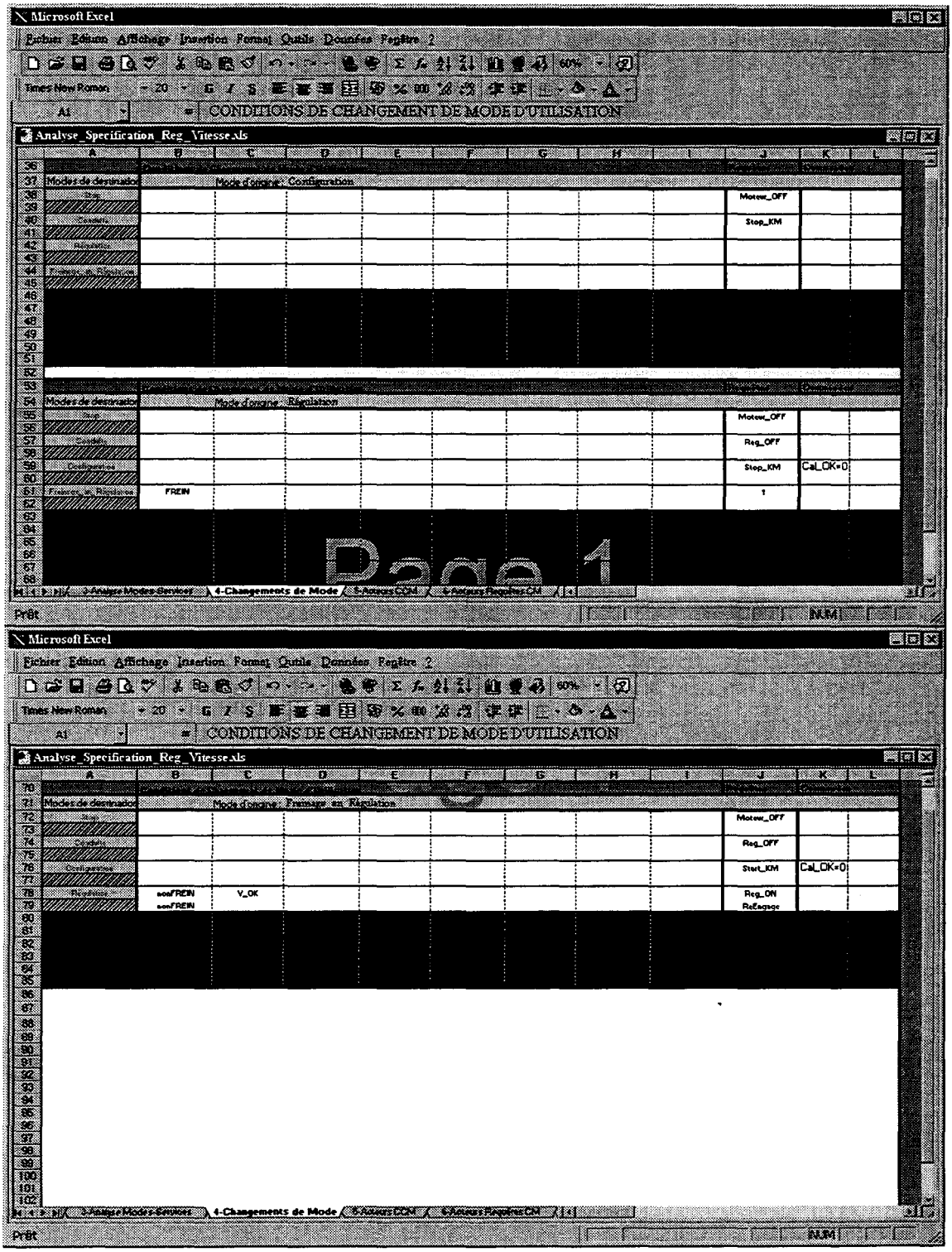

Microsoft Excel
VERIFICATION DE L'EXISTENCE DE PARAMETRAGE CROISE DANS L'EQUIPEMENT
Analyse_Specification_Reg_Vitesse.xls
COMMENTAIRES SUR LES PARAMETRAGES CROISES DE RANG 4 EXISTANTS
2-Analyse services, Paramétrage
Prêt
NUM

