Contribution à la méthodologie de conception système : application à la réalisation d'un microsystème multicapteurs communicant pour le génie civil

Rémy Maurice

N° d'ordre : *Année : 2005*

THESE

Présentée au

LABORATOIRE D'ANALYSE ET D'ARCHITECTURE DES SYSTEMES DU CNRS

en vue de l'obtention du titre de

Docteur de l'Institut National Polytechnique de Toulouse

Ecole doctorale : Génie Electrique, Electronique, Télécommunications
Spécialité : Conception de Circuits Microélectroniques et Microsystèmes

par

Rémy MAURICE

Ingénieur en Science des Matériaux

CONTRIBUTION A LA METHODOLOGIE DE CONCEPTION SYSTEME : APPLICATION A LA REALISATION D'UN MICROSYSTEME MULTICAPTEURS COMMUNICANT POUR LE GENIE CIVIL

Soutenue le 15 Décembre 2005, devant la Commission d'Examen :

Rapporteurs
- *Gaston CAMBON* — *Professeur à l'université de Montpellier II*
- *Isabelle DUFOUR* — *Chargée de recherche - HDR à l'université de Bordeaux I*

Examinateurs
- *Delphine BOUCHET* — *Ingénieur, EDF R&D*
- *Eric CAMPO* — *Maître de conférences - HDR à l'université Toulouse II, Directeur de thèse*
- *Daniel ESTEVE* — *Directeur de recherche CNRS, Co-Directeur de thèse*
- *Mario PALUDETTO* — *Professeur à l'université Paul Sabatier, Toulouse III*

Invité
- *Alain BASCOUL* — *Professeur à l'université Paul Sabatier, Toulouse III*

REMERCIEMENTS

Le travail présenté dans ce mémoire résulte de la collaboration entre la société Electricité De France Recherche et Développement (EDF R&D) et le Laboratoire d'Analyse et d'Architecture des Systèmes du Centre National de Recherche Scientifique (LAAS-CNRS).

J'exprime tous mes remerciements à Madame Claire Laurent, responsable du département Optimisation des Performances des Process (OPP) à EDF R&D et Messieurs Jean-Claude Laprie et Malik Ghallab, directeurs successifs du LAAS, pour m'avoir accueilli dans leur établissement respectif. Je remercie également Madame Anne-Marie Gué responsable du groupe Microsystèmes et Intégration des Systèmes du LAAS-CNRS.

Mes remerciements s'adressent tout particulièrement à mon directeur de thèse Eric Campo pour la confiance qu'il m'a témoignée en acceptant la direction de mes travaux ainsi que pour son soutien et ses conseils scientifiques.

Ils s'adressent aussi chaleureusement à Daniel Estève pour avoir co-encadré et guidé mes recherches. Sa grande disponibilité et ses conseils constructifs m'ont guidé tout au long de cette thèse.

Ils s'adressent également à Delphine Bouchet, ingénieur à EDF R&D, sans qui ce travail n'aurait probablement pas vu le jour. Ses conseils pertinents, sa détermination, son regard opérationnel a guidé le travail jusqu'au bout.

Je remercie sincèrement Monsieur Mario Paludetto qui m'a fait le très grand honneur de présider mon jury et de participer à l'amélioration de mes travaux par ses conseils judicieux. Je remercie également Monsieur Gaston Cambon et Madame Isabelle Dufour pour avoir soigneusement rapporté sur mon manuscrit et Monsieur Alain Bascoul pour avoir accepté avec beaucoup d'amabilité d'examiner mon travail dans sa partie applicative du Génie Civil.

Je tiens également à témoigner toute ma reconnaissance à l'équipe de recherche Ingénierie des Communications en informAtique Réseaux et Electronique sans fil (ICARE) de l'IUT de Blagnac, notamment à son directeur Jean-Jacques Mercier pour m'avoir permis de disposer des équipements de son laboratoire. Je remercie également les doctorants et particulièrement Salim El Homsi, Nicolas Fourty, Céline Guilleminot, et Adrien Van Den Bossche pour leur accueil très chaleureux et les échanges fructueux.

Je voudrais tout spécialement adresser mes remerciements aux doctorants investis sur les travaux thématiques de plate-forme de conception système, à savoir Hernan Duarte, David Guihal, Juan Carlos Hamon, Adel Ouardani, Petra Shmitt qui ont accepté de partager avec moi leur temps, leurs idées et leurs motivations sur ce projet de recherche.

Un grand Merci bien sure à mes collègues Gustavo Ardila Rodriguez, Patrick Abgrall, Eric Besson, Sylvain Bonhomme, Samuel Charlot, Ahmed Dkissi, Christophe Escriba, Craig Ferguson, Jan Sudor pour les discussions quotidiennes brillantes diverses et variées.

Enfin je voudrais remercier mes parents, mon frère, mes sœurs, ma belle famille et particulièrement mon épouse Marilyne qui ont œuvré dans l'ombre et qui n'ont cessé de m'apporter leur soutien au cours de ces trois années.

SOMMAIRE

Liste des figures

Glossaire

ASIC	Application Specific Integrated Circuit
CAO	Conception Assistée par Ordinateur
CDCF	Cahier Des Charges Fonctionnel
CMS	Composant Monté en Surface
CoB	Chip on Board
COTS	Commercial Off The Shelve
DMC	Dispositif Multicapteurs
FC	Flip Chip
FPGA	Field Programmable Gate Array
HiLeS	High Level Specifications
LVDT	Linear Variable Differential Transducer
MCM-V	Multi Chip Module - Vertical
MDA	Model Driven Architecture
MEMS	Micro ElectroMechanical Systems
OLC	Outils Logiciels pour la Communication
OMT	Object Modeling Technique
OOD	Object Oriented Design
OOSE	Object Oriented Software Engineering
PCB	Printed Circuit Board
PIM	Platform Independent Model
PLD	Programmable Logic Device
PSM	Platform Specific Model
RdP	Réseau de Petri
RF	Radio-Fréquence
SADT	Structured Analysis and Design Technique
SiP	System in Package
SoC	System on Chip
SoP	System on Package
STB	Spécification Technique du Besoin
SysML	System Modelling Language
TINA	TIme petri Net Analyzer
UML	Unified Modelling Language
VHDL	V(HSIC)-HDL : VHSIC Hardware Description Language
VHDL-AMS	VHDL - Analog and Mixed Signal
VHSIC	Very High Speed Integrated Circuits

Introduction générale

Imaginés à la fin des années 80 et mis au jour par le micromoteur électrostatique de R.S. Muller [FTM89], les microsystèmes sont aujourd'hui entrés dans une phase de développement industriel intense sous la forme de microsystèmes autonomes et communicants [BBB04], [GMW04], [YOD02]. La baisse des coûts et l'accroissement des performances facilitent en effet la définition et la mise en place de réseaux de capteurs connectés à une centrale de base (poste central) pour réaliser des tâches de surveillance continue, de détection d'alertes, de contrôle de procédés. On peut imaginer cette dynamique d'innovation sur une gamme d'applications très large et très diverses en sécurité industrielle, surveillance médicale, surveillance d'infrastructures, etc. Les arguments sont très forts dès qu'il s'agit de surveillance ou de maintenance. **Dans ce contexte, notre analyse est que les microsystèmes ont un rôle très important à jouer dans le développement d'une activité industrielle nouvelle sur la surveillance automatique des installations de toutes natures.**

Le microsystème est, dans notre analyse, un système miniaturisé, très compact, réalisé avec (autant que possible) des technologies de fabrications collectives pour avoir des prix de fabrication les plus bas possibles. C'est un système multifonctionnel avec des fonctions de mesure, de traitement du signal, de mémorisation, de communication et d'actionnement. Il comporte une capacité de calcul (processeur) qui peut embarquer des logiciels donnant une « certaine intelligence » à l'ensemble. Une dernière caractéristique est que ce microsystème doit être faiblement consommateur d'énergie pour communiquer « sans fil » sur des périodes longues (plusieurs années).

Malgré la diversité évidente des applications, l'architecture interne va être relativement générale ce qui autorise des efforts importants pour faciliter et accélérer les procédures de spécification, de conception et de fabrication, permettant de proposer de nouveaux produits rapidement avec une grande assurance qualité. Une part importante de notre motivation se situe donc dans la définition et la mise au point de **méthodes et d'outils de conception microsystèmes**. C'est un objectif d'une grande actualité dans le sens où les méthodes et les outils de la conception électronique ne sont pas suffisants pour traiter la **pluridisciplinarité** et donc les microsystèmes. **Nous voulons donc participer à cette extension « système » de la CAO électronique.**

Sous cet angle, notre travail s'inscrit dans la stratégie « système » du Laboratoire d'Analyse et d'Architecture des Systèmes et notamment du groupe Microsystèmes et Intégration des Systèmes, au sein duquel nous avons réalisé ce travail de thèse. Depuis quelques années, plusieurs travaux de thèse ont été axés dans cette perspective : N. HARCHANI [Har00], F. JIMENEZ [Jim00], J.C. HAMON [Ham05], D. GUIHAL [Gui03]. Ensemble, **nous avons construit une plate-forme « complète » de conception qui est la plate-forme HiLeS** que nous décrirons dans notre premier chapitre. Notre contribution a visé à définir les modalités de mise en œuvre sur un exemple précis d'application.

L'exemple choisi est celui d'un microsystème multifonctions, autonome et communicant destiné à la surveillance en Génie Civil. Nous avons appelé ce projet technologique : SmartGec. « Smart » pour d'une part, sa capacité à mémoriser et à traiter les informations provenant des paramètres physiques de son environnement, et d'autre part, sa capacité à les restituer sous une forme exploitable à une centrale de base distante, et « Gec » pour rappeler son orientation vers une application Génie Civil. L'idée est de disposer, sur une construction de structure, des capteurs pour détecter et mesurer les effets de fatigue. Il s'agit de capteurs répartis qui communiquent avec une centrale de base pour, en temps réel, faire la synthèse des informations et décider, si il y a lieu, d'alerter la maintenance. Le cahier des charges a été défini par l'entreprise EDF R&D qui a supporté ce travail et participé à sa conduite.

En résumé, le LAAS et EDF R&D se sont associés pour :

- Définir et développer des méthodes et des outils de conception pour les microsystèmes.
- Spécifier, concevoir et prototyper un microsystème spécifique pour une application de surveillance en Génie Civil.

Notre approche consacre une part importante du travail à définir une procédure détaillée de conception en nous appuyant sur cet exemple convenu. Elle **dégage une méthode générale et des outils adaptés**, pour finalement aboutir à un prototype réel dont les performances seraient « prédites » auparavant par un prototype virtuel. Nos objectifs suivants étaient de valider l'approche par la validation du prototypage réel vis à vis des spécifications initiale. Nous avons orienté notre travail simultanément sur la méthodologie et l'application. Cela nous conduit naturellement à corréler étroitement, dans la présentation, travail théorique et application. Avec ce choix, nous présenterons les résultats en quatre chapitres.

Un premier chapitre, fera le point sur la dynamique de développement microsystème d'une part, et l'état actuel des outils de conception d'autre part. Nous rappellerons plus en détail, l'état du développement de la **plate-forme HiLeS** auquel nous avons collaboré. Ce chapitre conclura sur la problématique microsystèmes telle que nous l'avons traité.

Le deuxième chapitre montrera, sur l'exemple, les étapes que nous avons conduites pour aller du stade des spécifications au stade des **spécifications validées**. Il abordera, dans une première partie, la méthode que nous avons suivie pour formaliser le cahier des charges. Dans une deuxième partie, notre système sera décrit dans le formalisme de l'outil HiLeS. Enfin, ce chapitre abordera les aspects de vérification des modèles de spécifications fonctionnelles.

Le troisième chapitre apportera, par l'exemple, les étapes de choix et d'analyse qui conduisent la transformation des spécifications en prototype virtuel. Il abordera les méthodes de transformation des représentations fonctionnelles en **prototype simulable**.

Enfin, le quatrième chapitre présentera les étapes d'intégration qui mènent au **prototype réel**. Il décrira les composants du système, ainsi que le test des principales fournitures et s'achèvera sur un test, en situation réelle, des fonctionnalités du système complet.

Chapitre 1

Problématique scientifique et industrielle

1.1 Introduction

Notre objectif de thèse est double :

- Concevoir et développer de nouvelles générations d'outils en conception système,
- Les expérimenter sur un exemple d'application finalisé.

Ce chapitre est consacré à exposer ces deux problématiques et à introduire les approches que nous avons choisies.

Sur les questions plus fondamentales des méthodes et outils de conception, notre travail s'inscrit dans une dynamique de groupe autour de la **plate-forme** HiLeS [Har00], [Jim00], [Ham05]. Nous avons participé aux spécifications de cette plate-forme en apportant nos expériences en relation avec les problèmes posés par la mise en œuvre. Ainsi, nous proposons, dans ce chapitre, un court état des pratiques et un état d'avancement de la **plate-forme** HiLeS. Dans le chapitre suivant, nous insisterons davantage sur les modes d'utilisation et de mise en œuvre, en essayant de dégager des pistes nouvelles pour compléter ou améliorer cette **plate-forme**.

Sur la question des applications microsystèmes, nous ferons un état des connaissances avant de rappeler très brièvement l'évolution historique du projet et les technologies accessibles pour le réaliser. Nous présenterons, ensuite, notre exemple d'application, le projet « SmartGec » orienté sur la conception et la réalisation d'un microsystème de surveillance en Génie Civil.

1.2 Etat de l'art des microsystèmes de surveillance

Il existe depuis toujours des capteurs servant la mesure et les automatismes associés. Nous nous intéresserons ici à des capteurs placés, en marge des automatismes, pour ne servir que des fonctions de surveillance. La surveillance des systèmes est une problématique générale dont la criticité est inhérente à la complexité toujours croissante. L'idée est d'implanter un niveau de surveillance supplémentaire (supervision) constituée des capteurs et d'un organe de diagnostic pour alerter l'utilisateur de toute anomalie de comportement, ou erreur de fonctionnement. La surveillance ainsi perçue est au cœur de notre problématique d'application. Conçue comme un niveau de surveillance externe au système principal, nous insisterons sur l'intérêt de l'autonomie en terme de communication et d'énergie.

1.2.1 Les capteurs de surveillance

Les micro-capteurs représentent les éléments sensibles qui vont détecter et convertir les phénomènes physiques en signaux identifiables et mesurables. Ils donnent une perception de l'environnement. Les capteurs peuvent être classés en deux grandes familles selon S.M. Sze [Sze94], les capteurs auto-générés et les capteurs de modulation. Prenons deux exemples pour illustrer ces deux familles de capteurs :

- un thermocouple où une variation de température (ΔT) produit une différence de potentiel directement mesurable. Il appartient donc à la famille des capteurs auto-générés car le signal capté est la source même d'énergie qui fournit le signal électrique mesuré en sortie.

- une résistance de platine alimentée par un courant constant où un ΔT produit une variation de résistance décelable par une variation de tension à ses bornes. Il

appartient donc à la famille des capteurs de modulation car le signal capté module un signal fourni par le système.

Les capteurs auto-générés ont le grand avantage de ne pas dépendre d'un signal externe et donc d'être plus économes en énergie, ce qui peut s'avérer important dans les systèmes de surveillance.

Les capteurs sont les outils habituels de la mesure et du contrôle automatique. Ils prélèvent l'information qui une fois filtrée sert la définition d'une action de commande. Dans les applications liées à la surveillance, les capteurs doivent être précis, fidèles et fiables, etc. Cela a définit une industrie de l'instrumentation et du contrôle automatique qui a joué un grand rôle dans le développement industriel.

Les capteurs sont utilisés « depuis toujours » à des fins de surveillance de procédés de fabrication industriels, par le biais de paramètres d'environnement comme la température, la pression, la vitesse, le couple, etc.

Les microtechnologies, basées sur l'utilisation du Silicium comme substrat, ont largement bénéficié des progrès faits dans le domaine de la microélectronique depuis les années 70. C'est depuis les années 80, que la communauté scientifique s'est intéressée aux usinages de volume du Silicium par voie chimique, physique ou mécanique. Ces techniques permettent de fabriquer des structures mécaniques mobiles sur Silicium de type poutre, pont ou membrane. Une réalisation majeure a été celle d'un micromoteur qui a démontré les possibilités de ces microtechnologies sur Silicium. D'un point de vue général, ces structures peuvent être utilisées selon deux axes :

- elles peuvent être passives et sujettes aux sollicitations de leur environnement, auquel cas elles représentent un élément sensible, qui sonde son environnement,
- elles peuvent être actionnées par des forces électrostatiques commandées, auquel cas, elles rejoignent alors la famille des actionneurs.

Nous nous intéresserons à la première famille qui propose une alternative ou de nouvelles applications au domaine actuel de la surveillance et de la mesure.

La réutilisation des procédés technologiques communs à la microélectronique confère aux microtechnologies les avantages du faible coût, lié à la fabrication de masse de l'industrie microélectronique et une compacité d'intégration qui a ouvert, depuis une dizaine d'années, le marché des microcapteurs. Un des premiers exemples de microcapteurs commercialisés en masse est l'accéléromètre d'Analog Device (ADXL202), utilisé pour détecter un choc et déclencher les « airbags » des voitures. Le Tableau 1-1 donne un exemple de quelques microcapteurs d'accélération qui sont commercialisés au coût moyen de 15 Euros l'unité.

Modèle	Consommation	Axes	Taille (mm)	Gamme de mesure	Signal de sortie	Accélération maximum (g)	sensibilité	Domaine de température
Analog Devices ADXL 202/210E	3-5,25V ; 0,6mA	2	5*5*2	± 2g à ±100g	Numérique (tps de cycle) et analogique	1000	312mV/g	0 ; 70°C
 **STMicroelectronics LIS2L01**	3-5,25V ; 1mA	2	10*15	± 2g à ±6g	analogique		2V/g	-40 ; 58°C
 **Delphi Intellek**	5V ; 5mA	2		± 0.75g à ±3g	analogique	10000	0.5V/0.75g 4.5V/3g	-40 ; +125°C
 **Tronic's**		2	3*3*1	± 2g à ±150g	analogique		220fF/g	
 **Si-Flex**	6V ; 45mA	3	58*75*79	±3g	analogique	1500	1.25V/g	-40 ; +85°C

Tableau 1-1 : Comparatif de microcapteurs d'accélération.

La compacité et le faible coût sont des avantages déterminants que les microcapteurs ont apportés au domaine de la mesure. Ces deux avantages ont permis aux microcapteurs d'être utilisés en très grand nombre et de manière quasi-invisible pour leur environnement direct. Une perspective ouverte par ces microcapteurs est donc la possibilité d'établir un diagnostic en local. Les microcapteurs sont alors dotés d'une « intelligence » par l'ajout d'une analyse du signal et d'une décision in-situ. Cette localisation du traitement de l'information au plus près de la source apporte une sécurité supplémentaire au système dans les domaines d'applications où la mobilité et l'autonomie, en terme de prise de décision, ont une importance considérable.

Dans le paragraphe suivant, nous nous appuierons sur l'exemple de quelques applications de systèmes de surveillance récemment développés et qui ont bénéficié des avancées des microtechnologies.

1.2.2 Les tendances vers la miniaturisation et l'autonomie

C'est au début des années 1990 que sont posées les bases de la problématique du regroupement de plusieurs microcapteurs sur une même puce par K.D. Wise [WN91]. L'étude porte d'abord sur la compatibilité des procédés technologiques de fabrication des parties capteurs et traitement du signal. Le but est de reporter, au plus près de la partie sensible, un premier étage de traitement du signal (amplification, multiplexage, filtrage) apportant aussi une première valeur ajoutée au conditionnement du signal. Quelques années plus tard, dans les années 2000, le lancement du projet Smart Dust par K. Pister [WLL01] tente de réaliser un regroupement de microcapteurs sur une même puce. L'objectif est d'intégrer un système de traitement du signal et d'y ajouter un moyen de communication. Les thèmes abordés par cette nouvelle problématique sont de natures différentes, ils concernent principalement, du point de vue informatique, la gestion de l'énergie que consomme le système lors de son activité de calcul autonome [HSW00], l'intégration de microcapteurs variés et les moyens de communication et de transmission de l'information. Les nombreuses recherches au plan international ont conduit, dans ces trois domaines, à l'obtention d'une première génération de systèmes avancés. Plusieurs systèmes sont développés sur la même base architecturale : capteur, conditionneur de signal, traitement du signal, mémoire, communication (Figure 1-1). L'existence d'une architecture de base invite à préparer une approche spécifique de la conception. C'est une des motivations de notre étude.

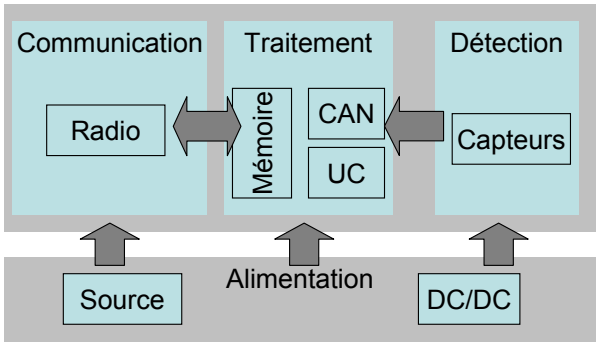


Figure 1-1 : Architecture de base des microsystèmes.

Le Tableau 1-2 présente cette première génération de systèmes d'étude qui ont pris pour base cette même configuration architecturale. Ces systèmes ont à peu près tous la même dimension. Ils sont contenus dans un cube de quelques cm de côté. Cependant, chaque équipe a choisi de concevoir son prototype sur une base microprocesseur/radio bien différente. Ces choix dépendent de paramètres tels que l'application cible, la quantité d'information à traiter, la vitesse de réaction souhaitée des nœuds du réseau, la durée de vie, l'étendue spatiale du réseau, etc.

Depuis, ces projets ont généré une créativité visible du point de vue industrielle par le biais de sociétés comme Chipcon, Ember, et CrossBow, qui proposent des services ou des produits liés aux domaines respectifs de la communication, du logiciel, et de l'intégration.

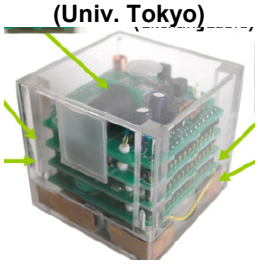
	µAMPS [MSB01] (MIT)	**Mica [HC02] (Berkeley)**	**U³ [KMM03] (Univ. Tokyo)**	**Smart-it [BG03] (Karslruhe)**
				
Radio	LMX3162 2.4Ghz	RFM TR1001 916.5 MHz	CDC-TR-02b 315MHz	RFM TR1001 868Mhz
Processeur	Strong ARM SA-1100	Atmega103	Pic18F452	Pic16F87X, 20MHz
Système d'exploitation	µOS	TinyOS		
Mémoire	512KB	512KB		8KB
Dimensions	55X55 (2niveaux)	30X60mm	50X50X50	50X50X10mm
Capteurs		Connexions pour 8 capteurs	Mouvement, Photo, Température	Lumière, pression, accélération, température

Tableau 1-2 : Première génération de systèmes multi-capteurs communicants.

Cependant, il reste un aspect important qui est le défi de demain, c'est l'autonomie en énergie. Nous voyons autour de nous de nombreuses applications multimédia liées à la portabilité et à la mobilité des objets. Ce concept est mature, et dans ce cas l'autonomie est souvent de relative courte durée (quelques jours). De plus, elle nécessite un passage par une source secteur régulière. Par contre, les applications impliquant des objets intelligents et autonomes répartis sur de larges étendues de notre environnement ne seront envisageables que si nous disposons de sources d'énergies suffisantes ou renouvelables pour les alimenter pendant toute leur durée de vie. Une grande variété de méthodes d'alimentation des microsystèmes en énergie a été étudié. Elles peuvent être classées en deux groupes [Mau03a] : d'une part, les sources locales d'une capacité nominale fixe (les microbatteries, les piles à combustibles, les accumulateurs), et d'autre part, les sources renouvelables qui sont associées à leur environnement pour se régénérer (l'énergie photonique, les vibrations, les ondes électromagnétiques, les rayonnements β, les flux éoliens). Ce dernier groupe est actuellement émergent, notamment pour les applications de petites dimensions qui nous concernent. Cette collecte d'énergie dans l'environnement ouvre de nouvelles problématiques de conversion d'énergie liées aux échelles millimétriques.

1.2.3 Les applications

Les applications concernent de nombreux domaines d'activité, notamment, l'agriculture, l'environnement, ou les réseaux de sécurité publique en général :

- pour l'agriculture : l'étude faite sur les réseaux de microsystèmes dans ce domaine [BBB04] a montré que de tels systèmes seront d'autant plus intéressants que les réseaux peuvent être déployés et modifiés à souhaits pendant les différentes saisons. Les réseaux apporteraient un intérêt dans la gestion de l'eau, la prévention de dérives climatiques, la surveillance de l'hygrométrie, et une base de données sur laquelle pourront discuter les différentes parties impliquées (propriétaires, exploitants, ouvriers agricoles).

- pour l'environnement, les écosystèmes : les réseaux de microsystèmes permettent de surveiller des zones reculées ou confinées [MPS02] sans en perturber les équilibres. La surveillance de zones aquatiques est aussi envisagée dans [YOD02].

- pour la santé : les réseaux d'assistance médicale [GMW04], permettent de suivre en direct, et de détecter les variations de paramètres physiologiques vitaux d'un patient. Ainsi, l'assistance médicale peut agir avant une crise, ou réagir vite face à une alerte (cas d'une chute par exemple).

Le LAAS a pour sa part contribué depuis les années 90 au développement d'applications microsystèmes liées à l'industrie automobile [EJT02], à l'aide aux personnes âgées [CCL02], au confort thermique dans l'habitat [CSE03]. Dans ce projet de collaboration avec EDF R&D, c'est le domaine du Génie Civil qui est le nouveau champ d'application.

Nous allons dans le paragraphe suivant présenter l'état des techniques d'intégration microsystèmes qui sont actuellement accessibles.

1.2.4 Les techniques d'intégration microsystèmes

La voie d'intégration monolithique choisie par Analog Device (via le composant ADXL) n'est pas toujours possible et le plus souvent il faut se tourner vers des techniques d'assemblage Système in Package (SiP). Cet assemblage, au sens large, joue un rôle très important dans le domaine de l'électronique pour la miniaturisation des systèmes. Son enjeu est de taille et la conception non triviale lorsque l'on doit intégrer des systèmes hétérogènes.

D'une manière générale, l'intégration ou l'assemblage des systèmes hétérogènes reste délicat à réaliser car il faut faire en sorte que chaque composant garde ses caractéristiques nominales de fonctionnement compte tenu des interactions avec d'autres éléments du système. Pour cela, la conception doit tenir compte d'un certain nombre de facteurs qui contraignent la fabrication de systèmes hétérogènes :

- la compatibilité technologique (interaction entre les procédés de fabrication),
- la compatibilité fonctionnelle (compatibilité électromagnétique, thermique, réactivité chimique),
- la fiabilité et les contraintes thermomécaniques (coefficient d'expansion),
- les propriétés mécaniques spécifiques (déformation, frottement, herméticité).

En plus des contraintes d'assemblage courantes connues pour les circuits intégrés électroniques, les microsystèmes nécessitent souvent le développement de nouvelles pratiques d'encapsulation qui tiennent compte des spécificités MEMS (Micro ElectroMechanical Systems). Nous citerons comme exemple la création d'une cavité de dimension suffisante pour permettre le mouvement de parties mobiles d'actionneurs ou de structures mécaniques suspendues. D'autres contraintes propres aux MEMS peuvent être citées : protection antivibratoire pour les structures libres de silicium, protection thermique, protection optique, protection chimique, bio-compatibilité pour les systèmes utilisés en biochimie, pharmacie, alimentaire.

L'assemblage de systèmes complexes bénéficie d'un effort de miniaturisation surfacique et volumique présent à tous les niveaux, de l'échelle nanométrique jusqu'à l'échelle centimétrique. L'optimisation de l'espace se fait au niveau des structures micrométriques sur la puce de silicium puis au niveau de l'arrangement des puces de silicium dans les boîtiers de dimension millimétrique, puis à l'étape d'assemblage de ces circuits intégrés sur circuits imprimés centimétriques.

Nous allons présenter les techniques actuelles utilisées pour le conditionnement et l'intégration d'une puce de silicium nue dans un système. Nous aborderons l'intégration monolithique (System on Chip, SoC), l'intégration hybride (Système in Package, SiP) et l'intégration émergente qui propose de fonctionnaliser le support (Système on Package, SoP). La Figure 1-2 présente une illustration de ces techniques. Nous allons ensuite aborder brièvement la technique classique de montage de composants sous forme de boîtiers à la surface d'une carte de circuits imprimés. Les solutions présentées se différencient par l'organisation des éléments passifs (capacités, résistances, filtres, etc.) qui forment le système complet attendu.

Nous conclurons par les intérêts et les possibilités qu'ouvrent ces moyens actuels d'intégration pour la conception d'un système compact.

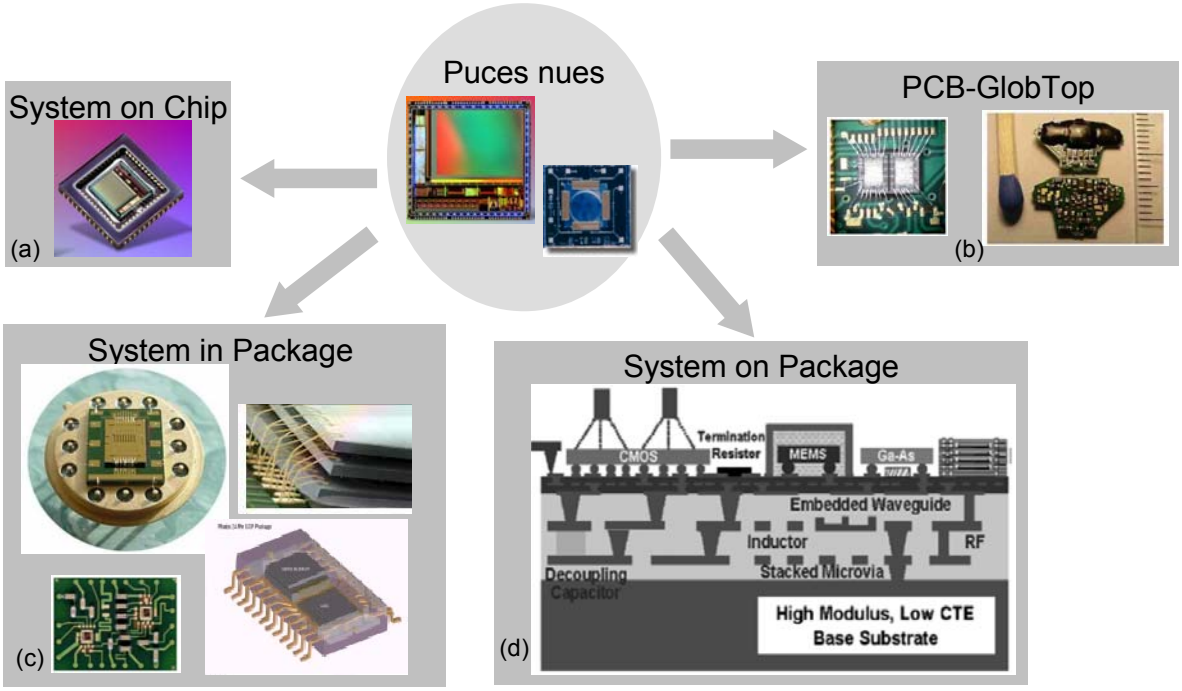


Figure 1-2 : Quatre techniques d'intégration de puces nues.

1.2.4.1 Intégration monolithique, Système sur Puce, SoC

Cette technologie d'assemblage Système sur Puce (System on Chip, SoC), illustrée par un capteur d'image Motorola MCM20014 (Figure 1-2(a)), est essentiellement utilisée pour la réalisation de systèmes dont la technique de fabrication est compatible avec la filière CMOS (Complementary Metal Oxide Semiconductor) de la microélectronique. Il s'agit de l'intégration, sur un même substrat de silicium de structures planaires de couches minces ou de structures volumiques de silicium réalisées par attaque chimique ou plasma. Ces techniques sont employées pour réaliser des fonctions analogiques, numériques, Radio-Fréquences, mécaniques ou optiques afin de produire des MEMS (Micro ElectroMechanical Systems), des MEMS RF (MEMS Radio-Fréquences), et des MOEMS (MEMS optiques). Cette intégration nécessite un grand investissement de conception et n'est viable que pour des systèmes produits en très grande quantité.

1.2.4.2 Intégration hybride « puces nues », SiP, SoP

Chip On Board (puce sur circuit, CoB)

La méthode, la plus commune d'interconnexion est le Chip on Board : la puce silicium est collée directement sur la carte de circuits imprimés, puis reliée à ce dernier par soudure de fils d'aluminium ou de fils d'or. Une résine d'encapsulation (Glop top) peut-être alors dispensée sur l'ensemble ou sur une partie du système pour garantir la protection vis à vis de l'environnement : contraintes thermiques et mécaniques. La Figure 1-2(b) [1] représente un exemple de réalisation d'assemblage CoB. La partie gauche de la figure représente une puce de silicium collée directement sur le circuit imprimé et la partie droite représente une puce enrobée dans une résine protectrice.

System in Package (SiP)

Il s'agit d'une méthode d'intégration bien maîtrisée aujourd'hui. Elle consiste en l'intégration de systèmes 2D dans un boîtier unique. Elle permet, lorsque la technique d'intégration monolithique SoC devient trop complexe et coûteuse, de regrouper à moindre coût, sur un même substrat organique, plusieurs circuits intégrés sur silicium. Ces puces de silicium peuvent être associées à des composants passifs, des capteurs et actionneurs. Les substrats à base de polymères souples ont apporté des fonctionnalités supplémentaires : la flexibilité mécanique et l'absorption de chocs

qui permettent la réalisation de modules plus ergonomiques et mieux adaptés aux applications portables.

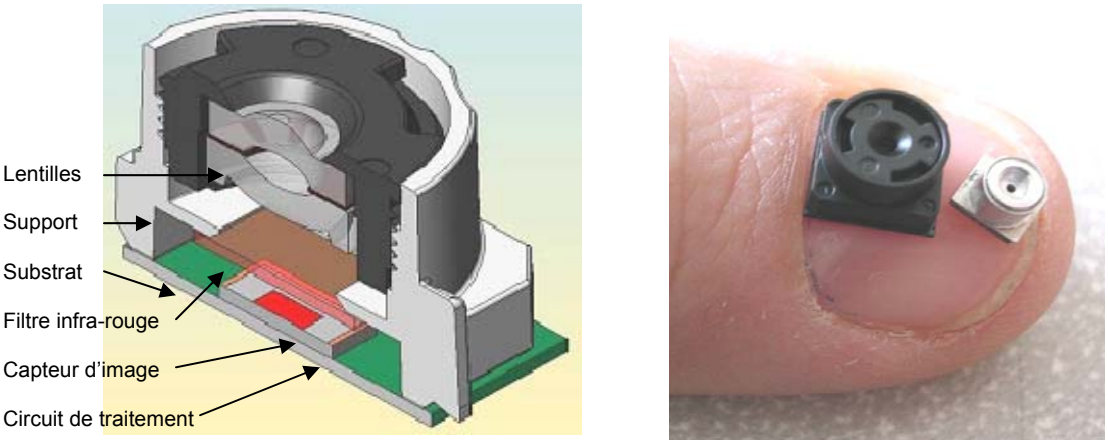


Figure 1-3 : Module d'une caméra intégrée, vue en coupe à gauche et vue du produit à droite. (*STMicroelectronics).*

La Figure 1-3 illustre l'utilisation de l'intégration SiP pour la réalisation d'une micro-caméra. Celle-ci est largement utilisée aujourd'hui dans les téléphones portables, les appareils de surveillance, les organiseurs personnels, certains jouets, etc. Nous voyons ici que les puces silicium du capteur d'image sont reportées sur un circuit de traitement qui est lui même reporté sur un substrat organique.

Multi Chip Module (module multi puces,MCM-V)

Lors de la mise en boîtier du composant silicium, il est possible d'interconnecter plusieurs composants dans un même boîtier, c'est l'intégration hybride multi-puces. Une fois mis sous boîtier standard, cet assemblage de puces de silicium peut être transporté sous la forme d'un sous-système compact.

L'intégration hybride vise l'utilisation de matériaux et composants d'origines technologiques très diverses. Le projet européen BARMINT [Est97], dirigé par le LAAS, a eu pour ambition d'identifier et de développer des méthodologies de conception, des outils et des technologies nécessaires à la fabrication de microsystèmes multifonctionnels : un démonstrateur a été réalisé (Figure 1-4) pour aborder les problèmes centraux de compatibilité de systèmes hétérogènes faisant intervenir plusieurs domaines technologiques. Il a associé des fonctionnalités optiques, mécaniques et chimiques pour envisager la résolution de problèmes que pose l'assemblage de ces diverses technologies. De plus, les techniques d'assemblage 3D et monolithiques, pour associer les parties MEMS et traitement électronique sur un même substrat de silicium, ont également été envisagées.

Le procédé consiste en un empilement de puces nues de silicium collées les unes aux autres et interconnectées entre elles. La densité d'assemblage est ainsi renforcée. Cette technique est appelée Multi Chip Module (MCM-V). La difficulté de l'empilement des puces est l'interconnexion des niveaux. L'assemblage BARMINT appliquait un procédé Thomson original basé sur l'enrobage en résine et des photolithographies latérales. La Figure 1-2(c), représente : en haut deux images d'un assemblage MCM-V, où les connexions électriques sont simplement filaires [1], en bas à droite une vue 3D d'un tel empilement monté dans un boîtier standard à monter en surface, et en bas à gauche un mini-circuit où les puces silicium sont associées à des éléments passifs avant la mise sous boîtier standard [2].

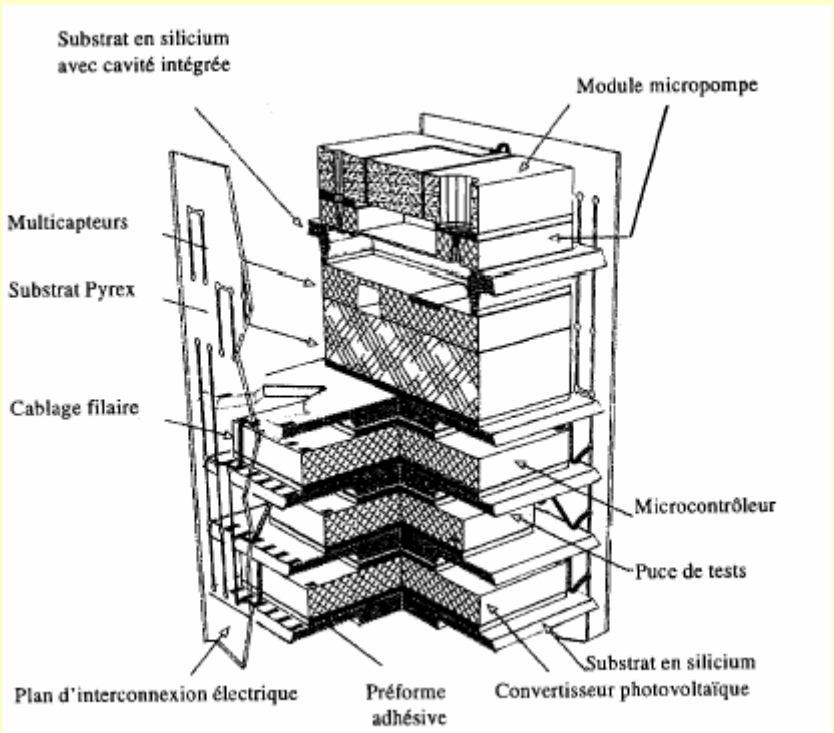


Figure 1-4 : Vue en coupe du prototype BARMINT.

