RNTL - Appel à propositions 2002

Description du projet

Acronyme du projet	**MORSE**

Titre du projet	**M**éthodes et **O**utils pour la **R**éalisation et la vérification formelle de **S**ystèmes interopérables **E**mbarqués critiques

1. Introduction

L'objectif du projet MORSE est de fournir une solution pour le développement d'applications **industrielles certifiables** dans le secteur de l'avionique et plus particulièrement pour les drones[1]. Cette classe d'applications est caractérisée par les contraintes suivantes :

- des exigences fortes de **fiabilité**,
- la nécessité de **certifier** l'application,
- la capacité à s'exécuter dans un environnement **réparti** ne disposant que de communications **asynchrones**.

La **fiabilité** est nécessaire pour garantir la sécurité des personnes survolées et des autres usagers de l'espace aérien. La **certification** vise à assurer que les méthodes de développement respectent le niveau de rigueur adapté au *niveau de criticité* du logiciel considéré. Elle impose des contraintes sur le cycle de vie du logiciel et sur la structure de l'application. La **répartition** de l'application et un mode de fonctionnement **asynchrone** caractérisent la gestion des liaisons au sol ainsi que le déroulement de missions combinées impliquant plusieurs drones. Sur ces points, la gestion du temps est moins critique. Par conséquent, MORSE ne visera pas la certification de contraintes temps réel. Cependant, une évaluation pourra être effectuée sur les composants obtenus selon la méthodologie MORSE à l'aide des techniques d'analyse du code généré disponibles sur le marché.

Compte tenu du contexte industriel du projet, MORSE doit fournir une *méthodologie outillée* guidant le développement des applications définies ci-dessus et permettant d'atteindre le niveau de qualité exigé dans l'avionique.

La méthodologie proposée et outillée par MORSE repose sur les cinq points suivants :

- une approche par **prototypage** centrée sur la notion de modèle,
- l'utilisation systématique des **méthodes formelles**,
- la **synthèse automatique de programmes** depuis les spécifications formelles,
- un **langage pivot** assurant la cohérence entre les étapes de la conception,
- la **réutilisation** de spécifications existantes.

L'approche par **prototypage** est courante dans l'industrie. La méthodologie construite dans le cadre de MORSE sera utilisée dans ce contexte, elle doit donc être intégrable dans un processus de développement maîtrisé. Le prototypage s'appuie sur la notion de *modèle* (exprimé dans un formalisme rigoureux) qui constitue un "fil conducteur" de la spécification d'une solution à sa réalisation. Néanmoins, dans le cadre d'applications réparties, cette méthode trouve ses limites. Il est en effet très difficile voire impossible de prendre en compte au niveau du prototype l'ensemble des comportements possibles du logiciel, la sémantique d'un tel modèle n'étant pas complètement opérationnelle.

Les **méthodes formelles** apportent une réponse à ce problème, ce qui justifie leur utilisation dans le cadre de MORSE. Leurs fondements mathématiques permettent d'exprimer les propriétés attendues du logiciel et de les prouver. Dans le cadre d'un processus de **certification**, les preuves de propriétés sont un moyen performant pour aider à la mise en conformité avec la norme. Le modèle formel sera le point de départ de l'analyse formelle et de la construction des prototypes.

L'analyse des modèles utilise principalement la technique du *model checking* **[16]**. Cependant l'explosion combinatoire induite par la répartition et l'asynchronisme est une limite dont il faut s'affranchir. La combinaison de plusieurs techniques est donc envisagée : représentation compacte de l'ensemble des états accessibles en utilisant les DDD (*Diagrammes de Décision de Données*) **[11]**, vérification modulaire et réduction du modèle en fonction des propriétés étudiées.

D'autre part, les *méthodes structurelles* offertes par un formalisme comme les réseaux de Petri permettent de vérifier certaines propriétés sans que le calcul de l'espace des états soit nécessaire. Ces deux approches (model checking et analyse structurelle) sont complémentaires.

[1] Avion de surveillance sans pilote.

La **synthèse automatique de programmes** consiste en la génération automatique du squelette distribué de l'application. Ce squelette est constitué des interfaces des différents composants et des programmes contrôlant les interactions entre ces composants. Le code produit permettra la construction rapide de *prototypes* pour l'évaluation immédiate de la solution proposée. À la fin du cycle de développement du logiciel, le code généré sera déployable en exploitation sur la plate-forme visée. MORSE se focalise donc sur la génération du code gérant la **répartition** et l'**asynchronisme**, propriétés reconnues comme difficiles à implémenter. D'autre part, un effort particulier sera porté sur la traçabilité entre les différentes phases du développement, conformément aux règles de certification.

UML est un standard pour la spécification de systèmes industriels. Cependant, il ne couvre pas tous les besoins requis pour la spécification de **systèmes embarqués répartis**. Notamment, les aspects comportementaux ne peuvent être capturés de manière non ambiguë au niveau du système complet. Pour cela, il faut disposer d'extensions permettant de capturer ces aspects et réutilisant l'architecture du système (classes et composants) qui est, elle, bien définie à l'aide d'UML.

Afin d'assurer le lien entre les phases de la méthodologie proposée, MORSE introduit un **langage pivot**. Ce langage permettra d'exploiter les informations d'une spécification UML. Il aura également une sémantique formelle pour supporter l'analyse de modèles et les capacités d'un *langage de description d'architecture* pour relier les composants de l'application. Le langage pivot offrira une structure modulaire favorisant la **réutilisation** de composants existants et la production de composants réutilisables. Les travaux autour de LfP *(Language for prototyping)* **[1]**, développé au LIP6 fournissent une première base pour ce langage pivot.

Le projet MORSE doit aboutir au développement d'une méthodologie de développement de logiciels certifiés répartis. Une application témoin dans le domaine des drones sera réalisée pour montrer la validité de l'approche proposée.

Le projet MORSE sera concrétisé par un **Atelier de Génie Logiciel** pilote (l'AGL MORSE) dont les retombées dans l'industrie seront appréciées. En effet, MORSE innove en proposant de nouvelles techniques dans le domaine des méthodes formelles (DDD, analyse structurelle avec les réseaux de Petri) afin de traiter un domaine très difficile où l'asynchronisme, la répartition (entre autres) sont prédominants. Ainsi, d'une certaine manière, **MORSE s'inscrit dans la ligne d'une nouvelle méthodologie incluant les méthodes formelles qui complète et prolonge les techniques déjà existantes**.

Le projet se situe complètement dans l'axe 1 définit par le RNTL (Concevoir des logiciels enfouis, critiques ou temps réel pour les objets et systèmes). Les mots-clefs qui caractérisent plus précisément MORSE sont indiqués dans le tableau ci-dessous.

Mot-clef	***Libellé***	***Justification***
1.1	*Systèmes / applications enfouis (embarqués)*	Le domaine d'application est celui des systèmes embarqués répartis critiques. MORSE vise la définition d'une méthodologie caractérisée par la construction d'une suite de prototypes exécutables obtenus par synthèse automatique de programmes et vérifiables formellement.
1.9	*Modélisation, vérification de comportement, validation*	Le projet MORSE propose de nouvelles techniques de spécification et l'utilisation des méthodes formelles dans le domaine d'application visé afin de maîtriser des systèmes de taille industrielle.
1.10	Transformations de spécifications/programmes	Les techniques de transformation de spécifications de haut niveau en programmes ou en représentations formelles (réseaux de Petri, DDD) sont très présentes dans le cadre de MORSE.

2. Objectifs de recherche technologique du projet

La recherche en France concernant les méthodes formelles est très active. De nombreux projets de recherche existent pour lesquels différentes approches sont explorées. Nous situons ci-dessous MORSE par rapport à d'autres projets proches,. Après une analyse minutieuse, nous avons principalement retenu les projets VASY **[17]** et le langage BDL **[14]** tous deux développés à l'INRIA.

2.1 Etat de l'art et intérêt du projet

On peut distinguer deux approches distinctes dans les techniques de vérification formelle : la vérification de modèle (*model checking*) d'une part, et les systèmes de preuves de théorème d'autre part.

La vérification de modèle propose d'étudier tous les états possibles d'un système et de vérifier des propriétés sur cet espace d'états. Le projet VASY constitue un bon exemple de cette méthode : il propose une approche asynchrone basée sur le langage de spécification LOTOS. Ce projet dispose de nombreux appuis industriels et d'outils très performants pour la vérification de modèles de grande taille. Cependant il n'a pas pour objectif la synthèse de programmes répartis à partir de spécifications. Par ailleurs, la structure de LOTOS reste éloignée du paradigme objet sur lequel repose UML, ce qui pose un problème dans le cadre de MORSE qui s'appuie sur ce standard (et donc le paradigme objet). Enfin, LOTOS n'est pas un langage de description d'architecture (ADL), ce qui est également pénalisant dans le cadre du projet car il serait nécessaire d'introduire un autre formalisme pour décrire l'architecture de l'application.

A contrario, la preuve de théorème propose de démontrer les propriétés par une suite d'étapes logiques constituant une preuve. Dans le cadre de ces approches, Coq développé par l'INRIA a fait ses preuves depuis longtemps. Ce type de technique permet de traiter des systèmes infinis, ou de trop grande taille pour que la vérification de modèle puisse être menée dans un temps raisonnable. Cependant, malgré ces avantages, cette technique est peu représentée dans l'industrie en raison de la difficulté de mise en application et de la formation très pointue qu'elle nécessite, même si différents travaux visent à en simplifier l'usage.