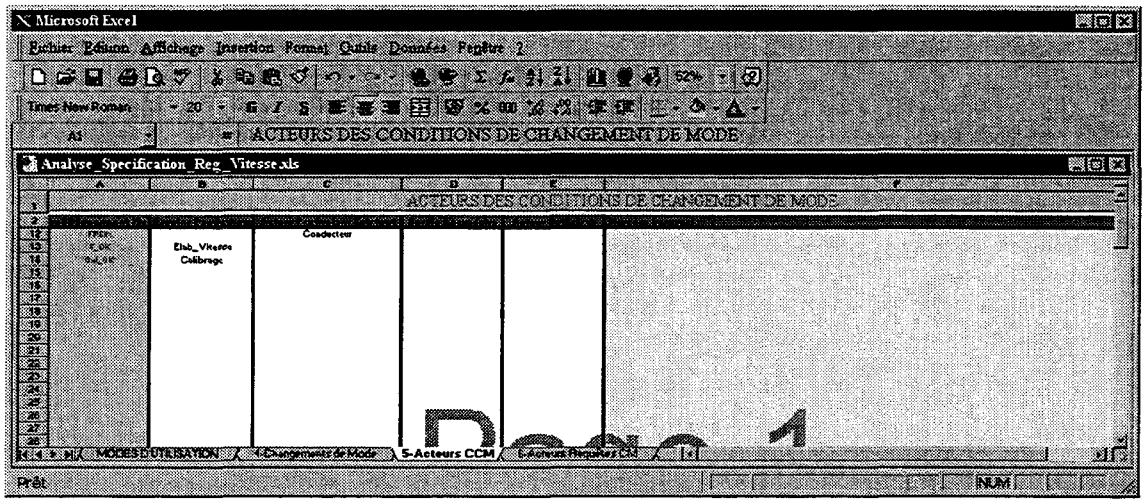

Microsoft Excel
Analyse_Specification_Reg_Vitesse.xls
MODES D'UTILISATION
Stop
Conduite
Configuration
Page 1
Régulation
Freinage_en_Régulation
MODES D'UTILISATION
Prêt
NUM