Au delà de la démonstration de faisabilité technologique de microsystèmes compacts basés sur des assemblages 3D, des résultats importants ont aussi concerné l'intérêt de démarrer et de développer les méthodologies de conception de systèmes hétérogènes : outils CAD pour les microsystèmes, simulation 3D thermo-électrique et thermo-mécanique.

System on Package (SoP)

Le concept de SoP amène l'idée d'intégrer dans le volume du substrat, jusqu'alors utilisé essentiellement pour les connexions électriques, des fonctions passives telles que des capacités, des inductances, des filtres qui participent aux fonctionnalités générales du système. La Figure 1-2(d) [SRS04] illustre le concept d'un substrat intégrant les fonctionnalités RF et optique, à la surface duquel les puces sont connectées renversées. Comme l'assemblage MCM-V, cette technique est une intégration en 3D. Cette dimension supplémentaire est formée par l'empilement de couches de substrats organiques qui ont été fonctionnalisées au préalable.

Parmi les moyens de densification des assemblages 3D, nous trouvons deux techniques : l'une classiquement employée : la puce renversée (Flip Chip), et l'autre à l'état de recherche : les puces amincies.

Flip Chip (puce renversée, FC)

L'innovation qui a marqué une rupture dans les dimensions des boîtiers et la densité d'intégration est certainement celle de l'assemblage puce renversée. La puce de silicium est retournée et les plots de contacts de la puce sont directement reliés aux connexions du boîtier. Cette connexion ultracourte permet de limiter les pertes résistives et capacitives dues aux fils ce qui est remarquable dans le cas des applications très hautes fréquences. La Figure 1-5 [3] illustre une vue en coupe d'une puce silicium montée renversée (Flip-Chip) dans un boîtier BGA (Ball Grid Array) qui repose aussi sur le concept Flip Chip. Nous voyons qu'ici, l'intégration Flip Chip a été réalisée à deux niveaux : puce silicium et composant pour limiter au maximum l'encombrement des fils de connexion.

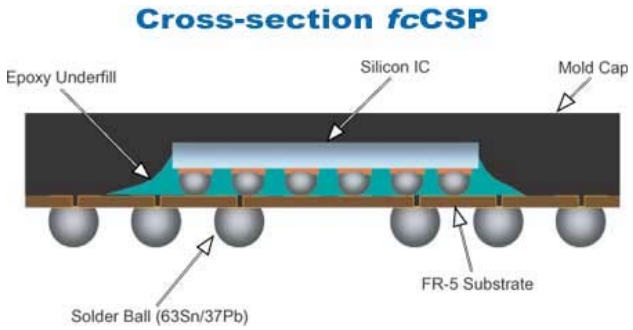


Figure 1-5 : Puce montée selon la technique Flip-Chip.

Intégration de structures amincies

L'assemblage de composants amincis est une technologie émergente présentant pour certaines applications de nombreux avantages : minimisation du poids et du volume, réduction de la puissance consommée, accroissement de la densité d'interconnexions.

Les technologies d'interconnexions en couches minces peuvent être appliquées aux assemblages électroniques pour des substrats amincis à quelques micromètres. La Figure 1-6 est une vue en coupe d'un assemblage de trois puces amincies sur un substrat de silicium.

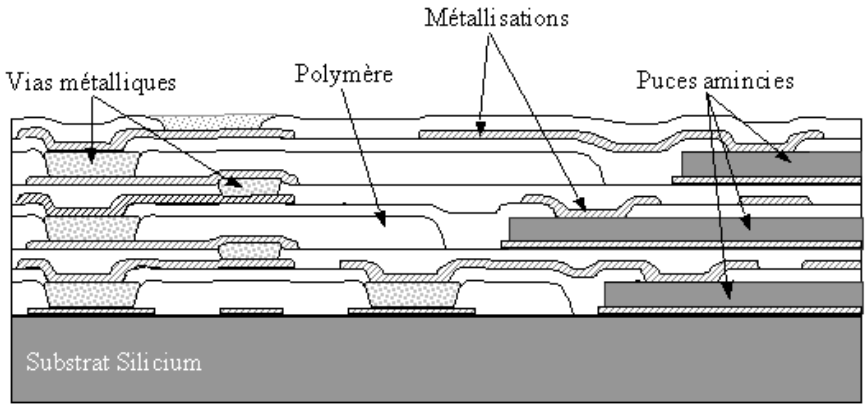


Figure 1-6 : Ultra Thin Chip Stacking.

Ces travaux ont été, en partie, effectués au LAAS dans le cadre de l'optimisation d'assemblages 3D de systèmes électroniques hybrides [PTB98].

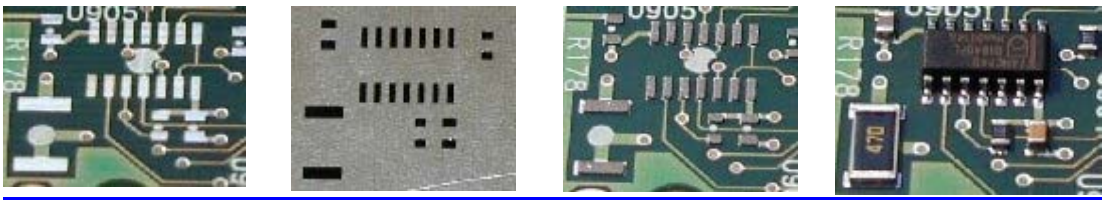
Ce paragraphe a montré que l'intégration est un travail à tous les niveaux d'échelles de dimensions pour arriver à optimiser la densité de fonctions d'un microsystème. Les technologies disponibles aujourd'hui sont multiples, nous avons vu le "Chip on Board", le "Multi Chip Module", le "Flip Chip", l'intégration de structures amincies et le "System on Chip". Chacune de ces techniques peut être considérée comme un outil pour réduire les dimensions d'un assemblage multipuces, ce qui donne la possibilité de densifier un assemblage compact en 3D.

1.2.4.3 Report CMS sur circuit imprimé

Au cours de ces dernières années, nous avons pu assister à un passage continuel de l'IMT (Insert Mount Technology) qui est le montage classique des composants par insertion sur les circuits imprimés, à la SMT (Surface Mount Technology) ou CMS (Composants Montés en Surface) qui est un montage des composants sur la surface du circuit imprimé.

Il est maintenant courant d'utiliser des circuits imprimés comportant une vingtaine de couches superposées de pistes conductrices pour une épaisseur totale de seulement 2,5 mm [4]. Cette architecture a été développée parallèlement aux boîtiers BGA (Ball Grid Array) qui ont plusieurs dizaines de connexions par centimètre carré. Le composant commercialisé par Synetics ayant 1232 contacts sur 4 cm² [5] est un exemple de la densité de connexions possibles sur un circuit imprimé.

L'avantage d'un CMS (Composant Monté en Surface) est qu'il permet de densifier l'implantation sur un circuit imprimé puisque l'on peut implanter des composants sur chacune de ses faces. Toutefois, il nécessite un outillage très spécialisé pour sa mise en œuvre, qui est plus contraignant et complexe que l'équipement utilisé pour les composants qui traversent le circuit imprimé. La soudure est posée par sérigraphie aux points de contact des composants, la Figure 1-7 montre les principales étapes de montage d'un composant CMS.



Circuit imprimé | Masque | Pâte à souder déposée | Composant après refusion

Figure 1-7 : Etapes de montage d'un composant CMS.

1.2.4.4 Notre positionnement

Les méthodes d'intégration exposées dans les paragraphes précédents traitent pour la plupart de systèmes électroniques complexes. La technologie SoC permet de réaliser de manière très compacte un système sur un seul substrat de silicium. Ce gain en compacité a un surcoût non négligeable lors de la phase de conception, puisque l'augmentation de la complexité du système nécessite plus de travail de conception en amont, comme des études de fiabilité et de compatibilité des circuits mixtes intégrés sur puce. Cette longue période de conception ralentit de fait la mise en vente du produit sur le marché. Aussi, c'est un produit qui reste difficilement modifiable une fois fabriqué. Cependant, dans une optique de production de masse, le prix unitaire de chaque composant diminue rapidement et devient alors rentable.

La problématique qui nous concerne est celle des systèmes hétérogènes qui consistent en un regroupement de fonctions de type logiques, analogiques, mécaniques, fluidiques, optiques, etc. Pour de tels systèmes hétérogènes, l'intégration sur silicium uniquement n'est plus envisageable, il faut se tourner vers l'intégration de type System on Package (SoP) qui implique l'utilisation de plusieurs types de substrats. L'avantage de l'assemblage SoP [TM99] est qu'il est basé sur une vision système de l'intégration, par l'agrégation de composants élémentaires simples et bien maîtrisés. Cette vision donne un double avantage : d'une part, une plus grande flexibilité pour envisager de faire évoluer les fonctionnalités d'un produit fini, par l'utilisation de bibliothèques de composants, et d'autre part, une réduction du temps de mise sur le marché du système dans sa version finale.

Cette technique sera mise en œuvre et adaptée à la réalisation de notre dispositif de surveillance multicapteurs communicant.

1.3 Les modes de conception microsystèmes

Nous venons de voir qu'il existe une grande diversité de composants et de modes d'assemblages qu'il conviendra de choisir durant la période de conception. Cette étape de conception part des spécifications du système ou du microsystème à réaliser jusqu'au lancement de la fabrication du prototype matériel. Les microsystèmes apportent une contrainte supplémentaire à la conception des systèmes électroniques standard qui est celle de la pluridisciplinarité. En effet, cette activité de conception met en relation un grand nombre d'acteurs, d'outils, de méthodes appartenant à une grande variété de domaines d'activités pour regrouper le savoir et arriver à concevoir juste et à temps. Dans ce paragraphe, nous allons exposer les motivations qui justifient le bien fondé d'une conception microsystème. Nous présentons également le cycle de développement en « V » dans lequel s'intègre la plate-forme de conception HiLeS.

1.3.1 Les motivations scientifiques

Les approches actuelles proposent une méthodologie descendante de conception passant par une étape intermédiaire de prototypage virtuel où le système est complètement représenté par un assemblage de modèles sur lesquels on pourra analyser le fonctionnement : faire des optimisations, des choix technologiques, des prédictions de performances sur la fiabilité ou la robustesse [SNP04].

Un microsystème est une entité fondamentalement multifonctionnelle et pluridisciplinaire : il pose des problèmes de représentation encore non résolues. Pour cette raison, nous avons choisi de donner à notre travail un double objectif :

- celui d'identifier les méthodes et les outils d'une conception efficace des microsystèmes,
- celui de concevoir un microsystème en appliquant ces méthodes et ces outils pour la validation et trouver les chemins d'amélioration. Dans ce chapitre introductif, nous allons brièvement situer la problématique et décrire les premières options choisies.

Les microsystèmes entrent dans une phase de maturité industrielle. Nous pensons qu'un microsystème pour une application de surveillance est constitué d'une base d'organes standards : capteurs, traitement du signal, processeur, communication qui a souvent été décrite dans la littérature [WN91], [CGK03]. Il y a donc un intérêt à imaginer une méthode assortie d'un ensemble d'outils de conception qui puisse aider le concepteur à rapidement établir les principaux paramètres qui caractérisent une application cible, tout en gardant cette trame standard.

La conception des systèmes électroniques propose aujourd'hui une grande variété d'outils qui permettent de concevoir et de simuler un système numérique ou mixte, etc. Chaque outil est adapté à un niveau d'abstraction choisi et à un domaine ciblé. Ces outils utilisent une variété de langages de conception MAST, VHDL-AMS, VHDL, C. Notre démarche tente de rapprocher les spécifications et les premières simulations fonctionnelles.

Dans le cadre d'une démarche de développement de systèmes et microsystèmes, le LAAS a lancé un projet coopératif MOCAS (« Méthodes et Outils de la Conception « Amont » des Systèmes et Microsystèmes) qui vise à regrouper les compétences développées au sein du laboratoire dans le domaine des outils et des méthodes associées à la robotique, l'automatique, la microélectronique, l'informatique et les microsystèmes. Les contributions sont diverses et concernent les domaines de la modélisation, la simulation, l'optimisation et les technologies dans un objectif d'accompagnement industriel ou de systèmes innovants. Nos travaux s'intègrent dans cette action et apportent comme illustration de cette démarche, la mise en œuvre de ces outils de conception pour la réalisation d'un microsystème de mesures communicant pour une application Génie Civil.

1.3.2 La conception descendante selon le cycle en V

Le modèle de cycle de développement dépend des informations à traiter, des connaissances, de la culture de l'entreprise, et des objectifs visés. En effet, certains cycles souhaitent favoriser l'innovation, d'autres favorisent la capitalisation et la réutilisation d'expériences. La représentation du cycle de développement la plus couramment utilisée en entreprise est celle du cycle en V. Il est représenté par la Figure 1-8. C'est un modèle couramment appliqué dans l'industrie classique. Il permet l'organisation générale du travail, la décomposition et la distribution de tâches. Sa structure en « V » met en regard des étapes de spécification/validation, conception/intégration et conception/tests autour du point rebond qui est le prototype virtuel. Cette structure offre une voie pour vérifier la conformité et valider des étapes clés dans le développement d'un produit. Ces étapes jalonnent le suivi d'un projet et garantissent une certaine qualité. D'une manière générale, ce cycle convient au développement de systèmes complexes dans lesquels interagissent un grand nombre de collaborateurs. En effet, chaque collaborateur, peut réutiliser cette même structure d'analyse en « V » pour décomposer le développement de son propre sous-système en des étapes du même type.

Cependant, deux inconvénients peuvent apparaître à l'usage de ce cycle. D'une part, ce type de développement, dans le cadre de l'ingénierie simultanée, n'apporte pas d'indications pour que

les parties prenantes d'un projet communiquent et échangent leurs travaux. Et d'autre part, dans le cas d'un besoin de changement de spécification en cours de projet, les modifications à envisager ne peuvent pas être facilement explicitées, ce qui mène souvent à redémarrer le cycle de développement depuis le début.

Dans le cadre de notre étude, nous serons donc attentif à, d'une part, apporter les outils pour faciliter les échanges entre les différents partenaires d'un projet, et d'autre part, rendre la conception plus réactive aux changements de spécification.

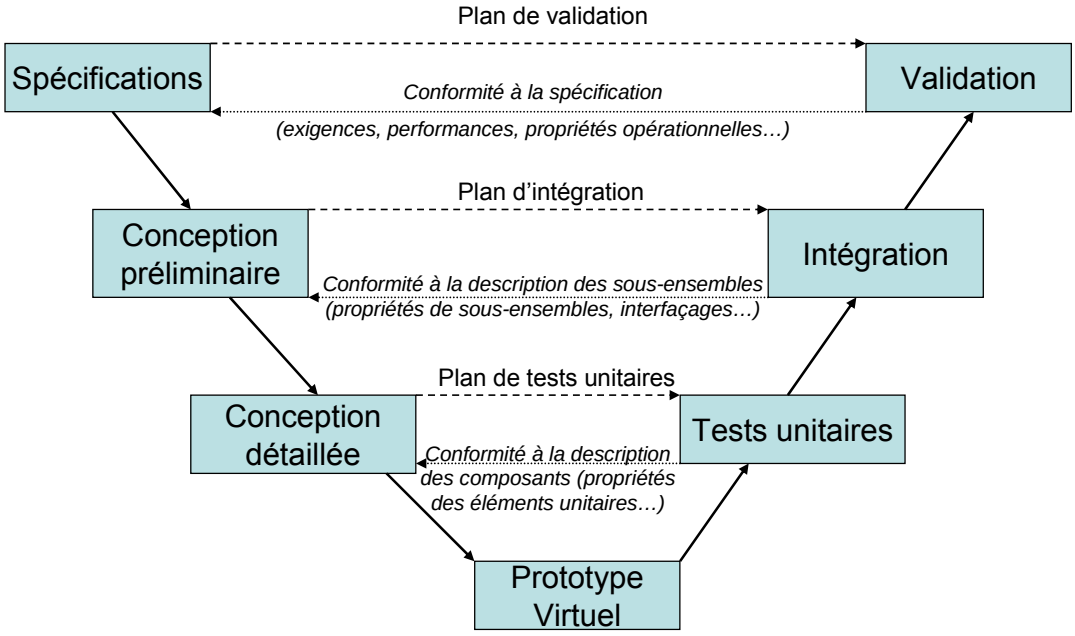


Figure 1-8 : Cycle en « V » applicable aux différents niveaux de décomposition du système.

A l'intérieur même d'un cycle de développement en V, la conception peut être vue sous deux angles : conception ascendante et descendante.

- La conception ascendante

Les approches de conception ont souvent été dans le passé de nature ascendante (Bottom-up) ou propre à la situation (Ad-hoc). Dans cette approche, la démarche se focalise sur le composant élémentaire, constituant l'innovation. Puis, une fois ces parties micro-mécaniques, micro-optiques, etc. optimisées, une électronique environnante y est rapportée. Les inconvénients associés à la méthodologie ascendante sont listés par [Kun00]. Il présente notamment comme défaut : le manque d'une vue architecturale du système et son optimisation, une nécessité d'effort important et un temps de calcul trop long pour une simulation du système à haut niveau.

- La conception descendante

Cette méthode est complémentaire de la précédente : elle s'efforce de rassembler au plus tôt toutes les données utiles à la conception et d'en faire une représentation amont aussi complète que possible. Cette représentation est ensuite décomposée en sous-systèmes qui seront ensuite décomposés à leur tour jusqu'à obtenir des modules simples. La conception se divise en trois phases principales :

- Les spécifications : Cette phase consiste à regrouper toutes les données concernant la description du système. Elles doivent apporter les informations relatives au fonctionnement, aux performances attendues, aux contraintes ou à toute autre forme d'information ou de recommandation non fonctionnelle telle que le coût, le poids, le volume, le délai de livraison, etc.

- Le modèle architectural du système global : Cette étape vise à établir une première base sur laquelle sera fondée le système. Le modèle est défini selon des fonctions décomposées en autant de sous-fonctions nécessaires pour obtenir des fonctions élémentaires exploitables.

- Le choix des composants et des technologies pour le prototypage virtuel : Cette phase constitue une étape de vérification qui consiste à faire une représentation informatique du système à réaliser. Ce prototype sera le moyen de tester, d'optimiser et de valider le système sans recourir à plusieurs prototypes réels.

L'étape suivante : l'intégration, est considérée comme postérieure à la phase de conception.

- Notre approche : une conception descendante intégrant des composants consignés

Nous avons vu que les microsystèmes ont une architecture qui se standardise, ce qui n'est pas pour autant opposé à l'augmentation de leur complexité. Or, comment gérer cette complexité croissante quand le marché demande une évolution constante des produits et ceci dans un temps très limité ? Nous décrivons ici le besoin d'une démarche de conception globalement descendante qui puisse être initiée dès les premiers temps du développement, c'est à dire lorsque les spécifications du système sont écrites. Cette approche de conception, très en amont de la réalisation du système physique, doit être utilisée pour réaliser des analyses du système, explorer différentes solutions architecturales, valider pas à pas la démarche de création du système.

La conception descendante pourrait viser un découpage du système en plusieurs sous-modules dès l'initiation du projet où chaque module concernerait un domaine spécifique, tel que l'électronique analogique, l'optique, la mécanique, et serait confié à une équipe de développement spécialiste du domaine. Puisque chaque domaine a ses propres outils de simulation, de vérification, et de validation, cette voie permet de créer des conceptions expertes pour chaque domaine. En pratique, elle limite aussi les possibilités d'échanges entre ces branches de conception parallèles. De plus, elle diminue les possibilités de vérification inter-domaines pour établir les premières étapes de validation du système complet.

Cette première voie, qui établit un premier choix très tôt dans la démarche, est adoptée par les concepteurs qui ont une vision experte de leur domaine, des technologies et des familles de produits.

Mais, nous savons que les technologies deviennent très hétérogènes et en constante évolution. Des nouveaux métiers pluridisciplinaires se créent. Par exemple, pour les domaines technologiques qui sont en phase de pérennisation comme les microtechnologies, les bases de données concernant leurs propriétés, leurs limites et leur cadre d'utilisation doivent se bâtir peu à peu. C'est pourquoi, dans une approche métier, dès lors que ces technologies de base sont paramétrées, répertoriées et classées, nous pouvons avoir une vision globale du système, sous la forme de « boîte noire » maîtrisable à haut niveau. Dans ce contexte, nous devons envisager d'effectuer un découpage fonctionnel du système en sous-système et identifier quelle « boîte noire » peut y apporter une solution.

En résumé, nous voulons considérer la conception comme une procédure permettant d'aller des spécifications du système à un partitionnement des tâches selon des métiers bien répertoriés, sachant que ces métiers peuvent être différents des découpages habituels en discipline. Ils vont comporter le plus souvent des fonctions pluridisciplinaires : on appellera par exemple métier = la mesure, qui va associer des fonctions, par exemple mécanique et électronique (électronique embarquée). Cette part de conception amont, associée au partitionnement métier, est illustrée sur la Figure 1-9, sur la base d'un cycle de développement en V.

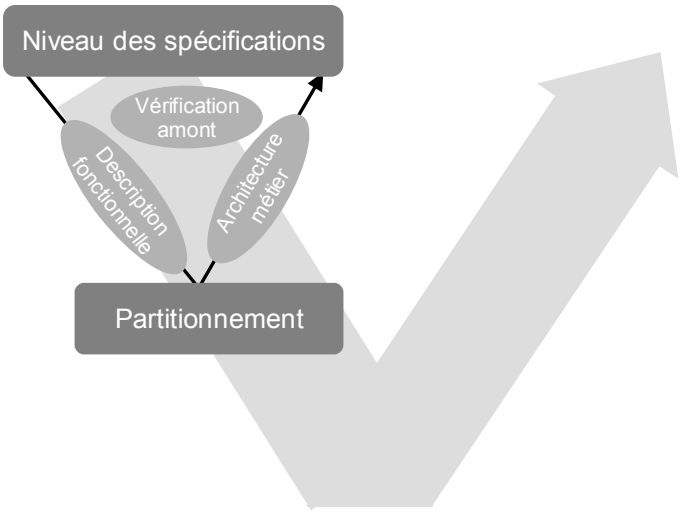


Figure 1-9 : Cycle en « V » et partitionnement.

Dans cette procédure, les spécifications doivent être la référence pour tous. Sur cette base, nous allons défendre l'idée de chercher, d'une part une décomposition fonctionnelle du système et d'autre part, à intégrer ces fonctions dans un découpage métier que l'on voit déjà se dessiner progressivement dans la standardisation des architectures comme nous l'avons signalé sur la Figure 1-1.

Dans cette démarche, nous proposons de réaliser cette segmentation seulement après avoir pu valider la description fonctionnelle du système. Ce concept de vérification au niveau du système est encore nouveau et pas encore bien cerné pour les applications microsystèmes [VD05].

1.3.3 La plate-forme de conception HiLeS

La plate-forme HiLeS est développée au LAAS depuis le début des années 2000 [Har00], [Jim00], [Ham05]. Elle s'inscrit dans une volonté collective, de spécifier et développer une base d'outils d'aide à la conception amont des microsystèmes.

Cette plate-forme s'intègre dans le cycle de développement en « V » tel que nous l'avons introduit. Dans la Figure 1-10 nous retrouvons la contribution de la plate-forme dans la zone entourée par des pointillés. Son positionnement central sur la branche descendante du « V » confère au concepteur un outil qui lui permet un lien étroit avec la partie plus amont des spécifications et la partie aval des domaines de conception spécialisés. L'objectif est qu'il puisse initier, distribuer et réguler les solutions, sous la forme de « prototypes virtuels » ainsi testés, aux équipes responsables de l'intégration. Nous voyons ici qu'à l'inverse de la démarche de conception Top-Down actuelle, décrite dans le paragraphe 1.3.2, cette démarche intègre un premier niveau de validation globale du microsystème avant de distribuer les modèles attendus concernant chaque spécialité de métier (électronique, mécanique, optique, thermique, etc.).

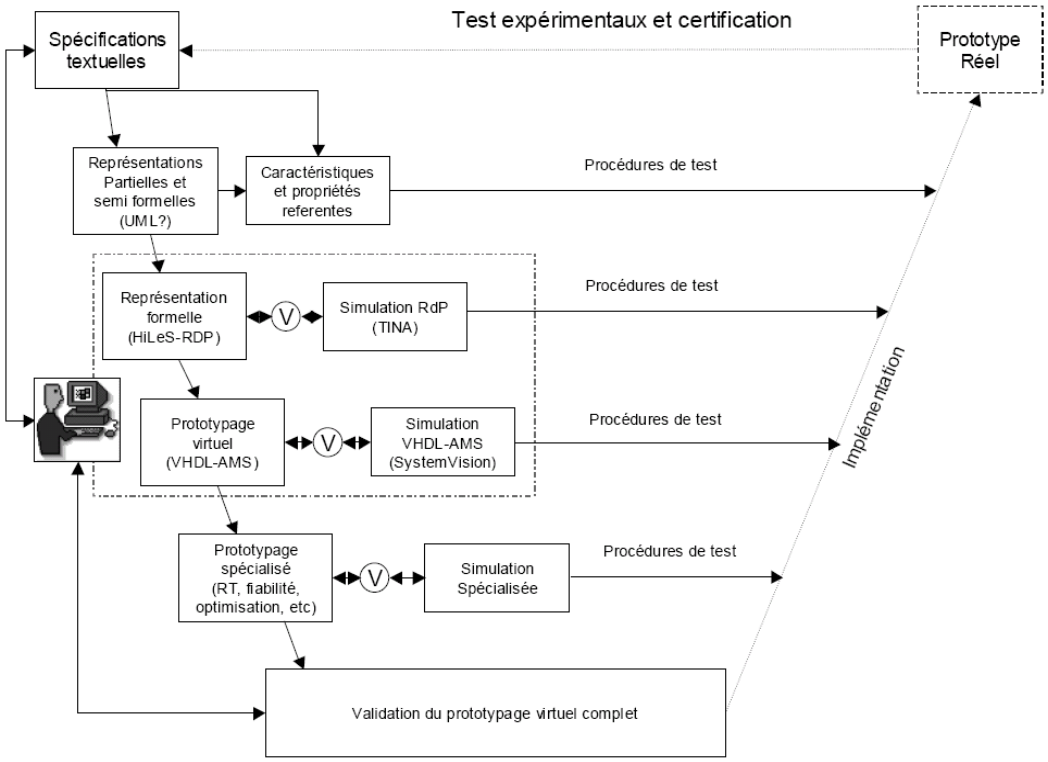


Figure 1-10 : La plate-forme de conception dans un cycle de développement en «V» [Mocas].

Les propriétés attendues de cette plate-forme sont celles de :

- générer une première représentation des spécifications (fonctionnalités, procédures, réseaux de commande) pour :
  - aider à l'établissement de l'architecture du système et à sa vérification,
  - aider à la temporisation logique de cette architecture et à sa vérification,
- simuler le système ainsi modélisé à haut niveau,
- aider les choix architecturaux,
- aider à faire une décomposition du système en sous-systèmes définissant des tâches distinctes (choix technologiques) applicables à la construction d'un plan de travail, pour l'intégration du système,
- proposer un prototype virtuel afin de vérifier, optimiser et valider le fonctionnement attendu du système.

Pour cela, la plate-forme HiLeS a adopté deux formalismes permettant d'avancer sur la voie de la vérification haut niveau et la simulation. Le premier formalisme est celui des réseaux de Petri qui permettent de modéliser des réseaux de commandes et des procédures complexes de fonctionnement du système, par le biais du logiciel TINA (TIme Petri Net Analyser) développé par le groupe OLC (Outils Logiciels pour la Communication) du LAAS-CNRS [BRV03]. Le deuxième est celui du langage VHDL-AMS qui assure la portabilité des modèles ainsi réalisés et apporte l'avantage de pouvoir couvrir la modélisation de systèmes multidisciplinaires. En effet, le langage VHDL-AMS a été choisi puisqu'il permet de décrire des systèmes pluridisciplinaires, grâce à l'application des lois généralisées de Kirchhoff. Cette propriété nous permet d'envisager la modélisation complète de microsystèmes complexes qui intègrent aujourd'hui des sous-ensembles optiques, mécaniques, ou fluidiques dits « passifs » conjointement à des sous-ensembles électroniques analogiques, numériques qui lui confèrent une « intelligence ».

Dans la conception système, le langage central VHDL-AMS, a permis le développement d'outils encore incomplets mais déjà diffusés commercialement, sur lesquels le LAAS se positionne en utilisateur. Par contre, le LAAS va être fournisseur de Recherche en apportant à ce nœud central

de conception : les modèles, les architectures, les algorithmes (commande, optimisation), les réseaux (communication, énergie), les règles de fabrication et d'assemblage,, etc.

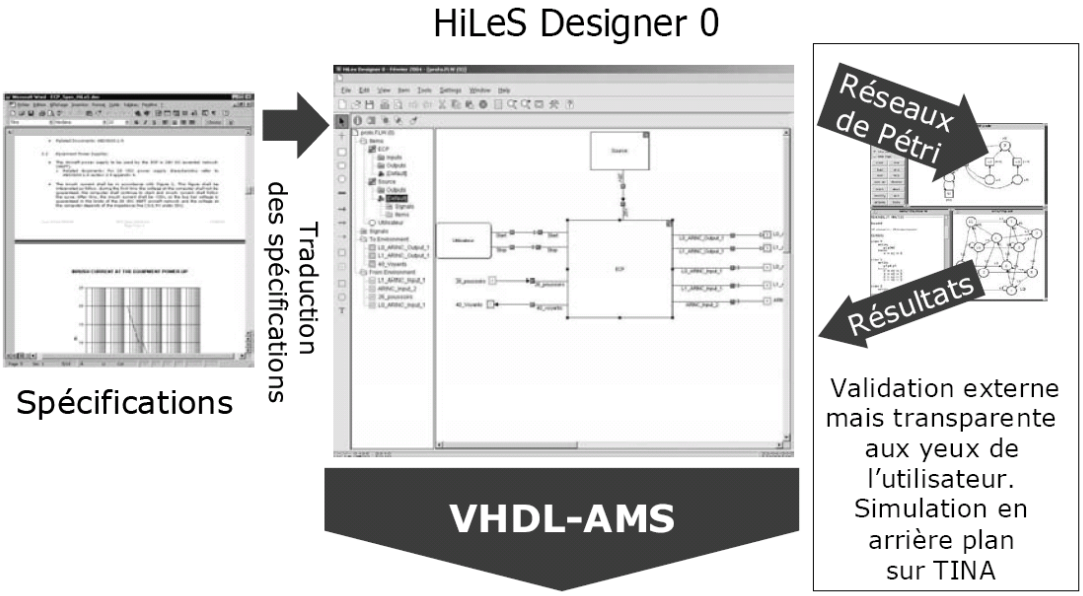


Figure 1-11: Diagramme général du concept de la plate-forme HiLeS Designer 0 [Ham05].

Dans la Figure 1-11, nous représentons le concept général de la plate-forme. L'entrée principale est la source d'information provenant du maître d'ouvrage à l'attention du maître d'œuvre qui se doit de produire un système. Le document d'entrée est généralement présenté sous la forme de spécifications textuelles. Nous verrons dans le chapitre suivant comment cette entrée peut être adaptée et structurée de manière à faciliter la représentation du système sous HiLeS. L'outil HiLeS aide le maître d'œuvre à générer une première représentation du système écrit en VHDL-AMS.

1.4 Notre projet

Notre projet vise la conception et le **prototypage d'un détecteur de vieillissement des bétons en Génie Civil. Nous l'avons appelé SmartGEC**. « Smart » pour d'une part, sa capacité à mémoriser et à traiter les informations provenant des paramètres physiques de son environnement, et d'autre part, sa capacité à les restituer sous une forme exploitable à une centrale de base distante, « Gec » pour rappeler son orientation vers une application Génie Civil. Ce projet a été proposé par EDF R&D pour des besoins internes.

Le cahier des charges de notre système peut se résumer par la spécification des caractéristiques suivantes :

- Mesure des variables d'environnement (Température ambiante, Humidité, Pression atmosphérique),
- Mesure d'une caractéristique de l'état de la surface d'un béton (Micro-déplacement).
- Mesure des variables d'environnement périodiquement (de 1 à 5 fois par jour).
- Transmission des données mesurées périodiquement vers une centrale de base déportée (1 fois par semaine)
- Réalisation de mesures supplémentaires sur demande de l'utilisateur.
- Emission d'une alerte, dans le cas où les paramètres dépassent un seuil.
- Autonome en énergie.

La mesure de micro-déplacement en surface du béton présente une difficulté liée à l'hétérogénéité du matériau. En effet, le béton, ou il faudrait plutôt parler des bétons tant leur composition est variée, sont une association de gravier, de sable, de ciment et de divers adjuvants.

Ces composants ont une dimension qui varie sur plusieurs échelles. Du centimètre pour les graviers au micromètre pour les adjuvants. Il est donc nécessaire de considérer les notions d'échelles pour étudier le comportement mécanique du béton. Le béton est un matériau « quasi fragile » qui se déforme par microfissurations. Pratiquement, il faudra différencier l'analyse locale des fissures discrètes, à l'analyse plus globale de leurs effets sur une longueur choisie. Nous nous plaçons dans le deuxième cas de figure où nous mesurerons la déformation du béton sur une distance choisie.

1.4.1 Le contexte de l'étude

L'application qui nous concerne pose plus généralement la problématique de l'usage des microsystèmes dans le Génie Civil. Quel peut être leur apport ?

Les questions posées par le domaine du Génie Civil sont multiples. Nous sommes plutôt concernés par le **vieillissement des structures de béton armé** [ATU04]. Sans entrer dans le détail de toutes les études menées dans ce domaine, le béton est un matériau « vivant » qui évolue tout au long de sa vie. Cette évolution des caractéristiques mécaniques du béton peut être due à deux phénomènes :

- d'une part, dans les premiers jours après la première coulée apparaissent la plupart des transformations chimiques qui influencent les propriétés mécaniques intrinsèques du matériau,
- et d'autre part, tout au long de leur vie, les structures de béton sont sujettes aux vibrations, aux déformations et à l'atmosphère environnante (humidité, température), ce qui provoque aussi des efforts mécaniques dans le béton. Cette vision duale de l'évolution des bétons permet de réaliser l'intérêt d'apporter des moyens de surveillance adaptés aux constructions Génie Civil sur les deux plans de vieillissement.

Le premier point concerne le jeune âge du béton. Cette période s'étend au delà de la prise du béton et jusqu'à ce qu'il atteigne 50% de sa résistance à 28 jours (ce qui correspond à 2 ou 3 jours). Ce premier plan de vieillissement est actuellement majoritairement traité par des systèmes de mesure par corde vibrante et par des systèmes optiques basés sur les fibres optiques. Ces deux catégories de systèmes sont noyées au cœur de la coulée pour donner une information sur les contraintes internes du matériau.

En effet, le retrait de séchage dépend de l'humidité relative présente aux limites de l'élément considéré. Il est d'autant plus fort que la quantité d'eau libre dans la pâte est plus grande. Le retrait endogène, lui découle de la composition de la pâte. La mesure de retrait s'effectue de façon volumétrique ce qui est non applicable en pratique. Le retrait d'autodessication de mortier est mesuré de manière linéique sur des éprouvettes retirées du moule 24h après mûrissement, couvert d'une pellicule imperméable. La technique de mesure consiste simplement à mesurer le déplacement relatif de 2 plots de références fixés sur la face latérale ou aux extrémités de l'éprouvette à différentes échéances de temps. Le temps zéro correspond à la fin de l'opération de scellement (les premières 24h sont négligées).

Le deuxième point concerne la vie à plus long terme du béton. Il est traité, d'une part, sur le plan du vieillissement dû aux vibrations, par des systèmes de mesure équipés de microcapteurs accéléromètres, et d'autre part, en ce qui concerne les déformations, par des mesures d'écartement par visée Laser ou par des jauges de contraintes de surface. En effet, le paramètre qui compte à l'usage, c'est l'ouverture des fissures, pour des raisons esthétiques mais surtout pour des raisons de durabilité de l'ouvrage. Il faut savoir que la peau du béton est toujours fissurée. La fissure sera invisible si elle est inférieure à 20µm, et jusqu'à 300µm elle sera considérée comme inoffensive. Au dessous de cette taille, les tensions superficielles sont supérieures aux forces gravitationnelles et empêchent tout mouvement d'eau en phase liquide. Si bien que l'eau qui a pu y pénétrer par capillarité ou par condensation ne peut en ressortir que par évaporation et par conséquent sans déplacer les ions. C'est donc cette ouverture maximale de fissure qui garantit en partie la durabilité des ouvrages en béton armé.

Notre objectif est ici de proposer un système de mesure pour le Génie Civil intégrant les dernières évolutions en matière d'intégration microsystème et répondant à un besoin de suivi des caractéristiques mécaniques d'une paroi de béton. Nous voyons qu'un tel système intéresse à la fois la réalisation d'essais menés en laboratoire pour l'étude du béton au jeune âge mais aussi et

surtout la surveillance des ouvrages de béton armé en cours de vieillissement. Nous allons donc, rappeler les pratiques de mesures actuelles dans les gros œuvres et voir comment nous avons spécifié notre dispositif en conséquence.

1.4.2 Les capteurs utilisés en Génie Civil

Nous proposons de classer les capteurs utilisés en Génie Civil suivant deux catégories : une première catégorie qui regroupe les capteurs installés avant la coulée du béton sur les armatures de renfort. Ils sont donc noyés dans la masse du béton tout au long de leur vie. Une deuxième catégorie regroupe tous les capteurs qui peuvent être posés après la coulée du béton, à sa surface. Ces derniers sont utilisés pour la mesure de déformation de joints (fissuromètres), la mesure de déformation sous sollicitations (jauges de contraintes), ou la mesure de micro-déplacements (extensomètres).

Nous évoquons ci-dessous les 3 principes de mesure les plus couramment employés dans ce domaine :

- Corde vibrante :

Cette mesure est basée sur l'utilisation d'une corde tendue entre deux points d'ancrage (Figure 1-12). Si cette corde est excitée par un champ électromagnétique, elle oscillera à sa fréquence propre de vibration. Toute variation de sa fréquence fondamentale témoigne d'une variation de tension. C'est en effet, cette variation de fréquence propre de vibration qui sera fonction du déplacement relatif de ces deux points d'ancrage.

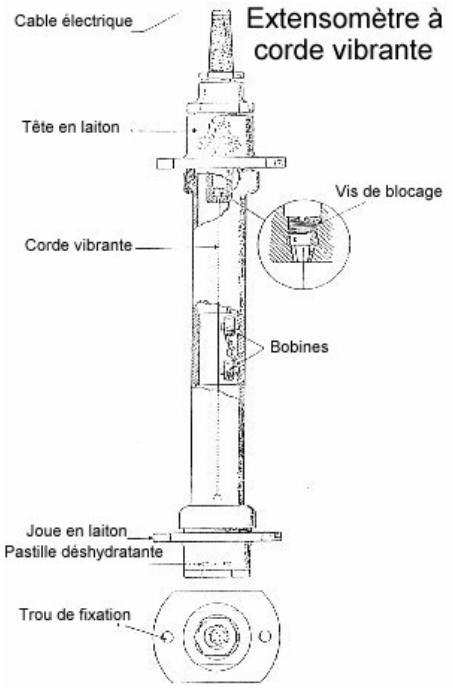


Figure 1-12 : Schéma d'un extensomètre à corde vibrante.

- Fibre optique :

La fibre optique est utilisée comme matériau de base pour évaluer les contraintes ou les micro-déplacements dans les structures. Les capteurs à fibre optique peuvent être divisés en deux familles : d'une part, les capteurs à modulation de phase, et d'autre part, les capteurs à modulation d'intensité. Nous nous intéressons ici à la première famille qui présente la plus grande précision de mesure de déplacement. Il s'agit de mesures d'interférométrie réalisées dans une cavité de Fabry-Pérot (Figure 1-13). Cette cavité est située au bout d'une longue fibre optique qui amène le signal lumineux. La déformation de cette cavité module la phase du signal réfléchi.