Les travaux considérant les méthodes formelles dans le cycle de vie du logiciel visent à donner une sémantique formelle aux spécifications UML. Dans le cadre de cette approche, citons le formalisme BDL (également étudié dans le cadre du projet industriel Reutel 2000) qui s'appuie sur une syntaxe proche des statecharts et permet de formaliser les aspects dynamiques de la spécification UML. Il est possible de modéliser des applications réparties en prenant en compte une sémantique synchrone ou asynchrone en fonction des informations extraites du diagramme de déploiement UML. Néanmoins, ces diagrammes restent imprécis dans la spécification du comportement des médias de communication ente composants car BDL ne constitue pas un langage de description d'architecture.

L'approche de BDL est différente de celle choisie dans le projet MORSE. BDL propose de formaliser UML, alors que le langage pivot vise à devenir un complément d'UML. En effet, comme le langage pivot doit proposer une modélisation plus fine des mécanismes de communication entre les composants de l'application, il doit également posséder les caractéristiques d'un langage de description d'architecture, ce qui est le cas de LfP.

L'approche MORSE

Les techniques de vérification proposées dans MORSE sont axées sur la vérification de modèles et la vérification structurelle qui présentent l'intérêt d'être complètement automatiques, ce qui simplifie leur mise en œuvre. Les techniques récentes introduites par le LaBRI avec les DDD (Data décision Diagram) laissent présager une bonne compression des états du système. Enfin, l'usage du calcul de propriétés structurelles, propres aux réseaux de Petri, permet également d'automatiser la recherche de propriétés sans calculer l'espace des états du système.

MORSE se propose ainsi d'encapsuler différentes techniques de vérification à travers un langage pivot qui fera le lien entre UML et les méthodes formelles. Cette solution permet de proposer une interface simple vers ces méthodes ce qui favorisera leur intégration dans un contexte industriel.

En résumé, l'intérêt principal du projet MORSE est d'abord de mettre à la disposition du monde industriel des techniques récentes concernant les méthodes formelles en les intégrant dans une méthodologie allant de la spécification au codage. C'est aussi permettre à ce dernier de traiter formellement des problèmes difficiles liés à la certification du logiciel auxquels il est maintenant de plus en plus confronté et pour lesquels il n'existe pas aujourd'hui de solutions applicables à un coût raisonnable. L'objectif majeur est d'amorcer un nouveau type de méthodologie focalisée sur la notion de modèle (approche de prototypage par raffinements) et utilisant plusieurs approches formelles complémentaires.

2.2 Innovation technologique et recherche développées pour le projet

L'objectif du projet MORSE est d'accroître la facilité de mise en œuvre des méthodes formelles **dans le cadre d'une méthodologie industrielle**. Pour cela, il innove dans les axes suivants.

Méthodologie orientée modèle.

La méthodologie qui sera proposée par le projet MORSE utilisera de manière conjointe des méthodes industrielles éprouvées et des **méthodes formelles**. De nombreux autres projets intégrent cette particularité méthodologique mais en général leur mise en œuvre sur des applications de taille industrielle s'avère laborieuse à cause de l'aspect formel et de son couplage avec les autres méthodes.

Le langage pivot proposé dans MORSE est intégré avec la méthodologie, il permettra d'exprimer les **spécifications formelles** issues d'un premier modèle en UML puis les propriétés. Il s'agit donc d'une encapsulation de méthodes formelles. Rappelons en effet que UML est semi-formel et par conséquent ne permet pas de vérifier les aspects dynamiques dans le cadre des systèmes répartis asynchrones qui nous intéressent ici. En revanche, grâce au langage pivot la **vérification des propriétés** pourra être effectuée ainsi que la **synthèse automatique de programmes**.

Le cycle en « Y » industriel sera conservé globalement, il sera néanmoins revu pour tenir compte, en plus des méthodes formelles, de **la démarche par prototypage** et de l'**impératif de certification** (entre autres, une traçabilité à tous niveaux sera introduite, y compris vers UML en tenant compte du contexte formel du langage pivot). De plus, il s'agit obtenir une méthodologie outillée (AGL MORSE) en appliquant les réflexions les plus récentes[2] sur ce sujet et dédiée aux systèmes embarqués de la même famille que celle des drones.

Utilisation de techniques formelles

L'approche des méthodes formelles dans le projet MORSE est novatrice sur plusieurs aspects : 1) l'utilisation conjointe de techniques de **vérification de modèle** (*model checking*) et de méthodes structurelles, 2) l'utilisation de techniques nouvelles pour la construction de l'espace des états accessibles lors de la vérification de modèle, 3) l'utilisation pour la modélisation d'un langage de haut niveau (le langage pivot) conçu pour être facile d'accès.

L'aspect formel de MORSE repose sur l'élaboration d'un modèle dans le langage pivot issu du modèle UML. La sémantique opérationnelle formalisée du langage pivot le rend utilisable pour la vérification et la synthèse de programme. D'autre part, ce langage de haut niveau est conçu pour être aussi **simple d'utilisation** que possible, afin de faciliter son déploiement dans un contexte industriel. Il dispose de tous les mécanismes permettant de modéliser aisément les contraintes liées à la **répartition** et à l'**asynchronisme**. De plus, le langage pivot assurera le lien entre le paradigme objet adapté à l'analyse et celui des processus communicants adapté au déploiement.

L'utilisation de techniques de vérification formelles est reconnue pour faciliter la **certification** car elles présentent le niveau de rigueur nécessaire pour le développement de logiciels critiques **[2]**. Malgré cela, l'utilisation de ces techniques dans un cadre industriel est fortement freinée par la difficulté de leur mise en œuvre et les problèmes de passage à l'échelle. Ces deux facteurs interdisent leur utilisation autrement que sur des sous-ensembles de taille réduite. L'approche MORSE apporte des solutions à ces problèmes. En premier lieu, les **techniques proposées sont automatiques** (par opposition aux systèmes d'aide à la preuve). D'autre part, MORSE utilisera deux types de méthodes

[2] cf. l'approche MDA «Model Driven Architecture» définie par l'OMG (Object Management Group), **[3]**

complémentaires : celles utilisant une approche structurelle, et celle basée sur la vérification de modèle (*model checking*).

Les **méthodes structurelles** permettent de calculer des propriétés générales sur les systèmes répartis (invariants, bornes…) qui doivent être interprétées pour correspondre à des propriétés de l'application. Elles présentent l'avantage de ne pas nécessiter la construction de l'ensemble des états accessibles ce qui facilite le passage à l'échelle. Ces méthodes ont été originellement développées dans le cadre des réseaux de Petri.

Parallèlement, la **vérification de modèle** permet entre autres de vérifier des propriétés de causalité entre événéments, spécifiques à l'application. Elle nécessite cependant la construction de l'ensemble des états accessibles et pose donc un problème de passage à l'échelle. Dans ce cadre, MORSE reposera sur de nouvelles techniques de représentation compacte de l'espace des états accessibles: les **DDD** (*Diagrammes de Décision de Données*). Ils **faciliteront le passage à l'échelle** des méthodes de vérification de modèles. La vérification de modèle sera appliquée directement sur le langage pivot, sans passage par un modèle intermédiaire. L'approche des DDD a été ébauchée dans le cadre du projet Clovis **[9]** réalisé en collaboration avec la DGA (Direction Générale pour l'Armement).

Asynchrone réel dans un contexte réparti

De nombreux ateliers (ESTEREL STUDIO, SILDEX, SCADE…) proposent la **synthèse automatique de programmes**, rares pourtant sont ceux qui le font dans un contexte à la fois asynchrone et distribué (SafeAir). Précisément, MORSE permettra une synthèse du programme de *contrôle* de l'application c'est-à-dire de son architecture en processus communicants en tenant compte des contraintes de **certification**. Ceci est réalisable car le **langage pivot est aussi un ADL** (*Architecture Description Language*) et sa sémantique est exécutable. Le programme généré implémentant le contrôle de l'application sera certifiable ce qui exige, entre autres, au moins une traçabilité avec le modèle dans le langage pivot et des contraintes sur la structure du code produit.

Réutilisation.

La **réutilisation** est un élément important dans les projets industriels. Elle se justifie par la nécessité de réduire le temps de développement et par les exigences de fiabilité. **La gestion de la réutilisation est prévue dès l'origine dans la définition du langage pivot**. Ce dernier possède une structure modulaire qui permet de réutiliser des parties de spécification dans de nouvelles applications. L'intégration des composants dans un nouvel environnement est vérifiée par la notion de **contrat comportemental** qui permet de spécifier le contexte d'utilisation du composant. Il est ainsi possible de vérifier que le composant déjà spécifié formellement est intégrable dans le nouvel environnement.

Considérant qu'il est actuellement impossible de construire automatiquement des spécifications formelles à partir d'un code existant (*reverse engineering*), MORSE ne propose pas de solution directe à ce problème. Cependant, l'approche développée dans ce projet permet d'intégrer l'existant à travers la description d'une abstraction de son comportement. Cette dernière ne valide aucunement le composant, mais permet de vérifier ses interactions avec l'application.

Application à un vrai projet industriel (drones)

L'AGL MORSE et sa méthodologie seront utilisés à la SAGEM dans le cadre du projet MORSE pour une application industrielle impliquant les drones. Le logiciel produit doit être certifiable conforme à la norme DO-178B **[10]** (reconnue internationalement comme la référence pour la certification de logiciels dans le domaine de l'avionique) Cette application est représentative d'un large domaine (les systèmes interopérables faiblement couplés). **Le projet MORSE fournira donc non seulement des résultats théoriques et des méthodes outillées mais aussi un premier retour d'expérience industrielle exploitable pour un premier raffinement de l'AGL proposé**.