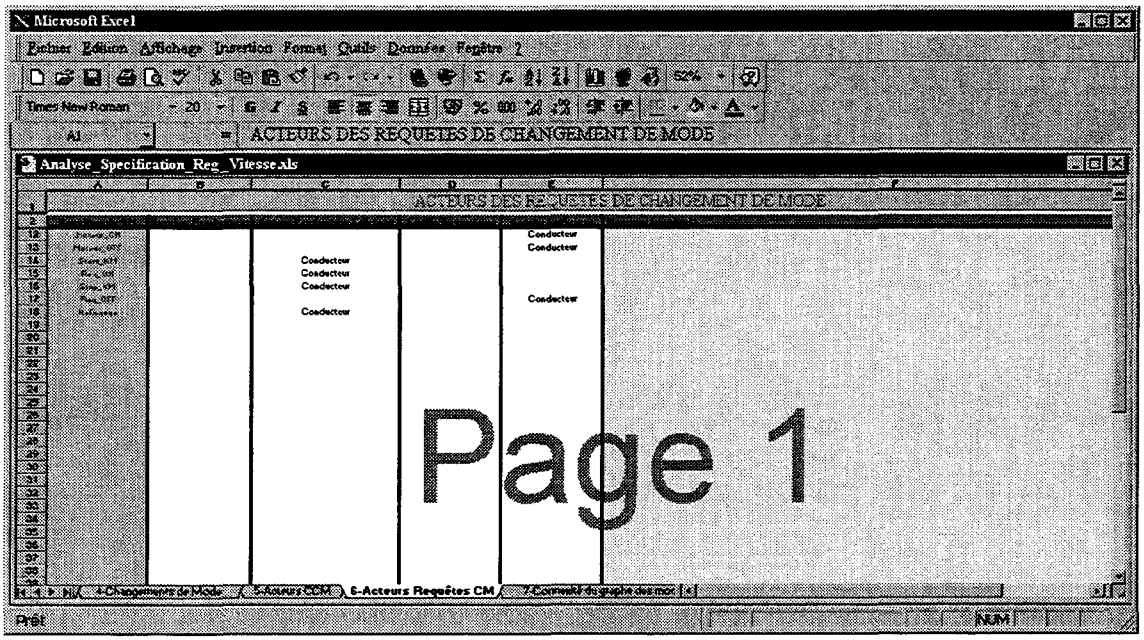

Microsoft Excel
APPARTENANCE DES SERVICES AUX MODES D'UTILISATION
Analyse_Specification_Reg_Vitesse.xls
APPARTENANCE DES SERVICES AUX MODES D'UTILISATION
Stop
Conduite
Configuration
Régulation
Freinage_en_Régulation
Etab_Consommation
Calibrage
Etab_Distance
Commande_Manuelle
Etab_Vitesse
Commande_Régulée
Etab_Vitesse_Moyenne
Saisie_Trajet
Saisie_VR
COMMENTAIRES SUR LE TABLEAU APPARTENANCE
3-Analyse Modes-Services
MODES D'UTILISATION
Prêt
NUM