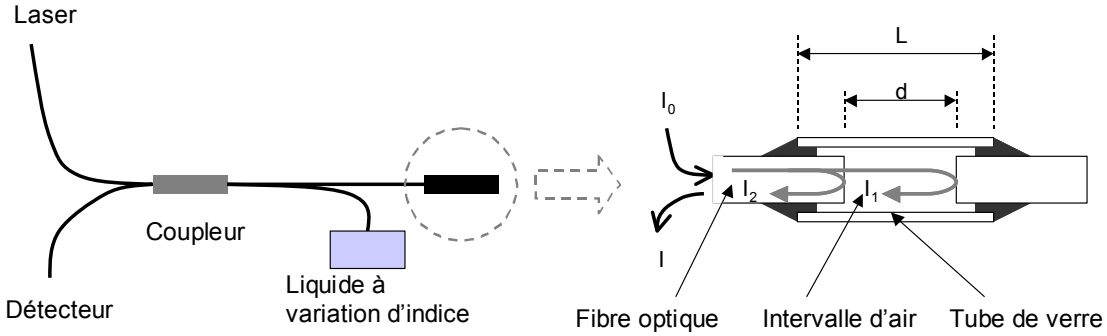


Figure 1-13 : Vue en section d'une cavité Fabry-Pérot.

Ce type de capteur peut être utilisé, soit collé en surface d'armatures métalliques coulés dans le béton pour évaluer leurs états de précontrainte, soit directement coulé dans le béton après avoir été enveloppé d'un conditionnement adapté (Figure 1-14).

Figure 1-14 : Jauge de Fabry-Pérot à immerger dans le béton.

- Linear Variable Differential Transducer (LVDT)

Le principe du LVDT est basé sur la mesure de la variation d'un champ magnétique transféré entre deux bobinages (Figure 1-15). Cette variation est provoquée par le déplacement d'un noyau mobile. Ces capteurs sont généralement utilisés accompagnés d'un système intégrateur qui permet de reporter le déplacement d'un point déporté sur le capteur. Cet intégrateur est donc scellé en deux points sur la structure à mesurer qui correspondent au point déporté et au point où est situé le capteur.

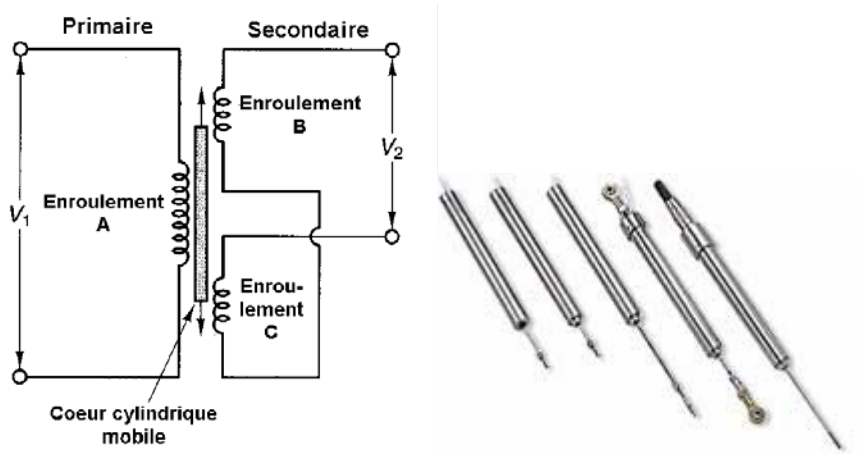


Figure 1-15 : Vues fonctionnelle et physique du LVDT.

Les précisions les plus fréquemment rencontrées pour ces capteurs sont reportées dans le Tableau 1-3.

Type de mesure	Corde vibrante	Fibre optique (cavité Fabry-Pérot)	LVDT
Précision en % de la pleine échelle	0,25	0,1	0,2
Résolution	10µm	1µm	1µm
Coût	300-600 €	600-700 €	600-700 €

Tableau 1-3 : Principales caractéristiques des capteurs de déformation couramment employés en Génie Civil.

Ces valeurs sont tirées de catalogues de produits commercialisés pour des applications Génie Civil [6]. Nous constatons que la précision et la résolution de la mesure par corde vibrante semblent en deçà des deux autres. Cependant, les résolutions dépendent aussi largement de l'électronique de traitement du signal associée au capteur. Il est donc difficile d'établir un classement sûr. De ce point de vue, les capteurs, qui ont été noyés il y a une dizaine d'années dans de grands ouvrages et dont l'électronique était obsolète mais déportée sur un poste central, ont pu être améliorés par la mise à jour de la partie traitement du signal.

Figure 1-16 : Capteurs pour la surveillance de monuments historiques.

Aujourd'hui, nous remarquons que la plupart des capteurs installés sur les ouvrages de Génie Civil sont câblés, au sens de la transmission d'énergie et de la transmission des données mesurées. L'exemple que nous présentons sur la Figure 1-16 concerne une installation mise en place par l'entreprise SITES [7] pour la surveillance de monuments historiques. Le capteur est installé sur un arc de voûte présentant une fissure.

Cependant, les grands ouvrages les plus récents bénéficient quelques fois des nouvelles technologies sans fil pour la communication de données. Il est donc intéressant de proposer un système sans fil et complètement autonome énergétiquement.

1.4.3 L'établissement du cahier des charges par les méthodes d'analyse fonctionnelle du besoin et des analogies

Le cahier des charges est le document de référence d'un projet. Il doit tout rassembler : les motivations, les idées, les exigences, les conditions d'utilisation et d'environnement, etc. Le cahier des charges a plusieurs volets :

- La description fonctionnelle du produit.
- La description physique du produit.
- Les exigences spécifiques du produit.

Nous nous sommes essentiellement intéressés, dans la démarche, aux aspects fonctionnels qui seront détaillés dans les prochains paragraphes.

Mais pour aller jusqu'au bout du prototype, il faut disposer de recommandations sur les deux autres volets. Les trois volets associés permettent au chef de projet d'organiser le travail des

équipes. La tendance aujourd'hui est une organisation en processus : processus de conception, processus de conduite de projet, sûreté de fonctionnement, qualité, etc.

Dans la mise en œuvre, cela dépend des exigences spécifiées. Dans notre cas, nous pouvons résumer les exigences à la démonstration de faisabilité des microsystèmes, dans un délai correspondant à la durée d'une thèse (Sept 2002 à Octobre 2005), sans exigences industrielles sur la qualité et la fiabilité du produit final.

Il reste donc la description physique du produit. Sur ce point, nous avons convenu de nous référer aux produits existants en apportant les innovations des produits autonomes et communicants. La base mécanique de l'extensomètre à corde vibrante de la Figure 1-12 peut donc guider nos choix terminaux d'implantation (systèmes mécaniques à deux points d'ancrage). La présentation suivante ne considère donc que les aspects fonctionnels.

Les études sont nombreuses pour aider à l'élaboration d'un cahier des charges avec deux ambitions principales :

1. Ne rien oublier,
2. Structurer le document de manière à favoriser la formalisation et le suivi des exigences.

Sur ce point (2), on trouve des travaux allant dans le sens d'une écriture encadrée par des mots clefs et par une structuration très rigoureuse du document. Cette approche paraît difficile à envisager dans le court terme car les participants à la rédaction d'un cahier des charges peuvent être nombreux et divers : difficile donc de leur faire parler un langage unique.

Notre projet a concrètement débuté sous l'impulsion d'une réunion qui a regroupé toutes les parties prenantes du projet. Si nous associons chaque personne présente à un acteur, nous pouvons alors dire que l'équipe de réunion était formée du maître d'ouvrage, et d'une équipe de concepteurs responsables de la maîtrise d'œuvre. Cette réunion a eu pour objectif de mettre à jour les principales spécifications et propriétés attendues du système. Cette étape est un premier pas vers le cahier des charges. Nous choisissons de la scinder en deux phases pour la compréhension de la démarche. Une première phase concerne la collecte d'informations latentes que détiennent les participants et d'autre part, une deuxième phase concerne l'innovation de conception et l'aide à la génération d'idées, sur la base de la méthode des analogies. Ces deux phases peuvent être conduites séquentiellement en débutant par l'une ou l'autre. Nous avons choisi de débuter la collecte des informations pour cadrer le projet et ensuite nous avons mené la phase de discussion créative. Nous allons détailler ces deux phases dans les paragraphes suivants.

Dans la première phase, nous sommes amenés à considérer **le problème de l'élicitation**, qui consiste à provoquer ou déclencher la formulation d'un maximum d'informations. Cette problématique touche essentiellement les études marketing pour sonder les informations latentes d'un groupe d'individus, mais il est important de la garder en tête dans le cadre de l'ingénierie et dans les rapports d'échanges entre différents partenaires d'un même projet. Dans ce cas, les informations proviennent de chaque partie prenante du projet et concernent différents domaines d'expertise relatifs à leur culture propre. Goguen présente et discute les démarches qui peuvent être entreprises [GL93]. Nous retiendrons l'introspection, les interviews orientés, les questionnaires, les discussions. Il s'agit d'un grand nombre de méthodes qui orientent plus ou moins l'information souhaitée en proposant un cadre. De cette manière, l'information est déjà pré-classée, en adéquation avec son contexte et pré-formatée pour son exploitation future. Il fait apparaître l'importance du contexte dans l'interprétation et l'appréciation des informations. Cette idée est fondamentale lorsque l'on souhaite garder le sens premier qui a été donné à l'information et ainsi limiter les risques de mauvaises interprétations. Cette idée de contexte est toujours valable une fois que l'information est formalisée, puisque le modèle ainsi créé n'est pertinent que dans un contexte bien défini par une syntaxe, des règles et des limites fixées. Cette étape concerne donc le regroupement d'informations qui sont latentes, mais qui ne demandent qu'à être extraites, formalisées et exploitées.

Nous avons, dans cette phase, utilisé la **méthode d'analyse fonctionnelle du besoin** [Afn96]. Nous nous sommes appuyés sur un guide constitué d'une liste de questions pour sonder un éventail large des besoins du maître d'ouvrage. Cette méthode constructive d'analyse fonctionnelle du besoin se déroule en six étapes qui se présentent sous forme de questions ou d'actions à expliciter :

1 - Quel est le besoin ?
2 - Quels sont les domaines et les exigences d'utilisation ?
3 - Identifier le système qui satisfait le besoin.
4 - Contrôler la validité du besoin.
5 - Quelles sont les séquences d'usage et les cas d'utilisation ?
6 - Quels sont les éléments de l'environnement, les fonctions et les contraintes ?

A ce moment de l'analyse, les réponses doivent être le plus étoffées possible. Il est important de se poser la question suivante : Comment apporter un maximum d'informations et comment enrichir ces informations par l'interaction des participants ? Certaines questions comme « Quel est le besoin ? » sont abruptes et doivent être traitées avec beaucoup de rigueur pour clairement définir les volontés de la maîtrise d'ouvrage. Il faut dans ce cas extraire le maximum d'informations latentes. Pour d'autres questions comme « Quelles sont les séquences d'usage et les cas d'utilisation ? », nous pouvons imaginer avoir recours à une phase de discussion créative d'idées.

Dans la deuxième phase, nous avons tenté d'apporter une plus-value à ces informations en y ajoutant la notion d'ouverture et de créativité. Cette démarche consiste à **imaginer des alternatives** pour la conception de notre système. Nous sommes ici dans une phase dite « créative ». Les méthodes d'aides à la génération d'idées sont variées. Elles sont en partie discutées par Hender [HRD01] qui compare l'efficacité de trois méthodes : le brainstorming, la méthode des analogies et l'inversion de postulats. Rappelons tout d'abord les bases de ces techniques.

Tout d'abord le **brainstorming**. Il est basé sur les deux principes fondamentaux suivants : un jugement différé et la quantité qui doit apporter la qualité. Il découle quatre lois de ces deux principes :

a. La critique n'est pas permise au cours de l'exercice, elle doit être différée car elle pourrait gêner la génération d'idées.
b. Les idées inconventionnelles sont bienvenues, elles sont souvent source de nouveauté.
c. Une grande quantité d'idées doit être listée ce qui augmente les chances de trouver la solution intéressante.
d. La combinaison et l'approfondissement d'idées qui ont été listées sont fortement conseillés.

Ensuite, la **méthode des analogies**. Cette méthode consiste à lister des mots clefs et à en détailler les attributs. Cette méthode est constituée des phases suivantes :

a. Décider du principe qui régit le problème.
b. Utiliser ce principe pour générer une liste d'analogies qui ont un concept similaire.
c. Sélectionner les analogies intéressantes et les décrire en détail.
d. Reconsidérer le problème posé vu de cette analogie pour générer des idées.

Pour finir, la **méthode d'inversion de postulats**. Les phases de cette technique sont les suivantes :

a. Lister toutes les assertions reliées au problème.
b. Retourner ces assertions dans tous les sens possibles, ce qui permet de changer de perspective.
c. Utiliser ces nouvelles phrases pour stimuler les idées.

L'étude se penche sur deux axes : la quantité et la pertinence des idées générées. La technique d'inversion de postulats et celle, très répandue, du brainstorming sont considérées les meilleures en terme de quantité d'idées générées. L'avantage est donné à la technique d'inversion de postulats. Cependant, en ce qui concerne la créativité, le classement des techniques a placé l'analogie en première position, suivie du brainstorming puis de l'inversion de postulats.

Nous avons choisi d'enrichir les réponses faites au questionnaire d'analyse du besoin en utilisant une méthode se rapprochant de celle des analogies. L'idée a été d'avoir une approche par mots clefs pour affiner les réponses aux questions posées. Ainsi, nous avons suivi les six étapes du questionnaire de la méthode d'analyse du besoin pour définir plus en détail le système et son environnement. Nous avons abouti à la liste suivante :

1 - Quel est le besoin ?

Suivre de manière **spatio-temporelle** les **caractéristiques** d'une **installation** dans son **environnement**.

Suivre pour surveiller (le niveau de risque), alerter (un service ad-hoc) et informer.
Spatio-temporelle : les mesures sont distribuées dans le temps et l'espace par un pas de temps T et le pas métrique P.
Caractéristiques : ce sont les grandeurs physiques.
Installation liée à la production, au transport et à la distribution d'énergie électrique.
Environnement : naturel, humain (gestion et utilisation), installation.

2 - Quels sont les domaines et les exigences d'utilisation ?

- Le système de suivi ne doit pas modifier ni perturber l'installation.
- Le système doit pouvoir être installé et enlevé (déplacé) aisément.
- Le système ne doit pas nécessiter d'intervention de maintenance de la part de l'utilisateur.
- Chaque élément du système doit être aussi miniaturisé que possible.
- Le système doit pouvoir être utilisé dans tout domaine d'utilisation terrestre.

3 - Identifier le système qui satisfait le besoin.

- Système comprenant des dispositifs multicapteurs (DMC) répartis/distribués.
- Les DMC doivent mesurer les grandeurs physiques à spécifier.
- Chaque dispositif doit communiquer ses données à une centrale de base d'informations.
- Système pouvant s'auto-diagnostiquer (contrôle du bon fonctionnement de ses organes).

4 - Contrôler la validité du besoin.

A qui, à quoi le système rend-il service ?

- Suivi de dérive des grandeurs physiques.
- Suivi de vieillissement des installations.
- Détection de danger.
- Suivi de maintenance.
- Détection de pannes.
- Cartographie.

Dans quel but ?

- Raisons de conformité à des normes de sécurité.
- Indirectement, d'amélioration de rendement.
- Réduction ou optimisation des dépenses.
- Cartographie de grandeurs physiques.

Sur qui agit-il ?

- Par une alarme, auprès du service ad hoc.
- Par l'envoi d'informations, auprès du service ad hoc, suite à sa demande.

Stabilité du besoin :

- Qu'est ce qui pourrait le faire disparaître ?
- Fiabilité et pérennité des installations.
- Banalisation de l'installation (jetable, etc.).
- Coût du suivi > coût du matériel.

Qu'est ce qui pourrait faire évoluer le besoin ?

- Les lois.
- Les normalisations.
- Les réglementations.

5 - Quelles sont les séquences d'usage et les cas d'utilisation du système ?

Séquence d'usage	Conception	Fabrication	Stockage	Mise en service, installation	Fonctionnement	Pas de maintenance	Enlèvement /déplacement
Utilisateur	Concepteur spécialisé			Expert	Service personnel ad hoc		Expert

6 - Quels sont les éléments de l'environnement, les fonctions et les contraintes ?

Les fonctions sont définies comme des services rendus par le système aux éléments de l'environnement. Ces fonctions F sont représentées graphiquement par des traits de liaisons traversant le système et reliant deux éléments de l'environnement. Les contraintes C, sont elles représentées par des flèches unidirectionnelles qui lient un élément de l'environnement au système.

Les fonctions et les contraintes sont d'abord listées puis elles sont représentées graphiquement sur la Figure 1-17.

Fonctions de service :

Usage :

F1 : Faire les mesures
F2 : Traitement des mesures
F3 : Enregistrer
F4 : Donner l'alarme
F5 : Transmettre les grandeurs physiques
F6 : Auto-test

Adaptation :

F7 : Distribution spatiale

Contraintes :

C1 : Etre fiable par rapport à l'environnement
C2 : Etre autonome (pas de maintenance)
C3 : Pas de modification, ni perturbation de l'installation
C4 : Taille la plus réduite possible

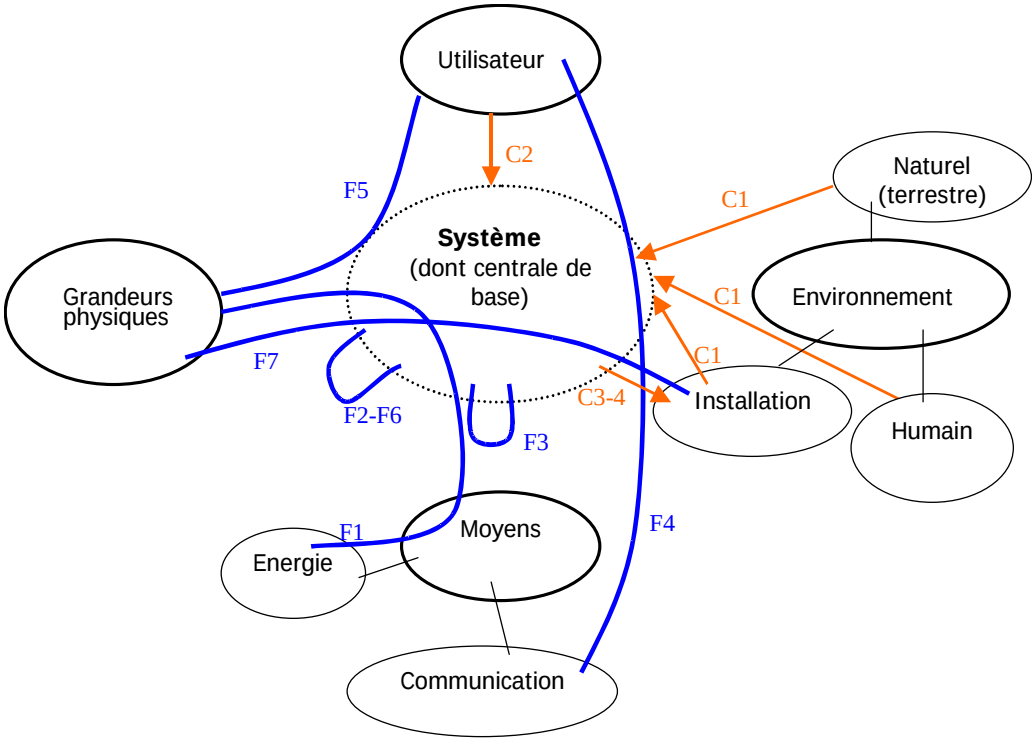


Figure 1-17 : Représentation graphique des fonctions et des contraintes du système.

La Figure 1-17 représente le système au centre, entouré par quatre domaines (Utilisateur ; Grandeurs physiques ; Environnement ; Les moyens) complétés par des sous-domaines de premier ordre qui interagissent avec le système par l'intermédiaire de fonctions F ou de contraintes C. Ici, nous obtenons la représentation d'un système d'acquisition de données classique et générique.

Dans le cadre de notre application, chaque sous-domaine de premier ordre est décomposé en attributs qui par leur nature, définissent plus en détail les besoins. Ainsi, nous obtenons un modèle

d'environnement particulier à notre microsystème. Les sous-domaines de second ordre ainsi obtenus sont illustrés sur la Figure 1-18.

Nous obtenons ici une première représentation détaillée de l'environnement du système qui fait apparaître les caractéristiques et les paramètres propres à notre application. Ce travail d'analyse du besoin est une base pour la rédaction du Cahier Des Charges Fonctionnel (CDCF) et de la Spécification Technique du Besoin (STB). Ainsi, nous obtenons une hiérarchie de fonctions et de contraintes du système par l'analyse du besoin. Cette hiérarchie est représentée par le diagramme arborescent de la Figure 1-19 et de la Figure 1-20.

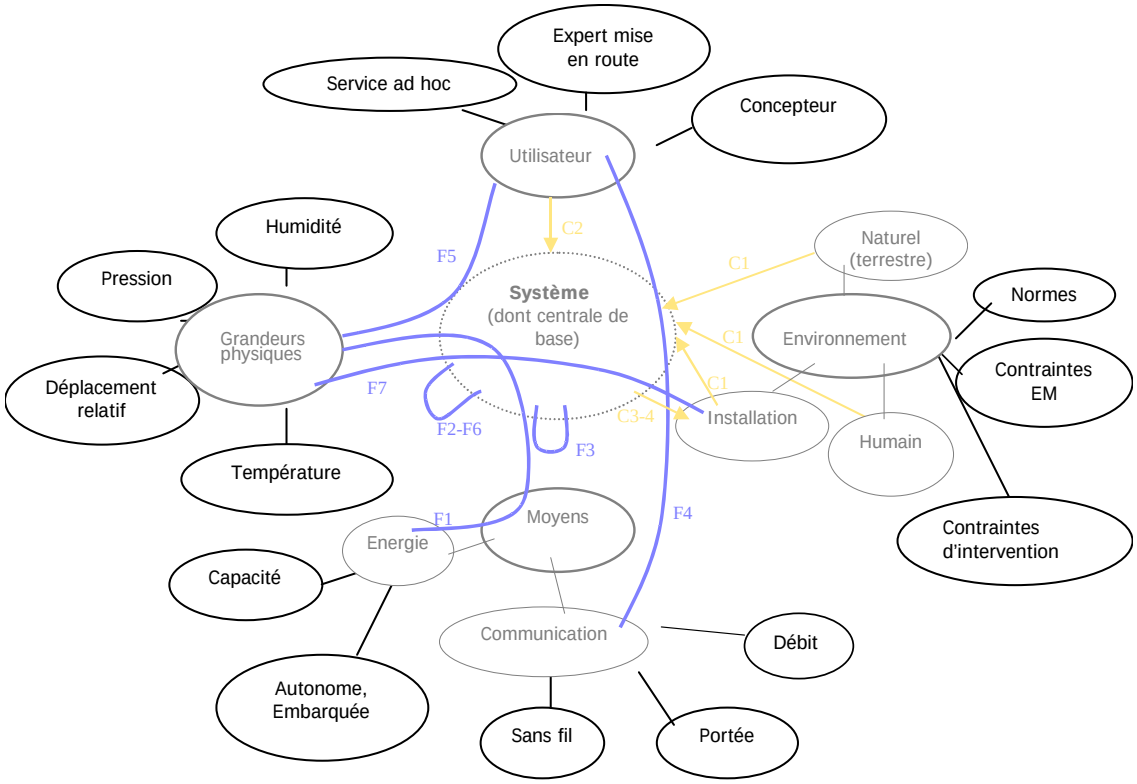


Figure 1-18 : Système et sous-systèmes propres à notre application.

- Le premier diagramme, relatif aux fonctions (Figure 1-19), énumère les quatre fonctions principales (Mesurer ; Traiter les mesures ; Communiquer ; Autotester) et leurs sous-fonctions associées que devra intégrer le système.

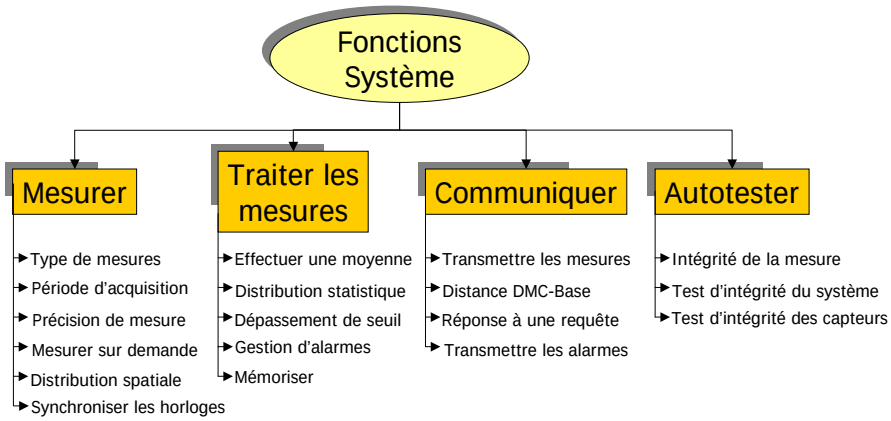


Figure 1-19 : Fonctions principales et sous-fonctions du système.

- Le second diagramme, relatif aux contraintes (Figure 1-20), énumère les deux contraintes principales (Etre autonome ; Ne pas perturber l'installation) et leurs sous-contraintes associées.

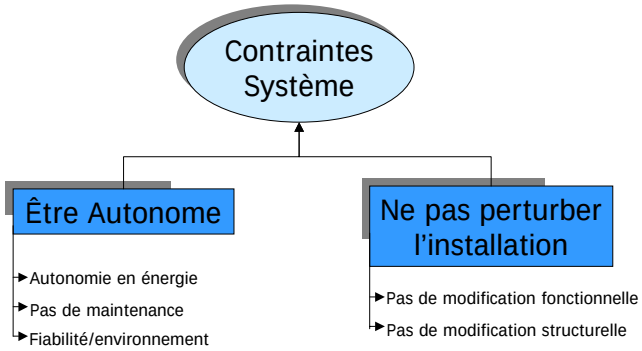


Figure 1-20 : Contraintes principales et sous-contraintes du système.

Dans le cadre du CDCF, nous avons détaillé par un « arbre des fonctions de services » chaque fonctionnalité élémentaire du système, en lui associant des **critères d'évaluation** et **d'acceptation** qui puissent par la suite être utilisés pour tester et valider le système réalisé.

Cette étape de rédaction des spécifications en langage naturel est nécessaire pour d'une part, chiffrer et paramétrer les fonctions et d'autre part, pour introduire des sous-fonctions supplémentaires qui ne sont pas apparues lors de l'analyse du besoin. Il s'agit d'obtenir ici une description la plus complète du système : incorporant à la fois les exigences de comportement externe (besoins clients) en terme d'entrées-sorties et leurs liaisons, mais aussi les exigences non fonctionnelles, telles que les contraintes, les performances, la durée de vie, la robustesse, la relation entre objets, la dynamique temporelle. L'écriture en langage naturel rend le document compréhensible par tous et donne une certaine liberté à la présentation du système, mais ne facilite pas toujours l'analyse à posteriori par une personne « étrangère ». Car en effet, le langage naturel induit des imprécisions et des ambiguïtés qui devront être levées lors de la phase d'analyse.

En effet, l'équipe qui écrit ce document peut facilement :

- Mélanger les besoins, buts, contraintes.
- Inclure de multiples concepts dans un même paragraphe (coupler diverses vues du système). Par exemple, les descriptions fonctionnelles peuvent être couplées à des informations relatives aux flux de données.

Ces problèmes peuvent avoir des répercussions néfastes sur les phases ultérieures du projet (maintenabilité, traçabilité). Il est conseillé de rédiger le cahier des charges en scindant le texte en paragraphes qui permettront une traçabilité plus grande du projet.

Nous souhaitons, à ce niveau, obtenir une représentation hiérarchisée des fonctions et sous-fonctions et être capable de suivre ou de tracer les exigences formulées dans le cahier des charges. Nous proposons d'intégrer à notre démarche un outil de suivi des exigences tel que DOORS, Requisite-Pro, IRqA, etc. qui jouera le rôle de pointeur de spécifications d'une représentation à une autre.

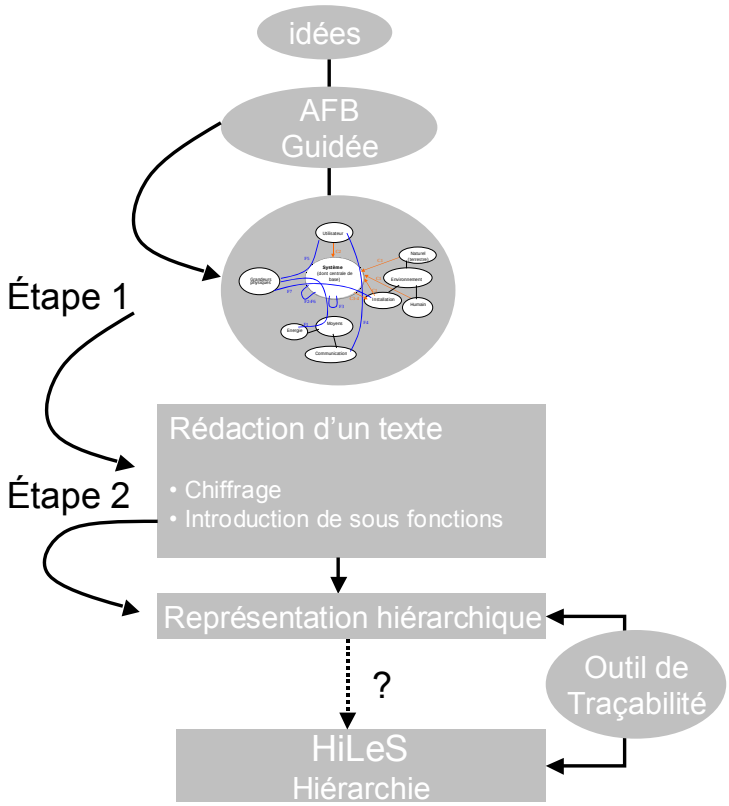


Figure 1-21 : Démarche générale de spécification du besoin et préparation préliminaire à la traçabilité des exigences dans le projet.

La structure générale de la démarche que nous avons suivie est illustrée par la Figure 1-21 :

- La première étape consiste en la représentation graphique du système global lié à son environnement par des fonctions et des contraintes.

- La deuxième étape consiste en la rédaction d'un document textuel devant être complet du point de vue des spécifications. Cette étape est nécessaire pour deux raisons : d'une part, pour chiffrer les paramètres, les performances attendues des fonctions du système, et d'autre part, introduire des sous-fonctions pour ainsi enrichir le système et établir ainsi une topologie hiérarchique. De plus, le format hiérarchique du document textuel facilite l'intégration d'outils de traçabilité des exigences dans notre démarche puisqu'ils permettront de faire des liens entre les fonctions classées dans le document textuel et les fonctions représentées par le modèle HiLeS.

Cependant, il reste un lien critique qui est le passage entre les spécifications textuelles et la représentation HiLeS. Nous aborderons, dans le deuxième chapitre les moyens et les outils que nous proposons pour définir le système HiLeS à partir des spécifications.

1.4.4 Le plan de travail

Nous avons voulu dans notre travail appliquer exactement le processus de développement de la Figure 1-8 en respectant autant que possible le caractère descendant de la conception. Pour cela, nous avons décidé d'engager l'intégration matérielle qu'une fois les tâches de modélisation et de simulation réalisées. Pratiquement, tout ne s'est pas fait aussi simplement mais, dans de nouveaux exemples, notre expérience peut être intéressante pour la conception.

Nous avons maintenant obtenu, après le travail préliminaire d'analyse du Besoin, un cahier des charges précis de notre microsystème. Nous rappelons, qu'un tel cahier des charges qui peut être distribué à plusieurs partenaires, doit regrouper les informations qui définissent tout le microsystème, du point de vue des attentes du maître d'ouvrage. A ce stade, les solutions techniques pour implémenter le système ne sont ni définies, ni réparties selon des métiers. Il existe alors plusieurs solutions possibles pour réaliser le système final. Avant de lancer l'implémentation du système, il nous paraît primordial d'être capable d'avoir une première évaluation d'un microsystème et de son architecture et ceci avant de confier la conception des sous-systèmes aux experts concernés. Pour cela, nous avons choisi de nous appuyer sur la conception amont pour proposer une esquisse fonctionnelle du microsystème. Ce premier modèle fonctionnel permet d'avoir d'une part une vision globale des fonctionnalités, des modes de fonctionnement, des domaines de validité d'exploitation du microsystème et d'autre part, il se veut être une base sur laquelle pourront s'appuyer les partenaires du projet pour échanger des informations et valider le développement des sous-systèmes dont ils ont la charge.

Nous proposons d'utiliser l'outil HiLeS pour définir une première architecture du système. Notre plan de travail suit une démarche descendante. A partir des spécifications générales du système, la conception amont va nous permettre d'explorer et d'évaluer différentes propositions d'architectures de « modèles amont » qui permettent de manipuler des fonctions génériques. La Figure 1-22 illustre le cycle de conception, modification, validation des architectures du microsystème.

Le passage entre le cahier des charges et la représentation du microsystème sous HiLeS est une transition délicate. Nous avons, sur cette partie, bénéficié des travaux du groupe ISI du LAAS qui nous a orienté vers une première étape de modélisation du cahier des charges par des diagrammes SysML (Cahier des charges formaté). Cette approche permet de semi-formaliser le système et de rendre les spécifications transportables, échangeables et de limiter les risques de mauvaises interprétations. Après avoir défini quelques règles simples du passage SysML-HileS, le système HiLeS peut être modélisé.

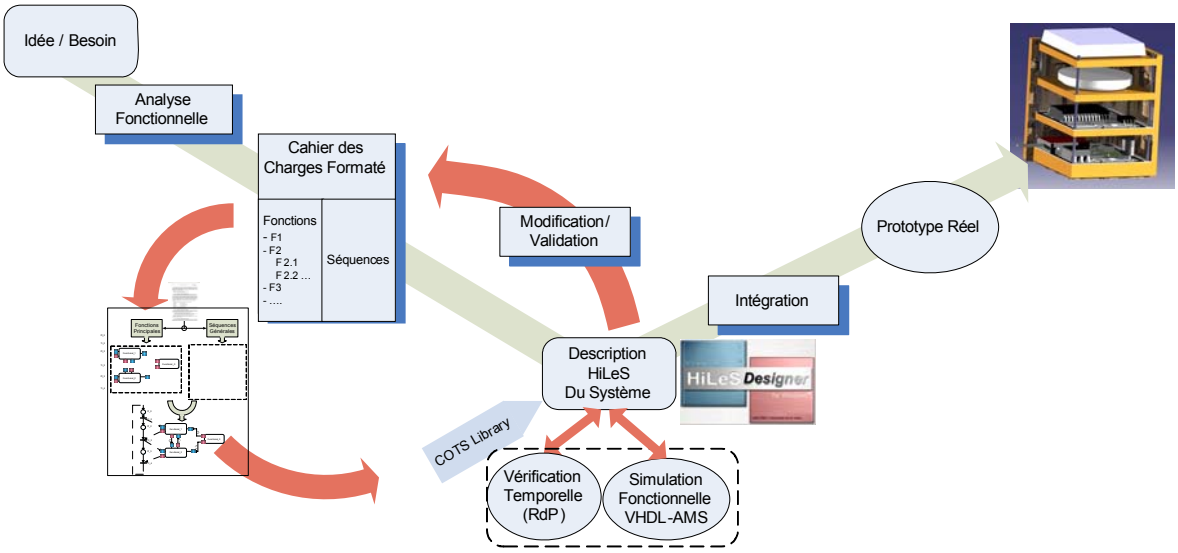


Figure 1-22 : Démarche globale pour la conception et l'implémentation d'un microsystème [Mau04b].

La plate-forme HiLeS va nous permettre de faire une première étude des caractéristiques temporelles du microsystème modélisé. Il sera de notre ressort de valider cette modélisation vis à vis des spécifications. A ce stade, le système est décomposé, par affinements successifs, en sous-parties fonctionnelles de plus en plus simples (dans le même esprit que SADT : Structured Analysis and Design Technique). Le but de cette étape est d'évaluer la faisabilité du projet et si besoin d'améliorer et détailler les spécifications. Cette boucle de retour apparaît sur la figure comme une flèche de liaison entre la description HiLeS du système et le cahier des charges formaté. Plusieurs itérations sont envisagées et l'on décide alors d'affiner l'architecture de manière à valider les caractéristiques temporelles attendues.

Dans un deuxième temps, nous obtenons une représentation architecturale fine et validée du microsystème complet, constituée de fonctions élémentaires. Chaque fonction ou composant pourra alors être remplacé par un modèle correspondant à un produit ou à une technologie existante. Le concepteur a alors ici le choix de l'implémentation des fonctions élémentaires. Il peut utiliser des composants élémentaires simples ou les agréger pour obtenir des fonctionnalités de composants réels et connus. Ces composants ou agrégats de composants devront être réalisés par une technologie appropriée. HiLeS gère la représentation complète de la structure en VHDL-AMS, afin de pouvoir réaliser les simulations comportementales du prototype virtuel.

1.5 Conclusion

La surveillance multisensorielle est une problématique de grande actualité. Les microsystèmes multicapteurs répartis communicants peuvent répondre à ce besoin s'ils sont connectés à un système d'analyse et de décision. Notre domaine d'application est la surveillance des bétons en Génie Civil : nous avons fait sur ces questions un état de l'art sur les technologies microsystèmes et sur les pratiques de mesure des contraintes dans les bétons.

Le deuxième point développé dans ce chapitre est d'ordre méthodologique. Quelle méthodologie de conception ? Quelle approche dans le cas des microsystèmes ? Après un tour d'horizon rapide des pratiques générales, nous proposons une démarche descendante, qui, partant des spécifications, viserait à définir des tâches et une architecture « métiers ».

Nous avons commencé dans ce chapitre par l'établissement du cahier des charges de notre système.

Les voies que nous avons explorées concernent :

1. le cahier des charges aux travers de réunions « dirigées » en appliquant la méthode d'analyse fonctionnelle du besoin.
2. l'organisation de la traçabilité des exigences en repérant dans le document et au fil du projet l'évolution et l'implémentation des exigences par les différentes parties du système.

Nous disposons à ce stade d'un cahier des charges et d'une démarche dont une première étape est la conception amont que nous traiterons dans le chapitre suivant.

A partir du cahier des charges, le processus de conception doit franchir les étapes de formalisation des exigences et de modélisation « amont » selon un formalisme compatible avec un objectif de vérification : nous y reviendrons dans le chapitre 2. Le processus de conception que nous avons retenu dans le cadre d'un travail collectif (projet MOCAS) a conduit à la proposition de la plate-forme HiLeS que nous avons très brièvement résumé. Pour davantage de détails, nous renvoyons le lecteur à la thèse de J.C. Hamon [Ham05].

Chapitre 2

Conception « amont » d'un microsystème multicapteurs

2.1 Introduction

La présentation du chapitre 1, nous a amené à la rédaction d'un cahier des charges et à la proposition d'une approche méthodologique consistant à mettre en œuvre la plate-forme de conception HiLeS.

Une première question s'impose : comment pratiquement passer du cahier des charges à une représentation formelle de ce cahier des charges ? Cette problématique non encore résolue rejoint ici celle de tout ingénieur souhaitant modéliser son système. Il faudra garder en tête quatre points importants de la pratique de la modélisation :

- Apporter une forme visuelle au système.
- Détailler sa structure et son comportement.
- Guider la construction, l'implémentation des tâches de sa réalisation.
- Documenter les décisions prises lors de l'implémentation finale.