3. Objectifs industriels du projet

L'atelier MORSE vise des projets dont les contraintes fortes sont entre autres, les suivantes : **certification**, **répartition** et **asynchronisme**. Il s'agit là d'une classe de domaines métier, bien représentée par le domaine des avions sans pilote ou drones, pour laquelle la communication entre les composants des systèmes est asynchrone et peu fiable. La communication entre les composants génère des difficultés pour lesquelles les hypothèses couramment admises comme l'endochronie[3] ou l'isochronie[4] ne sont plus applicables. Il est donc difficile voire impossible d'intégrer l'aspect asynchrone qui caractérise ces systèmes en partant d'un modèle synchrone ou partiellement synchrone (approche traditionnellement utilisée dans l'avionique).

La méthodologie par prototypage proposée par MORSE respectera le schéma suivant :

- Un modèle dans le langage pivot est construit par **élicitation** d'une spécification UML. Au cours de cette étape, le comportement des composants de l'application est explicité de manière non ambiguë à travers un contrat comportemental, Ce diagramme comportera également les propriétés qui doivent être vérifiées.
- Pour chaque propriété à vérifier identifiée dans le modèle pivot, des spécifications formelles seront produites afin de **vérifier formellement** la satisfiabilité de cette propriété. La technique de vérification (DDD ou Réseaux de Petri) sera choisie en fonction des critères de complexité du calcul de la propriété. En cas d'échec, une raison sera exposée à l'utilisateur dans les termes du langage pivot.
- Une fois les propriétés du modèle démontrées, on obtient une implémentation conforme du système par **synthèse automatique de programmes**. Cette opération utilisera des informations issues de la vérification (par exemple, le calcul de borne pour des tables ou des buffer) ainsi que les informations concernant le déploiement des composants modélisés sur l'architecture cible utilisée.

La spécification dans le langage pivot peut à nouveau être modifiée en fonction des observations effectuées lors des exécutions, et un nouveau cycle de production envisagé à partir du modèle ainsi raffiné (vérification + synthèse de programmes).

3.1. Marché et intérêt du projet

Nos investigations nous ont montré que les principaux produits industriels concurrents de MORSE sont l'atelier **SafeAir** (issu du projet Esprit SACRES), **l'atelier B** de Stéria et le projet REUTEL 2000 (INRIA, Alcatel). Nous les considèrerons ci-dessous successivement.

L'Atelier SafeAir

SafeAir **[19]**.vise le développement d'applications embarquées certifiées intégrant des contraintes de répartition dans le domaine de l'avionique, domaine proche de celui visé par MORSE.

Ce projet intègre plusieurs ateliers construits respectivement à partir des langages synchrones Signal (atelier SILDEX) et Lustre (SCADE) ainsi que les StateCharts (STATEMATE) qui amènent la composante asynchrone. Les spécifications écrites à l'aide de ces ateliers sont compilées dans un format d'échange unique *Scade_lang* (qui sera ensuite traduit en un programme Lustre). SafeAir propose ensuite de nombreux outils (sous la forme de "paquetages") utilisant *Scade_Lang* pour la vérification de modèle, de propriétés temporelles, la génération de code et la génération de tests.

L'une des caractéristiques de SafeAir est d'utiliser plusieurs formalismes complémentaires. Ces formalismes sont adaptés au domaine visé mais ce dernier ne correspond pas aux objectifs du projet MORSE sur les points suivants :

[3] Cette classe de programmes est définie ainsi : aucune décision de contrôle ne nécessite de tester l'absence de signaux dans l'environnement.

[4] D'après B. Caillaud, du projet Pampa, deux programmes forment une paire isochrone si le remplacement du protocole de communication synchrone par une communication asynchrone par file *non bornée* ne change pas les comportements de chaque composant (le timing global du système résultant n'est pas préservé).

- **L'asynchronisme** : Les communications considérées par SafeAir sont synchrones (héritage de Lustre, Signal). L'asynchronisme est construit à partir de la propriété d'*isochronie* qui permet de relier des composants synchrones par des liaisons asynchrones en s'appuyant sur des communications *fiables*. Cette hypothèse n'est pas applicable dans le cadre des drones car la majorité des canaux (hertziens) ne sont pas fiables et peuvent être interrompus et rétablis de manière asynchrone. .
- **La méthodologie** : UML est utilisé par la méthodologie MORSE durant la phase d'élicitation SafeAir utilise les formalismes des ateliers qu'il fédère. D'autre part, la démarche méthodologique dans MORSE, contrairement à celle de SafeAir, est basée sur le prototypage c'est-à-dire sur une suite de prototypes convergeant vers le produit final. Chaque prototype peut donc être validé par le client lorsqu'il le souhaite.

L'atelier B.

La méthode B **[13]** vise le développement d'applications critiques, elle est basée sur la génération de code depuis une spécification formelle du programme. La méthodologie utilisée repose sur un cycle regroupant plusieurs itérations, chaque itération raffinant la précédente. La dernière itération produit le code du programme.

Contrairement aux objectifs du projet MORSE, le code généré en utilisant B est strictement séquentiel (e.g. non parallélisable). Cette méthode ne peut donc pas adresser des applications réparties en mode de fonctionnement asynchrone ce qui est le cas des drones.

L'atelier B utilise un mécanisme de preuve assistée à partir de la spécification du système, il nécessite donc une formation aux méthodes de preuves de théorèmes, ce qui n'est pas courant dans l'industrie. B dispose également de bons environnements qui ont permis des développement de grande taille (projet Eole réalisé par Matra-Transport). Cependant, B ne gère pas explicitement le parallélisme qui doit donc être décrit de manière explicite, ce qui complique encore le modèle.

Le projet Reutel 2000

le projet Reutel 2000 **[18]** utilise le formalisme BDL (décrit au paragraphe 2.1) dans un cadre industriel. Ce projet, réunissant Alcatel et l'INRIA, a pour objectif la réalisation d'une plate-forme de développement pour le domaine des télécommunications. Le rôle de BDL dans ce contexte est de servir d'interface entre la chaîne de développement d'Alcatel et les outils d'analyse et de vérification de l'application. Les applications de ce formalisme sont surtout dédiées à des applications de télécomunication, ce qui est très différent du domaine visé par MORSE.

3.2. Type d'exploitation envisagée, diffusion des résultats et retombées économiques

L'utilisation des drones au sein des forces armées s'effectue majoritairement dans des volumes aériens réservés. Cependant, l'extension des activités des drones au secteur civil (surveillance de frontières, de forêts, des lignes électriques, gestion des catastrophes naturelles…), cumulée à la volonté européenne d'aligner les standards militaires sur les standards civils (pour homogénéiser le niveau de fiabilité des aéronefs du trafic aérien européen), implique une sûreté de fonctionnement sans faille pour les logiciels embarqués. La recherche d'un outil de développement et de vérification des logiciels critiques faisant appel aux méthodes formelles (reconnues par les autorités certifiantes) doit permettre d'augmenter très significativement la fiabilité de ces logiciels, et de "compenser" ainsi l'absence d'une présence humaine à bord. Ceci est un point de passage indispensable pour l'acceptation des drones dans le trafic aérien civil.

Le marché visé est clairement celui des drones mais la domotique (où le caractère critique est moindre mais l'aspect économique très important) à base de réseau hertzien constitue sans doute un autre bon débouché. Ces domaines ont pour caractéristique d'avoir des mécanismes de communication asynchrones dont les propriétés ne sont pas facilement capturables avec les outils actuels basés sur la propriété d'isochronie. Par exemple, pour les drones, les canaux de communication ne sont pas fiables (canaux hertziens). Il faut prendre en compte le problème des communications air/sol qui ne sont pas nécessaires dans les applications visées par des produits du marché : on ne peut faire d'hypothèse sur les communications hertziennes dont la qualité de service évolue dans le temps.

Finalement, MORSE n'est pas une fédération de produits déjà existants, bien au contraire il met à la portée des industriels de nouvelles techniques issues de recherches avancées dans le domaine des méthodes formelles tout en proposant un cadre méthodologique ré-actualisé. Il s'agit donc d'un premier pas vers une nouvelle génération de produits industriels qui prolonge celle qui existe déjà en tenant compte de contraintes industrielles difficiles, donc incomplètement traitées aujourd'hui.

Enfin, l'atelier présente un positionnement original sur le marché des AGL pour les caractéristiques de la méthodologie associée. Il intéressera tout besoin de conception et déploiement de logiciel embarqué réparti avec une approche méthodologique.

4. Organisation du projet

Le projet est organisé autour de deux industriels et deux laboratoires. Comme on va le montrer ci-dessous, ce consortium répond aux exigences qu'il s'est fixées : intégration de techniques très récentes en milieu industriel, élaboration d'un atelier et application sur un vrai projet industriel.

4.1. Partenariat

Les sociétés SAGEM SA et AONIX, le LIP6 et le LaBRI ont décidé d'associer leurs compétences. La complémentarité des partenaires motive cette association. L'union des compétences de chaque partenaire garantit à la fois la qualité technologique du projet, l'exhaustivité de l'analyse et un haut niveau de rigueur de la démarche.

L'équipe du projet MORSE est organisée de façon équilibrée (un industriel, un éditeur de logiciels et deux laboratoires) afin de répondre à la fois aux perspectives de retombées industrielles et aux exigences d'avances scientifiques. L'expérience accumulée par cette équipe lui permet de couvrir l'ensemble des domaines en relation directe avec le projet.

SAGEM SA

SAGEM est un groupe français de haute technologie présent dans les domaines des télécommunications, de la défense et de l'électronique professionnelle. Entièrement tourné vers la maîtrise technique et industrielle des technologies de pointe, le groupe SAGEM réalise un chiffre d'affaires de 4,3 GEuros (dont 60% à l'exportation) pour un effectif de 15 000 employés.

Dans la défense, SAGEM est le leader européen en optronique, en navigation inertielle, en rétrofit d'avions d'armes, en préparation de missions et en systèmes de drones, et le leader mondial en systèmes d'identification automatique d'empreintes digitales.