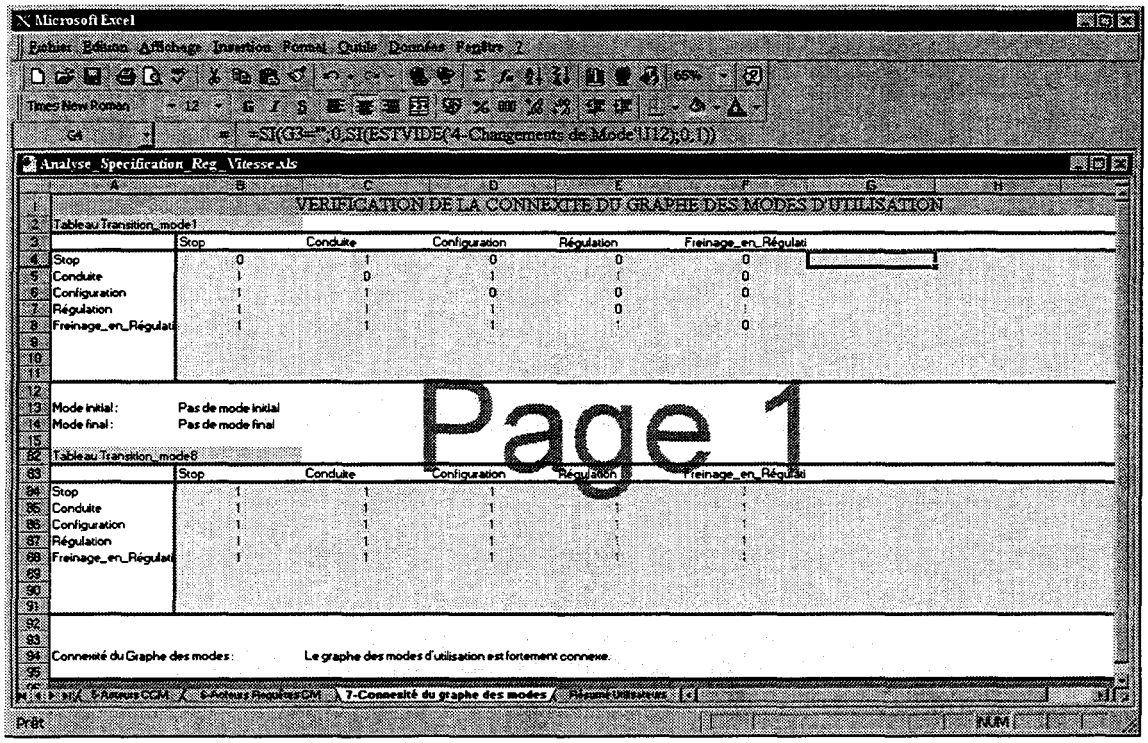
Page 1

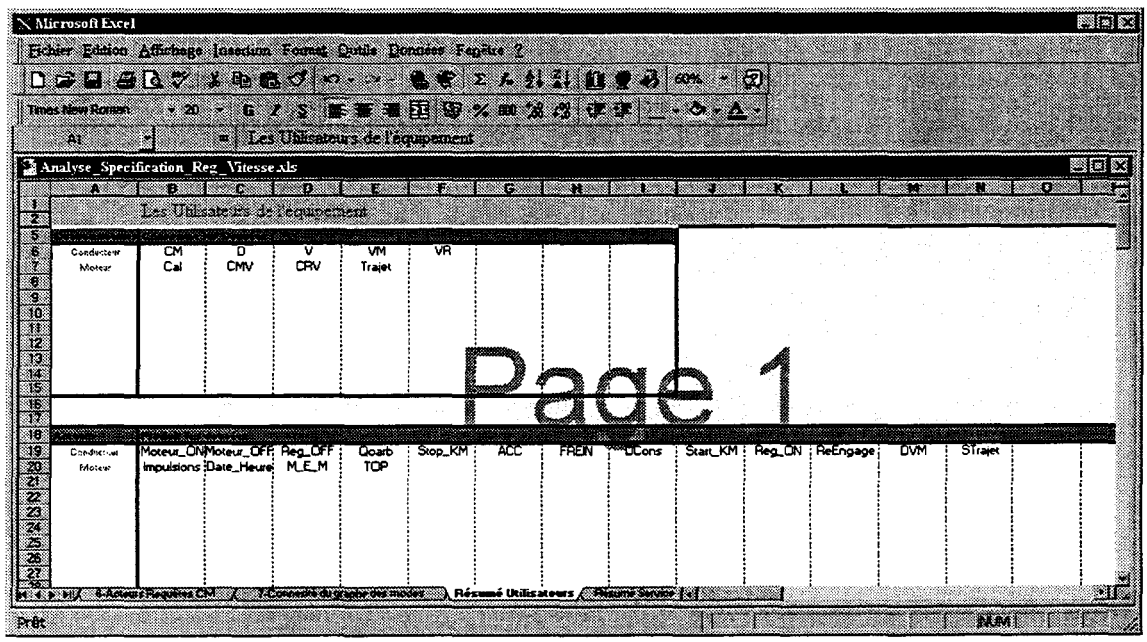

Microsoft Excel
CONDITIONS DE CHANGEMENT DE MODE D'UTILISATION
Analyse_Specification_Reg_Vitesse.xls
CONDITIONS DE CHANGEMENT DE MODE D'UTILISATION
Modes de destination
Mode d'origine : Stop
Moteur_ON
Mode d'origine : Conduite
Moteur_OFF
Start_KM
Cal_OK=0
Reg_ON
nonFREIN
V_OK
Cal_OK
4-Changements de Mode
Prêt
NUM