Nous avons donc, au départ, tenté de proposer une méthode originale basée sur l'identification de fonctions et de procédures par analyse fonctionnelle. Nous en présenterons les principes avant d'introduire une méthode complète structurée autour des représentations UML/SysML que nous avons défini grâce à la contribution du groupe Ingénierie Système et Intégration ISI, au sein d'un projet interne au laboratoire : le projet MOCAS.

Un autre point abordé dans ce chapitre est la traçabilité des exigences dans le contexte que nous venons d'évoquer : c'est un travail que nous avons insuffisamment approfondi mais sur lequel nous ferons des recommandations.

Le dernier point traité est celui de la vérification fonctionnelle avec le logiciel TINA développé au sein du laboratoire [BRV03] [8]. C'est une procédure en pleine évolution : basiquement elle consiste à extraire les réseaux de Petri de formalisme HiLeS et à vérifier les propriétés de ces réseaux par rapport au contenu du cahier des charges. Nous présenterons des résultats de vérifications sur des décompositions partielles de notre microsystème et expliciterons les stratégies de développement en cours.

2.2 Formalisation du cahier des charges

Nous avons vu que pour éviter au maximum l'ambiguïté de compréhension d'un cahier des charges rédigé en langage naturel, il convient de suivre une structure et des règles de présentation.

Ce document de base reste tout de même textuel. Il est donc « humain », c'est à dire qu'il dépend d'une culture, d'un contexte, ce qui amène souvent une rédaction subjective. De nombreux travaux tentent de limiter au mieux cette ambiguïté qui s'immisce dans les spécifications. L'idée généralement suivie est d'appliquer, au plus tôt, un formalisme pour créer des modèles de représentation de haut niveau à partir desquels, il sera possible de reformuler les spécifications du cahier des charges sous une configuration formelle. Nous pourrons ensuite appliquer des procédures de vérification pour s'assurer que ces représentations de haut niveau sont conformes au cahier des charges.

Pour atteindre cet objectif, nous proposons d'établir une démarche pour aller proprement du semi-formel au formel afin de bénéficier des avantages suivants :

- avoir une représentation claire des fonctionnalités et du comportement du système,

- générer une version formelle non ambiguë et complète des spécifications de ce système,
- permettre une première analyse des fonctionnalités et du comportement du système,
- établir une première validation des spécifications.

De nombreux travaux ont été réalisés, dans cet esprit, pour l'ingénierie logicielle. Aujourd'hui, les concepts de traçabilité et de validation des spécifications à haut niveau imaginés pour l'ingénierie logicielle émergent dans le domaine de la conception de systèmes hybrides qui englobent à la fois des parties logicielles et des parties matérielles multidisciplinaires. Nous nous situons dans ce cas de figure [Mau05b].

2.2.1 Position du problème

Dans l'absolu, il faut considérer qu'au delà du cahier des charges tel que nous l'avons introduit au Chapitre 1, le concepteur a accès à une base de données où l'on retrouvera tous les acquis antérieurs disponibles (Figure 2-1) :

- les composants réutilisables,
- les retours d'expériences récents,
- les données expertes,, etc.

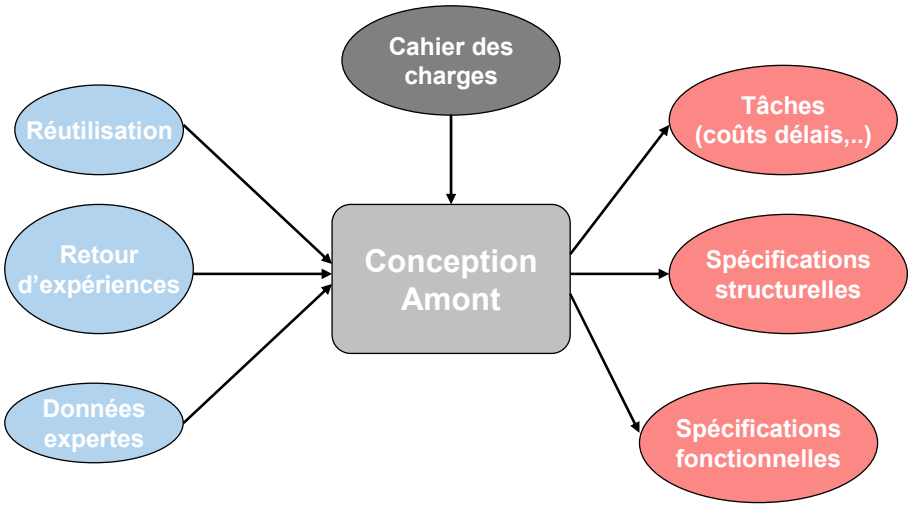


Figure 2-1 : Environnement de la conception amont.

Pour permettre une conception « globale » à la fois fonctionnelle, structurelle et économique, nous ne nous intéressons, dans ces travaux, qu'aux seuls aspects fonctionnels. Nous pensons que deux approches descriptives doivent être privilégiées :

- la description hiérarchique descendante des fonctions constitutives du système,
- la logique fonctionnelle débouchant sur un ensemble de scénarios.

Evidemment, nous voulons à partir de là, converger vers une représentation selon le formalisme HiLeS, pour y appliquer une démarche de vérification à l'aide de TINA et ouvrir la voie au prototypage virtuel VHDL-AMS.

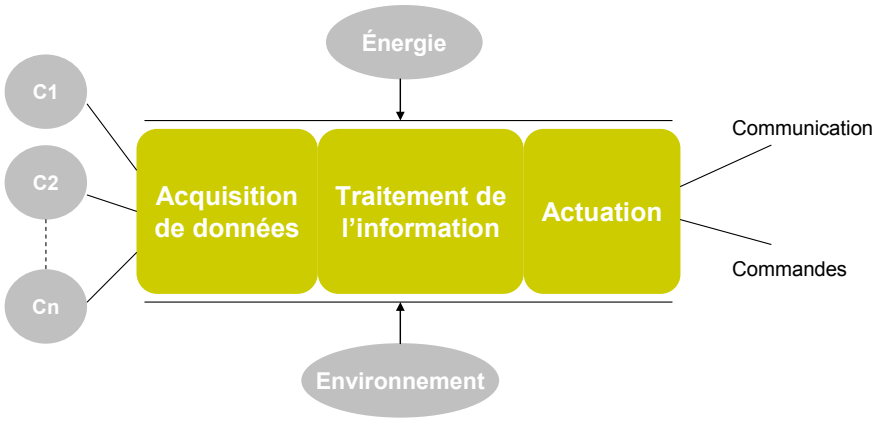


Figure 2-2 : Structure générique d'un microsystème multi-capteurs communicant.

Notre domaine applicatif est celui des microsystèmes. Nous souhaitons au terme de cette étude donner une démarche générique de conception avec une part importante de « re-use » en partant de l'idée que tous les microsystèmes ont une structure générique (Figure 2-2) : des capteurs de mesure (C1, C2, , ..., Cn), une fonction d'acquisition de données, de traitement de l'information (fusion multisensorielle, diagnostic, etc.), d'actuation (interface de communication et de commande), un besoin énergétique et des facteurs d'influences environnementales.

Pour décrire l'architecture du système proposé et pour rester conforme à l'esprit HiLeS, nous utiliserons le mot « bloc » pour fonctions et « procédures » pour la logique au sens des opérations et de l'ordre dans lequel les opérations sont menées.

2.2.2 Proposition et mise en œuvre d'une démarche de formalisation : fonctions et procédures

D'un point de vue fonctionnel, le cahier des charges doit définir clairement les objectifs fonctionnels du système tels que les entrées-sorties et les conditions d'environnement : énergie et environnement. Le système peut être complexe et il convient de rester au plus près des modes de raisonnement du concepteur.

Les vues représentatives des systèmes sont multiples, elles peuvent être : structurelles, fonctionnelles, comportementales (dynamique temporelle) ou orientées flux de données. A partir de ce point, nous proposons une approche de lecture du cahier des charges qui s'appuie sur les outils que nous développons également.

Par souci de clarté, nous avons choisi de réécrire textuellement les deux cas de description énoncés en 2.2.1 : à la fois la description statique fonctionnelle et la logique fonctionnelle aboutissant à la dynamique du système. Nous avons appelé ce document « Document de conception » [Mau04a].

La **description statique** définit une **hiérarchie de fonctions** qui se retrouve dans le plan du Document de Conception (Figure 2-3). Les « Fonctions objectifs » sont extraites du cahier des charges et deviennent des blocs.

Figure 2-3 : Plan du document de conception donnant les bases de la vue statique du système.

Chaque « bloc » constitutif du système est décrit par :

- une introduction descriptive de sa fonctionnalité,
- ses entrées/sorties et leur nature,
- sa structure interne.

La hiérarchie est ici symbolisée par des niveaux d'abstraction numérotés de 0 à n. Le niveau 0 est celui de plus haute abstraction, il représente le système dans son environnement. Le niveau 1 est celui de la description fonctionnelle du système, telle que proposée par le cahier des charges. A ce niveau, il n'y a pas véritablement de création architecturale de la part du concepteur mais une traduction claire des textes. On doit y trouver : les « Fonctions objectifs ». La décomposition se poursuit selon le besoin de détails qu'ont les concepteurs pour décrire chaque partie du système.

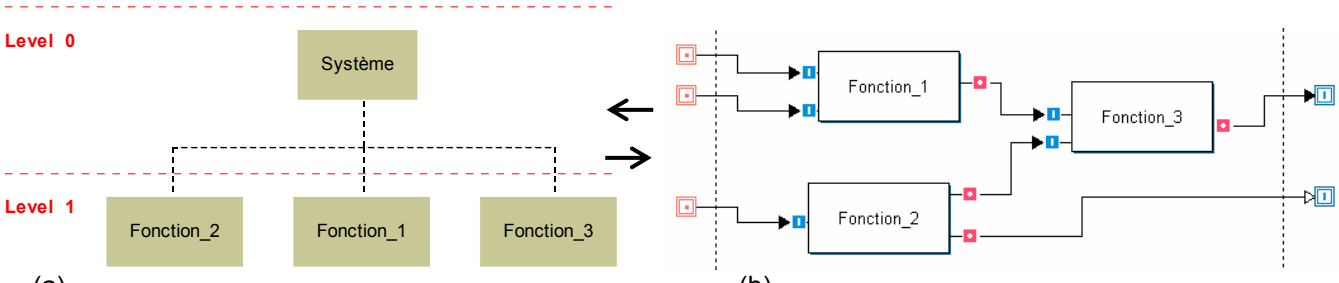


Figure 2-4 : (a) Représentation arborescente des niveaux 0 et 1, (b) Représentation fonctionnelle par bloc du niveau 1.

La représentation arborescente hiérarchique d'un système (Figure 2-4(a)) est la représentation la plus simple que l'on puisse imaginer : elle n'a aucune valeur opérationnelle sinon celle, lorsque le système est très complexe, de visualiser les fonctions.

La représentation fonctionnelle par blocs (Figure 2-4(b)) est une représentation des entrées/sorties, et met en évidence l'existence des relations entre les fonctions. Sans préjuger des technologies, on peut y lire un fonctionnement. Pour ces deux représentations statiques, le

caractère descendant des représentations est facile à imaginer, mais évidemment les fonctions nouvelles proposées à chaque niveau sont définies par le concepteur.

La **représentation dynamique** est la description des séries d'actions (logique fonctionnelle) qui forment les **séquences d'utilisation** (vue externe du fonctionnement du système). Ces représentations dynamiques sont une source d'informations importante pour bâtir le système puisqu'elles explicitent à la fois les aspects temporels de l'activité des blocs, les connexions entre les fonctions et les canaux d'échanges de données. Il y a cependant un risque de mélanger flots de données et liaison physique du système. A titre d'exemple, la Figure 2-5 illustre la forme de présentation textuelle que nous avons choisi pour décrire la succession d'actions propres à la procédure « Mémoriser ».

5.4.4 Procédure Mémoriser

*Le module **Memorise** stocke et date les résultats de mesure, de test et constitue une base de données de paramètres spécifiques pour l'évaluation des mesures et des tests.*

Les principales étapes se déroulent de la façon suivante :

- Activation de la fonction mémorisation « Start_memo » en début de test et après initialisation.
- Datation des données issues de la mesure.
- Stockage des données dans la mémoire. Les données sont sous la forme : MesureX/Qualité de la mesure X/Date (6 Octets).
- Transmission des mesures stockées en mémoire à la centrale de base à la période de transmission T_{tr} (mémorisée dans le bloc ***Timer***, Annexe 11) ou après demande explicite de mesures ciblées.
- Suppression des données dont le temps de résidence dans la mémoire est supérieur à T_{mem}.

Figure 2-5 : Exemple de la description textuelle de la procédure « Mémoriser ».

L'objectif est ici d'introduire le temps. Pour ce faire, nous avons défini :

1. Des **délais d'exécution** pour les fonctions réalisées dans chaque bloc : bien sûr, à ce stade de la conception, il s'agit d'une simple estimation proposée par le concepteur qui anticipe une solution technologique ou un élément de la base de données expertes déjà évoquée en 2.2.1.

2. L'**enchaînement logique** de ces fonctions en introduisant des notions de précédence ou parallélisme. ex : « cette fonction doit logiquement être créée avant celle-ci dans la procédure A ».

3. Des **points singuliers** dans le fonctionnement du système où le système est supposé être dans un état connu et précis : situation d'autotest, d'envoi de messages, etc.

Cette exploitation de la logique fonctionnelle ne définit pas la nature des signaux d'activités. Elle permet, dans un ensemble de chronogrammes à définir, de signaler quand une fonction (bloc) est active ou inactive. Un exemple de chronogramme est illustré par la Figure 2-6.

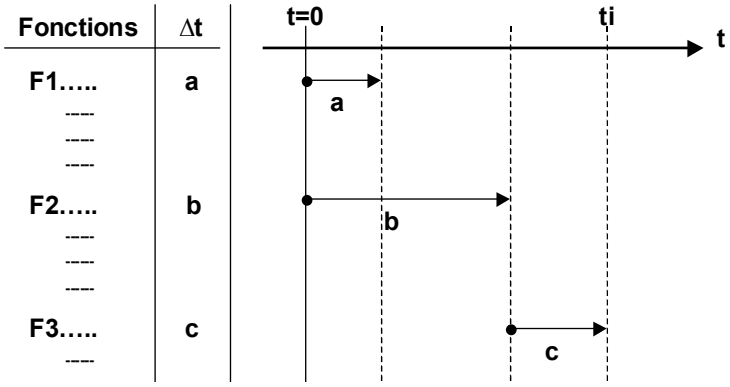


Figure 2-6 : Chronogramme d'activité des fonctions.

Ce chronogramme montre par exemple que :

- F1 et F2 sont activées en parallèle,
- F3 est activée après F1 et F2,
- F1, F2, F3 sont inactives après t=ti.

La mise en activité et l'arrêt d'un bloc sont provoqués par le changement d'état d'une de ses entrées. De cette description logique temporisée, des chronogrammes peuvent être construits qui serviront de base à l'exécution sous HiLeS. Ils peuvent prendre la forme de chronogrammes logiques, de représentation Grafcet, de diagrammes de Gantt, de diagrammes de séquence dans le graphisme UML/SysML. Selon notre démarche, au stade de la description haut niveau du comportement du système, nous avons fait le choix d'utiliser les diagrammes de cas d'utilisation et de séquence UML/SysML. Nous présentons ces étapes dans le paragraphe suivant (2.2.4).

Ici se pose la question du choix du bon niveau d'abstraction pour décrire la dynamique du système. Quel est le détail nécessaire pour décrire l'activité interne du bloc ? Selon notre expérience, le concepteur est tenté de passer à une description fine faisant quelque fois intervenir des éléments matériels qui représentent déjà une forme de choix matériel. Pour éviter cela, il faut garder à l'esprit que nous avons l'objectif de vérifier, par la simulation d'un prototype virtuel, les possibilités et alternatives d'une matérialisation du système. Même si le cahier des charges introduit le plus d'informations possible, pour définir le système, il faut ici faire attention de ne pas sauter le pas de la modélisation fonctionnelle et proposer une solution matérielle trop tôt (sauf si celle-ci est une contrainte de conception). Il faut donc rester dans l'abstrait le temps nécessaire pour « ouvrir » l'ensemble des fonctionnalités et envisager différentes hypothèses de matérialisation. Chaque hypothèse doit toujours satisfaire le cahier des charges. Nous allons, dans le paragraphe suivant, introduire la notion de traçabilité des exigences et l'illustrer dans notre exemple.

2.2.3 Traçabilité des exigences

L'ingénierie des exigences qui regroupe toutes les activités liées à l'identification, au suivi, à la gestion de l'évolution des exigences d'un projet a d'abord été déployée dans le domaine logiciel et se déploie aujourd'hui dans les sociétés produisant des systèmes matériels complexes. L'exemple qui peut être pris est celui de PSA Peugeot-Citroën qui a mis en place très récemment, depuis le milieu de l'année 1999, la première équipe d'ingénierie système [9]. Cette équipe a contribué à déployer un premier processus d'ingénierie des exigences.

Les outils d'ingénierie système s'inscrivent dans le contexte de cycle de développement en V, dans lequel, dans un premier temps les exigences sont identifiées et suivies dans la branche descendante du V, puis sont intégrées au traitement de la branche ascendante. Le principe de base de la traçabilité est d'établir des liens entre toutes les étapes de conception, ce qui correspond souvent à établir des liens inter documents et à vérifier la cohérence des exigences ou à valider un modèle à tous les stades de développement d'un projet. Le but est de garder une cohérence claire et directe entre les exigences spécifiées et les représentations du système.

D'une manière générale, les outils de traçabilité ont les fonctions suivantes :

- créer des liens inter documents,
- extraire des informations filtrées des documents texte (à l'aide de mots clefs contenu dans un lexique),
- éditer un nouveau document texte « spécialisé » (après un filtrage adapté),
- paramétrer l'état d'une exigence pour accéder à un suivi temporel de son implémentation (exemple de noms d'états associés à un lexique : en analyse, en négociation, en reformulation, référent, obsolète),
- visualiser une matrice d'impact,
- visualiser une matrice de conformité,
- visualiser une matrice de vérification.

La traçabilité permet de suivre une exigence formulée sur un cahier des charges tout au long d'un projet. Les outils de traçabilité actuels sont orientés « texte » et sont efficaces sur des documents textuels. Ils consistent à faire des liens entre des zones de textes marquées, ou des liens entre des zones de texte et des images, des graphiques, ou des tableaux qui illustrent un élément des spécifications. Il faut impérativement avoir une structuration claire du document pour faciliter l'indexation semi-automatique du document et la recherche de champs du document.

La Figure 2-7 illustre l'exemple d'un élément X que l'on retrouve dans 3 documents édités au cours d'un projet : le cahier des charges, la description du prototype et les procédures de tests. Ces 3 documents sont représentatifs de la structure en « V » du cycle de développement. Les liens sont ici représentés par des flèches bidirectionnelles. Le marquage de l'élément X est réalisé par le biais de « drapeaux » déposés à l'endroit voulu par le concepteur après l'analyse fine du texte. La plupart des outils du commerce utilisent le langage XML (eXtensive Markup Language) comme moyen d'établir ces liens.

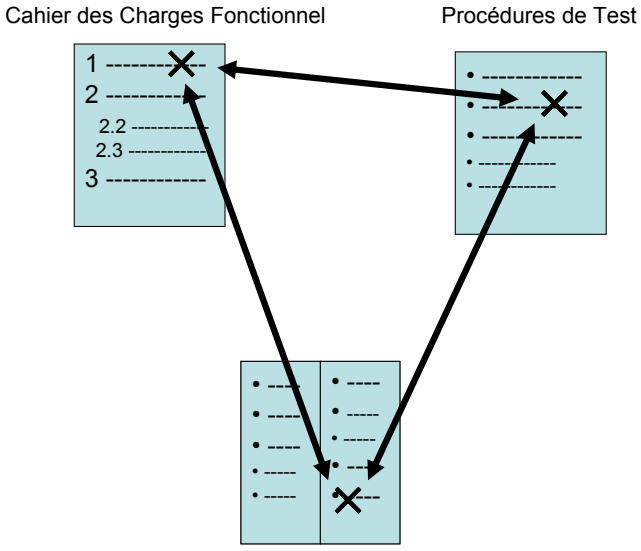


Figure 2-7 : Traçabilité obtenue par le biais de liens qui pointent un même élément X sur plusieurs documents.

Au delà de ce simple lien, la traçabilité des exigences permet de répondre à des questions telles que :

- Quels sont les constituants concernés en cas de modification d'exigences du client ?
- Quelles exigences systèmes sont mises en cause par un changement technique ou une non-conformité d'un constituant ?

Lorsqu'une modification de spécification du système a lieu, ces outils de traçabilité permettent de faire apparaître les correspondances relatives, l'impact de ce changement d'exigence sur les autres documents du projet qui décrivent le système. Cette caractéristique permet d'envisager

efficacement la réutilisation d'une partie ou de la totalité des spécifications déjà écrites dans de nouveaux projets et de fait de garder les liens vers d'autres dossiers de documentation associés.

Dans le cadre de notre projet, nous avons mis en œuvre ces considérations sur 3 documents à l'aide de l'outil DOORS [10] : le **cahier des charges fonctionnel**, le **document de conception**, et le **code VHDL-AMS** du prototype virtuel. Cet outil est à rapprocher de ses concurrents : Requisite-Pro [11] et IRqA [12], qui aident aussi à la gestion des exigences. Nous avons choisi DOORS pour une raison de facilité d'accès à EDF R&D. Il joue un rôle de pointeur d'exigences et établit des liens entre plusieurs documents qui instaurent, complètent, vérifient, ou définissent les exigences des spécifications d'un système. DOORS est le noyau qui permet de lier tous ces documents qui sont créés au cours d'un cycle de développement pour établir, à tout moment du projet, une matrice des liens d'interdépendances. Cet outil a toutes les fonctions des outils de traçabilité décrites en début de section.

Ces 3 documents se situent aux extrémités des branches du cycle de développement en V. Ils cernent ainsi la globalité du cycle. Avant d'utiliser un outil de traçabilité, le concepteur ou l'ingénieur système doit procéder à une phase d'analyse qui consiste à indexer chaque paragraphe pertinent d'un document afin de pouvoir ensuite les différencier et les classer par catégories. L'ingénieur est libre de créer les catégories qui sont pertinentes et qui correspondent au métier propre à chaque exigence. La liste des catégories formera un lexique que le concepteur et les autres intervenants utiliseront tout au long du projet.

Prenons dans un premier temps, l'exemple de l'analyse faite sur notre **cahier des charges**. Nous avons choisi de classer les lignes de texte par les catégories suivantes (visibles sur la colonne de droite de la Figure 2-8) :

- Fonction.
- Contrainte.
- Description.
- Méthodes d'évaluation.
- Critères d'acceptation.

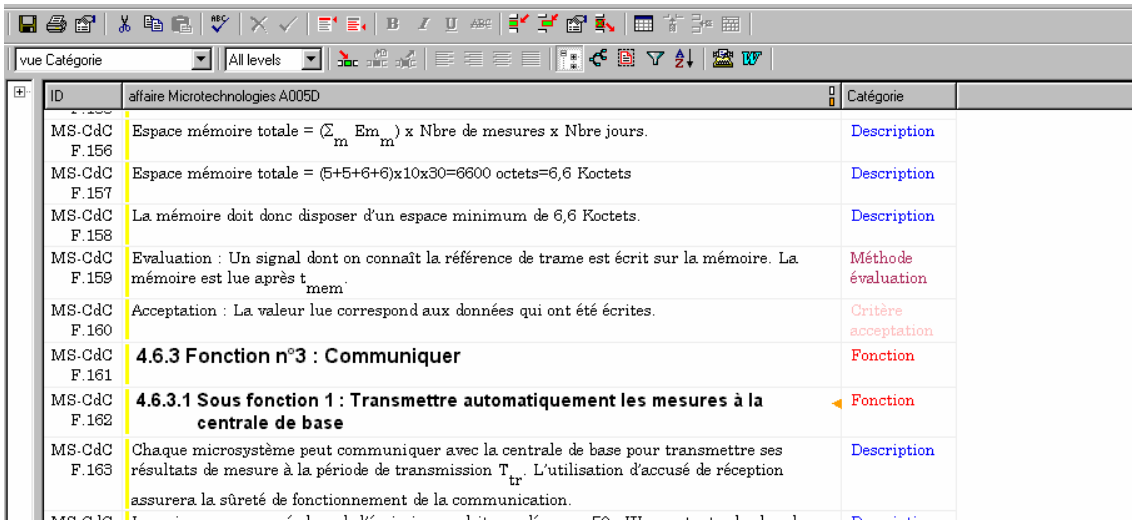


Figure 2-8 : Vue du cahier des charges et du classement des lignes indexées par catégories.

Si nous choisissons par exemple la catégorie « Méthodes d'évaluation » comme filtre : nous voyons apparaître uniquement les méthodes d'évaluation car toutes les lignes n'appartenant pas dans cette catégorie sont cachées. Cet exemple est simple mais il permet de montrer l'efficacité de la méthode. Nous voyons ici l'intérêt direct que peut avoir la génération d'un document texte. Par ce biais, nous obtenons un document allégé qui contient toutes les procédures de test et qui pourra être utilisé dans la phase d'intégration finale.

Le document suivant (Figure 2-9) que nous avons analysé sous DOORS est le **document de conception**. Il fait apparaître les fonctions qui sont modélisées par des blocs sous HiLeS. Chaque fonction satisfait une exigence particulière du cahier des charges. Ainsi, chaque bloc représenté sous HiLeS a été lié à une exigence fonctionnelle. Le lien est bidirectionnel, ce qui nous permet de

naviguer d'une représentation à l'autre en gardant une trace des spécificités de chaque bloc. Le suivi de l'impact est double : descendant (top-down) lorsqu'une modification des spécifications provoque une modification du système, et à l'inverse ascendant (bottom-up) lorsqu'une modification touche directement un constituant du système, nous pouvons remonter aux spécifications globales du système. Nous avons ajouté une propriété supplémentaire qui est d'indiquer les liens inter-blocs. La Figure 2-9 fait apparaître, dans la fenêtre grisée, les liens du bloc considéré avec les autres blocs : le bloc « Measure treatment » est connecté aux blocs « Transceiver », « Sensor test », et « Power ».

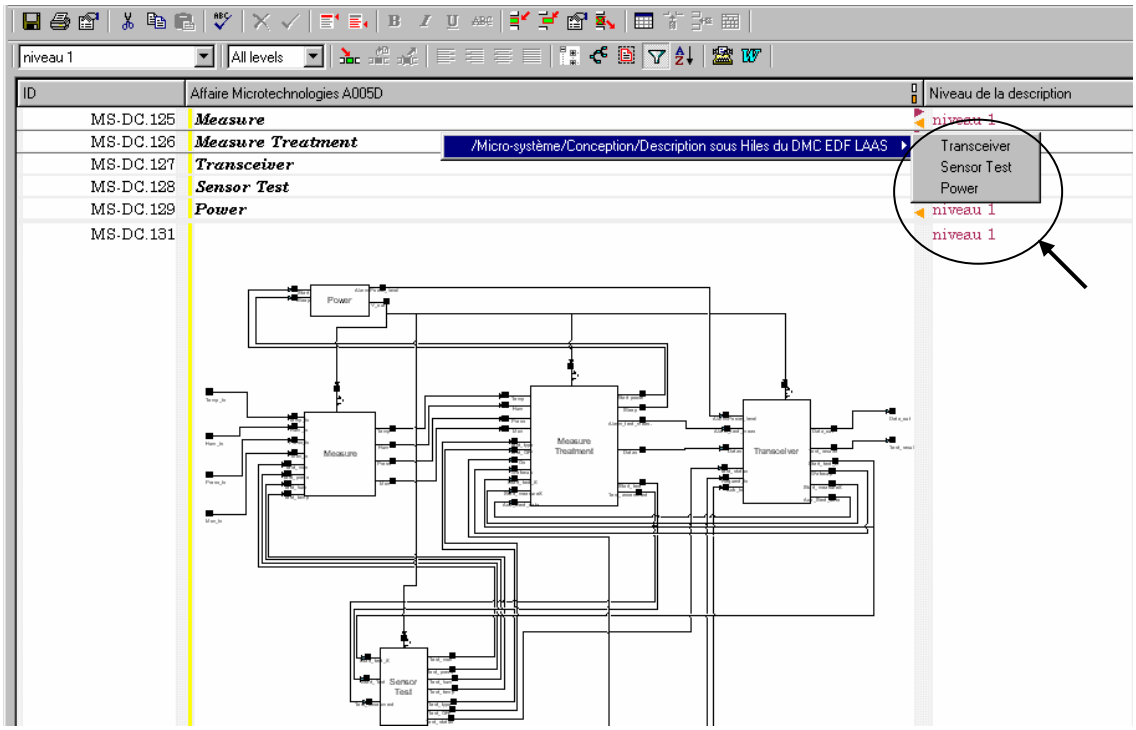


Figure 2-9 : Liens établis entre la description HiLeS et les exigences du cahier des charges.

Nous avons ensuite utilisé la matrice de traçabilité pour avoir une représentation des liens inter blocs, présentés sur la Figure 2-10. Les carrés gris foncés marquent la ligne et la colonne relative aux éléments sélectionnés. Les carrés noirs désignent l'existence d'un lien entre deux éléments. Ainsi, nous visualisons les connexions entre le bloc « Measure treatment » et les blocs « Transceiver », « Sensor test », et « Power ».

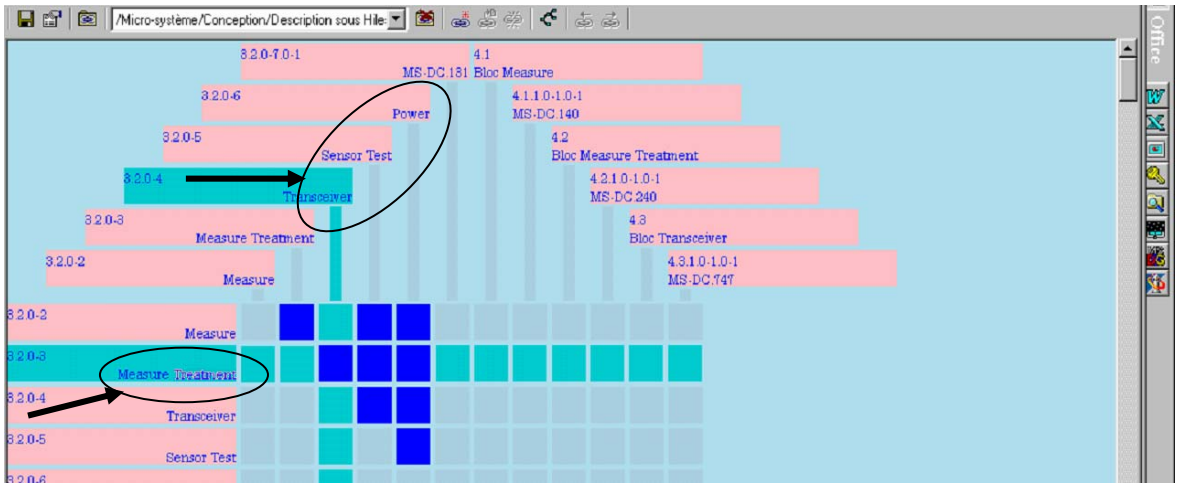


Figure 2-10 : Matrice de traçabilité entre les blocs HiLeS et les exigences du cahier des charges.

Le troisième document que nous voulons analyser est le document qui regroupe les codes VHDL-AMS du prototype virtuel. Nous y avons établi des liens de correspondance avec les deux documents précédents (cahier des charges et document de conception).

Nous avons démontré la faisabilité de la mise en place d'un processus de traçabilité qui permet de garantir la cohérence de la description du système, des spécifications au prototype virtuel. Nous avons franchi deux transitions successives pour arriver au prototype virtuel du système : le passage du cahier des charges au document de conception, et puis le passage du document de conception au code VHDL-AMS. Cet outil nous permet une première vérification relative à la cohérence de nos documents (Figure 2-11). Le défaut majeur de cette méthode est de faire reposer l'analyse sur des documents textes. Cependant, d'une manière générale, les codes informatiques évoluent rapidement et sont sujets à des modifications ou à des améliorations successives. Ici, il a fallu regrouper dans un document texte unique les codes VHDL-AMS, document qui devenait figé une fois édité. Tout l'intérêt de la flexibilité de la démarche est alors de fait diminué. Nous préconisons le pointage direct vers le fichier source, répertorié dans une bibliothèque appropriée.

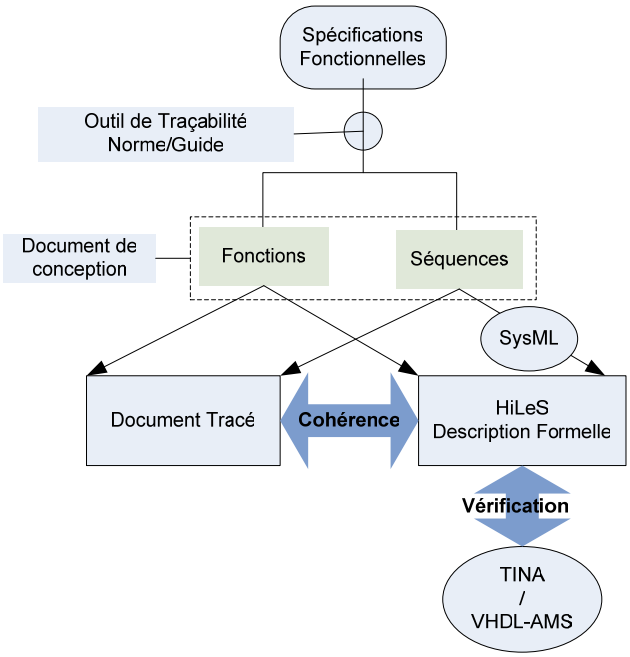


Figure 2-11 : Vue schématique de notre approche de conception.

Nous avons vu que pour éviter au maximum l'ambiguïté d'un cahier des charges rédigé en langage naturel, il convient de suivre une structure et des règles de présentation. Ceci n'est pas encore suffisant pour établir un moyen d'analyse fiable facilitant le passage vers une représentation formelle pour une première vérification. Il convient alors d'utiliser une représentation semi-formelle qui puisse atteindre les buts suivants :

- Aider l'analyse des diverses représentations du système selon la bonne abstraction,
- Faciliter la traduction en modèles HiLeS.

Nous aborderons ce point dans le paragraphe suivant.

2.2.4 Harmonisation avec l'approche UML/SysML

La conception logicielle, guidée par les processus « Model Driven Design » (MDD), a été un élément de référence pour développer des outils d'aide à la modélisation haut niveau de systèmes logiciels et les étendre à la conception système. Les analyses faites par Tse et Pong [TP91], sur plusieurs langages de spécifications, permettent de définir les principales caractéristiques de ces langages :

- Abstractions des modèles pour simplifier la représentation du système.
- Association de graphiques et de textes pour bénéficier de la simplicité de présentation des graphiques et la liberté d'expression du langage naturel.
- Cohérence des représentations maintenue après des manipulations.
- Indépendance de la conception et de la réalisation : la représentation doit être orientée vers l'expression du « quoi faire » sans prendre partie pour le « comment faire ».

Parmi les nombreuses méthodes de conception logicielles, trois d'entre elles ont su se regrouper et ajouter leurs qualités propres. D'une part, la méthode OOD [Boo93] gère la phase de conception et la construction du projet, d'autre part la méthode OOSE [Jac94] apporte les cas d'utilisation pour définir les exigences et pour l'analyse de la conception générale, et enfin, la méthode OMT [RBP91] est utile aux systèmes d'informations contenant une grande quantité de données.

La mise en commun des méthodes OOD, OMT et OOSE a permis de remodeler et d'unifier les représentations des systèmes logiciels dans une boîte à outil unique, proposée par Booch, Rumbaugh et Jacobson, appelée « Unified Modeling Language » (UML) [BRJ00]. L'essor d'UML dans le domaine des logiciels et le développement de plate-formes d'outils performants a naturellement conduit à envisager son utilisation dans l'ingénierie système. Pour cela, un effort d'adaptation et de normalisation a donné lieu à SysML (basé UML 2) en Janvier 2005 dont l'ambition est d'unifier la modélisation système et la modélisation logicielle. Cette norme pose les bases d'un langage qui ouvre une voie intéressante vers le prototypage virtuel.

En l'état, la porte d'entrée de la plate-forme HiLeS est la spécification textuelle. Il reste un pas important à franchir entre cette spécification textuelle et la représentation HiLeS tel que le signifie la flèche cerclée de noir sur la Figure 2-12. Ceci fait apparaître l'intérêt d'une démarche plus approfondie de traduction des spécifications textuelles.

Pour combler en partie ce saut, nous avons travaillé avec l'équipe Ingénierie des Systèmes et Intégration (ISI) du L.A.A.S. pour proposer une approche complète de conception système vers le prototypage virtuel, basée sur les composants [EOP05]. Elle s'appuie sur une première modélisation des spécifications par le biais de diagrammes SysML.

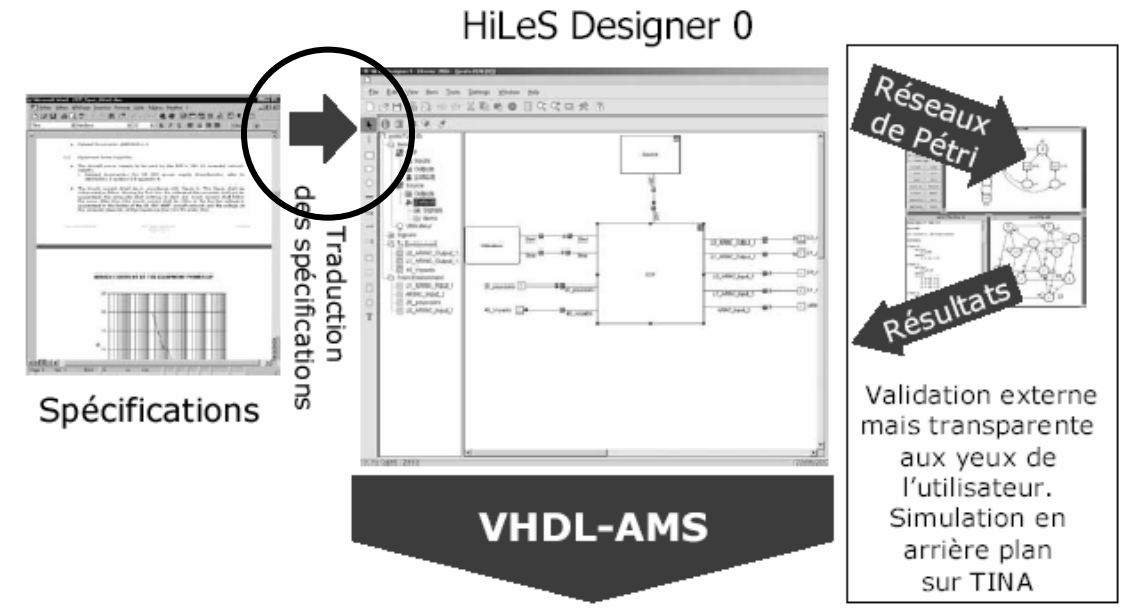


Figure 2-12 : La plate-forme HiLeS Designer [Ham05].

En fait, cette modélisation privilégie une approche par composants qui adopte une démarche globalement descendante. Les avantages sont les suivants :

- Distinguer des éléments du système, en les regroupant, par exemple, par domaines responsables de l'hétérogénéité du système, ce qui permet une diminution de la complexité de description du système en diminuant le nombre d'éléments, pour envisager d'obtenir : 7 plus ou moins 2 éléments visibles [Mil56].
- Augmenter les possibilités de réutilisation en encapsulant des solutions techniques par composant.
- Permettre, par l'approche descendante, de décrire les composants à différents niveaux d'abstraction. Ceci mène à une structure composite de composants eux mêmes constitués d'assemblage de composants.
- Décrire les composants de plus bas niveau par des modèles comportementaux (tels que VHDL-AMS) qui permettront de valider, par simulation, la globalité du système.