SAGEM est le pionnier mondial de la certification aéronautique des drones, avec la certification (unique au monde) obtenue par le drone SPERWER (vendu dans 4 pays européens).

SAGEM a été l'un des premiers industriels (dès 1994) à appliquer systématiquement une méthodologie orientée objet par prototypage. La démarche a été utilisée pour de gros projets : systèmes d'authentification automatique à partir d'empreintes digitales (AFIS), systèmes de préparation de mission, cartographie et également pour des logiciels embarqués dans les drones.

AONIX

AONIX (75 personnes en France) est partenaire de Grands Comptes de l'industrie et offre à ses clients des solutions technologiques pour le développement et la gestion d'applications stratégiques autour de ses outils et d'une gamme de services, depuis le support simple jusqu'aux solutions complexes (temps réel critique Ada, Java, C, C++), test et certification, modélisation orientée objet basée sur UML (outil StP de modélisation UML et outil ACD de génération de code), cycle de développement en Y.

AONIX a commencé à adresser le marché de la sûreté de fonctionnement dès 1989 avec la fourniture d'un exécutif séquentiel certifiable jusqu'au niveau A du standard DO178B. AONIX a fourni des solutions (produits) et de l'expertise dans le cadre d'un grand nombre de projets " safety critical " dans les domaines de l'Avionique (Sextant Avionique, Airbus), du Ferroviaire (CSEE Transport, Alstom), de l'Espace (Aérospatiale, Matra Marconi Space), et de l'Energie (CEA, Sema Group).

AONIX participe également à des projets de systèmes embarqués utilisant UML, à la fois par ses outils (outil de modélisation StP, méta générateur de code ACD) et par son centre de compétence Objet dont une des particularités est la mise en place d'un processus "itératif" et "incrémental" qui découple les aspects" métier" des "aspects techniques" sur les projets (mis en place chez Sagem , Alstom, Sextant, Compaq, Alcatel...).

LIP6 (Thème Systèmes Répartis et Coopératifs, Université Paris 6)

Le thème SRC travaille depuis plus de dix ans dans la conception, la vérification et la réalisation de systèmes répartis. Le LIP6 apporte dans le projet MORSE le savoir-faire suivant:

- une expertise dans la vérification de propriétés dans les systèmes embarqués acquise à travers des coopérations **[4].**
- une expertise dans l'interopérabilité de services: LIP6/SRC a participé à la rédaction de plusieurs livres collectifs sur l'application des réseaux de Petri dans cette problématique **[5, 6]**.
- une expertise dans la définition de méthodologies basées sur des modèles de haut niveau : des retours d'expérience nous ont permis d'identifier les problèmes liés à la traçabilité entre spécifications et de réfléchir à de nouvelles stratégies de vérification formelle par synthèse de réseaux de Petri
- une expertise dans la génération de code: notre équipe a expérimenté pendant sept années la génération de programmes répartis à partir de réseaux de Petri.
- une expertise dans la définition de plates-formes d'intégration systèmes: notre équipe diffuse via Internet depuis 1994 CPN-AMI, un AGL réseaux de Petri construit à partir de FrameKit **[7]** et intégrant de nombreux outils développés dans différents laboratoire universitaires.

Les activités scientifiques du thème Systèmes Réparti et Coopératif, d'où sont issus les participants LIP6, ont donné lieu ces quatre dernières années à la soutenance de 14 Thèses d'Université, 5 Habilitations à diriger des Recherches, 7 publications en revues internationales, 56 communications internationales.

LaBRI (équipe MVTSI, Université de Bordeaux)

L'équipe «Modélisation, Vérification et Test de Systèmes Informatisés» a centré son activité de recherche sur l'ingénierie des systèmes complexes et critiques. Elle apporte un savoir faire reconnu au projet MORSE dans les domaines suivants :

- *Modélisation et vérification formelle* : Historiquement, l'équipe s'est bâtie autour d'André Arnold sur les systèmes de transitions. Le produit synchronisé de systèmes de transitions a été introduit par A. Arnold et M. Nivat en 1982. La théorie des systèmes de transitions est donc au centre de nos préoccupations, ainsi que les différentes logiques temporelles.
- *Calcul booléen et symbolique* : l'équipe a développé une compétence sur le calcul propositionnel (l'algèbre de Boole) qui s'avère être un excellent langage de modélisation. Nous menons de nombreux travaux sur les diagrammes de décisions binaires et leurs extensions aux diagrammes de décisions : les diagrammes de Décisions de Données. Contrairement aux BDD, ces derniers permettent de manipuler des variables non bornées, ainsi que des relations infinies. L'originalité de cette démarche est de fournir à l'utilisateur la possibilité de définir ses propres opérateurs ou analyses, permettant ainsi de traiter des systèmes complexes tels que les programmes VHDL.
- *Atelier d'aide à la conception et à l'évaluation de systèmes critiques* : Pour faciliter l'emploi des méthodes formelles pour le développement logiciel, nous avons défini le formalisme AltaRica **[9]**, plus proche de la façon dont les industriels voient les systèmes, et qui permet cependant, par traduction dans d'autres formalismes, de disposer de la puissance des outils existants.

L'activité scientifique de l'équipe MVTSI à donné lieu, ces quatre dernières années à 18 revues, 45 colloques avec comité de rédaction, 9 conférences invitées, 5 participations à des projets nationaux (dont 3 projets RNTL), 13 contrats avec des établissements publics et privés et 2 contrats internationaux.

Collaborations antérieures

Ces partenaires ont déjà engagé des collaborations :

- LIP6 et LaBRI ont participé ensemble au projet Clovis (DGA) **[9]**. Ces deux laboratoires travaillent ensemble depuis plusieurs années des thèmes de recherche en réseaux de Petri.
- LIP6 et SAGEM collaborent sur la problématique de la synthèse automatique de programmes dans le cadre d'une thèse CIFRE
- SAGEM utilise depuis longtemps les outils Aonix, notamment dans le cadre du projet M51 (Défense Nationale).
- depuis la soumission du projet au programme RNTL 2001, les partenaires ont continué de réfléchir sur la base des remarques des experts. Le groupe a en particulier consolidé la spécification du langage pivot en intégrant des contraintes industrielles.

4.2 Organisation du partenariat dans le projet

SAGEM, chef de file du projet MORSE, assurera la gestion du projet et la coordination des activités des quatre partenaires :

- SAGEM SA fournira l'application témoin pour la validation du projet : le développement de logiciels certifiables pour drones. Il sera également le correspondant du réseau et veillera au bon déroulement des étapes contractuelles (i.e bilans d'avancement, états d'avancement des travaux).
- AONIX sera responsable de la méthodologie MORSE pour le développement de logiciels embarqués.
- Le LIP6 est responsable de la définition du langage pivot et sur les techniques de synthèse de programmes pour le prototypage.
- Le LaBRI est responsable de la mise en œuvre des méthodes formelles.

Les partenaires ont des compétences complémentaires, ce qui motive leur participation à l'ensemble des sous-projets. Les développements seront assurés par tous les partenaires en fonction de leur domaine de compétence ; l'ensemble sera intégré sous la responsabilité du LIP6 dans la plate-forme d'intégration logicielle FrameKit **[7]** afin de constituer l'AGL pilote.

5. Description technique du projet

Le projet est découpé en 5 sous-projets découlant des objectifs précités :

- Sous-projet 1 (responsable Aonix) : Méthodologie de développement de systèmes répartis embarqués asynchrone
- Sous-projet 2 (responsable LIP6) : Langage pivot et expression de propriétés sur le modèle
- Sous-projet 3 (responsable LaBRI) : Vérification formelle de propriétés
- Sous-projet 4 (responsable LIP6) : Synthèse automatique de programmes
- Sous-projet 5 (responsable SAGEM) : Intégration dans l'AGL MORSE et application témoin

5.1 Sous-projet 1 : Méthodologie de développement de systèmes répartis embarqués asynchrone

Responsable : Aonix

Partenaires

SAGEM	33% : compétences en certification
AONIX	53% : compétences en méthodologie objet
LIP6	10% : compétences en prototypage & méthodes formelles
LaBRI	5% : compétences en méthodes formelles

Objectifs

Trois buts essentiels :

- Tâche 1.1 : Définir la méthodologie MORSE.
- Tâche 1.2 : Définir les relations entre UML et le langage pivot.
- Tâche 1.3 : Expliciter et formaliser le matériel de qualification nécessaire.

Détail des réalisations et échéances

Échéance du sous-projet 1 : T0-T0+35 durée : 35 mois

Tâche 1.1 : Définir la méthodologie

La définition de la méthodologie sera faite principalement dans cette tâche à travers la production d'un guide méthodologique. On ne peut cependant prétendre tout traiter ici sans considérer les autres travaux. Ainsi nous procéderons par itérations tout au long du projet MORSE. C'est pour cette raison que l'on trouvera dans les autres sous-projets des tâches intitulées "retour sur la méthodologie".

L'objet de la tâche 1.1 est d'adapter une méthodologie éprouvée commune à SAGEM et AONIX : le cycle de développement en «Y». Dans cette approche, les aspects métier et les contraintes techniques sont traités dans deux «branches» distinctes. Les produits de ces deux banches sont fusionnés pour l'implémentation.

Il faudra intégrer les méthodes formelles (i.e. langage pivot, recherche de propriétés, etc.) et le contexte particulier du projet (certification) dans la méthodologie qui comportera les phases suivantes :

- L'**Elicitation** a pour objectif l'élaboration du modèle pivot à partir d'une spécification UML.
- La **Vérification** a pour objectif la vérification du modèle pivot sur une «spécification quotient» obtenue par optimisation du modèle en fonction des propriétés à vérifier (e.g. élimination des éléments de la spécification qui sont neutres vis-à-vis de la propriété). L'objectif avoué est de réduire ainsi la complexité de la preuve.
- La **Définition de l'architecture** concerne la génération des composants et la gestion des éléments de l'architecture du système utilisation de «containers logiques» pour les composants-métiers.
- L'**Implémentation** est réalisée par **Synthèse automatique de programmes** qui synthétise du code à partir des entités du modèle pivot et permet de contrôler leur déploiement sur une architecture cible d'exécution (matériel + middleware).