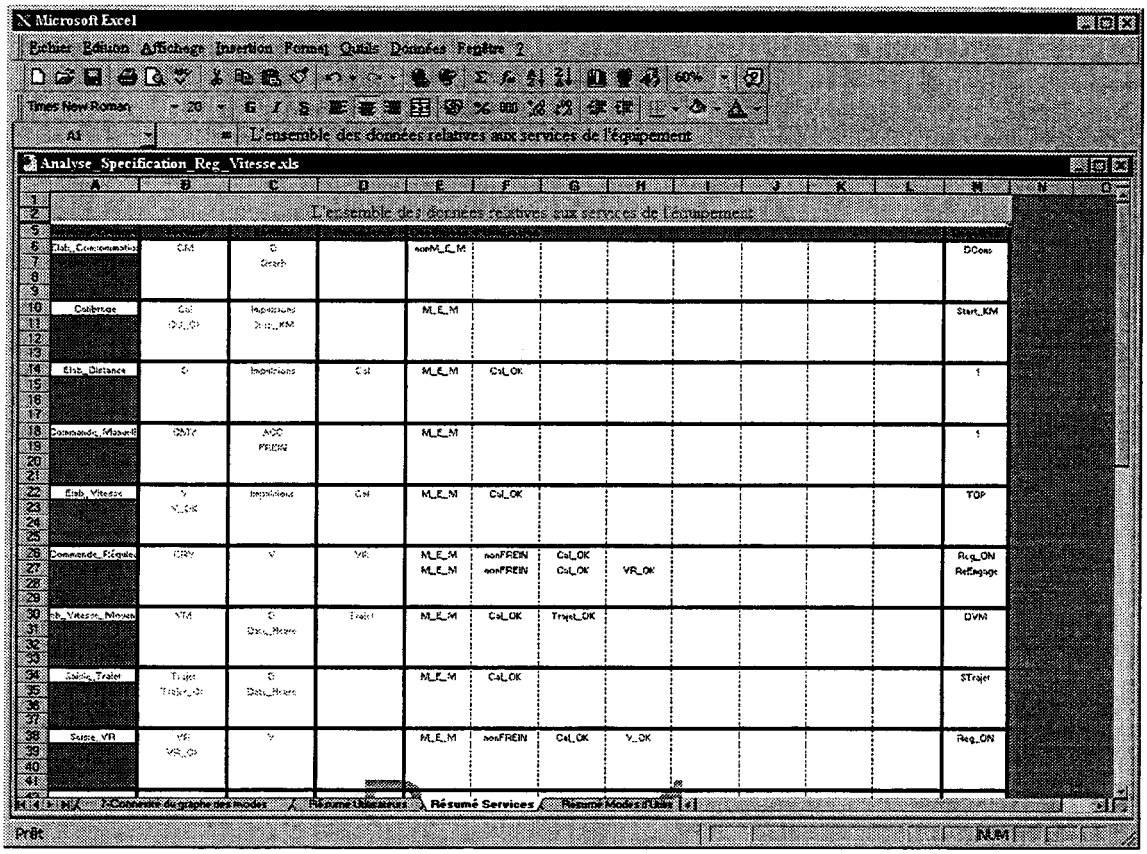

Microsoft Excel
CONDITIONS DE CHANGEMENT DE MODE D'UTILISATION
Analyse_Specification_Reg_Vitesse.xls
Mode d'origine : Configuration
Motor_OFF
Stop_KM
Mode d'origine : Régulation
Motor_OFF
Reg_OFF
Stop_KM
Cal_OK=0
FREIN
1
4-Changements de Mode
Prêt
Microsoft Excel
CONDITIONS DE CHANGEMENT DE MODE D'UTILISATION
Analyse_Specification_Reg_Vitesse.xls
Mode d'origine : Passage en Régulation
Motor_OFF
Reg_OFF
Start_KM
Cal_OK=0
non FREIN
non FREIN
V_OK
Reg_ON
ReEngage
4-Changements de Mode
Prêt