Sur la base des représentations UML/SysML, la démarche proposée applique les étapes suivantes :

La première étape concerne l'identification des limites et des éléments du système pour définir son contexte environnemental.

A) Analyse des exigences de premier niveau

1) Description physique du problème.
2) Identification et description des éléments de l'environnement.
3) Diagramme de contexte général.
4) Identification des acteurs à l'aide des éléments de l'environnement.
5) Analyse des exigences temporelles.
6) Finalisation du diagramme de contexte.

La seconde étape est liée à la définition des interactions des éléments du système et de leur comportement interne.

B) Analyse fonctionnelle

1) Définir les cas d'utilisation.
2) Construire le diagramme de classes structurel (basé sur les éléments physiques de l'environnement).
3) Identifier les objets.
4) Décrire chaque Cas d'Utilisation (CU) :
   - Description textuelle du scénario.
   - Description dynamique par le(s) diagramme(s) de séquence et diagramme de collaboration.
5) Enrichir le diagramme de collaboration au fur et à mesure de la description des CU.
6) Décrire les comportements :
   - Identifier les objets actifs et passifs.
   - Décrire le comportement des objets actifs à l'aide de Réseaux de Petri.

La troisième étape consiste à définir des éléments de transitions vers la représentation HiLeS.

C) Traduction en des modèles HiLeS

1) Fiche composant (objet) sur le principe de classe.
2) Traduction SysML-RdP vers Hiles :
   - Objets → blocs.
   - Comportements discrets → RdP HiLeS.

2.2.5 Utilisation de la méthode UML/SysML sur notre exemple

Nous allons suivre et appliquer à notre exemple les trois phases de la méthode décrite dans le paragraphe 2.2.4.

L'étape A de cette méthode rejoint nos travaux sur l'élaboration du cahier des charges par la méthode d'analyse du besoin. Nous retrouvons notamment la notion de présentation du système dans son environnement, dans le but de faire ressortir les parties du système y étant liées ou interagissant. Cette première analyse aboutit au diagramme de contexte général (étape A.3). Ce diagramme fait apparaître tous les éléments de l'environnement en liaison avec le système (Figure

2-13). La méthode des composants suggère d'identifier les acteurs (A.4) et d'en déduire un diagramme de contexte final associé (A.6). Les acteurs consistent en une abstraction des éléments en les regroupant par exemple selon les critères suivants :

- Rôle de l'élément,
- Interaction entre éléments,
- Nature des éléments,
- Etc.

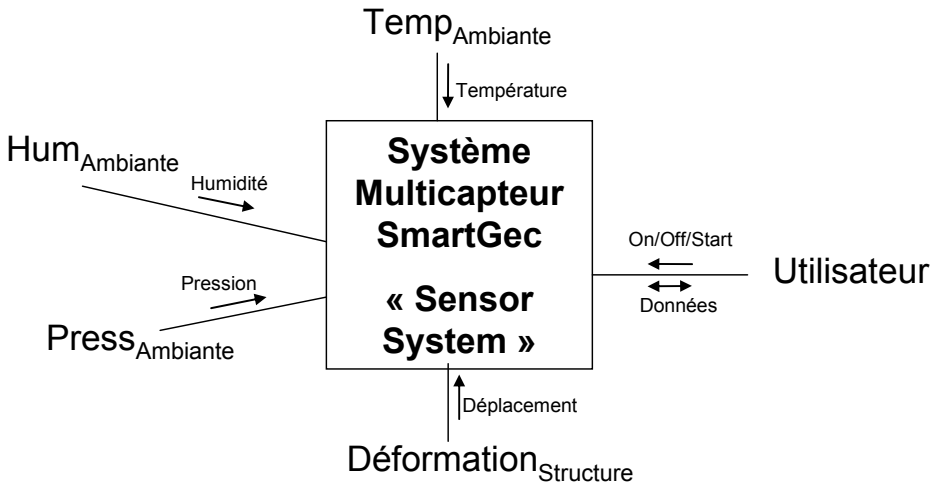


Figure 2-13 : Diagramme de contexte général de notre système.

Si nous appliquons les critères suivants : le rôle, la nature, l'interaction, nous regroupons les éléments de l'environnement et obtenons une vue du système à plus haut niveau (Tableau 2-1) qui regroupent les éléments relatifs à l'ambiance par un acteur unique : Structure béton.

Eléments de l'environnement	Eléments haut niveau (acteurs)
$Temp_{Amb}$	Structure Béton
Hum_{Amb}	
$Press_{Amb}$	
$Déformation_{Structure}$	
Utilisateur	Utilisateur

Tableau 2-1 : Eléments de l'environnement et éléments haut niveau.

Nous faisons ainsi apparaître 2 acteurs (cellules grisées):

- Structure Béton.
- Utilisateur.

Cette vue haut niveau du système permet de représenter le diagramme de contexte finalisé (A.6) représenté sur la Figure 2-14. Les principaux acteurs apparaissent autour du système.

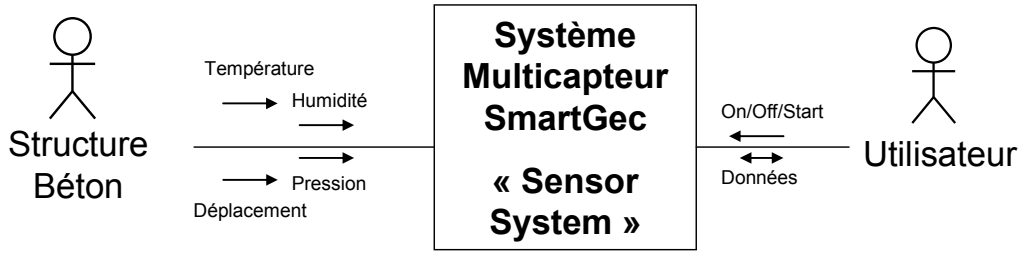


Figure 2-14 : Diagramme de contexte finalisé.

L'étape B a pour objet d'analyser le comportement dynamique du système. Tout d'abord, les cas d'utilisation ou les modes de fonctionnement doivent être définis (B.1). L'analyse, faite dans le cahier des charges et reportée dans le paragraphe 2.2.2, nous fournit quatre cas d'utilisation principaux qui décrivent l'activité du système par des procédures textuelles : *Initialiser*, *Tester*, *Gérer l'énergie*, et *Surveiller Structure*. De plus, le scénario *Surveiller Structure* est décomposé en trois cas d'utilisation : *Mesurer*, *Mémoriser* et *Communiquer*. La Figure 2-15 fait apparaître ces cas d'utilisation et leurs liaisons avec les acteurs en gardant une représentation simple, sans ordonnancement temporel.

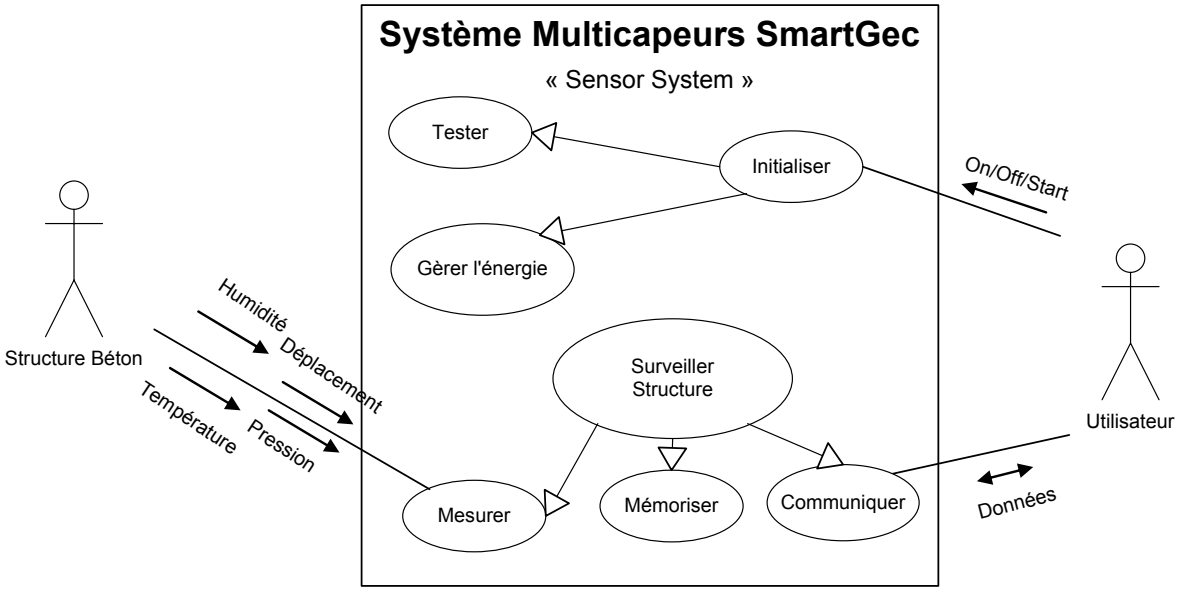


Figure 2-15 : Cas d'utilisation du fonctionnement du système.

Nous sommes dans la configuration particulière d'un système qui intègre ses capteurs, sa communication et son alimentation et qui est noyé dans un environnement sur lequel il ne peut pas agir. L'acteur *Structure Béton* a une interaction faible avec le système dans le sens où son interaction est unidirectionnelle puisqu'il ne fait qu'apporter des paramètres au système. A ce stade de la description, nous devons tenir compte de la contrainte forte du cahier des charges qui est de définir le système selon des classes d'éléments génériques. En nous appuyant sur la Figure 1-1 du premier chapitre qui décrit un microsystème générique, nous pouvons faire apparaître dans le système, les classes matérielles standard suivantes :

- Capteurs.
- Alimentation.
- Médium de communication.
- Unité de traitement.

Etant donné que ces classes matérielles sont intégrées au système (par exemple la classe Capteurs), nous pouvons créer des classes virtuelles associées (par exemple CapteurV) qui regroupent les fonctionnalités propres de ces objets. Nous sommes toujours dans l'analyse fonctionnelle du système (nous ne considérons pas les aspects non fonctionnels), ainsi toutes les instances virtuelles des classes se retrouvent dans le composant *Unité de traitement* qui est l'image des fonctionnalités du système. Ces classes sont utilisées pour définir les objets qui interviennent dans les diagrammes de séquence.

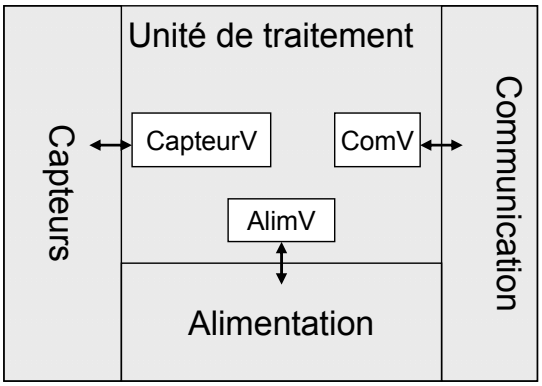


Figure 2-16 : Lien entre classes physiques et classes virtuelles.

La méthode suggère de présenter un diagramme structurel (Figure 2-17) qui est basé sur l'analyse des éléments de l'environnement du système (Tableau 2-1) et les classes définies plus haut, et d'en identifier les objets (B.3). Nous faisons ainsi apparaître dans notre cas les classes virtuelles qui composent la classe *Unité de Traitement* (UT).

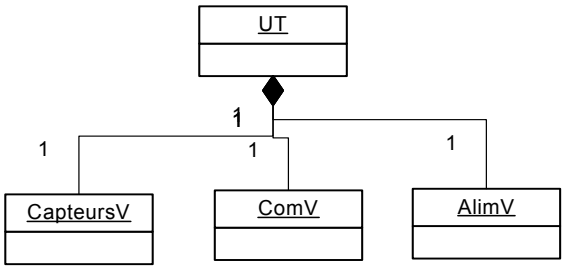


Figure 2-17 : Diagramme structurel de l'unité de traitement.

Une analyse plus fine, consiste à considérer les scénarios et à les attribuer à divers objets (liaisons des cas d'utilisation). Cette allocation nous conduit à un découpage différent et plus fin de la classe Unité de Traitement, représenté sur la Figure 2-18. Elle est composée des quatre objets suivants : Mémoire, Gestionnaire temporel, Analyseur, Initiateur.

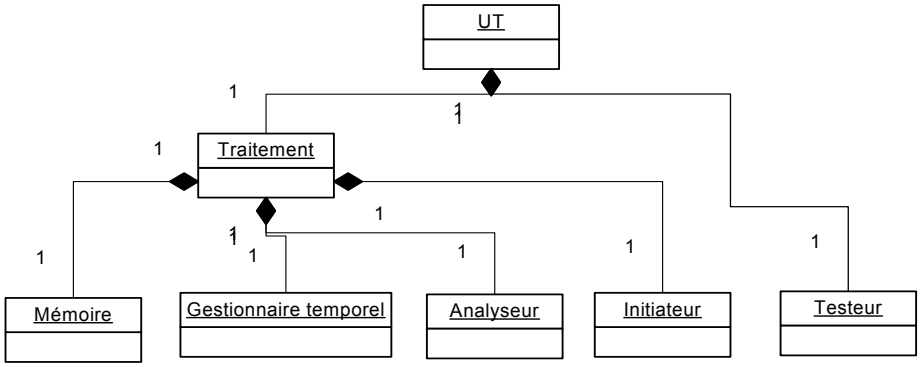


Figure 2-18 : Décomposition de l'Unité de Traitement (UT).

Il est nécessaire de poursuivre la méthode pour décrire la dynamique du système (B.4) par la représentation des **diagrammes de séquence et de collaboration**. Chaque scénario est détaillé par un **diagramme de séquence** propre qui fait apparaître les objets concernés, les messages transférés entre ces objets et l'aspect temporel du séquencement. Les contraintes de temps peuvent ici être renseignées par la dimension des barres d'activation. La Figure 2-19 illustre le diagramme de séquence du scénario « Mémoriser ».

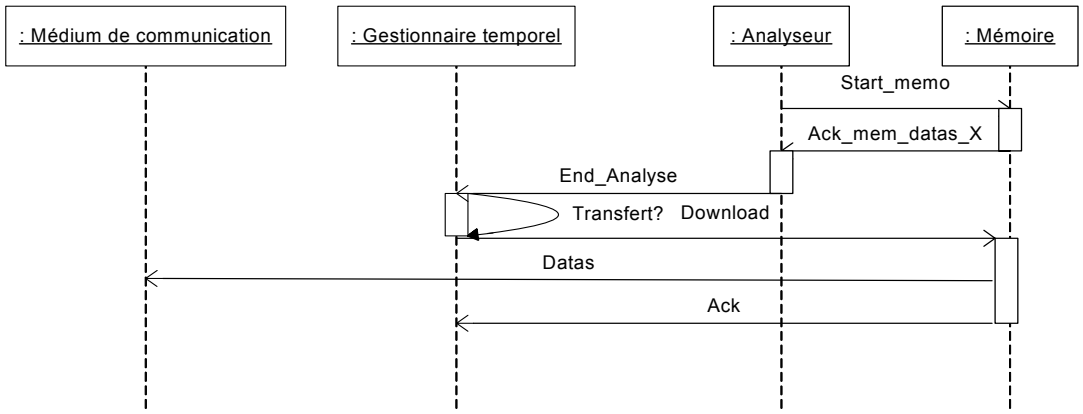


Figure 2-19 : Diagramme de séquence du scénario « Mémoriser ».

L'autre vue proposée pour représenter la dynamique du système est le **diagramme de collaboration**. Il met à plat les rôles que jouent les objets pour chaque scénario et présente leurs interactions réciproques. Le diagramme de collaboration correspondant au scénario « Mémoriser » est représenté sur la Figure 2-20.

Nous pouvons ainsi définir le diagramme de collaboration de chaque scénario. Aussi, comme le suggère la méthode (B.5), nous pouvons constituer un diagramme de collaboration complet correspondant à tous les scénarios. Cette vue a l'avantage de regrouper tous les messages échangés entre les objets mais de ce fait souffre d'un manque de lisibilité lorsque le système s'étoffe. Nous préférons ainsi différencier chaque diagramme de collaboration.

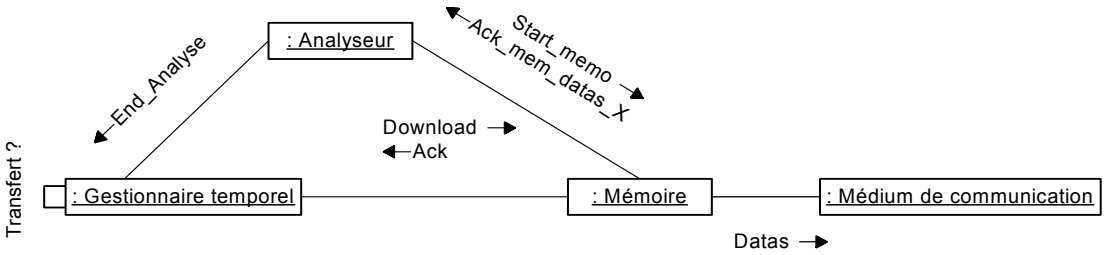


Figure 2-20 : Diagramme de collaboration du scénario « Mémoriser ».

Nous disposons maintenant d'une représentation de la dynamique d'interaction des objets. L'étape qui suit (B.6) pose la question suivante : Comment décrire le comportement interne des objets, les services rendus au système ? Nous proposons ici aussi de nous appuyer sur les diagrammes de séquence qui font apparaître les échanges de messages pour chaque scénario. Nous pensons qu'il est important de garder la vision séparée de chaque scénario pour bâtir le comportement global des objets à l'aide de réseaux de Petri. En effet, nous avons montré que le comportement de chaque objet est une superposition de réactions relatives aux scénarios joués. Ainsi, pour définir le comportement global d'un objet, il faut composer les diagrammes de séquence pour déployer les interactions de chaque objet avec ses voisins, tout en gardant la notion du temps, de la synchronisation et des pré et post conditions logiques qui caractérisent le déroulement d'une séquence.

Les objets qui apparaissent dans les diagrammes de séquence et qui agissent dynamiquement avec le système sont dits « actifs » alors que les autres sont dits « passifs ». Il y a ici un choix de conception à faire. Nous sommes dans le cas d'un système hétérogène qui contient des messages logiques mais aussi des messages analogiques. Le concepteur doit les différencier en prévision du passage vers le prototypage virtuel. La description de l'objet est complète après avoir consigné les propriétés de la classe à laquelle il appartient :

- Exigences non-fonctionnelles :
  - Données publiques.
  - Données privées.

- Comportement :
  - Externe (relations avec les objets de même niveau).
  - Interne (décrit à un niveau inférieur).

- Méthodes :
  - Traitements de données fournies aux autres classes.

A ce stade, nous devons considérer les contraintes dures du cahier des charges. C'est dans le but de réaliser un système générique que nous nous appuyons sur des fonctions standard communes à ce type de système.

L'étape C nous présente une voie de transition vers la représentation du système dans HiLeS. L'analyse SysML nous a permis de faire apparaître et de classer les différents comportements du système dans des composants adéquats (C.1). Ainsi, le concepteur a choisi les propriétés de chaque composant du système. Les diagrammes de collaboration sont très proches de la représentation HiLeS, du moins sur la partie architecturale du système. En effet, la méthode propose de faire l'analogie directe entre un objet et un bloc HiLeS, et entre un comportement discret et un réseau de Petri (C.2). Sur cette base, nous bâtirons le système sous HiLeS.

2.2.6 Bilan et recommandations

Nous avons posé le problème général de la conception amont en 2.2.1. Nous l'avons dans un premier temps abordé par la voie du bon sens en proposant une réécriture du cahier des charges en fonctions et procédures. Cette première approche intuitive, orientée « fonctions » nous a permis d'une part, de classer et ordonner les « fonctions objectifs » du système et d'autre part, de clarifier la logique temporelle de son comportement. Ce travail a abouti à un document textuel de référence pour la définition du système. Il nous restait un pas important à franchir pour converger vers la représentation HiLeS. Nous avons alors fusionné notre première approche à l'approche orientée objet SysML. Cette dernière permet de combler une partie du saut qui existe entre le cahier des charges et la représentation HiLeS. Elle aide à structurer l'analyse du système, à en dégager ses principaux acteurs et son fonctionnement dynamique. La démarche suit trois étapes qui regroupent les représentations de diagrammes UML suivants :

- Diagramme de contexte général.
- Diagramme de cas d'utilisation.
- Diagrammes de séquence (illustrent les scénarios des cas d'utilisation).
- Diagramme de collaboration.

L'avantage de l'approche fonctionnelle est le découpage clair des fonctionnalités attendues du système, en éléments fonctionnels simples. Ce qu'apporte en plus l'approche objet, c'est de considérer à la fois ce que fait le système mais aussi ce qu'est le système. En effet, par cette méthode, on définit les objets constitutifs du système qui peuvent être tangibles ou immatériels. Cela permet de classer et d'encapsuler les constituants élémentaires d'un système selon leurs nom, attributs et comportements internes. Ainsi, la définition claire de l'objet, en fait une unité faiblement couplée à son environnement et facilement portable et réutilisable dans d'autres contextes. L'objet ainsi créé échange les services rendus par ses fonctionnalités (méthodes) internes avec les autres objets et reçoit de la même manière les services rendus par les autres objets. Lorsque nous parlons d'objets, il ne s'agit pas de faire un choix matériel sur une solution possible, mais simplement de définir ce que devraient être les constituants du système. Ainsi, l'agencement et l'interconnexion entre ces objets définiront la première architecture du système.

Le passage d'une représentation d'un modèle haut niveau UML/SysML du système vers HiLeS est donc clair sur le point des objets. Ceux ci seront définis par l'association de blocs structurels et fonctionnels.

Ces analyses ont ajouté un certain nombre d'étapes supplémentaires à la phase de conception amont. Sont-elles toutes pertinentes ? Dans notre cas, nous pensons avoir bénéficié du nombre des descriptions ou vues du système pour étoffer sa compréhension et détailler son fonctionnement. Il faut décomposer, scinder, détailler pour ensuite mieux regrouper, agréger les

parties élémentaires et bâtir le système complet. Nous retrouvons ici typiquement l'idée de déconstruction de la branche descendante du cycle en « V » (top-down) pour ensuite réunir et implémenter dans la phase ascendante du cycle.

Il reste la question complexe du passage du modèle amont vers le modèle de comportement qui intègre à la fois l'aspect logique séquentielle et analogique. Dans HiLeS, nous disposons de réseaux de Petri et de blocs fonctionnels. L'analyse faite ici nous aide à construire un réseau de commande dicté par les messages échangés dans les diagrammes de séquence. La partie analogique du comportement du système n'est pas traitée mais peut-on l'apparenter aux méthodes fournies par les classes de composants ? Nous aborderons ce point dans le chapitre 3 par la simulation.

2.3 Modélisation HiLeS du microsystème

Avant de présenter la modélisation du système dans HiLeS, nous allons exposer le formalisme HiLeS.

2.3.1 Le formalisme HiLeS

Nous présentons ici les principaux éléments qui définissent le formalisme HiLeS. Ce formalisme est le fruit des travaux successifs accès sur la théorie [Jim00] et sur le développement et l'utilisation du logiciel HiLeS [Ham05].

Ainsi, dans le contexte industriel de conception des microsystèmes pluridisciplinaires que nous avons décrit dans le chapitre précédent, la plate-forme de conception HiLeS a un double objectif : d'une part, être un outil d'aide à la conception pour l'équipe chargée de l'étude de faisabilité du système et d'autre part, être une plate-forme ouverte aux fournisseurs pour qu'ils puissent valider leur propre contribution dans le système global. Nous évoquons ici deux points importants qui sont le cœur de la problématique du prototypage virtuel ; concevoir suivant les spécifications, puis simuler et vérifier les propriétés du système.

Pour cela, HiLeS fournit les éléments de base pour décrire le système pluridisciplinaire à haut niveau. Nous avons un ensemble de blocs (structurels et fonctionnels), des canaux de communication et des réseaux de Petri qui composent la description d'un système sous HiLeS. Les blocs et les canaux de communication permettent de présenter de manière hiérarchique la structure fonctionnelle du système alors que le formalisme des réseaux de Petri permet de modéliser les états de fonctionnement de manière formelle.

2.3.1.1 Les blocs

Les **blocs fonctionnels** permettent de décrire le comportement du système sous la forme d'équations algébriques, logiques ou différentielles. Ces équations sont modélisées en langage VHDL-AMS. Ces blocs représentent la plus fine description fonctionnelle du système. Ils sont représentés par des rectangles aux coins arrondis (Figure 2-21.1).

1. Bloc Fonctionnel 2. Bloc Structurel

Figure 2-21 : Blocs HiLeS.

Les **blocs structurels** permettent la décomposition hiérarchique du système. Ils peuvent être composés de blocs structurels, de blocs fonctionnels ou de réseaux de Petri. Ils sont représentés par des rectangles anguleux (Figure 2-21.2). Ces blocs sont utilisés pour la décomposition

hiérarchique de système et pour l'agrégation de blocs pour former un composant matériel exploitable.

2.3.1.2 Les canaux

Les canaux de communication sont le moyen d'échanger des données inter blocs et entre les blocs et les réseaux de Petri. Les canaux transportent des signaux continus, des signaux discrets et des arcs de Petri (Figure 2-22).

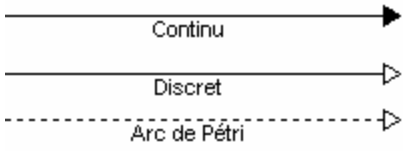


Figure 2-22 : Canaux de communication HiLeS.

Les signaux continus sont utilisés dans le cas de la modélisation de signaux de données analogiques tels qu'un déplacement, une pression ou une tension. Tandis que les signaux discrets sont utilisés pour leurs attributs logiques (0 ou 1), pour modéliser par exemple l'événement de détection de fautes, d'un seuil ou tout autre événement discret.

Les Arcs de Petri sont des signaux discrets particuliers, utilisés d'une part pour la description du réseau de commande (liens entre les places et les transitions du réseau) et d'autre part, pour établir le lien entre le réseau de commande et les processus gérés par les blocs.

2.3.1.3 Le modèle de commande

HiLeS utilise le formalisme des réseaux de Petri ordinaires qui permettent la représentation graphique de systèmes concurrents et parallèles. Le réseau comporte trois éléments particuliers : un ensemble fini de places symbolisées par des cercles, un ensemble fini de transitions symbolisées par des tirés et un ensemble fini d'arcs orientés qui assurent la liaison d'une place vers une transition ou d'une transition vers une place.

Les places constituent la matérialisation d'une condition (état du système) alors que les transitions représentent l'événement qui modifie l'état du système. HiLeS utilise les réseaux dits ordinaires puisque le tir d'une transition est validé à la condition que les places amont soient marquées d'un jeton et seulement un.

Les arcs du réseau de Petri connectés aux blocs seront considérés comme les « connecteurs » par lesquels les processus internes au bloc sont déclenchés. Le déclenchement effectif a lieu lorsqu'un arc amène un jeton dans le bloc cible.

2.3.1.4 Représentation type dans HiLeS

A partir des éléments définis dans les paragraphes précédents, nous pouvons ainsi représenter des systèmes à très haut niveau. La Figure 2-23 illustre un système type élaboré sous HiLeS. Ainsi, nous faisons apparaître sur la Figure 2-23.1 trois blocs structurels interconnectés par des canaux continus (in_1, in_0 et out_1), des canaux discrets (out_0) et des canaux discrets relatifs aux arcs de Petri (C1, C2, S1, S2). La Figure 2-23.2 illustre la structure interne du bloc Structural_0. Ce bloc structurel est composé d'un bloc structurel (Structural_5), un bloc fonctionnel (Functional_0) et d'un ensemble de places, de transitions et d'arcs de Petri qui forment un réseau de commande.

Nous voyons ainsi la transition entre une représentation du système par un assemblage de blocs vers une représentation fonctionnelle temporisée par le biais de réseaux de Petri. Cette construction hiérarchique peut être réalisée comme ici, dans le but de **décomposer** un bloc structurel, ou à l'inverse, elle peut être utilisée de manière à **agréger** plusieurs blocs et réseaux de commande en un bloc structurel.

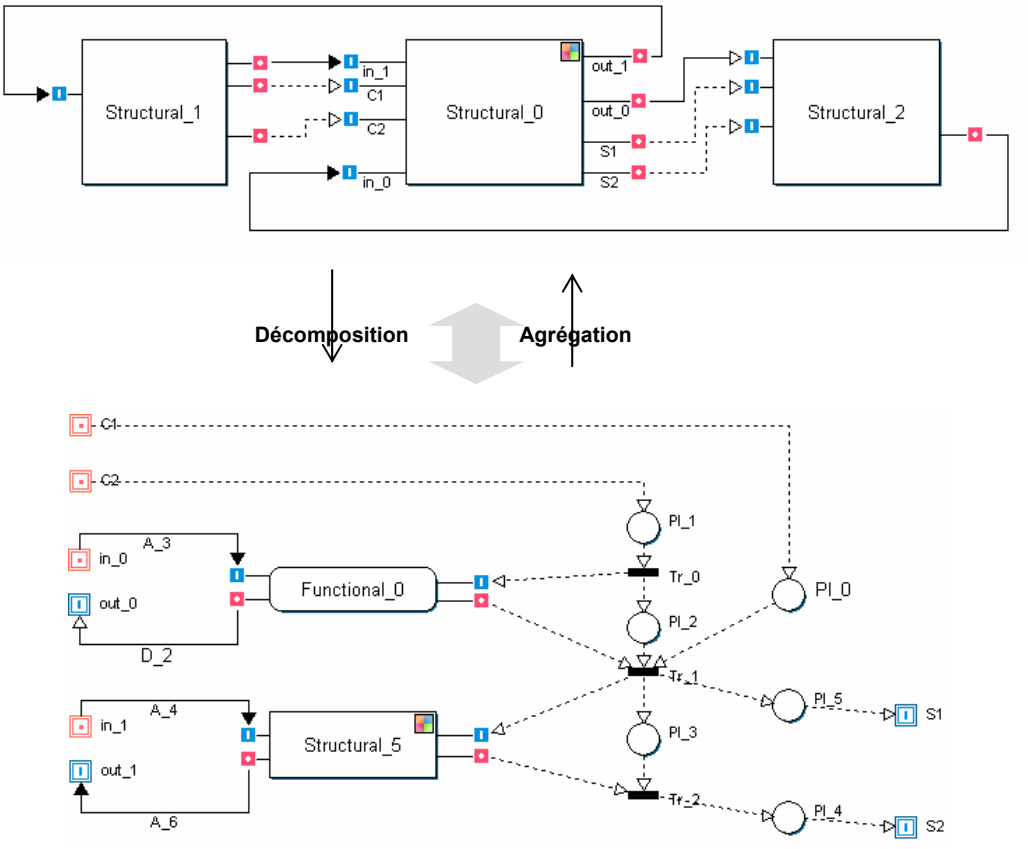


Figure 2-23 : Représentation type sous HiLeS :

Agrégation et décomposition du bloc Structural_0.

Dans HiLeS, le formalisme de communication entre le réseau de Petri et les blocs structurels ou fonctionnels peut être comparé au mode de communication asynchrone avec retour différé. Prenons l'exemple du bloc Stuctural_5 de la Figure 2-23.2. Nous avons en effet dans un premier temps la transition Tr_1 qui émet un ordre d'initiation d'activité. Cet ordre est instantané et indépendant de l'activité du bloc Structural_5. Dans un deuxième temps, la fin d'activité est donnée par le tir de la transition Tr_2 qui vaut un temps non nul correspondant au temps nécessaire à la terminaison de l'activité du bloc Structural_5.

2.3.2 Mise en œuvre sur l'exemple

La définition du système s'est nourrie d'une approche double : à la fois de l'analyse fonctionnelle et de l'approche UML/SysML. Nous avons ainsi élaboré la représentation du système sous la forme de blocs encapsulant des fonctions et de la logique séquentielle, à la manière du formalisme HiLeS.

Dans notre démarche, HiLeS est une passerelle qui permet deux étapes de conception :

- La première consiste à décrire le système, sa hiérarchie, une hypothèse architecturale, sa dynamique de logique temporelle et le type des données échangées entre les composants, sans en préciser leur nature particulière.
- La seconde est le prototypage virtuel qui introduit la notion de données et de manipulation des données par le biais de la simulation VHDL-AMS.

Nous illustrons la première étape dans ce paragraphe. La description complète du système est présentée a fait l'objet d'un rapport interne [Mau04a].

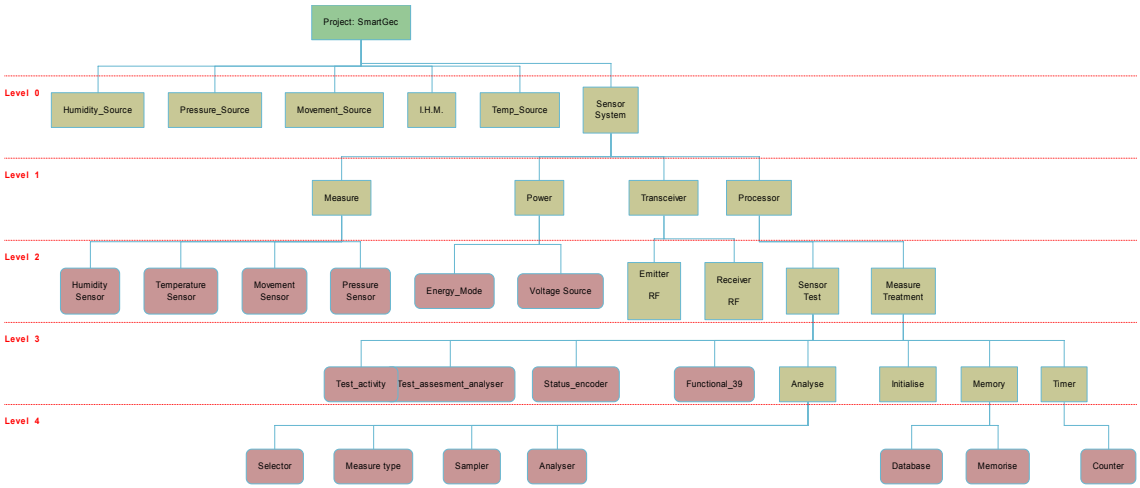


Figure 2-24 : Décomposition hiérarchique du système SmartGec.

La hiérarchie globale du système est représentée sur la Figure 2-24. Cette représentation est extraite automatiquement du projet décrit dans HiLeS.

Notre **système multicapteurs SmartGec** est symbolisé par le bloc **« Sensor System »**. Il est constitué de quatre niveaux hiérarchiques auxquels s'ajoute un niveau 0 qui représente l'environnement du système.

2.3.2.1 Le niveau 0 : L'environnement du système

Nous exploitons le diagramme de contexte décrit par la Figure 2-14 de la section 2.2.5 pour définir ce niveau 0. Nous avons choisi de représenter les acteurs « structure en béton » et « utilisateur » d'une manière différente. La « structure en béton » est présentée sous la forme de ses paramètres physiques qui intéressent le système (Température, Humidité, Pression, Déplacement) et que l'on détaille pour des raisons de compréhension. L'« utilisateur » quant à lui est vu à travers l'« Interface Homme Machine » qui communique avec le système au moyen de signaux de commande (Figure 2-25).

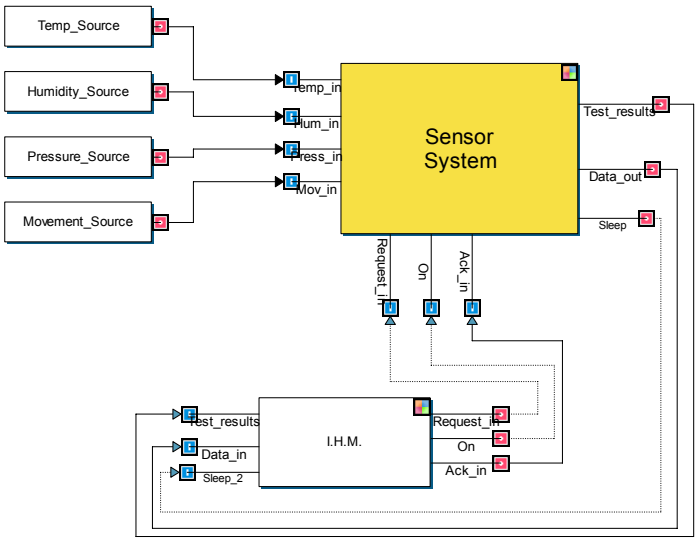


Figure 2-25 : Environnement du système.

2.3.2.2 Le niveau 1 : Les composants ou les acteurs principaux

Le niveau 1 est la représentation des acteurs principaux qui découlent de l'analyse SysML. Nous devons préciser que la classe SysML « médium de communication » est remplacée par le composant « Emetteur/Récepteur ». Cette correspondance est guidée par les contraintes

technologiques imposées par le CDCF qui amènent des solutions techniques génériques. Il s'agit ici d'un choix sur du médium de communication Radio-Fréquence.

Nous distinguons sur la Figure 2-26 les quatre composants principaux du système représentés par des blocs structurels :

- **« Alimentation »** : constitue la source d'énergie du système et gère la distribution d'énergie vers le système,
- **« Capteurs »** : intègre l'ensemble des fonctions relatives à la détection et à la conversion de phénomènes physiques de l'environnement en signaux identifiables et mesurables,
- « **Processeur »** : regroupe l'ensemble du traitement et de l'analyse des signaux,
- **« Emetteur/Récepteur »** : constitue l'interface de connexion du système vers « l' I.H.M. ».

Nous comparons cette vue aux diagrammes de collaboration définis en section 2.2.5. En effet, un diagramme de collaboration doit représenter les liens qui existent entre chaque composant (modélisation des échanges de messages) pour un cas d'utilisation. Nous avions donc considéré autant de diagrammes que de cas d'utilisation. Cette vue est la composition de tous les diagrammes de collaboration. Ainsi, elle intègre tous les liens qui apparaissent sur chaque diagramme, ce qui rend la lecture de la figure peu aisée.

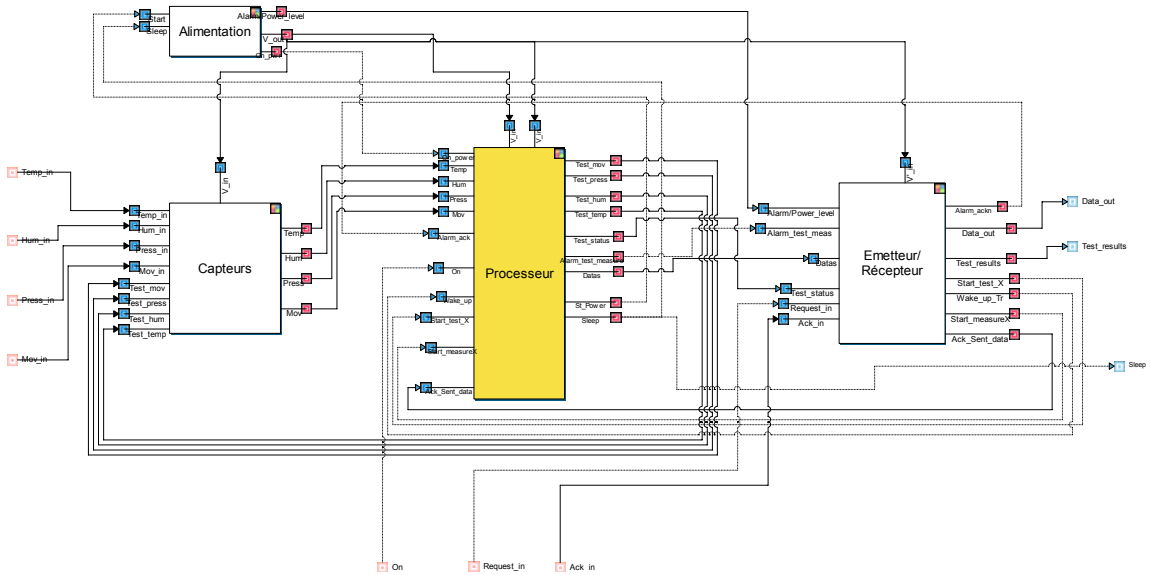


Figure 2-26 : Composants principaux du système = niveau 1.