Un risque identifié du cycle de développement en «Y» est le manque de communication entre les deux «branches». Dans le cadre du projet MORSE, l'usage d'un même formalisme pour les aspects techniques et métier, favorisera la possibilité d'effectuer des vérifications croisées et contribuera à réduire ce risque de manière significative.

Tâche 1.2 : Définir les relations entre UML et le langage pivot

L'objectif est d'exprimer les règles de transformations entre une expression UML concernant l'aspect métier de l'application et une expression dans le langage pivot intégrant de manière formelle les contraintes techniques. Ces règles concernent les échanges d'information identifiés dans la tâche 1.1 (phase d'élicitation).

Dans la pratique, on réalisera un méta-modèle du modèle pivot. Ce méta-modèle se situe de fait au même niveau que le méta-modèle utilisé par UML. La consolidation des deux permettra de spécifier les règles de transformation de modèles. Cette approche est largement inspirée du Méta Object Facilities spécifiée par l'OMG **[12]**.

Tâche 1.3 : Expliciter et formaliser le matériel de qualification nécessaire

Le but de cette tâche est d'identifier les exigences des organismes de certification vis-à-vis des composantes logicielles et méthodologiques de l'AGL MORSE, puis de définir les actions à entreprendre (documentation, qualité des outils, etc...) afin de conformer cet atelier aux contraintes de qualification. Il est important de noter que l'AGL MORSE ne sera pas «qualifié» (e.g. qui respecte les contraintes définies par la norme et qui peut donc être utilisé pour le développement de logiciels certifiés) mais fournira le maximum du matériel de qualification (documents, tests...) nécessaires à la certification d'un logiciel embarqué.

5.2 Sous-projet 2 : Langage pivot et expression de propriétés sur le modèle

Responsable : LIP6

Partenaires

SAGEM	9% : compétences en certification
AONIX	14% : compétences UML
LIP6	50% : compétences en prototypage & spécifications formelles
LaBRI	27% : compétences en méthodes formelles & vérification

Objectifs

Quatre buts essentiels :

- Tâche 2.1 : Définition du langage pivot.
- Tâche 2.2 : Expression des propriétés attendues.
- Tâche 2.3 : Intégration des propriétés au modèle et au processus de vérification.
- Tâche 2.4 : Retour sur la méthodologie

Détail des réalisations et échéances

Échéance du sous-projet 2 : T0+3-T0+35 durée : 32 mois

Tâche 2.1 : Définition du langage pivot

Il s'agit principalement : 1) de formuler l'information de manière à permettre un traitement semi-automatisé pour la vérification comme pour la synthèse de programmes, 2) d'insérer des éléments permettant de définir des propriétés à vérifier par le système, 3) de définir le sous-ensemble d'UML propice à l'utilisation des méthodes formelles proposées dans MORSE.

Le premier objectif sera de définir précisément la sémantique comportementale du langage pivot et de la formaliser en précisant la notion d'état et d'évolution d'un état. Le langage pivot sera compatible avec le standard UML et s'appuiera sur les mécanismes d'extension de ce langage tels que les « tagged values » et/ou les « stéréotypes ».

Le langage pivot aura une déclinaison textuelle en XMI permettant ainsi de relier l'atelier MORSE à tout outil UML respectant ce standard. XMI permettra également d'assurer la liaison entre les outils UML et l'AGL pilote.

Il faudra ensuite définir les notations utilisées pour spécifier les propriétés attendues sur le modèle qui sont de deux types : des propriétés globales sur le système ou un sous-système et des assertions. La tâche 2.2 identifiera les propriétés pertinentes pour le type de système considéré et la tâche 2.3 précisera l'intégration de la définition et de l'exploitation de ces propriétés dans le cadre méthodologique retenu.

Enfin, le langage pivot devra intégrer des caractéristiques d'un ADL, notamment au niveau de la description des caractéristiques des communications entre composants afin de décrire finement les caractéristiques des protocoles utilisés pour assurer les communications. De telles caractéristiques sont déjà expérimentées dans LfP et devront être étendues dans le cadre des applications visées (en particulier sur la description de la fiabilité des communications).

Tâche 2.2 : Expression des propriétés attendues

Dans le but de faciliter la vérification de propriétés générales, on considèrera la construction de formules «prêtes à l'emploi» pour les propriétés très communes du domaine d'application. Il est cependant nécessaire que la propriété ait une expression générique indépendante du modèle. Nous proposons donc d'étudier les propriétés couramment étudiées sur les applications visées et de définir une forme générale applicable au plus grand nombre de modèles.

L'aspect hiérarchique du langage pivot (déjà présent dans LfP) permet d'envisager l'écriture de propriétés à différents niveaux de la spécification. Il faut donc étudier à quels niveaux il est possible de définir une propriété donnée en fonction des parties de la spécification qu'elle met en jeu. On étudiera également la possibilité de définir des propriétés qui couvrent plusieurs niveaux de la spécification. Le sous projet 3 fournira les techniques de vérification de ces propriétés, il sera alors important de prendre en compte les résultats obtenus pour faire évoluer l'écriture des propriétés.

Tâche 2.3 : Intégration des propriétés au modèle et au processus de vérification

Nous proposons d'exprimer les propriétés à vérifier sous forme de logique temporelle et d'assertions. Il sera nécessaire de définir les règles d'écritures des propriétés pour un modèle dans le langage pivot, notamment en raison de la nature hiérarchique des modèles exprimés dans ce langage. Les propriétés seront liées à la spécification. Ainsi, le modèle ne pourra être considéré comme correct que lorsque l'ensemble des propriétés incluses seront vérifiées.

Il faudra intégrer des propriétés directement dans la spécification comportementale afin de vérifier que les comportements des différents composants de l'application sont compatibles entre eux. Ces propriétés prennent la forme de contrats comportementaux ou de composants dédiés, ils sont regroupés sous le terme d'*observateurs*. Des techniques pour la vérification des propriétés exprimées par ce moyen seront étudiées par le sous-projet 3.

Tâche 2.4 : Retour sur la méthodologie

A partir des travaux d'avancement concernant la définition du modèle pivot explicités dans les tâches précédentes, il est nécessaire que soit raffinée la méthodologie élaborée en tâche 1.1. Le but de la tâche 2.4 est de prendre en compte ce travail.

Ce retour se fait par lecture croisée de type auteur/lecteur. Les acteurs de tâches font part de leurs remarques sur la méthodologie à la lumière de leur expérience. L'auteur (ici Aonix) intègre ou non ces remarques dans le guide méthodologique. Dans tous les cas, une réponse doit être fournie.

L'ensemble des versions du guide méthodologique avec les remarques et les réponses seront archivées par le projet.

5.3 Sous-projet 3 : vérification formelle de propriétés

Responsable : LaBRI

Partenaires

SAGEM	8% : observation vis-à-vis des besoins du domaine d'application
AONIX	2% : observation vis à vis de la méthodologie (retour sur la méthode)
LIP6	45% : compétences en prototypage & spécifications formelles
LaBRI	45% : compétences en méthodes formelles & vérification

Objectifs

Quatre buts essentiels :

- Tâche 3.1 : Vérification par traduction en réseaux de Petri de haut niveau.
- Tâche 3.2 : Vérification symbolique à base de diagrammes de décisions de données
- Tâche 3.3 : Développement des outils de vérification de l'atelier MORSE
- Tâche 3.4 : Retour sur la méthodologie

Détail des réalisations et échéances

Échéance du sous-projet 3 : T0+9-T0+35 durée : 26 mois

Tâche 3.1 : Vérification par traduction en réseaux de Petri de haut niveau

Des techniques structurelles de vérifications ont été développées pour les réseaux de Petri depuis de nombreuses années. Le LaBRI et le LIP6 ont participé activement à leur l'élaboration **[8]**. L'objet de cette tâche consiste donc en la spécialisation de ces techniques à la vérification structurelle du langage pivot. Trois types d'analyses sont envisagées :

1 Des simplifications seront calculées en fonction de la propriété à étudier. Cette dernière peut être locale à un ou plusieurs processus, sans nécessairement impliquer l'intégralité du système. Il s'agit donc de créer un modèle équivalant au modèle de départ du point de vue de la propriété étudiée. Le but de cette analyse étant de faciliter l'analyse effectuée au cours de la tâche 3.2 en obtenant un modèle présentant un graphe des états accessibles réduit.

2 Certaines propriétés peuvent être vérifiées en étudiant seulement la structure du modèle. Ce type de vérification présente l'avantage d'être souvent plus rapide que l'étude exhaustive des états accessibles. De nombreuses erreurs de structures du modèle peuvent ainsi être rapidement détectées.

3 Les techniques de programmation linéaire permettent d'obtenir certaines mesures générales sur les composants du système étudié. Des bornes, par exemple, peuvent être obtenues quant à l'utilisation de certaines ressources, permettant ainsi de configurer au mieux l'application générée.

Tâche 3.2 : Vérification symbolique à base de diagrammes de décisions de données (DDD)

Pour chacune des propriétés identifiées dans les tâches 2.2 et 2.3, on procédera de la manière suivante :

1 Représentation du modèle et des propriétés en termes de diagrammes de décision de données (DDD). Cette technique consiste à établir un codage des états du système en DDD et à représenter les transitions du système en termes d'opérateurs sur les DDDs.