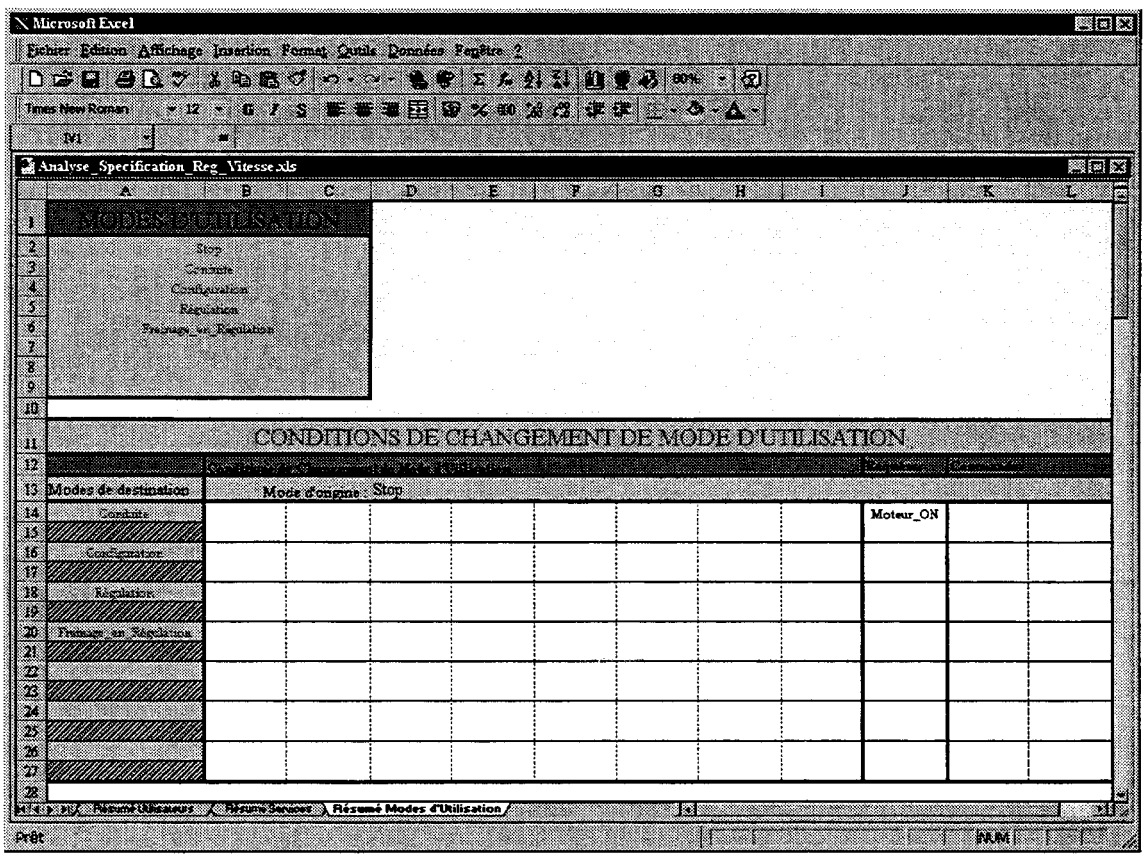

Microsoft Excel
ACTEURS DES CONDITIONS DE CHANGEMENT DE MODE
Analyse_Specification_Reg_Vitesse.xls
ACTEURS DES CONDITIONS DE CHANGEMENT DE MODE
Conducteur
Etat_Vitesse
Calibrage
MODES D'UTILISATION
4-Changement de Mode
5-Acteurs CCM
6-Acteurs Requêtes CM
Prêt
NUM