Nous pouvons imaginer utiliser un filtre pour réduire le nombre de signaux visibles et ne sélectionner que les signaux selon leur nature. Ceci serait intéressant dans le cadre de la première analyse de la logique séquentielle faite par TINA, qui n'utilise que les signaux de commandes (arcs de Petri). La représentation du système serait alors allégé et l'on ne visualiserait plus que les arcs de Petri.

2.3.2.3 Les niveaux 2, 3, 4 : La décomposition des composants principaux

Le découpage des « composants blocs » principaux est, comme nous l'avons vu dans la section 2.2.2, dicté par le cahier des charges et a été guidé par l'analyse des cas d'utilisation et l'élaboration des diagrammes structurels vus en section 2.2.5. Il s'agit d'allouer les fonctionnalités ou les méthodes exigées à chaque composant bloc. La décomposition peut se faire sur autant de niveaux que nécessaires afin d'obtenir une fonction élémentaire. Le découpage est guidé par les cas d'utilisation mais reste ouvert et laisse au concepteur le choix de l'architecture. A ce niveau, le concepteur doit s'appuyer sur une bibliothèque de composants et sur des données expertes qui le guideront dans ses choix.

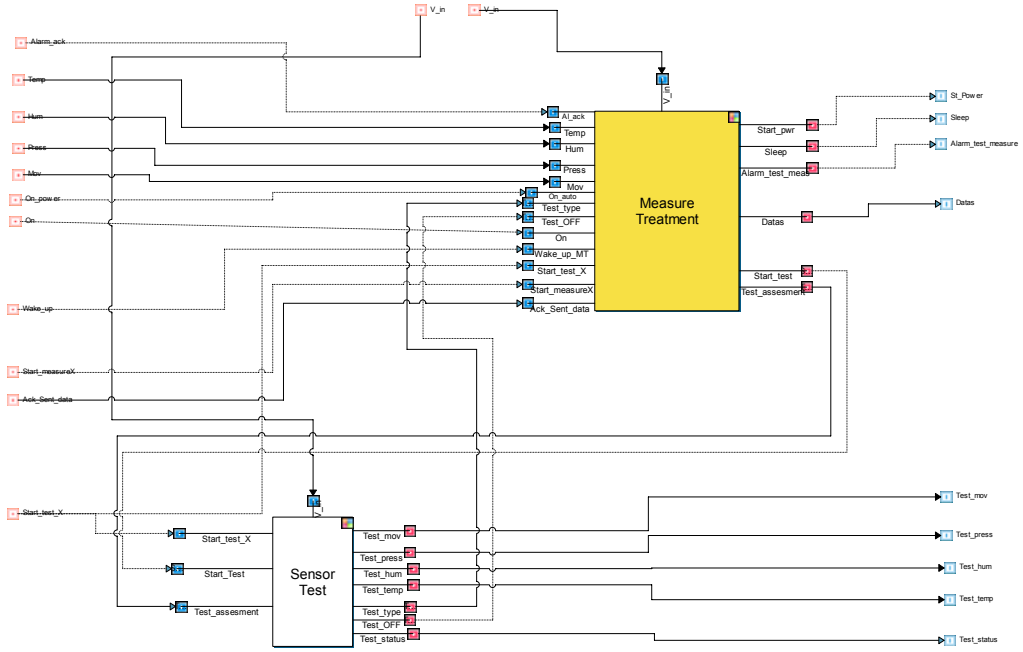


Figure 2-27 : Décomposition du bloc Processeur = niveau 2.

Nous prenons comme exemple la décomposition du bloc structurel principal « Processeur » : La Figure 2-27 illustre le niveau 2 qui intègre deux sous-blocs :

- **« Measure Treatment »** : un bloc structurel de traitement et d'analyse des mesures,
- **« Sensor test »** : un bloc structurel de génération et d'analyse des signaux test.

Tous les signaux interfaces du bloc « supérieur » sont reportés et alloués aux blocs de niveaux inférieurs, pour garder la cohérence avec les diagrammes structurels.

La décomposition se poursuit afin d'obtenir une représentation par blocs des cas d'utilisation. Ainsi, nous retrouvons quatre sous-blocs dans le bloc « Measure Treatment » (Figure 2-28) :

- **« Timer »** : synchronise l'activité et les modes de fonctionnement du système,
- **« Memory »** : mémorise les données nouvelles et celles de référence,
- **« Initialise »** : gère le déroulement et l'enchaînement des activités du système,
- **« Analyse »** : sélectionne, échantillonne et analyse les signaux.

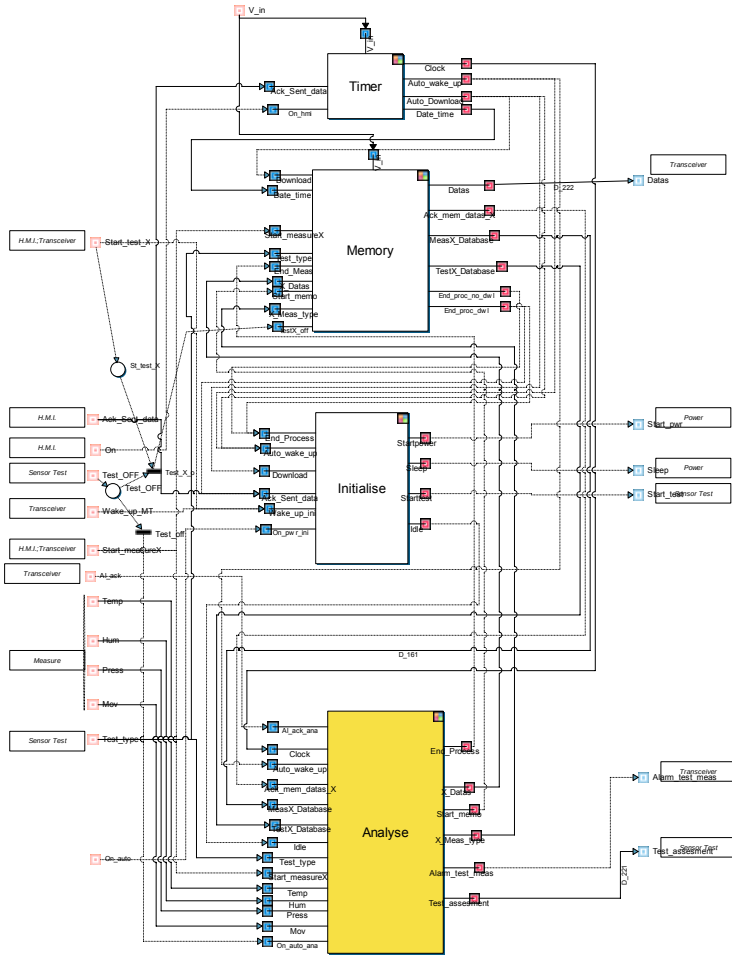


Figure 2-28 : Décomposition du bloc Measure treatment = niveau 3.

Dans la décomposition supplémentaire du bloc « Analyse » nous faisons apparaître ses fonctions internes élémentaires (blocs fonctionnels). La Figure 2-29 illustre la composition du bloc « Analyse » en fonctions élémentaires et réseaux de Petri. Le bloc « Analyse » est appelé composant élémentaire.

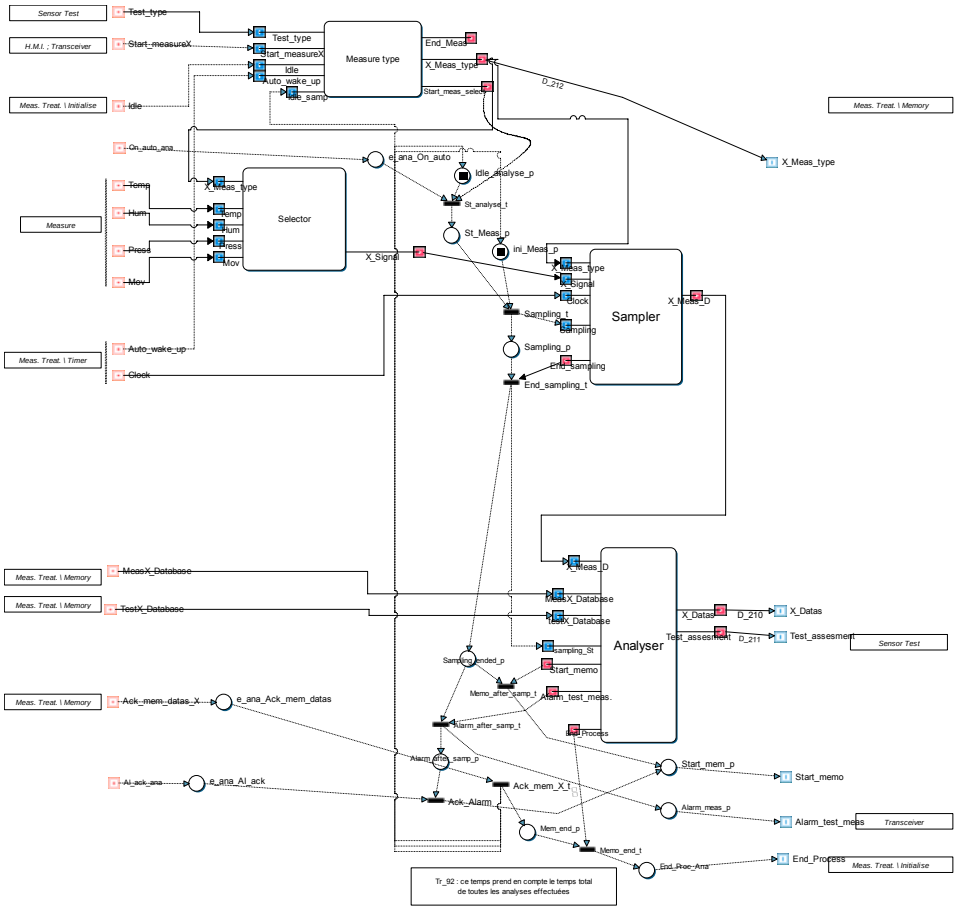


Figure 2-29 : Décomposition du bloc Analyse = niveau 4.

Le bloc « Analyse » est doté de quatre fonctionnalités élémentaires :

- sélectionner le type de mesure : **« Measure_Type »**,
- sélectionner le signal à mesurer : **« Selector »**,
- échantillonner le signal : **« Sampler »**,
- analyser la mesure : **« Analyser »**.

Ces quatre fonctionnalités sont activées par le biais d'un réseau de Petri déterminant leur début et leur fin d'activité. Ce stade de décomposition est le plus fin de la description puisque nous n'avons plus de blocs structurels à décomposer.

2.3.3 Difficultés rencontrées et recommandations

Nous avons montré comment HiLeS permet de construire un système par la décomposition successive de blocs structurels. Ce processus est efficace et déjà bien orienté par l'étude SysML préalable. Notre démarche rejoint les orientations des recommandations MDA (Model Driven Architecture), dans le caractère descendant de la modélisation fonctionnelle et l'intérêt de distinguer les modèles indépendants de la plate-forme (PIM : Platform Independent Model) et les modèles intégrés sur les supports techniques (PSM : Platform Specific Model).

L'application successive des formalismes UML/SysML puis des formalismes des réseaux de Petri peut être compris comme une succession de transformation de modèles. Il y a, entre le processus de conception MDA en « Y » et les processus en « V » que nous avons détaillés dans les paragraphes précédents des différences qui mériteraient d'être commentées davantage, mais cela n'apporterait rien à la démarche appliquée aux microsystèmes qui ont des problèmes de complexité moyenne. Nous donnons sur la Figure 2-30 le processus de base MDA. Nous pouvons constater la part de modélisation amont dans notre démarche : UML/SysML pour la formalisation

des exigences et HiLeS pour la modélisation fonctionnelle amont est la même que celle de la figure. Cette modélisation amont conduit, lorsqu'on lui apporte une description du support technique d'implémentation, à la modélisation technologique (prototype virtuel) par le biais du langage VHDL-AMS.

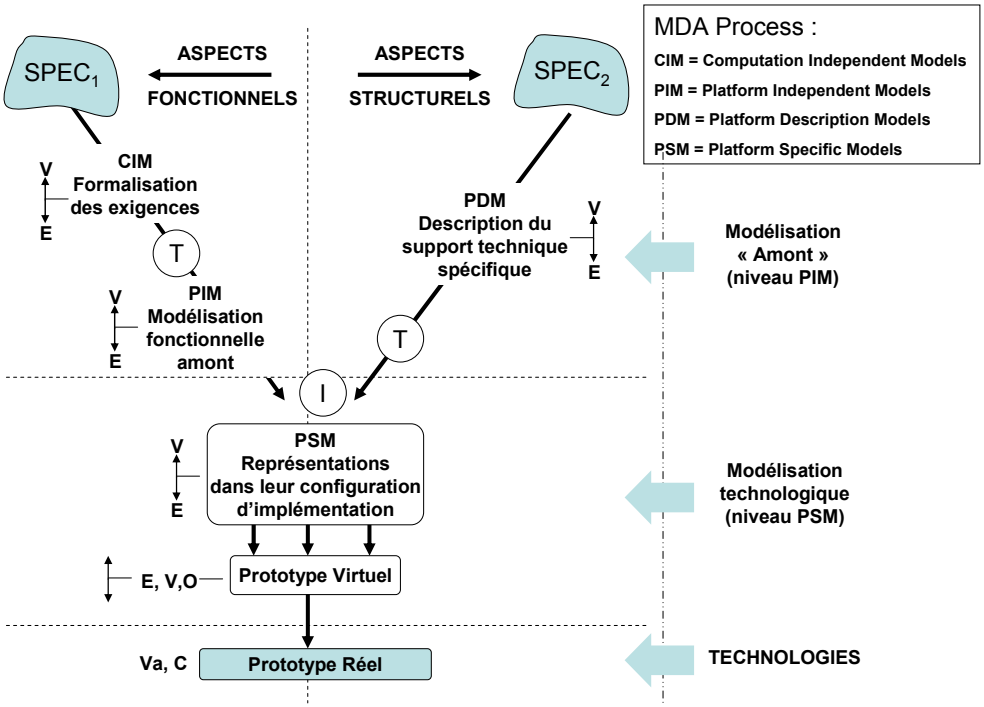


Figure 2-30 : Processus de conception MDA en « Y ».

Selon cette démarche, nous avons obtenu une première représentation statique structurelle hiérarchique du système. La partie la plus complexe de la construction est celle de l'intégration de la dynamique du système. Pour cela, nous nous sommes appuyés sur les diagrammes de séquence qui animent des composants élémentaires et définissent les signaux échangés. Nous constatons ici qu'il est plus simple d'établir les connexions des réseaux de commande qui ont été déterminées par les diagrammes de séquence une fois que toutes les fonctions élémentaires ont été allouées aux blocs du système. La construction statique du système doit donc précéder la construction dynamique. En effet, nous sommes dans une approche de modélisation pour la réalisation d'un prototype. Pour cela, il faut injecter au modèle haut niveau des contraintes de choix de modèles ou d'architectures qui puissent être implémentées, c'est équivalent à l'avis d'un expert ou d'une base de données experte.

Lors de la construction « top-down » du système, les ports de connexion doivent être nommés de manière explicite pour permettre de suivre le cheminement des informations dans le système. Dans l'approche SysML les messages échangés entre chaque composant sont nommés par un verbe d'action. Nous proposons de nommer non pas les arcs mais les ports relatifs aux arcs de Petri pour qu'ils soient représentatifs des actions qu'ils induisent (ex Figure 2-29 : 'Sampling', 'End_sampling', 'Start_memo', 'End_Process').

Dans cette section, nous avons montré comment établir une représentation du système sous HiLeS en affichant clairement l'aspect statique et dynamique du système.

2.4 La vérification sur TINA

Nous avons vu dans le paragraphe précédent le cheminement suivi pour obtenir une représentation graphique HiLeS de notre microsystème. Cette représentation associe un ensemble de blocs fonctionnels et structurels, des connexions discrètes et continues et des réseaux de Petri. Nous allons présenter dans les paragraphes suivants comment ces réseaux de Petri apportent le

moyen de vérifier. La vérification est l'argument clef de la démarche descendante de conception. Elle est le moyen de s'assurer que le contenu de la modélisation est conforme aux exigences du cahier des charges. Elle s'applique donc à toutes les étapes (transformation de modèles). Elle complète les opérations de suivi des exigences dont l'objet est de s'assurer que l'on a bien pris en compte une exigence repérée dans le cahier des charges. Idéalement, l'opération de vérification porte sur l'exigence initiale du cahier des charges transformé en propriétés exprimées dans le formalisme de vérification (réseau de Petri) que l'on va vérifier par simulation (TINA). La plate-forme HiLeS intègre un pont pour communiquer avec le logiciel TINA pour la simulation des réseaux de Petri. TINA est un logiciel développé par le groupe OLC (Outils Logiciels pour la Communication) du LAAS-CNRS [8]. Cette vérification peut conserver trois niveaux de vérification, d'une part la vérification des propriétés intrinsèques des réseaux de Petri, pour montrer la cohérence de la description, d'autre part la bonne temporisation du système, et enfin la structure architecturale du système.

2.4.1 L'objectif de la vérification formelle dans la plate-forme HiLeS

La vérification TINA porte sur une représentation du système réduite aux réseaux de Petri (RdP). Sur cette base, on peut accéder à une première vérification fonctionnelle comportementale du système. On veut avoir une vérification formelle qui seule apporte la preuve mathématique qu'une propriété du modèle est vérifiée avec un niveau de confiance de 100%.

Trois niveaux de vérifications sont accessibles :

- Les propriétés classiques générales du réseau de Petri (bornitude, blocage, réinitialisation).
- Des propriétés particulières exprimées par le concepteur à partir des exigences du cahier des charges et de son analyse du système tel qu'il le conçoit.
- Des propriétés architecturales pour valider ou éviter des erreurs de choix de découpe architecturale, en préparation au partitionnement notamment hard-soft.

Sur cette base, la démarche de vérification consiste à :

- Identifier puis exprimer des propriétés du système que l'on veut vérifier dans le formalisme HiLeS-RdP.
- Poser les questions à TINA.
- Analyser les retours.

Il s'agit, dans notre travail, d'une étape d'exploration des couplages HiLeS-TINA qui méritera de nombreux développements pour simplement identifier et exprimer les propriétés à vérifier. Nous remercions ici tous les collègues créateurs de TINA qui nous ont guidé dans cette première étape.

Le **premier niveau de vérification** aborde les propriétés générales d'accessibilité (espace d'états fini, absence de code mort, possibilité de réinitialisation du système).

Le nombre fini d'états de marquage (**aspect borné**) des places d'un réseau, global ou partiel, doit être vérifié avant tout autre propriété sans quoi le réseau est inexploitable pour l'application des méthodes formelles. Cette étape pose les bases à partir desquelles les vérifications des propriétés particulières sont possibles. Nous verrons l'importance de ce point dans le choix de la démarche de vérification d'un modèle de réseau de Petri. L'aspect non borné se traduit souvent par la présence d'une place du réseau dont le marquage contient une infinité de jetons. TINA fournit le moyen de localiser ces places et permet ainsi au concepteur d'analyser la modélisation localement, aux abords de cette place.

Le **code mort** (absence de vivacité ou blocage de certaines transitions) signifie que certaines transitions n'ont pas été sensibilisées à partir d'un état de marquage quelconque et accessible du réseau. Cette caractéristique est la marque d'un disfonctionnement de la modélisation. En effet, nous souhaitons avoir toutes les transitions du système vivantes et il serait inutile que la modélisation comporte des transitions mortes ne jouant aucun rôle dans la description des opérations du système. Il est donc intéressant de localiser de telles transitions afin de révéler une éventuelle erreur de modélisation. Aussi, dans une autre mesure, la détection d'une branche entière sans vivacité permet de visualiser le disfonctionnement du processus modélisé par cette

branche. Ceci est à mettre en parallèle à l'analyse faite par le concepteur qui doit, dans sa modélisation, associer des tronçons de réseaux aux propriétés du système.

La **réinitialisation** d'un système est liée à sa capacité à trouver une séquence de tir pour retrouver un état d'origine (marquage initial) en partant de n'importe quel état de son fonctionnement. La vérification de ce caractère est essentielle pour valider le fonctionnement cyclique d'automates.

Le **deuxième niveau de vérification** concerne les propriétés spécifiques du système. Pour cela, **il faut que le concepteur puisse confronter sa modélisation à des exigences de fonctionnement**. Ces exigences consignées dans le cahier des charges doivent décrire les réactions du système pour tous ses cas d'utilisation. Cela peut prendre la forme d'un texte descriptif ou d'un modèle de fonctionnement du système en suivant le formalisme SysML. Nous avons choisi dans un premier temps la solution textuelle mais nous avons travaillé sur la deuxième voie qui permet d'imaginer la vérification automatique du modèle par le biais de l'approche SysML. Le fondement de cette approche est de reprendre les modèles amonts du système pour valider l'implémentation fonctionnelle du système et ainsi garder une ligne cohérente dans la conception. Ainsi, on valide les propriétés du système par rapport au cahier des charges.

Le **troisième niveau de vérification** s'applique à l'architecture du système. Cette vérification permet au concepteur de visualiser de façon abstraite le comportement d'un bloc architectural ou de choisir un ensemble de blocs du système. L'analyse faite par TINA **permet l'observation du comportement au travers des entrées et des sorties des blocs**. TINA fournit le moyen de visualiser le comportement de blocs d'intérêt par une représentation par projections. Le concepteur doit les interpréter et les analyser pour les comparer au comportement attendu.

2.4.2 Mise en œuvre sur l'exemple

La modélisation HiLeS du système intégrant les réseaux de Petri s'est appuyée sur le découpage hiérarchique du système et sur sa description dynamique du comportement par les diagrammes SysML (séquence et collaboration, 2.2.5).

L'approche décrite dans [PDB05] propose une première étude de traduction de diagrammes de séquence en RdP pour lier UML à la vérification formelle. Dans l'exemple, un large éventail de types de liens entre objets UML sont traités : liens de communication synchrone, asynchrone, asynchrone avec retour différé, liens de contrôle d'itération, chien de garde. Ces modélisations types sont les éléments de base qui permettent d'accéder à la modélisation de systèmes fonctionnels complexes. L'analyse faite dans [PDB05] manipule et met en forme les objets par leurs interactions et les méthodes qu'ils appliquent sur les messages qu'ils reçoivent et émettent. Cette méthode concerne aussi les aspects temporels liés au traitement et à la propagation des messages entre les différents objets du système.

Dans notre exemple, nous n'avons pas utilisé de mécanismes automatiques pour la génération des réseaux de Petri. Mais nous nous sommes basés sur la description des réseaux de commande exposés en 2.2.5. En effet, nous avons choisi, dès la représentation SysML, de nous focaliser sur la représentation des messages de commande échangés dans le système, en les différenciant des signaux de données, pour faciliter le passage vers le formalisme HiLeS.

L'élaboration des réseaux de Petri sous HiLeS se base sur la description unitaire des cas d'utilisations par leurs diagrammes de séquence associés. Nous avons dû ainsi traduire n diagrammes de séquence en n réseaux de Petri. Cependant, tous ces réseaux de Petri sont intimement liés par le partage et l'utilisation de messages communs. De ce fait, la description HiLeS nous donne la possibilité de reconstruire la complexité du système. Nous avons vu que la vérification formelle par TINA doit s'effectuer à la suite de deux processus : d'une part, la succession des étapes détaillées en 2.2.5 dans le but de décomposer le système pour en simplifier la représentation (branche descendante de la Figure 2-31) ; d'autre part, HiLes donne le moyen de recomposer pas à pas la complexité du système par la composition du réseau global.

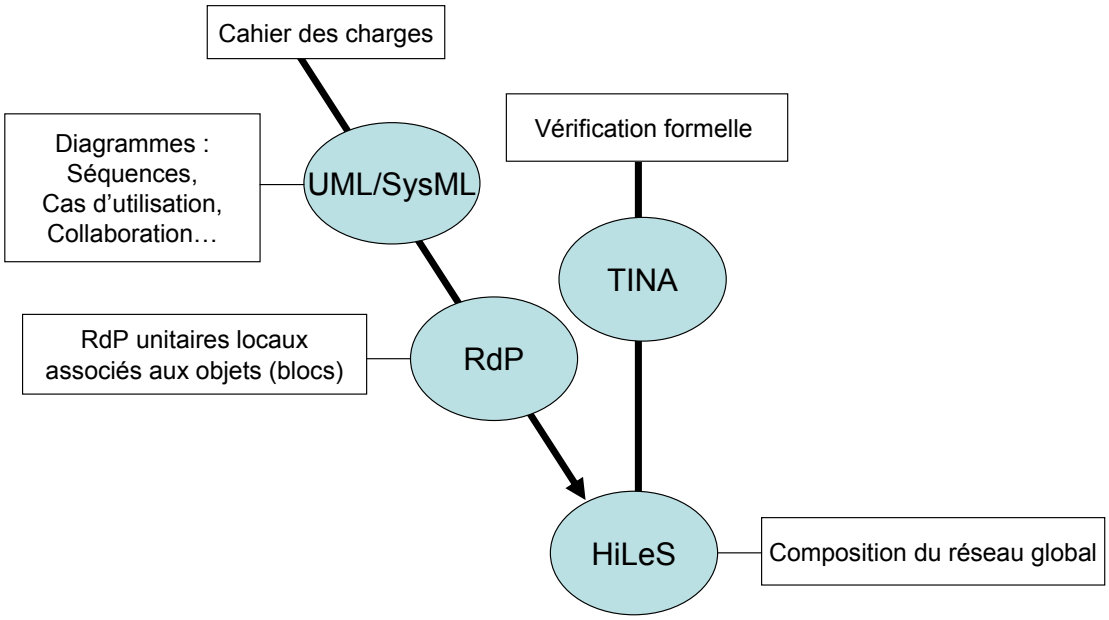


Figure 2-31 : Transformations successives conduisant à la vérification formelle.

Nous serions tenté de réaliser une première vérification formelle des réseaux de Petri unitaires locaux, relatifs aux blocs structurels, pour valider plus tôt le comportement dans son contexte local. Cette question ouvre la problématique de la vérification locale et globale. Si nous prenons chaque réseau de Petri indépendamment, nous nous apercevons qu'il est discontinu puisqu'il ne reflète qu'une partie du processus complet. La simulation de tels réseaux est complexe, elle peut être envisagée si le concepteur parvient à recréer à la fois l'environnement des signaux et leur dynamique.

La composition de tous les réseaux du système nous permet de former un tout simulable et vérifiable. Nous avons travaillé dans cette direction pour obtenir un réseau global représentant le comportement du système dans sa totalité. Le concepteur doit à cet instant introduire dans la modélisation une valeur à tous les intervalles de tir de transition, pour ainsi faire apparaître le facteur temporel de la modélisation.

La mise en œuvre des vérifications formelles sous HiLeS a regroupé 3 parties prenantes : le concepteur du logiciel HiLeS, l'analyste TINA, et le concepteur du microsystème. Nous avons mis en place une collaboration transversale au sein du LAAS [Cha05].

2.4.2.1 Vérification des propriétés classiques

Nous avons conduit les premières analyses classiques du réseau global extrait sous HiLeS. Ces analyses formelles du système ne peuvent être menées qu'après avoir obtenu un réseau borné. Pour cela, la première étape de vérification s'est portée sur cette propriété.

Cette analyse a permis de déceler d'une part des problèmes d'ordre sémantique lors de la procédure d'extraction du réseau depuis HiLeS vers TINA. Ceci a conduit à des modifications des règles du processus d'extraction. D'autre part, des problèmes de modélisation ont été décelés par la présence de nœuds bloquants dans le réseau (par exemple : l'accumulation de jetons sur des places). TINA nous a fourni le moyen de les localiser. Plusieurs corrections ont alors été apportées au modèle pour aboutir au réseau borné. Nous avons obtenu un réseau de 77 places et 69 transitions. Les transitions marquant la fin du processus d'un bloc fonctionnel sont temporisées $[t_{fonctionX\ min};t_{fonctionXmax}]$ alors que les autres transitions sont temporisées par défaut [1;1]. Ce choix contraint un enchaînement séquentiel des transitions, contrainte que l'on aurait pas eu si la temporisation avait été instantanée [0;0].

L'analyse du code mort nous a permis de mettre en évidence que certaines branches du système étaient constituées de transitions mortes. Nous avons fait le lien entre ces branches et les différents cas d'utilisation. En effet, le système peut fonctionner dans 3 modes distincts :

Mode 1 : Fonctionnement automatique,
Mode 2 : Réveil forcé par l'Interface Homme-Machine,
Mode 3 : Réveil forcé par la mise en route manuelle (bouton).

La modélisation telle que représentée sous HiLeS prend en compte ces trois modes mais nécessite une initialisation particulière pour signifier le mode choisi. Il faut donc faire autant de simulations TINA que de modes de fonctionnement possibles de notre système. Dans ce cas, il est normal d'observer des branches de réseau mortes qui correspondent aux modes non couverts par l'état d'initialisation.

```
     STRONG CONNECTED COMPONENTS:

     1 : 0
     0 : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119
120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137
138 139

     SCC GRAPH:

     1 -> Pulse/0
     0 -> Auto_wake_up_on/0, Download_on/0, Tr_Auto_wake_up/0,
Auto_Wup_t/0, W_Up_t/0, Start_power_t/0, Tr_Startpower/0, Start_on/0,
Start_done/0, Tr_On_pwr/0, Pwr_Ok/0, Tr_Starttest/0, Start_select/0,
Start_test_i/0, Test_started/0……
```

Figure 2-32 : Analyse de réinitialisation autour de la transition "Pulse".

La dernière vérification qu'apporte l'analyse classique des réseaux de Petri sous TINA est la réinitialisation du système. Cette propriété nous concerne particulièrement puisque notre système doit être réinitialisable en mode 1, avec une cyclicité commandée par la transition « Pulse » du bloc « Timer ». Le résultat d'analyse (Figure 2-32) montre qu'à partir de tous les états du mode 1, l'état initial peut être atteint. La transition « Pulse » active toutes les transitions du mode 1. Seul le mode 1 est réinitialisable. Les modes 2 et 3 ne sont que transitoires. C'est à dire que la fin des modes 2 et 3 est marquée par un retour en mode 1.

2.4.2.2 Vérification des propriétés particulières du système

Cette étape fait intervenir le concepteur, d'une part pour lister les propriétés à vérifier d'une manière non ambiguë mais aussi pour analyser les résultats de simulation qui sont sujets à interprétation. Cette vérification ne se fait qu'après la vérification des propriétés classiques du réseau. Les propriétés particulières doivent être traduites dans la syntaxe de la logique temporelle linéaire avec l'aide d'un « vérificateur » formé aux techniques formelles.

Pour l'exemple, nous nous sommes intéressés aux propriétés particulières du bloc « Initialise ». Nous avons listé quelques propriétés types impliquant : l'ordonnancement des actions, l'occurrence conditionnelle, etc.

L'analyse faite par TINA nous propose un moyen de visualiser les séquences d'événements par le biais des mécanismes de projection. La projection observationnelle génère un graphe qui peut être très vite trop chargé et peu lisible. Ainsi, pour améliorer la lisibilité, nous pouvons masquer certaines transitions pour ne sélectionner que les transitions d'intérêt. Nous avons ainsi limité le nombre de transitions en nous focalisant sur l'environnement du bloc « initialise ». La projection obtenue est donnée en Figure 2-33. Elle fait apparaître les transitions impliquées dans les événements successifs qui interviennent dans ce bloc. La figure montre l'exemple du mode 1 dans lequel deux « chemins » sont possibles après la mise en veille « Tr_Sleep » : soit le système fonctionne classiquement (mise en route « Tr_Auto_Wake_up », test, mesure, mise en veille), soit le système démarre en spécifiant une demande de téléchargement des données en fin de cycle (« Tr_Auto_Download »).

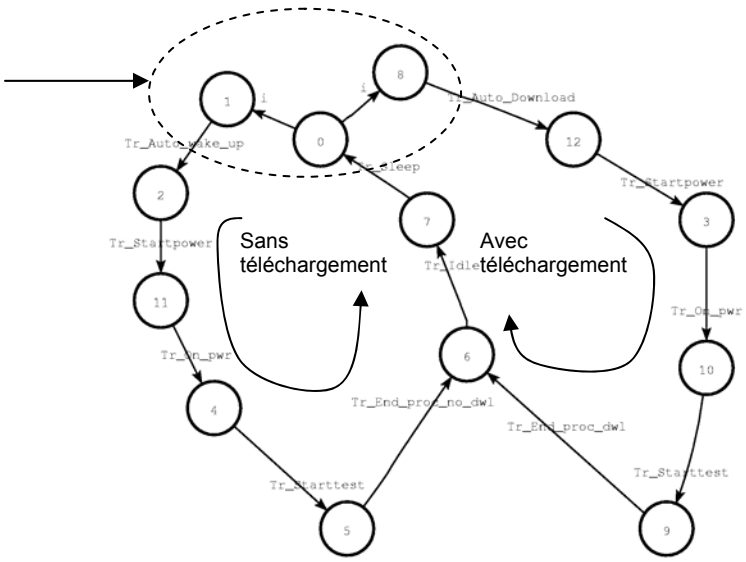


Figure 2-33 : Projection du mode 1.

Ainsi, dans cette modélisation, la probabilité d'occurrence de l'un des deux cas de figure (avec ou sans téléchargement) est identique. Or, nous souhaiterions rendre la boucle « avec téléchargement » accessible seulement après n boucles « sans téléchargement ». Nous retrouvons ici une insuffisance des possibilités de modélisation due à l'utilisation de réseaux ordinaires dans HiLeS, pour lesquels les arcs orientés reliant les places aux transitions ont par défaut un poids d'une unité.

Prenons un deuxième exemple de propriété : le mode 2 provoque un réveil (Tr_Auto_wake_up_rec) qui sera toujours suivi d'un accusé de réception d'envoi de données (e_ini_Ack_Sent_data) avant de retourner en mode de veille (Tr_Sleep).

En logique temporelle linéaire, cela est transcrit par :
[] (Tr_Auto_wake_up_rec => <> (Tr_End_proc_dwl => <> Tr_Sleep));

La projection présentée sur la Figure 2-34 illustre l'enchaînement des 3 événements dans la boucle inférieure. Cette propriété est donc vérifiée sur le graphe projeté. Il apporte aussi l'information supplémentaire de retour du système en mode 1 à la suite du mode 2. Nous retrouvons ainsi les caractéristiques de la Figure 2-33.

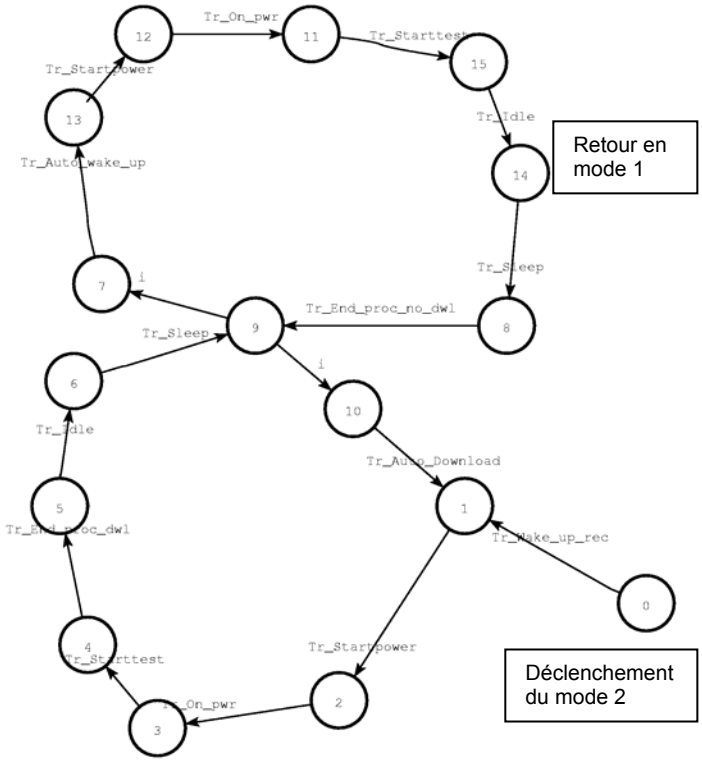


Figure 2-34 : Projection du mode 2.

2.4.2.3 Vérification des propriétés architecturales

Afin de réduire la distance entre le concepteur HiLeS et les techniques formelles de vérification, l'équipe OLC a proposé une approche de « vérification architecturale » permettant d'analyser un système HiLeS en mettant à profit les choix architecturaux (structuration et hiérarchisation du système en niveaux et en blocs) du concepteur. Cette approche est basée sur l'utilisation du mécanisme de projection. Il permet de visualiser le séquencement des différents événements d'un sous ensemble du système que l'on veut observer. Ainsi, l'équivalence de comportement permet de visualiser de façon abstraite le comportement du sous-ensemble défini par le concepteur ou l'enchaînement d'événements entre certains d'entre eux.

La vérification architecturale a porté sur l'observation des entrées et des sorties de certains blocs choisis par le concepteur et qui ont été définis dans HiLeS. Cette méthode est facilement automatisable si les règles de hiérarchisation/interfaçage d'HiLeS peuvent être employées de façon à discerner les limites des blocs structurels décrites par leurs transitions d'entrées et de sorties. Dans notre étude, les transitions de sorties des blocs étudiés ont pu être repérées aisément, car HiLeS génère automatiquement ces transitions, selon le formalisme « Tr_Nom du port de sortie du bloc ». Par contre, les transitions d'entrée ont du être désignées manuellement, puisqu'elles sont externes au bloc concerné et ne font pas encore l'objet d'un automatisme d'HiLeS.

Prenons un exemple de propriété faisant intervenir les blocs « IHM », « Measure_Treatment » et « initialise » :

La demande de réveil « Tr_Wake_up_rec » doit attendre que le système soit en état de veille « Sleep_mode_p » (place interne du bloc initialise) (ou déclenchement de « Tr_Sleep ») avant de démarrer un test « Tr_Start_test_X ».

En logique temporelle linéaire, cela est transcrit par deux formules :

(1) - [] (Tr_Wake_up_rec => <> Tr_Start_test_X);

Le vérificateur de modèle renvoie la réponse TRUE.

(2) - [] (Tr_Sleep => - Tr_Wake_up_rec);
Le vérificateur de modèle renvoie la réponse FALSE et le prouve par un contre-exemple.