2 Vérification symbolique du modèle. La première étape consiste à donner une représentation symbolique de l'ensemble des états accessibles du système. Cette représentation permet de vérifier de manière élémentaire les propriétés de sûreté attendues du système. Nous mettrons en œuvre des algorithmes de points fixes pour la vérification des propriétés de vivacité du système.

3 Expression du résultat dans les termes du modèle pivot. Dans le cas d'une détection d'erreur, il s'agit de fournir des indications concernant les éléments de spécification correspondants dans le modèle pivot

Nous savons par expérience que les facteurs liés à la représentation du modèle en termes de DDD peuvent influer grandement sur les performances des vérifications. Par exemple, le choix du codage des états est de même nature que le choix des variables dans un BDD (Binary Decision Diagram). Mais aussi, la manière de coder les transitions du système par des opérateurs peut influer les performances de l'outil de vérification dans des rapports de 1 à 1000. Nous consacrerons donc un effort important à la première étape et les outils développés dans le cadre de la tâche 3.3 permettront sa mise au point.

Tâche 3.3 : Développement des outils de vérification de l'atelier MORSE

L'objet de cette tâche consiste à outiller les techniques de validations étudiées dans les tâches 3.1 et 3.2 :

1 Une bibliothèque implémentant les DDD a déjà été réalisée dans le cadre du projet CLOVIS. Les premiers résultats obtenus sont très positifs, cependant des optimisations sont nécessaires pour obtenir un outil suffisamment performant pour traiter des problèmes de la taille requise.
2 Au-dessus de cette bibliothèque, les différents algorithmes «classiques» de vérification de propriétés temporelles (CTL) seront implémentés.
3 En s'inspirant des implémentations déjà réalisées dans le cadre des réseaux de Petri, des outils de vérifications structurelles pour le modèle pivot seront réalisés. De nombreuses réutilisations seront possibles, en particulier en utilisant des «solvers» de contraintes existants pour les techniques de programmation linéaires.

Tâche 3.4 : retour sur la méthodologie

Les tâches déjà définies dans le sous-projet concernent l'outillage de la vérification. Comme l'un des objectifs de MORSE est de rendre "transparente" l'utilisation des méthodes formelles et de prendre en compte les aspects de certification, il est nécessaire de raffiner à nouveau la méthodologie élaborée en tâche 1.1 à la lumière de son utilisation dans l'application témoin (sous-projet 5).

Ainsi, cette tâche s'inscrit dans une aide au développement de l'application-témoin: il s'agit de créer un conseil disponible et compétent pour la réalisation de la tâche 5.4

5.4 Sous-projet 4 : Synthèse automatique de programmes

Responsable : LIP6

Partenaires

SAGEM	37% : compétences en systèmes embarqués
AONIX	33% : compétences en exécutifs embarqués et en génération de code
LIP6	26% : compétences en système répartis & génération de code
LaBRI	3% : compétences en sémantique opérationnelle

Objectifs

Quatre buts essentiels :

- Tache 4.1 : règles de traduction de programmes
- Tache 4.2 : intégration du prototype dans un environnement d'exécution
- Tache 4.3 : implémentation de l'outil de synthèse automatique de programmes
- Tâche 4.4 : retour sur la méthodologie

Détail des réalisations et échéances

Échéance du sous-projet 4 : T0+10T0+35 durée : 25 mois

Tâche 4.1 : règles de synthèse de programmes

Lorsqu'un modèle dans le langage pivot satisfera toutes les propriétés demandées, MORSE permettra la synthèse automatique des programmes répartis correspondant au squelette de contrôle de l'application. Les programmes produits seront traçables avec le modèle afin que l'on puisse déterminer à quelle partie du modèle correspond toute partie du code source. Il sera alors possible de faire remonter au niveau du modèle des informations concernant le code synthétisé. Cela permettra de satisfaire deux objectifs :

- la conformité avec les exigences de la norme choisie comme référence (DO-178B) qui impose la traçabilité entre les données produites par le cycle de vie,
- la possibilité de déterminer quelle partie du modèle il convient de modifier si le code produit ne satisfait pas des contraintes imposées par le cahier des charges.

Afin d'accroître la réutilisation des techniques de synthèse automatique de programmes depuis des spécifications dans le modèle pivot et expérimenter plusieurs types de déploiement, on formalisera les règles de transformation en deux temps :

1. La définition de règles de **transformations sémantiques** associant des «patterns» de code réparti aux constructions offertes par le langage pivot. Ces règles visent à produire des arbres de syntaxe abstraite.

2. La définition de **règles de génération de la syntaxe concrète détaillée** appliquées aux arbres de syntaxe abstraite construits par les règles de transformations sémantiques.

Pour effectuer ce travail, on définira la «grille de lecture» d'une spécification dans le langage pivot et l'architecture générique type qui lui sera associée. Pour que les composants soient interopérables et réutilisables, le modèle d'architecture générique doit abstraire ses besoins vis-à-vis de l'environnement qu'il exprimera sous la forme de **services requis**. Il s'agit de cataloguer les éléments d'un «exécutif» nécessaire à l'exécution de la sémantique opérationnelle du langage pivot. Ces derniers pourront être classés suivant les fonctionalités du langage pivot utilisés dans une spécification (on peut ainsi optimiser l'exécutif associé à l'application en n'y embarquant que ce qui est strictement nécessaire à l'exécution du modèle). Ce travail conduira donc naturellement à définir un modèle d'interface pour l'environnement d'exécution, qui sera pris en compte pour la construction de l'application.

Tâche 4.2 : intégration du prototype dans un environnement d'exécution

L'intégration dans son environnement d'exécution du squelette de contrôle réparti d'une application spécifiée avec la méthodologie MORSE s'appuie sur la notion d'exécutif (au sens d'un langage de programmation). Ainsi, il faudra produire une API (Application Program Interface) offrant les mécanismes permettant d'assurer le contrôle d'une application répartie (RPC, communication par message, communication par mémoire partagée, gestion des erreurs de communication, etc.) à partir des services requis définis dans la tâche 4.1.

Cette API se comportera comme un intergiciel (ou middleware) entre l'application elle-même et son environnement d'exécution. Dans le cadre du projet MORSE, nous fournirons une implémentation de l'API sur l'environnement cible d'exécution choisi pour l'application témoin.

L'environnement cible choisi sera fourni par la société Aonix pour les logiciels embarqués critiques.

Tâche 4.3 : implémentation de l'outil de synthèse automatique de programmes

À partir des règles définies dans la tâche 4.1 et des informations sur l'exécutif produit dans la tâche 4.3, nous construirons un outil de synthèse automatique de programme à partir du langage pivot pour l'environnement cible visé.

L'outil ainsi développé sera principalement dédié à l'environnement d'exécution concerné. Cependant, l'utilisation de meta-générateurs de programmes et l'interfaçage propre entre les programmes générés et l'environnement d'exécution devraient nous permettre de traiter à terme (après validation de l'approche dans le cadre de l'application témoin) d'autres environnements d'exécution que celui originellement choisi.

Tâche 4.4 : Retour sur la méthodologie

La raison essentielle de cette tâche est surtout d'être un soutien en parallèle aux tâches précédentes et à la tâche de réalisation 5.4. Le raffinement de la méthodologie étant supposé acquis au moment du sous-projet 4.

5.5 Sous-projet 5 : AGL MORSE et application témoin

Responsable : SAGEM

Partenaires

SAGEM	69% : compétences en systèmes de drone
AONIX	15% : compétences en méthodes de développement
LIP6	9% : compétences en spécification formelle
LaBRI	7% : compétences en vérification formelle

Objectifs

Cinq buts essentiels :

- Tâche 5.1 : Définition de l'interface type des outils pour intégration dans l'AGL pilote.
- Tâche 5.2 : Intégration des outils.
- Tâche 5.3 : Définition de l'application témoin

- Tâche 5.4 : Réalisation de l'application témoin avec l'AGL MORSE
- Tâche 5.5 : Mesures et comparaisons

Détail des réalisations et échéances

Échéance du sous-projet 5 : T0+5-T0+36 durée : 31mois

Tâche 5.1 : Définition de l'interface type des outils pour intégration dans l'AGL pilote

Le LIP6 dispose de FrameKit **[7]**, une plate-forme d'intégration logicielle qui a fait ses preuves sur de gros projets universitaires (AGL basé réseaux de Petri comportant un demi million de lignes de code développés dans le cadre de différents projets universitaires dans le monde puis intégrés par le LIP6). Cette plate-forme constitue une base intéressante pour la construction de l'AGL pilote du projet MORSE par intégration de composants logiciels développés par les différents partenaires.

A cette fin, la définition des standards propres au projet MORSE sera nécessaire. Ces standards s'appuierons sur ceux de FrameKit d'une part, sur XMI et des formats d'échanges pertinents pour la représentations des modèles formels (réseaux de Petri et DDD) d'autre part.

Tâche 5.2 : intégration des outils

Cette tâche consiste à intégrer les outils produits par les différents partenaires au sein de l'AGL MORSE constitué sur FrameKit et de décrire la méthodologie de manière à fournir un guide aux ingénieurs qui utiliseront l'AGL pour réaliser l'application témoin.

Tâche 5.3 : définition de l'application témoin

On propose de développer un logiciel certifiable d'un système de drone (avion sans pilote). Cette tâche comprend principalement les étapes suivantes :

- Définir le périmètre précis de l'application : il s'agira d'une application réelle déjà en partie réalisée (ceci est nécessaire si on veut une comparaison finale pertinente).
- Montrer que cette application permettra effectivement de tester MORSE. Ceci nécessite de :
  a) dégager des propriétés pertinentes du système en fonction des critères de la tâche 3.1,
  b) vérifier que cette application exploite les possibilités de la métodologie MORSE,
  c) définir les scénarios de fonctionnement nominal et dégradé (pouvant mettre en cause la sécurité de la mission).
  d) préciser les composants ou entités conceptuelles réutilisables
  e) définir plusieurs plate-formes de déploiement
- Identifier les paramètres à mesurer pour caractériser à la fois la méthodologie (temps de développement, qualité, etc.) et l'application (robustesse, maintenabilité, réutilisabilité, testabilité, etc.).