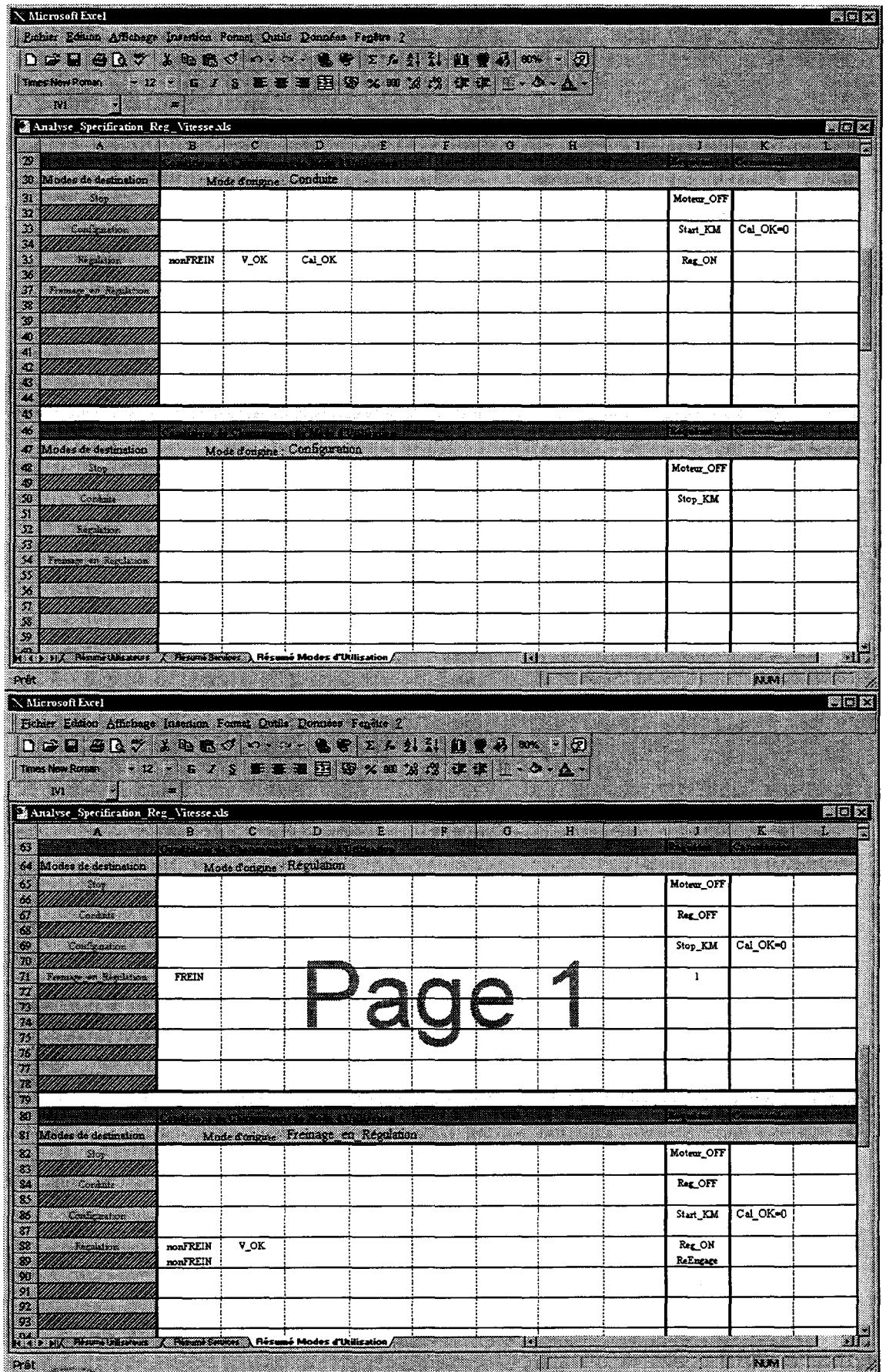

Microsoft Excel
ACTEURS DES REQUETES DE CHANGEMENT DE MODE
Analyse_Specification_Reg_Vitesse.xls
ACTEURS DES REQUETES DE CHANGEMENT DE MODE
Conducteur
Conducteur
Conducteur
Conducteur
Conducteur
Conducteur
Conducteur
Page 1
4-Changement de Mode
5-Acteurs CCM
6-Acteurs Requêtes CM
7-Conseil du graphe des mot
Prêt
NUM


Microsoft Excel
Analyse_Specification_Reg_Vitesse.xls
VERIFICATION DE LA CONNEXITE DU GRAPHE DES MODES D'UTILISATION
Tableau Transition_mode1
Stop
Conduite
Configuration
Régulation
Freinage_en_Régulati
Mode initial : Pas de mode initial
Mode final : Pas de mode final
Tableau Transition_mode8
Connexité du Graphe des modes : Le graphe des modes d'utilisation est fortement connexe.
Page 1


Microsoft Excel
Analyse_Specification_Reg_Vitesse.xls
Les Utilisateurs de l'équipement
Conducteur Moteur
CM Cal
D CMV
V CRV
VM Trajet
VR
Moteur_ON Impulsions
Moteur_OFF Date_Heure
Reg_OFF M_E_M
Qcarb TOP
Stop_KM
ACC
FREIN
QCons
Start_KM
Reg_ON
ReEngage
DVM
STrajet
Page 1


Microsoft Excel
Analyse_Specification_Reg_Vitesse.xls
L'ensemble des données relatives aux services de l'équipement
Résumé Services
Prêt
NUM


Microsoft Excel
Analyse_Specification_Reg_Vitesse.xls
MODES D'UTILISATION
Stop
Contrôle
Configuration
Régulation
Passage en Régulation
CONDITIONS DE CHANGEMENT DE MODE D'UTILISATION
Modes de destination
Mode d'origine : Stop
Moteur_ON
Résumé Modes d'Utilisation
Prêt
NUM



UNIVERSITÉ DES SC. ET TECHN. BIBLIOTHÈQUE UNIVERSITAIRE LILLE