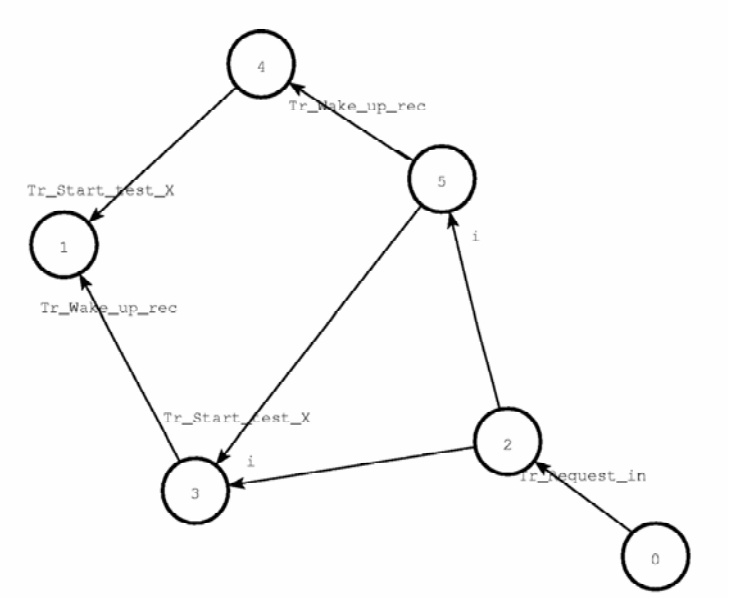
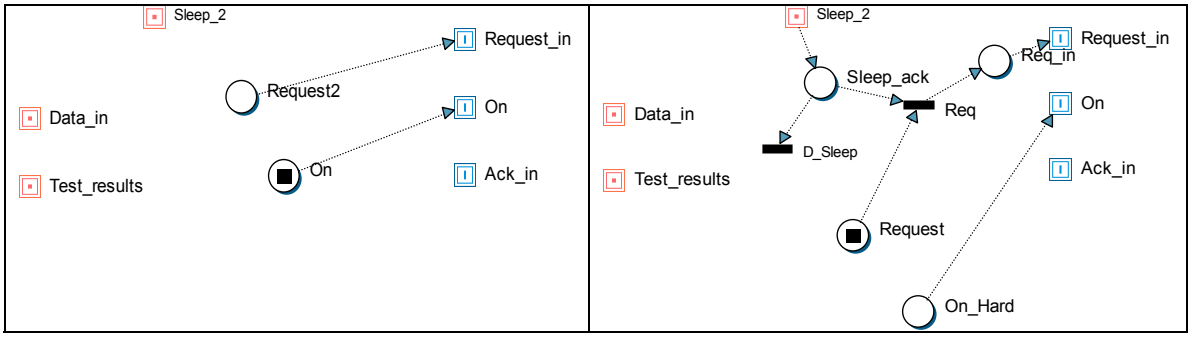


Figure 2-35 : Projection illustrant le contre exemple donné par TINA.

Ce contre-exemple est donné sous la forme graphique (Figure 2-35) grâce à une projection observationnelle sur le graphe de classes d'états en ne gardant que les transitions « Tr_Request_in », « Test_X », « Tr_Start_test_X » et « Tr_Wake_up_rec ».

Ce contre exemple donné par TINA montre que l'on peut avoir un réveil forcé sans qu'il y ait une mise en veille au préalable. L'événement « Tr_Sleep » n'apparaît pas sur la projection. La formule (2) n'est donc pas satisfaite par notre modèle.

Nous avons alors modifié la modélisation du bloc IHM afin d'introduire une pré-condition supplémentaire sur le déclenchement de la requête de mesure « Tr_Request_in ». La Figure 2-36(b) illustre le lien qui a été ajouté entre le port « Sleep_2 » et le port « Req_in » pour introduire cette pré-condition.



(a) IHM première version (b) IHM deuxième version

Figure 2-36 : Modification du modèle de l'IHM.

Ainsi, la propriété donnée en exemple est vérifiée. La prise en compte de la requête externe se faisait sous n'importe quelle condition sur la Figure 2-35. La condition est maintenant visible sur la projection illustrée dans la Figure 2-37, par les états qui nécessitent la précédence de l'état 0 et de « Tr_Sleep » pour poursuivre le processus.

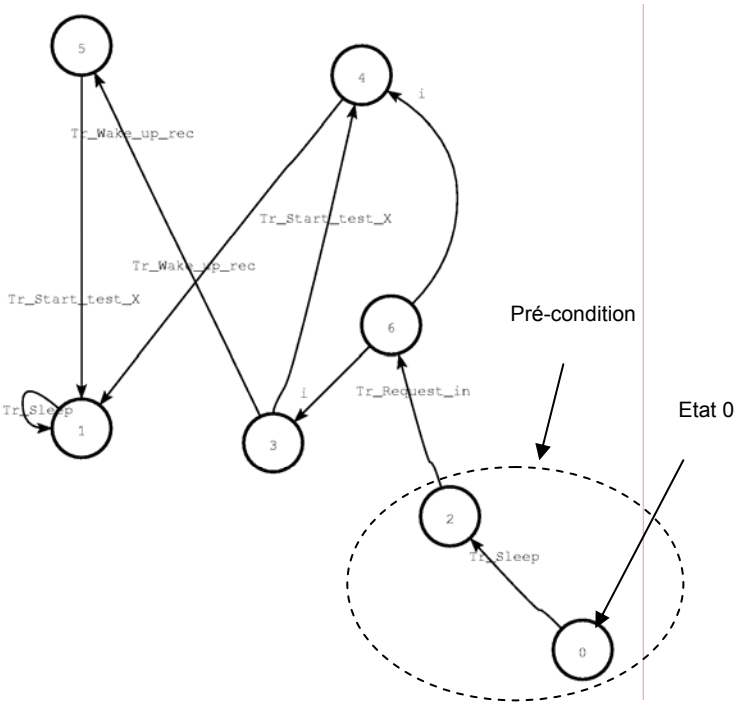


Figure 2-37 : Projection après modification.

Nous avons illustré la démarche de vérification architecturale en nous intéressant en particulier au bloc « Measure_Treatment », « IHM » et « Initialise ». Nous avons pu, ainsi, discerner des erreurs de comportement dus à des problèmes de modélisation et finalement valider le comportement attendu, décrit par des séquences d'événements.

2.4.3 Difficultés rencontrées et recommandations

D'une manière générale, les tests intéressants que l'on a pu envisager sur notre exemple de système multifonctionnel sont de l'ordre du comportement ou du séquencement des activités.

Du point de vue de la vérification, nous avons observé certains cas bloquants d'extraction des réseaux depuis HiLeS, notamment à l'interface des blocs. L'avantage principal que nous attribuons à HiLeS est de proposer un moyen de construction de la complexité du système à partir de blocs élémentaires. HiLeS doit constituer un réseau total et borné du système. La méthode de vérification TINA sous HiLeS permet alors de repérer les erreurs éventuelles de modélisation ou d'assemblage de ces sous-blocs.

Au niveau de la modélisation, HiLeS ne permet pas de modéliser des boucles itératives ayant un nombre d'itération fini. Il peut en outre modéliser des boucles infinies. Ainsi, dans la Figure 2-33, les deux boucles apparaissent équivalentes, ce qui montre que la modélisation ne fait pas apparaître la condition selon laquelle la boucle avec téléchargement ne peut avoir lieu seulement si n boucles sans téléchargement ont été exécutées.

D'un ordre général, la modélisation du comportement suppose une bonne connaissance des réseaux de Petri et de leur fonctionnement. Afin de réduire cette difficulté, des méthodes de transformation de modèles par la méthode des méta-modèles (passage d'une modélisation de type UML à une modélisation de type réseaux de Petri) sont en cours de développement au laboratoire à la suite de travaux sur la transformation de modèles particuliers. La Figure 2-38 indique comment s'élabore la transformation d'un modèle (MOD_1) à un modèle (MOD_2) de moindre abstraction par 'addition' d'un Formalisme (F) (META Model) et d'une Transformation (T).

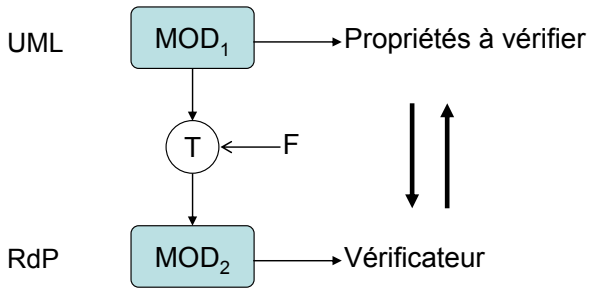


Figure 2-38 : Transformation T de modèles par un formalisme F.

L'utilisation d'un outil de transformation de modèle permettra, à partir d'un modèle qui représente les propriétés du système à un haut niveau d'abstraction (MOD_1) de lui associer un modèle vérifiable (MOD_2) et de valider les propriétés du système.

2.5 Conclusion

Nous avons montré dans ce chapitre le rôle et l'importance d'une conception « amont » pour des systèmes complexes. Cette conception s'appuie sur les outils tels que les diagrammes UML, les réseaux de Petri et les méthodes de vérification formelles par les réseaux de Pétri.

Cette démarche nécessite trois étapes essentielles :

I. Une étape de formalisation du cahier de charges, indispensable car elle formalise des exigences qui pourront être tracées et vérifiées.
II. Une étape de conception fonctionnelle puisque nous avons du formalisme HiLeS une représentation fonctionnelle du système dont nous pouvons vérifier les choix par simulation sous l'outil TINA.
III. Une étape préparatoire au partitionnement que nous allons développer dans le chapitre suivant.

Les points des propositions à retenir pour ce chapitre sont les suivants :

- L'analyse fonctionnelle du cahier des charges est indispensable pour extraire les cas d'utilisation du système. Aussi, nous avons vu que l'intégration d'une méthode objet (UML/SysML), pour la formalisation des spécifications nous a permis d'obtenir un premier modèle portable qui peut être échangé et commenté par divers acteurs. Ces modèles peuvent être manipulés comme des briques de base. Cependant, une transition doit être comblée, celle de la transformation de ces modèles pour intégrer une plate-forme de vérification formelle.

- Sur le plan de la vérification, nous avons pu montrer l'intérêt d'introduire un moyen de vérification formelle des propriétés des RdP pour la première validation du comportement du système global.

Chapitre 3

Le prototypage virtuel

3.1 Introduction

Au terme du chapitre 2 nous disposons d'une représentation formelle, fonctionnelle, « vérifiée » du microsystème. Jusqu'ici, nous n'avons fait aucune hypothèse explicite sur les choix technologiques : Chaque fonction du système est représentée par un bloc, dans une représentation globale hiérarchique. Seuls les délais d'exécution fonctionnelle sous-tendent des solutions pratiques. Evidemment, il y a un grand intérêt à exploiter cette simplicité et cette flexibilité pour explorer différents chemins de conception : différentes évaluations « précoces » sont possibles en explorant l'espace des délais d'exécution. Nous pouvons, par exemple, évaluer l'intérêt et les limites de l'architecture fonctionnelle choisie en terme de robustesse. Nous pouvons encore analyser des effets de dégradation fonctionnelle (fonctionnements dégradés). Notre propos ici est plutôt d'approcher les fournisseurs par la définition et la spécification de Sous-Ensembles « physiques » correspondant aux modules à assembler. Pour ce faire, la tâche essentielle que nous allons explorer est celle du partitionnement :

- Partitionnement en Sous-Ensembles architecturaux cohérents,
- Partitionnement Matériel-Logiciel.

Nous aborderons ensuite la façon d'approcher les fournitures, récupérer les propriétés et les modèles de ces propriétés pour obtenir un véritable prototype virtuel en langage VHDL-AMS.

Nous considèrerons enfin, une deuxième étape de vérification par simulation VHDL-AMS.

3.2 Les langages

Chaque domaine d'ingénierie utilise les outils les mieux adaptés pour concevoir et optimiser les produits qu'il développe. Le domaine de l'électronique est celui qui a du, très tôt, savoir gérer au mieux la complexité due au nombre important d'éléments constitutifs d'un système. Ainsi, le besoin d'un plus haut niveau d'abstraction est naturellement né dans la démarche de conception. La modélisation « amont » opère donc au niveau des concepts sans référence explicite aux technologies. Elle applique, dans notre démarche, les standards UML/SysML de description fonctionnelle et les réseaux de Petri pour une représentation temporisée et vérifiable. Ces outils doivent permettre au concepteur d'accéder à la définition de systèmes complexes, sans devoir définir les briques de base des systèmes et se soucier de l'implémentation physique. Au delà, les langages d'implémentation sont attachés à la nature des fonctions à manipuler = numérique, analogique, mixte, pluridisciplinaire, etc.. Pour les systèmes numériques, les industries et les scientifiques du domaine se sont consultés et ont regroupé leurs expériences pour travailler et définir ensemble la norme VHDL. Ainsi, ce langage de description de très haut niveau a été standardisé en 1987 par la norme VHDL IEEE 1076-1987. Il est l'outil qui a ouvert la voie de la conception et de la synthèse automatique de circuits logiques. Cette norme a été réactualisée et complétée régulièrement. Elle est définie à ce jour par la norme VHDL-IEEE 1076-2004. Aujourd'hui, dans un contexte international fortement concurrentiel, les concepteurs doivent être capable de penser leur produit le plus rapidement possible en fournissant une solution viable et robuste.

La conception de systèmes hétérogènes, mettant en œuvre à la fois l'électronique numérique, l'électronique analogique, la thermique, l'optique, la mécanique ou encore d'autres disciplines, apporte une complexité supplémentaire au domaine de la conception et de la simulation système. Ces besoins ont entraîné l'émergence de nouveaux outils pour le prototypage virtuel. La finalité est d'aboutir à la validation de systèmes pluridisciplinaires au plus tôt de la conception.

La simulation analogique des composants et des circuits est née à Standford avec des outils comme Spice. C'est en 1993 que les travaux ont commencé pour associer un sous-ensemble analogique au noyau numérique. L'extension de la norme VHDL a permis d'intégrer au langage les signaux analogiques et mixtes « Analog and Mixed Signal – AMS ». La norme a été ratifiée en 1999 (VHDL-AMS IEEE 1076.1-1999) et est définie à ce jour par le standard IEEE 1076.1.1-2004. Cette normalisation apporte aussi une réponse attendue à la modélisation multi-domaines, en conservant le même cœur de calcul (résolution d'équations de Kirchhoff) mais en changeant des variables selon le Tableau 3-1.

Domaines	Electrique	Mécanique de translation	Hydraulique	Thermique	Economique
Variables de flux	Courant (i)	Vitesse (v)	Débit Volumique (q_v)	Flux d'entropie (q_s)	Flux des produits
Variables d'effort	Tension (u)	Force (F)	Pression (P)	Température (T)	Prix unitaire

Tableau 3-1 : Tableau d'équivalence de variables de flux et d'effort pour différents domaines.

La simulation systèmes n'a pas attendu cette extension de norme et a été très vite abordée par plusieurs variantes de langages de modélisation multi-domaines. Un certain nombre de langages propriétaires ont été développés avec succès, comme MAST, Verilog-AMS et diverses passerelles ont été élaborées pour relier ces langages de programmation différents. Les propositions les plus récentes sont présentées aux utilisateurs comme conformes à la norme VHDL-AMS. C'est le cas de Simplorer [13] et de System Vision [14]. Nous avons choisi d'utiliser ce dernier dans nos travaux, lequel s'inscrit dans le cadre d'une collaboration entre Mentor Graphics et le LAAS-CNRS.

Les systèmes, qui étaient jusqu'ici modélisés sous Matlab, VHDL et Spice peuvent aujourd'hui être modélisés sous VHDL-AMS, puisqu'il offre la possibilité de faire travailler simultanément un simulateur à événements discrets et un solveur d'équations différentielles. Bien sûr, les habitudes de conception restent et il faut s'attendre à de nouvelles évolutions via des interfaces de langages Matlab/VHDL-AMS, SystemC/VHDL-AMS.

Pour plus d'informations sur le langage VHDL-AMS, le lecteur pourra se référer à l'ouvrage de Y. Hervé [Her02] et aux documents d'A. Vachoux [Vac03].

3.3 La traduction du modèle amont en modèle VHDL-AMS

La logique de conception veut que, la représentation HiLeS du système étant vérifiable, la transformation HiLeS/VHDL-AMS reste conforme et soit donc si possible automatique. La voie de la génération automatique d'un code complet VHDL-AMS du système, d'après un modèle haut niveau tel que décrit dans le chapitre 2, doit respecter deux aspects du modèle : d'une part, la génération automatique de la logique temporelle qui articule les événements et les décisions conditionnelles de fonctionnement du système, et d'autre part, la génération automatique de la représentation structurelle des enveloppes (interfaces et interconnexions) des blocs définies par l'architecture :

- Le premier point a fait l'objet de plusieurs travaux axés sur la transformation de modèles RdP vers des modèles VHDL-AMS. Nous retiendrons historiquement, la méthode ConPar [Mac98] et la méthode des composants proposée par [ABG04] à l'implantation de laquelle nous avons participé.

- Le deuxième point est traité dans HiLeS par la génération d'un fichier VHDL_AMS pour chaque bloc (entity), accompagné de son interface.

Nous avons donc ici deux générateurs de code distincts qui séparent nettement le monde logique et le monde analogique. Cependant, nous souhaitons les coordonner pour créer un modèle complet de système analogique commandé et dynamique.

3.3.1 Application de la méthode ConPar

Dans le cadre de notre application, nous avons utilisé la méthode ConPar. Cette méthode a en premier lieu été conçue pour simuler des réseaux de Petri à vocation d'analyse de systèmes numériques (VHDL). Nous l'avons complétée afin de considérer l'interaction entre des RdP et des blocs fonctionnels, traitant des signaux analogiques du VHDL-AMS, et ceci sans recours à des automatismes d'écriture de code. Ainsi, pour un bloc structurel composé de blocs fonctionnels et de RdP, le code VHDL-AMS comprend une partie séquentielle et une partie continue. La partie séquentielle est un « process ». Dans cette méthode, dans le cas d'un réseau de Petri synchrone, le calcul du marquage du réseau est fait par l'intermédiaire d'un « process » qui est sensibilisé par une horloge. Cependant, il est préférable d'avoir une représentation asynchrone des réseaux de Petri, dans le cas d'une modélisation haut niveau, pour cela, cette méthode propose de sensibiliser les « process » par une liste de sensibilité. Nous avons modifié cette liste, de telle sorte qu'elle contienne les événements du réseau de Petri (tir d'une transition) mais aussi les événements des signaux d'interconnexion entre RdP et blocs. Ce « process » affecte les valeurs de marquage des places. La partie temps continue, quand à elle, décrit les règles d'interconnexion du réseau et les affectations de valeurs aux signaux de sortie.

Nous prenons comme exemple le RdP du bloc « Sampler » illustré par la Figure 3-1. Ce réseau est constitué de 5 places et de 4 transitions et réagit aux commandes de 3 signaux provenant de blocs fonctionnels : « Start_meas_select », « End_sampling » et « Ack_mem_datas_X. ».

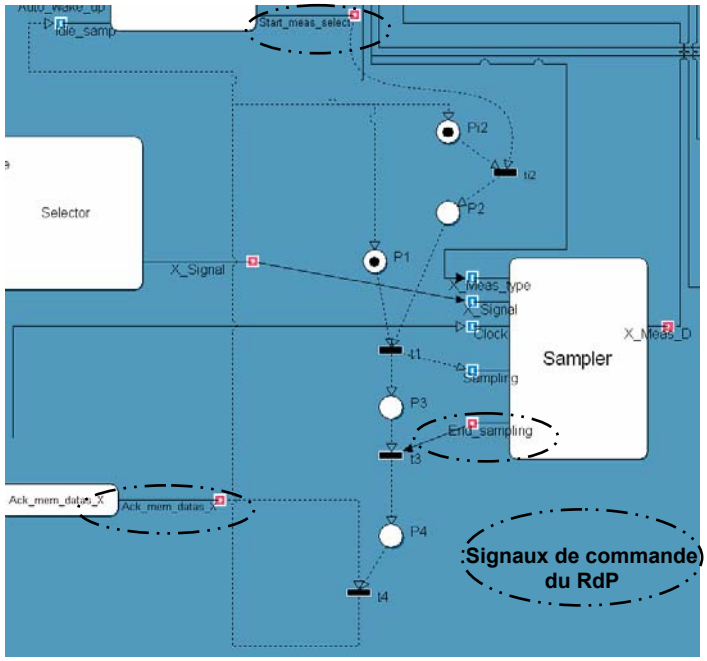


Figure 3-1 : Représentation HiLeS du RdP de la structure interne du bloc MT\Analyse (niveau 3).

La traduction VHDL-AMS de ce réseau est représentée sur la Figure 3-2. Nous pouvons observer 3 parties dans la description ConPar. D'une part, la déclaration d'un « process » actualisant les valeurs affectées aux places du réseau (représenté par l'encadré supérieur). Les mises à jour des valeurs sont réalisées chaque fois qu'un des signaux de la liste « wait on » est activé. D'autre part, le deuxième encadré regroupe les équations conditionnelles simultanées qui définissent les lois de validation des places et des transitions. Pour finir, la troisième partie sert de test pour alerter les cas de conflits entre transitions lors de la simulation.

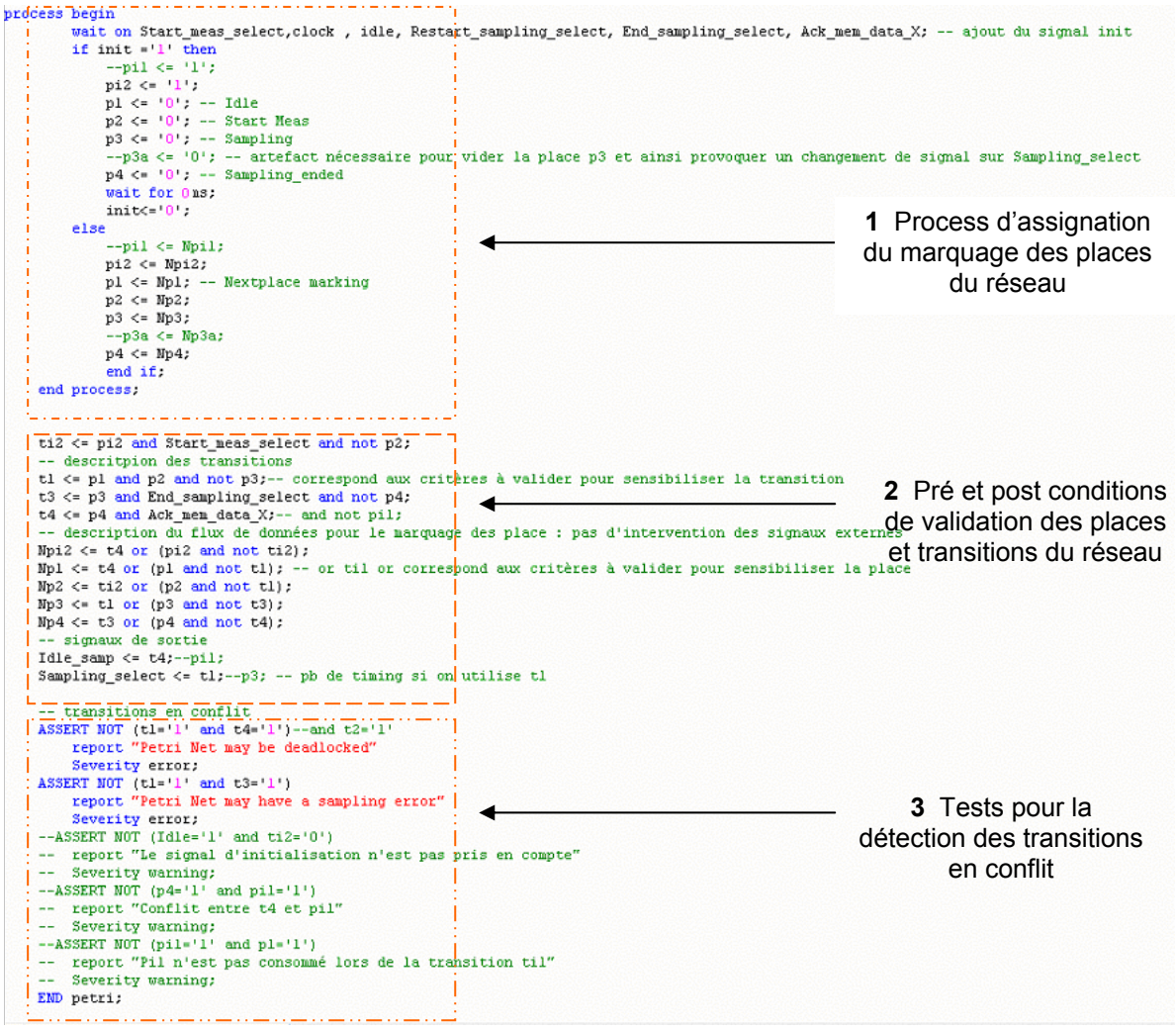


Figure 3-2 : Description du RdP de la structure interne du bloc MT\Analyse (niveau 3) par la méthode ConPar.

Cette méthode a l'avantage d'être simple de mise en œuvre et propose une organisation du code lisible, facilement identifiable et certainement automatisable. Cependant, elle impose une traduction supplémentaire qui favorise l'introduction d'erreurs dans le cas d'une méthode non automatique.

Nous nous sommes ainsi intéressés à une approche basée sur la représentation des places et des transitions des réseaux de Petri comme composants élémentaires. Nous présentons cette méthode et les modifications que nous lui avons apporté dans le paragraphe suivant.

3.3.2 Application de la méthode des composants

La méthode des composants propose de recréer un réseau de Petri en utilisant des composants VHDL qui représentent chaque élément d'un réseau : les transitions et les places. Chaque composant est représenté par une entité VHDL et comporte des ports d'entrées-sorties pour ses interconnexions. Ainsi, la description topologique du réseau laisse apparaître clairement tous les éléments constitutifs et leurs interconnexions. Cette méthode a été développée au LIRMM, en collaboration avec le LAAS-CNRS [ABG04].

Cette approche permet la traduction en VHDL d'un réseau de Petri complet. Le code associé est généré automatiquement et peut ainsi être simulé pour valider son comportement. Cependant, ce qu'il manque à cette méthode de traduction pour être totalement exploitable dans la plate-forme HiLeS, c'est le lien entre le réseau de commande et les blocs fonctionnels. Nous allons exposer les

modifications que l'on a apporté sur ce point. Tout d'abord, nous présentons les composants élémentaires définis dans cette approche.

Le **composant Place** est présenté selon deux formats : synchrone ou asynchrone. Dans le cas de notre application de modélisation haut niveau de système dynamique, nous nous intéressons au cas asynchrone. L'interface de ce composant décrite sur la Figure 3-3 est composée par :

- des paramètres génériques : le nombre d'entrées et de sorties sont des variables dynamiquement mises à jour lors de l'extraction du réseau sous HiLeS selon la topologie du réseau.
- des ports :
  - « init » : donne la possibilité d'initialiser l'état de la place.
  - « ini_jetons » est la valeur du marquage de la place à l'instant initial.
  - « aj_jetons » est un vecteur de bit d'activation de la place (ajoute un jeton).
  - « ret_jetons » est un vecteur de bit de désactivation de la place (retire un jeton).
  - « marque » est le bit de marquage de la place.

```
component Place_Async
generic(    nb_entrees    : natural:= 1;
            nb_sorties    : natural:= 1);
port   (init          : in std_logic;
        ini_jetons    : in std_logic;
        aj_jetons     : in std_logic_vector(nb_entrees-1 downto 0);
        ret_jetons    : in std_logic_vector(nb_sorties-1 downto 0);
        marque        : out std_logic);
end component;
```

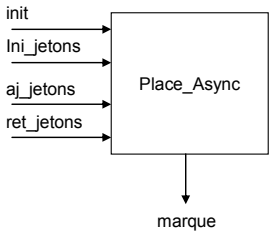


Figure 3-3 : Les ports d'interface du composant Place Asynchrone.

L'interface du **composant Transition** est décrite sur la Figure 3-4 :

- Les paramètres génériques : le nombre d'entrées et de sorties sont des variables dynamiquement mises à jour lors de l'extraction du réseau sous HiLeS.
- Les ports :
  - « c_t » : est la condition associée à la transition.
  - « type_transit_e » et « type_transit_s » précisent le type d'arc utilisé en entrée et en sortie (classique, inhibiteur, test).
  - « marque_tie » marquage des places d'entrée.
  - « ret_amont » marque à retirer des places amont.
  - « aj_aval » marque à ajouter dans les places aval.

```
component Transition
generic(    nb_entrees    : natural:= 1;
            nb_sorties    : natural:= 1);
port   (c_t               : in std_logic;
        type_transit_e    : in std_logic_vector(nb_entrees-1 downto 0);
        type_transit_s    : in std_logic_vector(nb_entrees-1 downto 0);
        marque_tie        : in std_logic_vector(nb_entrees-1 downto 0);
        ret_amont         : out std_logic_vector(nb_entrees-1 downto 0);
        aj_aval           : out std_logic_vector(nb_sorties-1 downto 0));
end component;
```

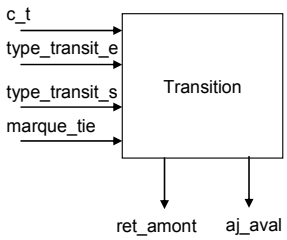


Figure 3-4 : Les ports d'interface du composant Transition.

Ces deux composants permettent de définir un réseau de Petri complet. Son marquage est calculé par un algorithme, intégré dans un process, qui tient compte des jetons ajoutés et retirés de la place à chaque événement des signaux d'entrée de la liste de sensibilité (Figure 3-5).

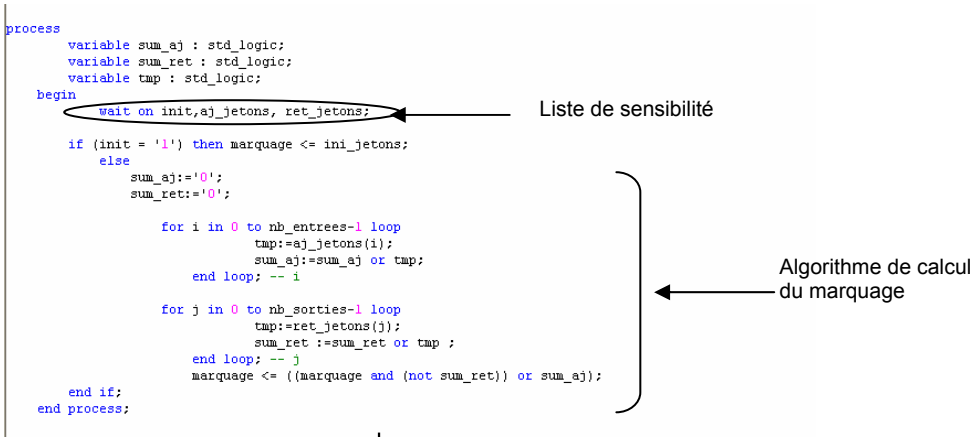


Figure 3-5 : Process de calcul du marquage d'une place.

Cependant, dans le cadre de l'approche de conception de systèmes pluridisciplinaires dans HiLeS, nous devons penser à garder le lien fort entre les aspects discrets et continus du système. Cette liaison est représentée dans HiLeS par un arc de Petri reliant une transition et un bloc. Le modèle VHDL-AMS du système généré par HiLeS doit intégrer cette dimension. Pour cela, nous proposons d'ajouter un composant de liaisons **PnetCompatibility** dont l'interface est décrite sur la Figure 3-6 :

- Les paramètres génériques : le nombre de signaux de contrôles de sortie est une variable dynamiquement mises à jour lors de l'extraction du réseau sous HiLeS.
- Les ports :
  - « ret_jeton » : désactive la fonction en retirant un jeton.
  - « End_of_process » .
  - « Control_Outputs » .

```
component PnetCompatibility

generic (nb_control_outputs : natural:=1);

port(   ret_jetons      : in std_logic_vector(nb_control_outputs-1 downto 0);
        End_of_process  : out std_logic;
        Control_Outputs : out std_logic);
end component;
```

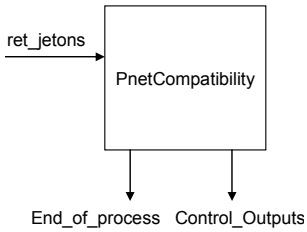


Figure 3-6 : Les ports interface du composant de Liaison (PnetCompatibility).

Dans HiLeS, nous aurons alors un composant PnetCompatibility, intégré au bloc fonctionnel ou structurel (Figure 3-7). Il permet de gérer le flux de jetons dans le réseau en déclenchant le début d'activité du bloc et en détectant sa fin d'activité.

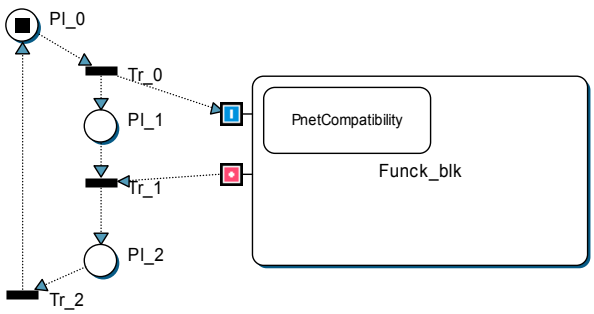


Figure 3-7 : Intégration du composant PnetCompatibility dans HiLeS.

Cette étude est en cours de test et de validation. L'étape suivante est d'intégrer un automatisme de génération de ce composant en VHDL-AMS, depuis la plate-forme HiLeS au même titre que les

places et les transitions. Nous voyons qu'ici il s'agit de générer un code VHDL-AMS qui sera l'interface de connexion, pour le concepteur, entre le réseau de commande généré automatiquement et les codes fonctionnels qu'il doit définir et intégrer. Ceci implique de considérer les aspects tels que la visibilité des signaux, la clarté et la structure du code.

3.3.3 Les approches en cours

D'autres travaux concernent les réseaux de Petri hybrides dans le sens où ils peuvent gérer le temps discret et continu. Notamment, l'approche récemment proposée dans [Alb05] est un premier pas qui ouvre la voie de la génération automatique de code mixte (analogique-numérique). Le modèle proposé permet de générer un code source (VHDL-AMS) pour une modélisation réalisée à partir de RdP. La partie « analogique » du système est représentée par des places et des transitions particulières qui peuvent être hiérarchiques. Les signaux tel que nous les concevons dans HiLeS (continus et discrets) sont ici apparentés à des variables partagées entre des équations différentielles contenues dans les éléments hiérarchiques du RdP. Cette méthode se différencie de celle d'HiLeS par le fait qu'elle est purement orientée vers la manipulation d'objets. Ainsi, l'intégration de cette méthode dans HiLeS impliquerait des changements dans l'approche de la modélisation sous HiLeS, mais permettrait de rapprocher la démarche à la méthode de description UML, aussi orientée objet.

3.4 L'agrégation fonctionnelle

Nous avons, à ce stade de la modélisation amont, le modèle fonctionnel graphique HiLeS qui présente des blocs interconnectés par des réseaux de Petri. Le lien entre ce modèle fonctionnel et le modèle structurel repose sur le choix du concepteur, aidé et conseillé par les experts techniques, qui peuvent orienter l'agrégation ou le partitionnement de blocs et ainsi faire apparaître des éléments connus des fournisseurs. Ainsi, nous pouvons bénéficier de la réutilisation de modèles existants et insérer les modèles correspondant à chaque entité en vue de la simulation globale du système.

Ici, nous pensons que le partitionnement a à la fois un aspect « hard / soft » mais aussi un aspect métier correspondant à un domaine d'expertise particulier. Dans notre exemple, nous avons quatre fonctions de mesure : pression, température, humidité, déplacement. La question peut être posée quand au choix de leur implémentation physique. Allons nous les considérer comme quatre entités distinctes ou allons nous envisager de les regrouper ? Nous pourrions par exemple envisager de regrouper la fonction température et humidité. Cet aspect garde l'avantage du haut niveau d'abstraction du prototypage virtuel. Il consiste à déterminer les frontières et inventorier les parties du système qui sont relatives à une expertise particulière.

3.4.1 La procédure préconisée par HiLeS : agrégation – « building blocks »

HiLeS permet de réaliser des tâches d'agrégation de blocs à tous les niveaux de description afin d'obtenir des sous-ensembles cohérents qui puissent être traités par un domaine d'expertise particulier. Le traitement consiste ici en l'introduction, pour chaque bloc fonctionnel, d'un modèle comportemental constituant la base de représentation d'une fourniture. Il peut s'agir de réutilisation ou de développement de nouveaux modèles de fournitures. Ainsi, nous passons de la vue fonctionnelle à la représentation du système basée sur des technologies, des structures et des organes matériels.

Nous revenons ici à la description du système dans HiLeS telle que nous l'avions à un stade préliminaire, précédent la description faite dans le paragraphe 2.3.2. Ainsi, le niveau 1 de la description HiLeS comprenait cinq blocs (Figure 3-8). Nous avons ici fait le choix d'agréger les blocs « Test » et « Traitement des mesures » pour faire apparaître un bloc « Processeur ». Ainsi, nous obtenons une architecture standard comportant un processeur, un organe de communication, une alimentation, et une entité de mesure.

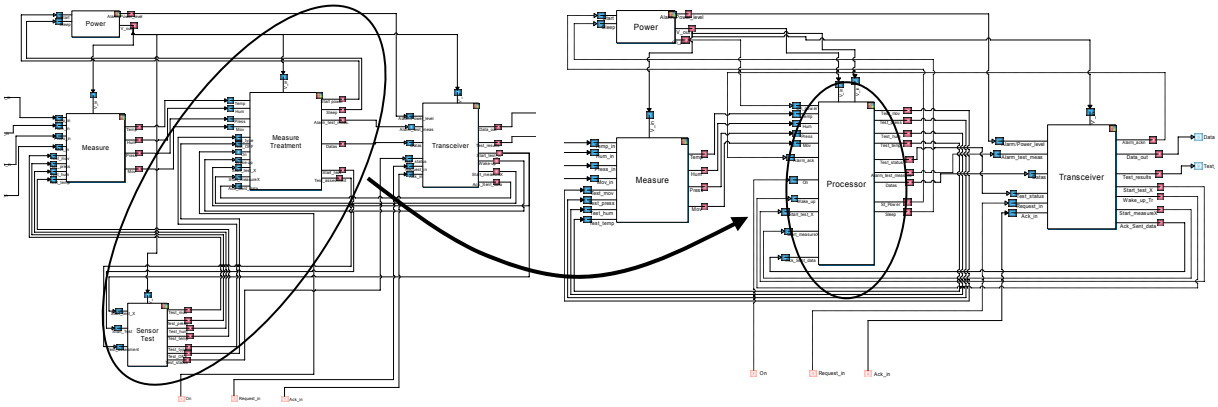


Figure 3-8 : Agrégation des blocs «Test» et «Traitement des mesures » en un bloc «processeur».

HiLeS donne le moyen de remanier la structure d'un système et de réaliser des agrégations sur tous les niveaux hiérarchiques, par le biais d'un outil de visualisation de la structure hiérarchique du système (Figure 3-9). Sur cette vue, nous pouvons faire le choix des blocs à agréger et donner un nom pour le nouveau bloc ainsi formé.