Tâche 5.4 : réalisation de l'étude témoin avec l'AGL MORSE

L'application témoin sera développée par SAGEM en utilisant la méthodologie MORSE et en s'appuyant sur l'AGL associé. Un *cahier d'évaluation* sera mis à jour durant tout le développement afin d'enregistrer les problèmes rencontrés et d'assurer la traçabilité des différentes opérations.

- La première étape du développement vise à la définition de l'architecture du système exprimée dans le modèle pivot.
- La génération de spécifications formelles doit permettre la vérification des propriétés énoncées dans le modèle pivot.
- la génération de l'application-test permettra de la déployer et d'analyser son comportement dans un environnement opérationnel.

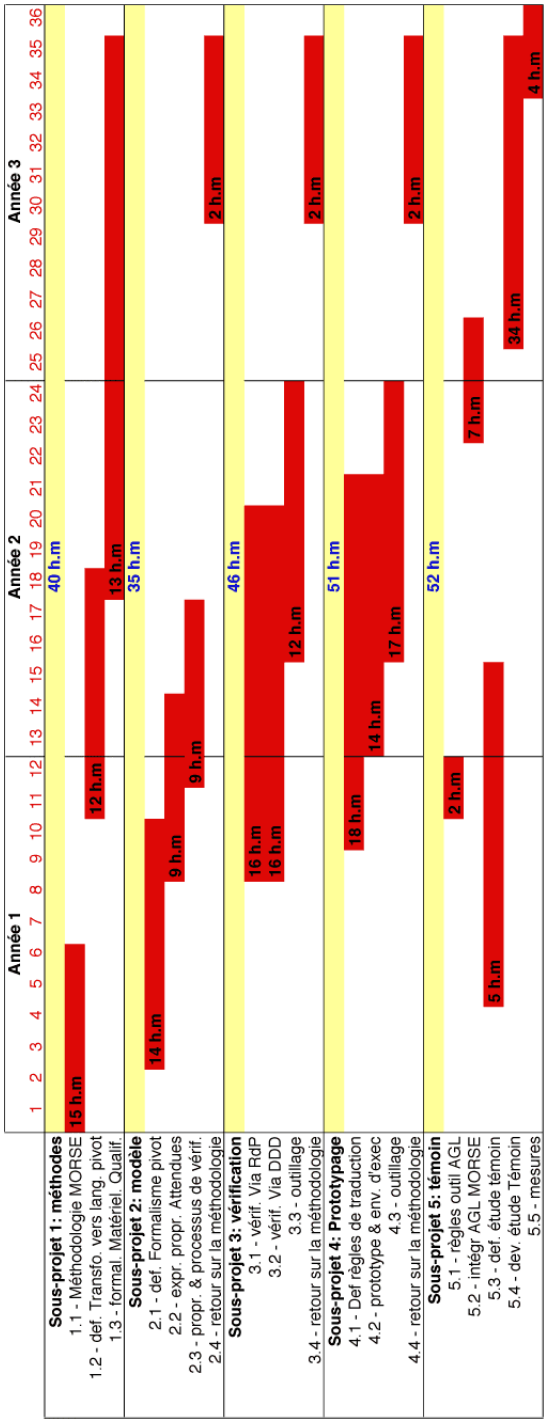
Ces deux dernières opérations seront appliquées à plusieurs reprises dans le cadre d'un processus de raffinement du modèle pivot intégrant progressivement toutes les contraintes d'exécution du système.

Tâche 5.5 : mesures et comparaisons

Il s'agit, dans cette tâche, d'étudier le cahier d'exploitation afin de comparer le développement «classique» et le développement suivant la méthodologie MORSE. À l'issue de cette tâche une synthèse sera réalisée et dressera une liste des anomalies constatées. On proposera ensuite les améliorations envisageables.

5.6 échéancier des sous-projets

Le tableau en page suivante récapitule une prévision d'échéancier des tâches dans les sous-projets ainsi que d'une estimation des coûts en homme x mois.



6. Résultats du projet et retombées

Le besoin de certifier les logiciels est une demande de plus en plus pressante des clients. Certains domaines (aéronautique, médical, ferroviaire, etc…) sont impliqués depuis dans des processus de certification mais on assiste aujourd'hui à une généralisation de la demande. Par exemple l'automobile (en particulier le contrôle moteur) et les télécoms (équipements de communication à fortes composantes logicielles) sont de plus en plus concernés. De plus, le processus de vérification pour ces domaines porte de plus en plus sur des propriétés ayant de multiples facettes. Elles obligeront à recourir à plusieurs formalismes, plusieurs méthodes lesquels pourraient à terme être tous unifiés au sein de MORSE.

Pour la plupart des projets logiciels "courants", l'enjeu n'est pas la criticité mais la qualité et la maintenance à moindre coût. MORSE pourrait être utilisé pour la réalisation de logiciels de qualité approchant le "zéro-faute". Cette tendance pourrait être renforcée du fait que MORSE intègre un ensemble de procédés qui généralise les méthodologies industrielles déjà pratiquées.

En conclusion, l'objectif du projet est d'aboutir à un AGL pour le développement de systèmes répartis et devant être certifiés. AONIX en tant qu'éditeur logiciel envisage d'exploiter les résultats de ses travaux sous forme de licences commerciales en partenariat avec les autres membres du consortium.

Retombées scientifiques

Le retour d'expérience sur l'utilisation de méthodes formelles dans un contexte de développement par prototypage concerne principalement les deux points suivants :

- **Méthodes formelles**: a) L'application de techniques novatrices pour réduire des spécifications formelles en utilisant des informations d'ordre sémantique (les constructions définies dans le langage pivot) et structurelles, b) L'étude de la traçabilité des informations entre le modèle dans le langage pivot et les spécifications formelles et, en particulier l'assistance à l'identification des problèmes impliquant la non-vérification d'une propriété.
- **Méthodologie**: a) l'analyse de l'impact d'une approche complètement basée sur la notion de spécification et non de programme, b) l'étude des possibilités de raffinement du système au niveau du langage pivot, c) la définition d'une méthodologie de vérification formelle «croisée» à partir de deux approches complémentaires (structurelles et model checking), d) la certification des liens entre les différents niveaux de description du système (UML, langage pivot, programmes), e) vérifier la qualité des programmes obtenus par synthèse automatique de programmes répartis (depuis le langage pivot) dans le contexte d'applications certifiables.

Retombées industrielles et économiques

Pour une société comme SAGEM, qui développe des produits de haute technologie (à fort contenu logiciel) sur des marchés très variés (télécoms, transports, sécurité, défense), l'enjeu est global, l'AGL MORSE étant susceptible d'être utilisé pour la plupart des applications logicielles embarquées.

Aonix est présent et actif sur le marché des AGL pour les systèmes embarqués certifiés et sur les services liés aux méthodes et processus. La méthode développée et l'atelier associé lui permettront de se positionner mieux sur ces marchés comme un précurseur.

Participation aux organismes de normalisation et de standardisation

Les travaux sur le langage pivot pourront servir de contribution au groupe de travail «UML certifiable » dont AONIX est le leader, groupe de travail dont le but est d'établir une proposition dans ce sens auprès de l'OMG.

Pour les aspects de certification, SAGEM envisage d'impliquer les autorités de certification aéronautique, à la fois au plan français (DGAC) et européen (JAA), dans la validation de la méthodologie et des outils issus de MORSE.

Résultats qui pourront être réutilisés dans d'autres projets de recherche

Le problème majeur des méthodes formelles est la maîtrise de l'explosion combinatoire due à la taille des systèmes industriels. Les techniques et outils définis dans le but d'accroître cette maîtrise tiendront compte de caractéristiques propres au domaine d'application considéré. De tels résultats

devraient être généralisables à des domaines d'application voisins (comme les systèmes répartis dynamiques).

Principe d'accord et propriété industrielle

Les partenaires du projet gardent la propriété intellectuelle des composants technologiques qu'ils possèdent au commencement du projet. La liste non limitative de ces composants est :

1. AONIX: Les produits StP et ACD ainsi que les schémas de génération de code existant à partir d'UML,
2. SAGEM : l'ensemble des cadres applicatifs (algorithmes, codes, documents de conception, …) mis à disposition du projet
3. LIP6 : l'environnement CPN-AMI, les environnements FrameKit (méta-générateur d'AGL) et MetaScribe (méta-générateur de code),
4. LaBRI : la bibliothèque pilote des Diagrammes de Décision de Données.

Durant la réalisation du projet, chaque partenaire a accès aux technologies des autres partenaires, et cela de façon non-transférable à des tiers, non utilisable en dehors du cadre du projet, et non-illimitée en termes de copies de ces composants technologiques mais en nombre raisonnable pour le bon déroulement du projet. Il sera procédé à un échange d'accord de confidentialité si un ou plusieurs partenaires le demandent.

À la fin du projet, la propriété intellectuelle sera définie comme suit:

1. Chaque partenaire reste propriétaire des composants qu'il aura fournis ainsi que des enrichissements qui y seront apportés durant le projet.
2. AONIX pourra commercialiser l'AGL issu de ces travaux. Des accords concernant les montants des *royalties* seront signés entre les partenaires pour tenir compte des outils que chacun aura intégrés dans l'AGL MORSE.
3. Les industriels conservent l'accès aux composants technologiques et le droit d'utilisation illimité dans le temps du produit délivré à la fin du projet. Ils devront cependant, le cas échéant acquérir toute version ultérieure ou dérivée de ce produit, auprès des sociétés qui en assurent la commercialisation. Cette acquisition se fera toutefois à un tarif préférentiel.
4. Les laboratoires, universités ou autres organismes publics gardent l'accès aux composants technologiques et au produit délivré à la fin du projet sans limitation de temps mais sous condition de ne les utiliser que dans le cadre de travaux de recherche ou d'éducation.

7. Tableaux de financement

7.1 Tableau I

Partenaire	Personnel				
	sous-projet 1	sous-projet 2	sous-projet 3	sous-projet 4	sous-projet 5
LIP6	8 545,66 €	42 728,29 €	38 083,91 €	22 620,86 €	7 395,28 €
LaBRI	3 861,37 €	20 961,71 €	34 416,54 €	2 271,39 €	5 197,99 €
AONIX	375 198,92 €	102 094,95 €	15 536,19 €	238 221,54 €	109 948,40 €
SAGEM	271 894,27 €	71 708,38 €	72 747,63 €	311 673,67 €	579 183,05 €
TOTAL	**659 500,22 €**	**237 493,32 €**	**160 784,26 €**	**574 787,46 €**	**701 724,73 €**

Partenaire	Equipement	Fonctionnement	Frais de gestion
LIP6	35 000,00 €	30 462,44 €	14 786,92 €
LaBRI	20 000,00 €	42 302,54 €	10 320,92 €
AONIX	40 000,00 €	30 000,00 €	80 900,00 €
SAGEM	35 000,00 €	0,00 €	122 006,00 €
TOTAL	**130 000,00 €**	**102 764,98 €**	**228 013,84 €**

Partenaire	Total personnel	Autofinancement	Aide demandée
LIP6	119 374,00 €	138 738,00 €	199 623,36 €
LaBRI	66 709,00 €	86 754,00 €	139 332,46 €
AONIX	841 000,00 €	595 140,00 €	396 760,00 €
SAGEM	1 307 207,00 €	878 527,80 €	585 685,20 €
TOTAL	**2 334 290,00 €**	**1 699 159,80 €**	**1 321 401,02 €**

7.2 Tableau II

Le total des moyens nécessaires au projet est la somme des colonnes « Total personnel » et « Autofinancement » du tableau I, pour les universitaires, et la somme des colonnes « Total personnel », « équipement », « fonctionnement » et « frais de gestion » pour les industriels.

Partenaire	Total des moyens nécessaires	Aide demandée
LIP6	457 735,36 €	199 623,36 €
LaBRI	292 795,46 €	139 332,46 €
AONIX	991 900,00 €	396 760,00 €
SAGEM	1 464 213,00 €	585 685,20 €
TOTAL	**3 206 643,82 €**	**1 321 401,02 €**

8. Références bibliographiques

[1] **F. Kordon, I. Mounier, E. Paviot-Adet & D. Regep,** « Formal verification of embedded distributed systems in a prototyping approach », in proceedings of the International Workshop on Engineering Automation for Software Intensive System Integration , Monterey, June 2001

[2] **J. Rushby,** « formal methods and their role in the certification of critical of critical systems », Computer Science Laboratory, SRI International, technical report CSL 951 March 1995

[3] **OMG,** « Model Driven Architecture (MDA)», Document number ormsc/2001-07-01, http://www.omg.org/cgi-bin/doc?ormsc/01-07-01.pdf, July 2001

[4] **M. Doche, I. Vernier-Mounier & F. Kordon,** « Modular approach to specify and validate an Electrical Fligh Control System », in proceedings of Formal method Europe 2001 (LNCS)

[5] **E. Paviot-Adet & I. Vernier-Mounier**, « Modélisation et Vérification de l'interopérabilité de services de télécommunication », À paraître chez Hermes Edition, éditeur Scientifique Michel Diaz

[6] **F. Kordon,** « Code Generation from Petri Nets », à paraître dans « Petri Net and System Engineering », editeurs scientifiques, C. Girault & R. Valk

[7] **LIP6-SRC,** « Le projet Mars », http://www.lip6.fr/mars

[8] **D. Poitrenaud & J-F. Pradat-Peyre,** «Pre and post-agglomerations for LTL model checking», In Proceedings of the 21th International Conference on Applications and Theory of Petri Nets, Lecture Notes in Computer Science, Aarhus, Denmark, June 2000. Springer-Verlag

[9] **LaBRI & LIP6,** Marché 99-34-024 de la DGA – «Extension des diagrammes de décision binaire pour le traitement des types finis. Application à la vérification de systèmes complexes», rapport final d'activité

[10] **Requirements and Technical Concepts for Aviation (RTCA),** «Software Recommandations in Airborn Systems and Equipment Certification (DO-178B)», 1992

[11] **J-M. Couvreur, E. Encrenaz, E. Paviot-Adet, D. Poitrenaud and P-A. Wacrenier,** «Data Decision Diagrams for Petri nets Analysis», to appear in Petri Net '2002

[12] **OMG,** «The MOF specification», document OMG, réf. 00-04-00, http://www.omg.org/cgi-bin/doc?formal/00-04-03.pdf

[13] **J. Abrial,** «the B-book», Cambridge University Press, 1995

[14] **B. Caillaud, J-P. Talpin, J-M. Jézéquel , A. Beveniste & C. Jard,** «BDL : A Semantics Backbone for UML Dynamic Diagrams auteurs». Rapport de recherche n°4003, 5 Septembre 2000.

[15] **A. Arnold, A. Griffault, G. Point, & A. Rausy,** «The AltaRica formalism for describing concurrent systems. Fundamenta Informaticae», 40:109--124, 2000

[16] **Ph. Schnoebelen, B. Bérard, M. Bidoit, F. Laroussinie, & A. Petit,** «Vérification de logiciels : Techniques et outils du model-checking», Vuibert, 1999

[17] **H. Garavel**, «VASY», Inria Rhones-Alpes, http://www.inrialpes.fr/vasy/

[18] **Alcatel,** «REUTEL-2000 : Outils et techniques pour applications de télécommunications réutilisables», http://www.inrialpes.fr/vasy/reutel/

[19] **SAFEAIR Consortium,** «SAFEAIR : Advanced Design Tools for AirCraft Systems and Airborne Software» http://www.safeair.org

9. Contacts

Partenaires	Contact scientifique	Contact administratif
SAGEM	Jean-Pierre Velu Jean-Pierre.Velu@sagem.com tel : 01 34 30 57 85 fax : 01 34 30 50 40	Jean-Thierry Audren Jean-Thierry.Audren@sagem.com tel : 01 58 12 41 35 fax : 01 58 23 79 27
AONIX	Christian Romano romano@aonix.fr tel : 01 41 48 11 43 fax : 01 41 48 10 24	Christophe Faurère Faurere@aonix.fr tel : 01 41 48 10 36 fax : 01 41 48 10 21
LIP6	Fabrice Kordon Fabrice.Kordon@lip6.fr tel : 01 44 27 88 20 fax : 01 44 27 74 95	Véronique Varenne Véronique.Varenne@lip6.fr tel : 01 44 27 74 96 fax : 01 44 27 74 95
LaBRI	Jean-Michel Couvreur couvreur@labri.u-bordeaux.fr tel : 05 56 84 69 34 fax : 05 56 84 66 69	Jean-Louis Lassartesses jll@labri.fr tel : 05 56 84 65 80 fax : 05 56 84 69 24

10. Glossaire

AGL	Atelier de Génie Logiciel.
ACD	*Architecture Centric Development* ; méta-générateur de code d'AONIX sur lequel se basera la génération de code et de spécifications formelles du projet MORSE.
ADL	Architecture Description Language : langage de description d'architecture.
CPN-AMI	Environnement de Génie Logiciel issu de la plate-forme d'intégration *FrameKit* pour la modélisation et l'évaluation des systèmes distribués au moyen de réseaux de Petri.
CTL	computational Tree Logic : logique temporelle à temps arborescent.
DDD	Diagramme de décision binaire ; méthode de représentation de l'espace des états accessibles d'un modèle.
MDA	Model driven Architecture ; approche de développement en cours de définition à l'OMG, entièrement basée sur la notion de modèle.
FrameKit	Méta-outil de création d'ateliers logiciels développé par le LIP6.
LaBRI	Laboratoire Bordelais de Recherche en Informatique
LIP6	Laboratoire d'Informatique de Paris VI.
LfP	Language for Prototyping ; langage défini au LIP6 pour la spécification de systèmes concurrents asynchrones.
LUSTRE	Langage de la famille des langage formels synchrones au même titre que ESTEREL ou SIGNAL.
MetaScribe	Méta-générateur de code développé par LIP6.
MORSE	Méthode et Outils pour la Réalisation et la vérification formelle de Systèmes interopérables Embarqués critiques.
OCL	*Object Constraint Language* ; langage de définitions de contraintes d'UML sur des entités du modèle.
OMG	Object Management Group.
Pattern	Démarche proposée qui identifie, puis formalise une solution à un problème récurrent.
RTCA	Requirements and Technical Concepts for Aviation ; organisme à but non lucratif qui développe des recommandations pour les organismes de régulation du trafic aérien.
Scade_lang	Langage basé sur LUSTRE défini par l'atelier SCADE pour la spécification de systèmes synchrones.
Tagged Values	Etiquettes UML ; mécanisme d'extension prédéfini d'UML.
UML	*Unified Modeling Language* ; formalisme standard de description graphique de modèles objets.
XMI	*Extended Model Interface* ; description sous forme XML du méta-modèle UML.
XML	*Extended Markup Language* ; langage standard dérivé de SGML permettant de décrire sous forme de tags des structure de données.