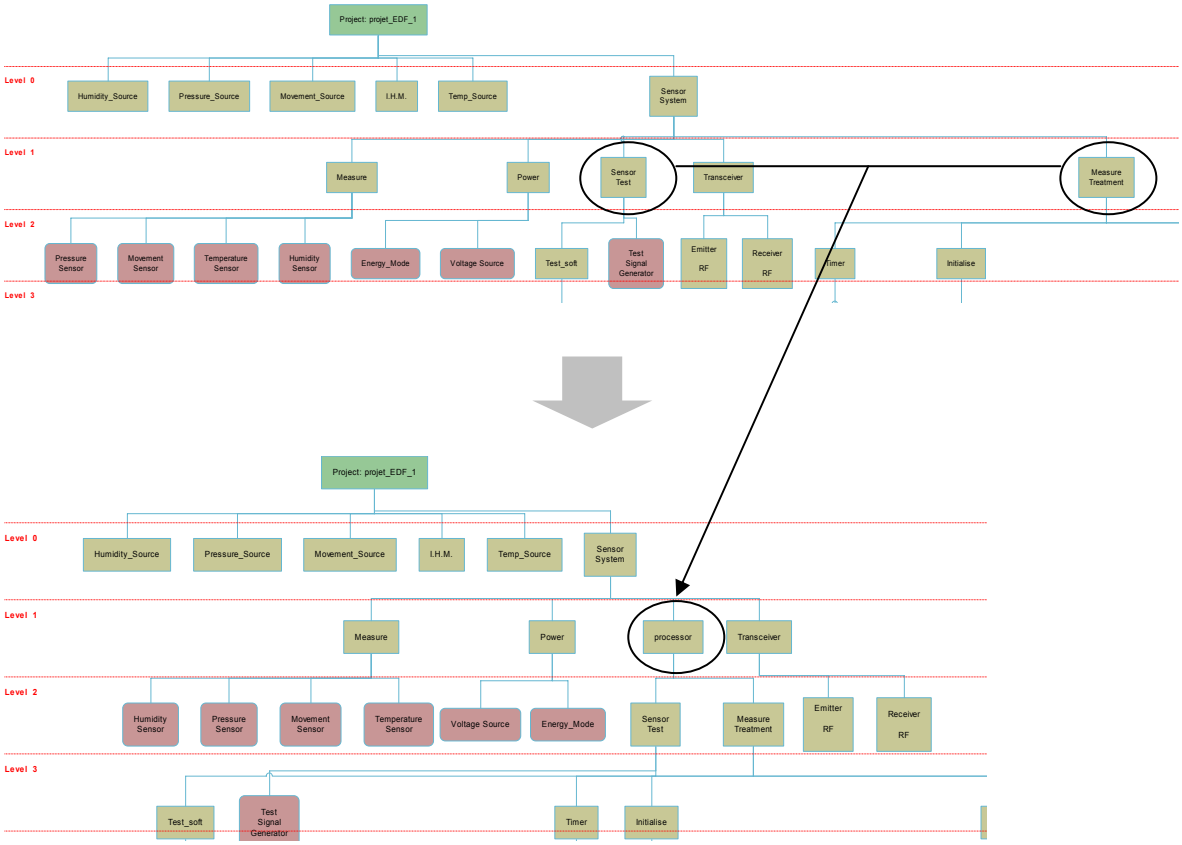


Figure 3-9 : Vue hiérarchique de l'agrégation.

Du point de vue de la modélisation, l'encapsulation ajoute un niveau hiérarchique par la création d'un nouveau « groupe » dans le système. Cependant, elle ne doit pas modifier la structure hiérarchique fonctionnelle pour pouvoir conserver la validité des étapes de vérification réalisées par TINA dans le chapitre 2.

Le partitionnement en modules physiques (building blocks) s'opère par synthèse entre la représentation fonctionnelle hiérarchique qui découle de la représentation HiLeS « vérifiée » et les exigences venues des considérations structurales, organiques et technologiques. Cette démarche est rendue explicite par une représentation de ce double processus en Y (Figure 3-10)

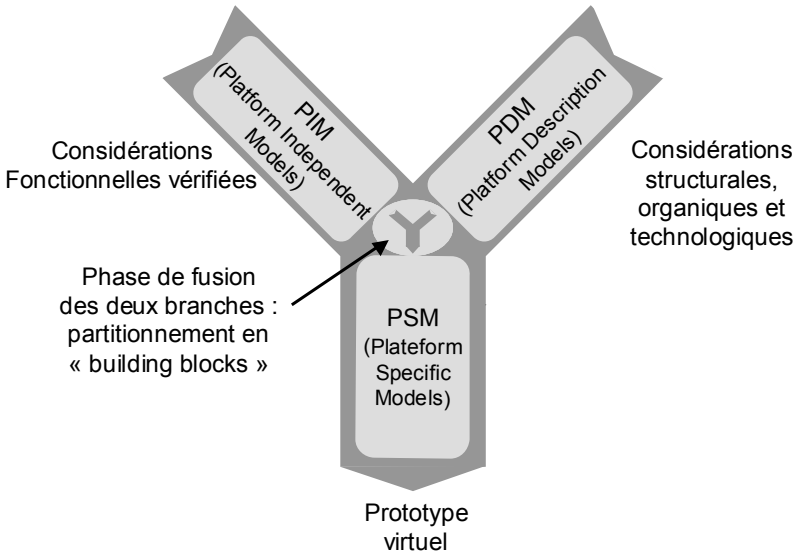


Figure 3-10 : Processus en Y de partitionnement du système en « building blocks ».

Dans une démarche complémentaire à la notre [GBG05], il est proposé que cette transformation : fonction – building blocks, s'opère manuellement sur une matrice de liaisons : fonctions – organes.

Une fois ces modules (Building Blocks) établis, ils seront l'objet :

- de l'établissement d'un cahier des charges spécifique qui découlera de l'exploration des représentations HiLeS et VHDL-AMS,
- de l'établissement d'un planning de réalisation par le développement d'un processus « conduite de projet ».

On aura ainsi réalisé le pont entre conception produit et conduite de projet qui est un objectif partagé par plusieurs groupes de travail [BRE04] sur les processus d'ingénierie système, mais dont le détail de la démarche sort du cadre de notre contribution.

3.4.2 La question du partitionnement Hard/Soft

Le partitionnement tel que nous venons de le présenter ne distingue pas le caractère matériel ou logiciel de la solution qui sera finalement choisie dans l'implantation terminale. Deux cas peuvent se présenter :

1. Le système comporte une infrastructure informatique propre et l'option de partitionnement consiste à affecter la fonction à cette infrastructure ou pas. Dans le cas positif, le fournisseur écrira l'algorithme dans le langage informatique dédié à cette infrastructure d'accueil en validant les spécifications fonctionnelles voulues dès la description système « vérifiée » et les interfaces systèmes-infrastructure informatique.

2. Les spécifications laissent ouvert le choix entre une réalisation matérielle et un algorithme embarqué sur un microprocesseur. Le microprocesseur a son langage interne propre et pour l'instant, il est proposé, pour la plupart des microprocesseurs, une transformation automatique à partir d'un algorithme écrit en C.

C'est dans le cas 2 que nous nous plaçons. Dans la démarche que nous essayons de proposer, il faut écrire l'algorithme en C et disposer d'une traduction VHDL-AMS si l'on veut l'inclure dans une procédure de prototypage virtuel. Voici la procédure que nous avons suivie.

Nous avons dû tout d'abord considérer le problème de la répartition des fonctions à implémenter en "Hard" ou en "Soft". Cette question est récurrente pour tous les systèmes électroniques mixtes et reste pour le moment ouverte.

Pour cela, nous avons pris l'option d'expliciter l'architecture de notre système pour en dégager une partie numérique par agrégation de blocs et de réseaux de Petri de commande. Le choix est fait par expertise dès le niveau de description HiLeS, de manière à faire émerger une architecture classique, de type mémoire-µprocesseur (Figure 3-11). Nous montrons ici un exemple de marche à suivre.

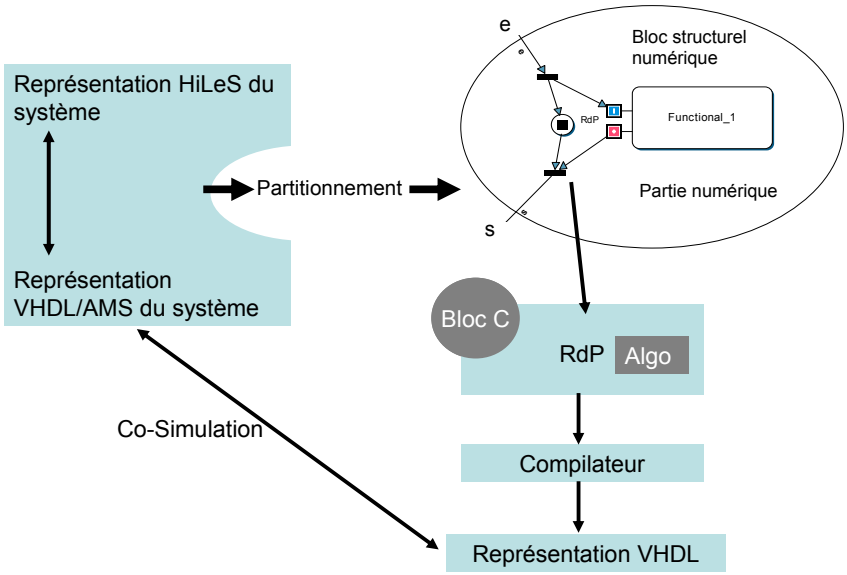


Figure 3-11 : Implémentation de la partie « Soft » de notre modèle.

(i) On a extrait de la représentation HiLeS, ce que l'on souhaitait réaliser par Soft. On l'appellera Bloc Structurel numérique.

(ii) Ce bloc est composé de deux entités :

- L'entité RdP.
- L'entité fonctionnelle.

(iii) La question qui se pose est la possibilité d'avoir une traduction automatique RdP/Langage C. L'entité fonctionnelle doit être écrite en C = Bloc C.

(iv) Procédure suivante habituelle : on « compile » le Bloc C en assembleur compatible avec le simulateur VHDL-AMS.

(v) On vérifie le tout par cosimulation sur simulateur VHDL-AMS.

Partant de cette idée, nous souhaitons vérifier le bon fonctionnement de notre code sur l'organe de calcul choisi du dispositif.

La Figure 3-12 décrit ce que nous proposent les outils actuels, en supposant que nous sachions transformer du VHDL-AMS en C et du VHDL-AMS en VHDL synthétisable.

Il apparaît, après discussions internes au laboratoire que les outils disponibles pour la simulation de processeurs offrent deux voies suivant le produit qu'ils visent à simuler (option 1 et 2 sur la Figure 3-12).

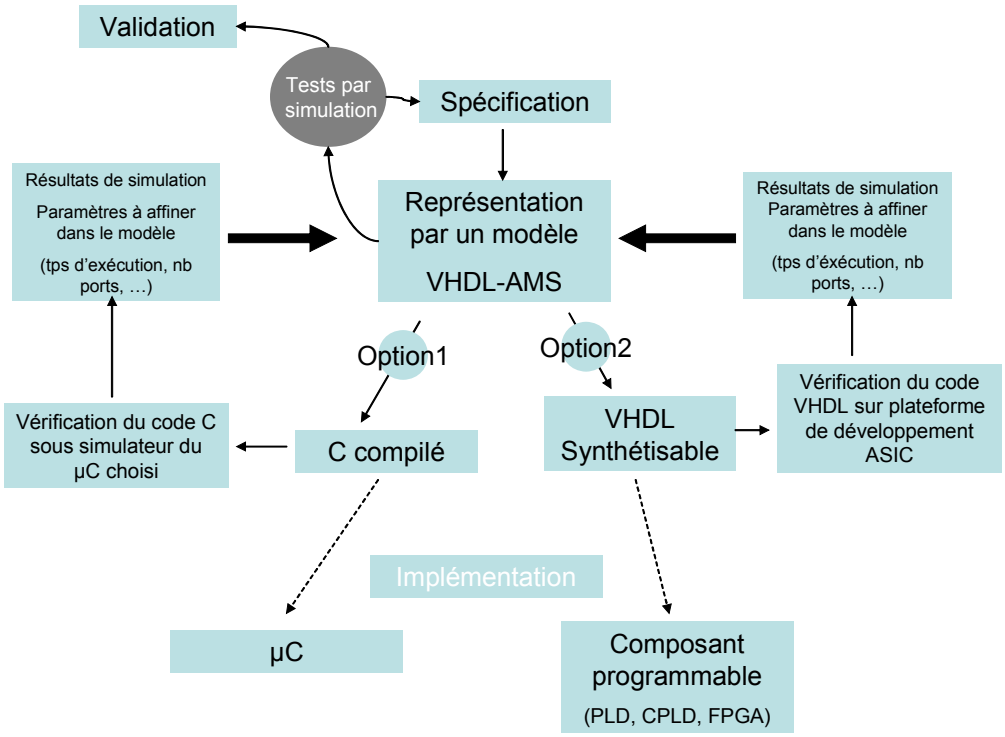


Figure 3-12 : Les deux voies proposées par les outils actuels pour la vérification de la partie Soft de notre modèle.

Le choix entre ces deux options se fait sur des critères tels que la vitesse d'exécution du code, le besoin de parallélisme, le coût, la taille mémoire, la consommation, etc.

La Figure 3-12 incite à deux commentaires :

- Dans le cas de l'option 1 : L'utilisation d'un microcontrôleur implique d'extraire du code VHDL-AMS un code C purement séquentiel. Ce code est ensuite vérifié dans un simulateur spécifique au microcontrôleur. Les résultats de simulation peuvent être réinjectés dans le modèle pour optimiser le modèle du système complet.
- Dans le cas de l'option 2 : Le modèle de commande VHDL garde ses propriétés de parallélisme d'exécution de tâches. Le rebouclage des résultats de vérification sur la représentation VHDL-AMS permet d'affiner les paramètres du modèle.

Le partitionnement logiciel-matériel est toujours une problématique d'actualité. Nous y apportons de surcroît une considération nouvelle qui est celle de l'inclure dans un système pluridisciplinaire où cohabitent non seulement des éléments électroniques et logiciels mais aussi mécaniques, fluidiques, thermiques, etc. Bien sûr, le partitionnement n'affecte que les parties fonctionnelles du système qui apportent une intelligence de calcul au système. Ce sont ainsi les éléments actifs qu'il faut trier et partitionner. Mais il est probable de voir non pas un processeur unique pour le système mais des processeurs dédiés à des tâches propres à chaque domaine (thermique, fluidique, mécanique), ce que l'on voit déjà apparaître sur les capteurs intelligents qui embarquent désormais un premier traitement numérique des signaux.

Nous avons ici mené une étude d'ordre général. Les outils disponibles, pour cosimuler matériel et logiciel sont très lourds à manipuler (Seamless de Mentor Graphics) et nécessitent une bonne expertise. Ces logiciels permettent de simuler un code logiciel sur une architecture processeur particulière décrite en VHDL. Le point que nous estimons difficile est celui d'incorporer le code VHDL du processeur intégrant l'algorithme logiciel dans le système, en remplacement de la partie numérique représentée sur la Figure 3-11. Cette intégration devrait alourdir considérablement la modélisation et augmenter le temps de simulation. En effet, l'échelle de description du processeur est celle des portes logiques, avec un temps de l'ordre de la microseconde, alors que la vue système doit être simulable à des temps de l'ordre de plusieurs secondes voir plusieurs heures.

Ainsi, nous avons mis en lumière quelques points bloquants qui devront être approfondis pour aboutir à la simulation de notre système complet intégrant une architecture processeur particulière.

3.4.3 Les options dans notre exemple

Au regard des exigences spécifiées dans le cahier des charges (faible coût, pas de mesures temps réel, très faible consommation), nous choisissons de nous orienter vers l'utilisation d'un microcontrôleur (option 1), car il est plus adapté à notre application qu'un composant programmable (FPGA, PLD, etc..) de par son faible coût, sa mise en œuvre simplifiée, sa faible consommation et ses capacités de reconfiguration dans l'optique de modifications ultérieures. Dans cette même optique, nous avons fait le choix d'un microcontrôleur plutôt que d'un microprocesseur qui développe certes une puissance de calcul plus importante mais qui est inappropriée à notre application. De plus, il est davantage consommateur d'énergie et n'incorpore pas de convertisseurs analogiques-numériques.

3.5 Spécifications et choix des fournitures

La modélisation du système global est élaborée à partir du choix de fournitures, sur catalogue ou à partir d'une bibliothèque standard. La phase d'agrégation a ainsi fait apparaître dans le système plusieurs « building-blocks » correspondant à différents domaines d'expertises technologiques. Cette partie est constituée d'un certain nombre de blocs fonctionnels qui décrivent les composants élémentaires du système et précisent leurs interfaces, leurs fonctionnalités attendues, les liens de parenté avec d'autres composants. Ce recueil d'information, qui concerne une partie du système, est mis à la disposition des fournisseurs qui peuvent ainsi faire une offre appropriée en réponse à ces spécifications. Nous discernons deux cas de figure : soit le bloc a une réalité physique qui correspond à une partie matérielle (ex. : un capteur), auquel cas nous procédons à la description comportementale de ce matériel en prenant en compte les propriétés que nous donne le constructeur (à terme les modèles VHDL-AMS devront être fournis), soit il correspond à un composant du système qui n'est pas encore développé, auquel cas, nous définirons un modèle de comportement théorique qui sera pris comme référence et bâti à partir des spécifications attendues qui ont été précisés sous HiLeS.

3.6 Le prototype virtuel du système

Le prototypage virtuel est la représentation informatique du système complet qui permet d'accéder à une première validation comportementale par simulation [Mau05a]. Au stade de nos analyses, le langage VHDL-AMS est sûrement le langage le mieux adapté pour cette mission.

Le prototypage virtuel commence avec la traduction des modèles amonts HiLeS en VHDL-AMS grâce à l'introduction de considérations technologiques « expertes » ou de « re-use ». Evidemment, à ce stade, plusieurs options sont possibles en fonction des technologies choisies, des fournisseurs, de la fourniture. Ce que nous avons dit précédemment est que cette première étape d'analyse des possibilités se termine par un (ou des) partitionnement(s) en « building blocks » pour lesquels la conception amont écrit un cahier des charges spécifique. On peut imaginer, dans la pratique industrielle, que ceci conduit au lancement par les concepteurs d'un appel d'offre à fournitures.

Le prototypage virtuel terminal ne pourra être réalisé que sur la base de la fourniture par chaque fournisseur des modèles VHDL-AMS de son produit. Une fois ces modèles intégrés dans la représentation virtuelle complète, on pourra efficacement anticiper les propriétés finales du système qui a été conçu.

Nous avons bien sûr tenté de suivre cette logique en faisant un premier prototype virtuel utile aux spécifications. Au delà, la réalisation que nous présenterons dans le chapitre 4 résulte d'une collaboration étroite qui a mêlé nos outils avec ceux de la société AREG [15], intégrateur de notre prototype matériel.

3.6.1 La modélisation VHDL-AMS dans l'exemple

L'exemple que nous avons traité décrit le système par les modèles suivants :

- Le modèle d'un environnement simple qui intègre les variations des quatre paramètres étudiés (Température, Humidité, Pression, Déplacement).
- Le modèle comportemental de chaque capteur qui génère pour chacun un signal analogique correspondant à sa tension de sortie.
- Le modèle d'un cœur de traitement de l'information (convertisseur A/N, calculs statistiques, mémoire).
- Les modèles de processus de fonctionnement (RdP) permettant de gérer : des alarmes, une séquence de mesure automatique, une séquence de téléchargement, un processus de mémorisation.

Nous avons déclaré certains paramètres de ces modèles en tant que paramètres génériques pour qu'ils puissent être manipulés et modifiés avant chaque simulation. Ces paramètres apparaissent généralement en en-tête de chaque fichier « .vhd » correspondant à un modèle particulier. Nous avons regroupé ces paramètres dans un « package » au sens VHDL, pour faciliter l'accès à la modification de l'ensemble de ces paramètres, à travers l'utilisation d'un seul fichier.

3.6.2 Les résultats de simulation

Afin d'illustrer la modélisation du système, nous présentons dans ce paragraphe, les simulations faites sur le fonctionnement du cœur de traitement de l'information, notamment la modélisation du fonctionnement du bloc « Analyse ». Selon le découpage HiLeS (Figure 2-29 du paragraphe 2.3.2.3), le bloc « Analyse » est composé d'un réseau de Petri et de 4 blocs fonctionnels : « Measure type », « Selector », « Sampler », « Analyser ». Les codes VHDL-AMS relatifs à ces blocs sont consultables en Annexe 4.

Nous illustrons ici les simulations relatives à chaque bloc, en faisant apparaître leurs ports d'entrée et de sortie et les signaux correspondant. Ceci, dans le cas d'un fonctionnement du système en mode automatique.

Le bloc « Measure_type »

Le bloc fonctionnel « Measure_type » (Figure 3-13) a pour but de déclencher le début de la mesure par le signal « Start_meas_select » et de déterminer le type du signal à mesurer « X_Meas_type » parmi les quatre provenant des capteurs.

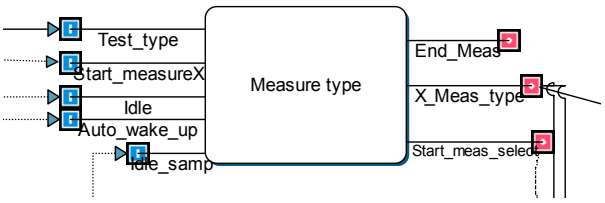


Figure 3-13 : Interface du bloc « Measure_type ».

Le chronogramme de simulation illustré sur la Figure 3-14 présente le déclenchement de la routine de mesure par le signal « auto_wake_up ». Simultanément, le signal de début de mesure «start_meas_select » et le signal déterminant le type de mesure « X_meas_type » sont émis. Pour chaque impulsion « end_meas », la valeur de « X_mea_type » est incrémentée, jusqu'à atteindre le nombre de capteurs.

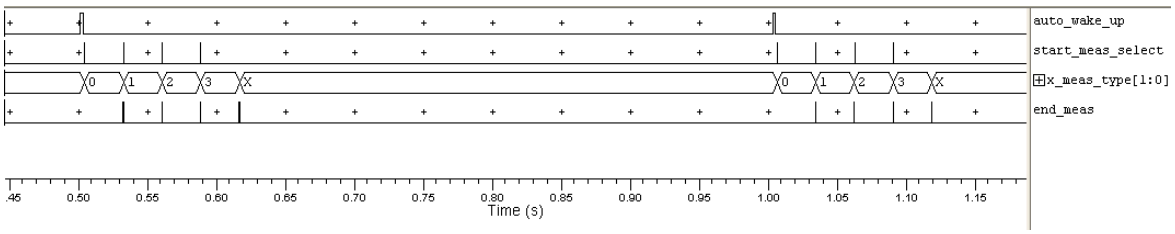


Figure 3-14 : Simulation du bloc "Measure_type".

Le bloc« Selector »

Le bloc « Selector » (Figure 3-16) joue un rôle de multiplexeur qui sélectionne l'un des 4 signaux analogiques relatif aux ports d'entrée (« Temp » : température, « Hum » : humidité, « Press » : pression, « Mov » : déplacement), selon la valeur du signal de référence « X_Meas_type ». Le port de sortie « X_Signal » prendra donc successivement les valeurs imposées par « X_Meas_type ».

Figure 3-15 : Interface du bloc « Selector ».

La simulation illustrée sur la Figure 3-16, présente les 4 signaux analogiques « press_v », « temp_v », « mov_v », « hum_v ». Le signal « x_meas_type » prend successivement les valeurs (0,1,2,3) pour définir le type du signal de sortie « x_v_signal » (respectivement la température, l'humidité, la pression et le déplacement).

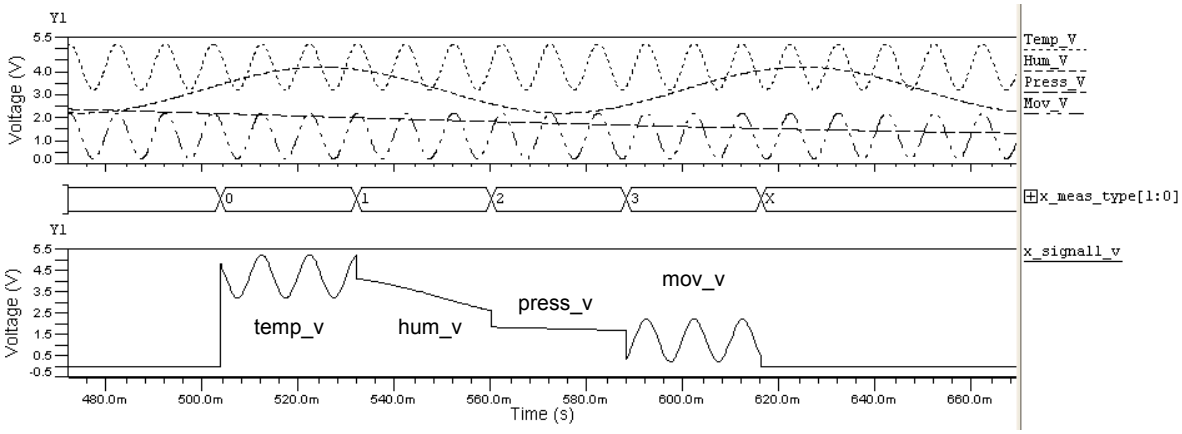


Figure 3-16 : Simulation du bloc « Selector ».

Le bloc « Sampler »

Le bloc « Sampler » (Figure 3-17) réalise l'échantillonnage du signal analogique d'entrée. L'échantillonnage doit être fait pour les quatre signaux distincts provenant des quatre capteurs (« X_Signal »). Le système doit réaliser six échantillonnages pour chaque signal. La conversion analogique se fait sur huit bits mais peut être modifiée par l'intermédiaire d'un fichier « Package » qui regroupe les paramètres génériques du système. Ces paramètres peuvent être modifiés avant chaque simulation.

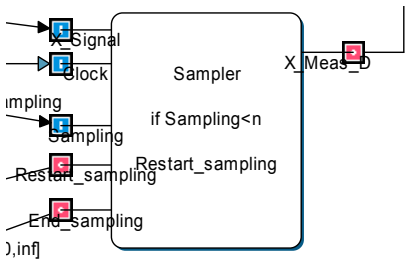


Figure 3-17 : Interface du bloc « Sampler ».

La simulation du bloc « Sampler » est illustrée sur la Figure 3-18. Nous proposons 3 vues qui illustrent différentes échelles de temps :

- La Figure 3-18.1 présente la vue d'ensemble du fonctionnement du bloc : le signal « sampling_select » déclenche l'échantillonnage du signal « X_Signal » à périodes régulières fixée par un paramètre générique.

- Sur la Figure 3-18.2, nous observons quatre demandes d'échantillonnages successives (« sampling_select ») qui correspondent au début d'échantillonnage de chaque signal de capteur.

- La Figure 3-18.3 présente les six échantillonnages faits sur le signal provenant du premier capteur.

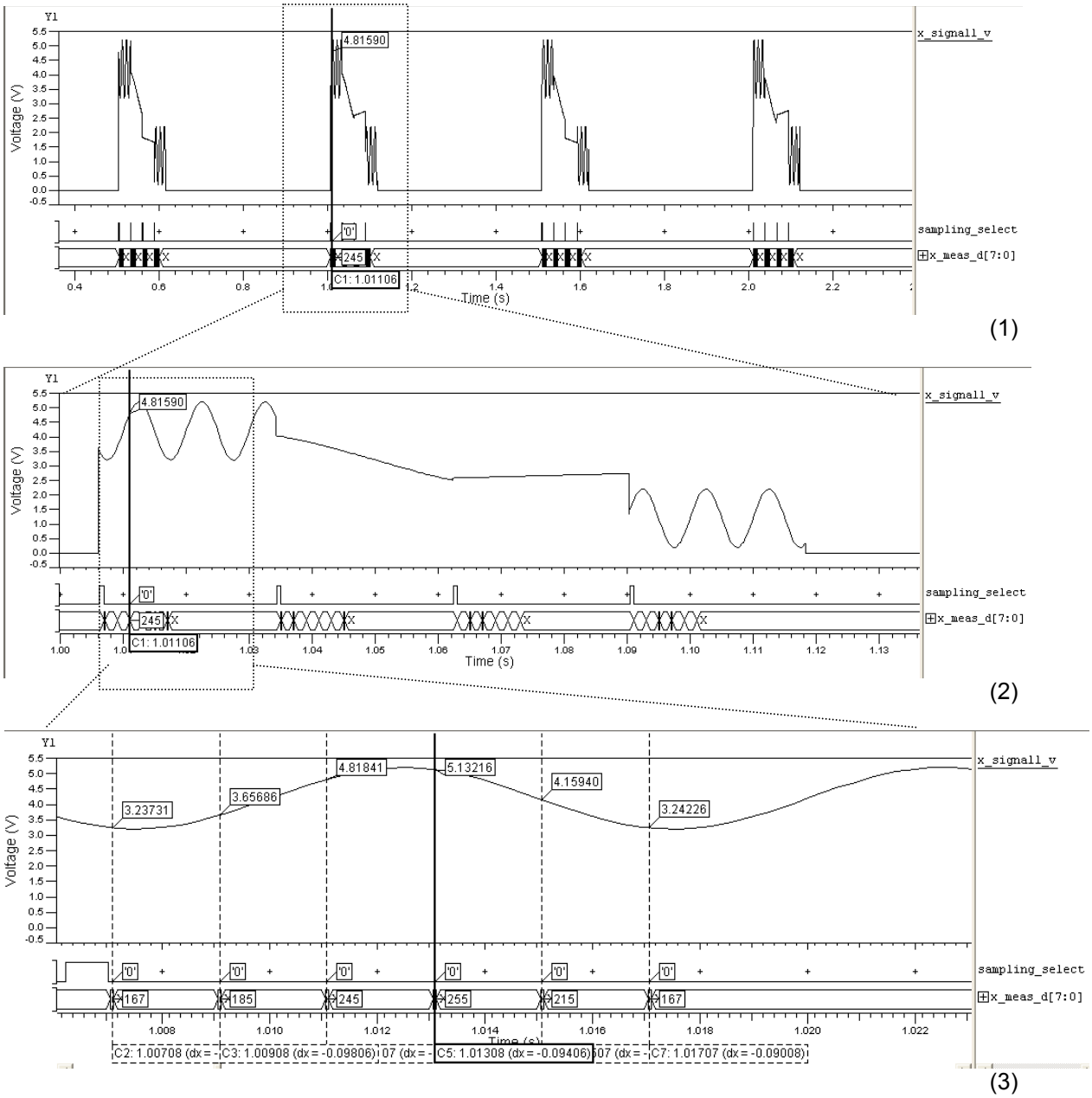


Figure 3-18 : Simulation du bloc « Sampler ».

Le bloc « Analyser »

Le bloc « Analyser » (Figure 3-19) calcule la moyenne des valeurs numériques reçues par le port « X_Meas_D », transmet le résultat sur le port « X_Datas » et signifie le début de mémorisation par le port « Start_memo ». Il compare le résultat aux valeurs portées par les signaux « MeasX_Database » et dans le cas d'un dépassement, il donne une alerte par le port « Alarm_test_meas ».

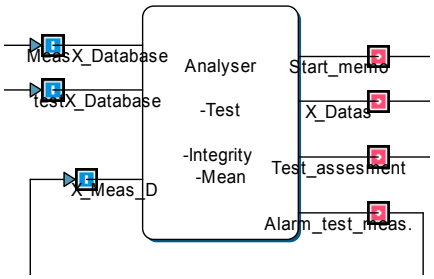


Figure 3-19 : Interface du bloc « Analyser ».

La simulation du bloc « Analyser » est illustrée sur la Figure 3-20. Nous observons le signal « x_meas_d » prendre successivement les six valeurs échantillonnées, ceci pour les quatre capteurs. La valeur moyennée est transmise sur le port « X_datas » en même temps que la demande de début de mémorisation « start_memo ».

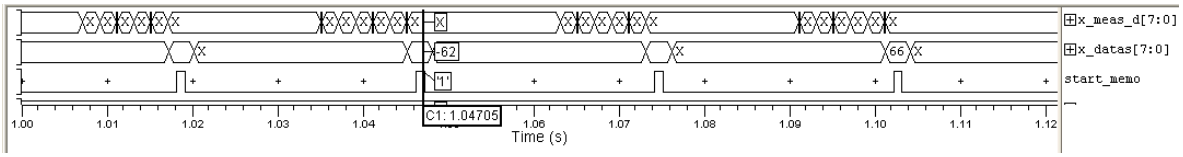
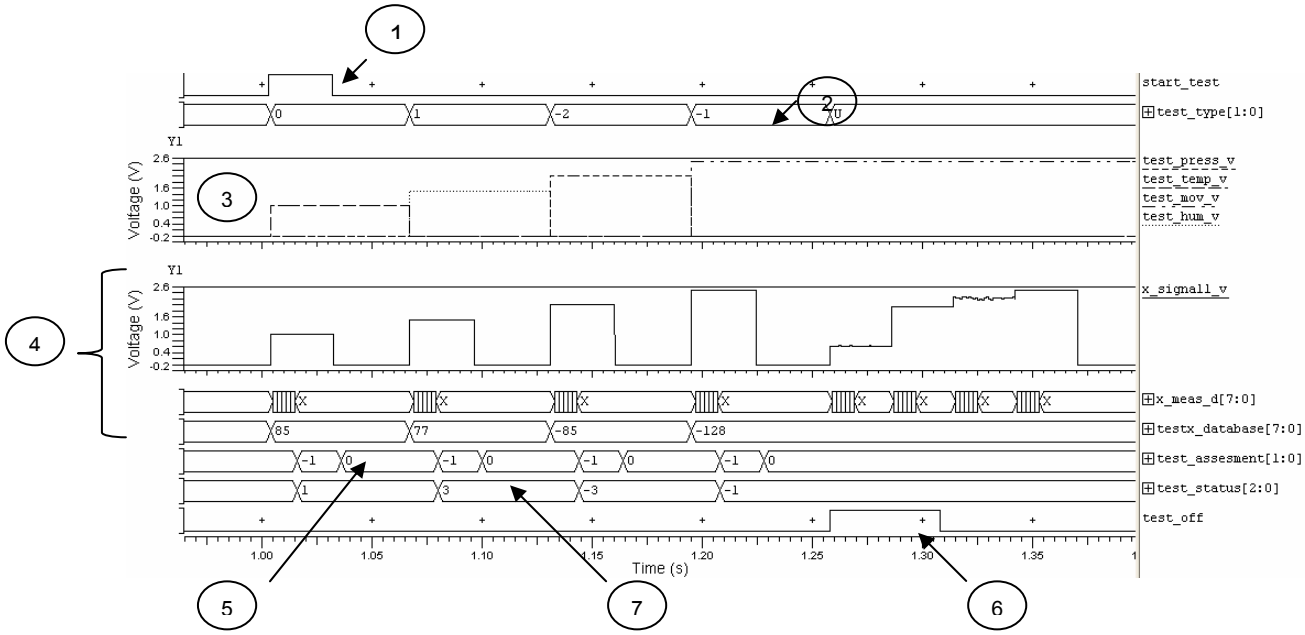


Figure 3-20 : Simulation du bloc « Analyser ».

3.6.3 Tests et validations

Le code VHDL-AMS de chaque bloc a été validé individuellement par simulations successives sur un banc de test « testbench » approprié. Nous avons ensuite agrégé ces blocs pour constituer pas à pas les blocs de plus haut niveau.

Du point de vue de la modélisation de la dynamique du système, les modèles ont été validés sur la base de la confrontation des chronogrammes de simulation et des événements particuliers décrits dans les procédures. Nous présentons sur la Figure 3-21 le cas de l'analyse de la simulation de la procédure « Tester ». La vérification se fait pas à pas, en suivant la description textuelle. Ainsi, chaque signal évoqué dans le texte doit être visible sur la simulation. Il doit répondre aux contraintes temporelles liées à l'ordonnancement des événements, et doit aussi porter les informations correspondantes aux fonctionnalités attendues par la description textuelle qui en est faite.



5.4.2 Procédure Tester

Le module ***Sensor Test*** *génère les signaux de test pour chaque type de microcapteur.*

1 - Initiation du test sur impulsion du module ***Initialise*** « Start_test » (cas d'un test séquentiel automatique de tous les microcapteurs) ou sur demande de la centrale de base ***H.M.I.*** « Start_test_X » (cas d'un test ciblé d'un microcapteur X).

2 - Configuration de l'unité de mesure en mode test : émission de « Test_type » vers les modules ***Analyse*** et ***Memory*** du bloc ***Measure Treatment***.

3 - Emission séquentielle d'un signal de test spécifique à chaque microcapteur.

4 - Echantillonnage du signal « X_signal1_V » par la fonction ***Sampler*** du module ***Analyse*** (« x_meas_d[7:0] ») et comparaison à un signal de référence provenant de la mémoire «TestX_Database ».

5 - Analyse du signal « Test_assesment » (résultat de test) provenant du module ***Analyse*** par la fonction ***Test assesment analyser*** :

- Un résultat négatif provoque le redémarrage de la procédure de test p fois (--)
- Si p est dépassé, un signal d'échec (e) est envoyé à la centrale de base ***H.M.I.*** et un signal de fin de test « Test_OFF » est envoyé vers le module ***Initialise***.
- 6 Si le résultat est positif (++), un signal de fin de test « Test_OFF » est envoyé vers le module ***Initialise***.

7 - Emission des résultats de test vers la centrale de base par le signal « Test_status » vers le ***Transceiver*** puis « Test_results » vers ***H.M.I.***(centrale de base).

Figure 3-21 : Vérification de la procédure textuelle « Tester », consignée dans le cahier des charges par confrontation du texte et du chronogramme de simulation.

Nous avons fait apparaître sur cet exemple sept événements qui apparaissent dans la simulation et qui valident la description textuelle de la procédure. Cette vérification doit toutefois être automatisée. L'utilisation de diagrammes de séquences UML peut être une voie de remplacement des textes descriptifs. Cependant, le lien qui doit être fait entre chaque représentation, dans un but de validation, reste un aspect délicat et non abouti pour l'automatisation de la démarche.

3.6.4 Difficultés rencontrées et recommandations

Les difficultés rencontrées sont principalement liées aux étapes de transition qui consistent à changer de langage de modélisation. Les outils de traduction de modèles sont encore au stade du développement. Ils sont semi-automatiques. Ils nécessitent encore une intervention importante du concepteur dans la phase d'agrégation du modèle global.

L'outil de transformation du modèle global du système HiLeS en VHDL-AMS est essentiel pour accéder à la modélisation d'une architecture globale stable du système. Si cette étape est réalisée par la voie manuelle, elle s'avère très longue et complexe quand le système prend de l'ampleur.

La modélisation de la hiérarchie fonctionnelle du système est considérée comme satisfaisante si l'on peut subdiviser un bloc en sous-blocs pour apporter des précisions supplémentaires dans son modèle de fonctionnement, et ceci, sans devoir modifier l'interface de ce bloc. Cela revient à dire que la démarche top-down doit considérer avec pertinence, l'intégralité des signaux d'interface à chaque niveau d'abstraction. Cette étape n'est pas toujours évidente, notamment dans la phase de développement d'un système où certains blocs doivent être définis très tôt sans connaître nécessairement l'intégralité du réseau des connexions environnantes. Dans ce cas, il faut envisager d'éventuelles modifications aux interfaces au fur et à mesure que le processus de description affine l'architecture du système.

3.7 Extensibilité de la méthode aux microsystèmes de surveillance

Cette méthode de conception permet de modéliser, du point de vue comportemental, des systèmes multi-domaines. Ainsi, nous pensons pouvoir approcher assez rapidement d'autres domaines technologiques d'applications nécessitant le cas échéant l'intégration de divers capteurs.

Nous avons suivi une démarche générique pour décrire notre système et avons choisi, du point de vue de la modélisation, d'utiliser une architecture de base (Figure 1-1, 1.2.2) qui puisse être réutilisée. La modularité du modèle réside aussi dans le fait d'utiliser des variables génériques qui peuvent être modifiées selon l'application voulue. Les paramètres accessibles sont par exemple le temps de réaction du système, la capacité de stockage, le nombre de capteurs, etc.

3.8 Conclusion

Le prototypage virtuel est une étape incontournable et un support de vérification dans la conception d'un système complexe. Nous avons détaillé les étapes pour y parvenir à partir de la représentation amont du chapitre 2.

A ce stade, nous étions en mesure de proposer un modèle « amont » du système : Blocs et interconnexions. La part fonctionnelle du système est définie par les allocations temporelles sur les blocs.

L'étape suivante du processus de conception est la vérification. Pour ce faire, nous avons interfacé la représentation HiLeS et la vérification TINA. Avec l'appui de nos collègues OLC-LAAS, nous avons engagé le processus de vérification et montré l'intérêt de la démarche par des exemples de propriétés que nous avons vérifié. A l'évidence, il y a un travail de recherche fondamentale à faire pour générer plus systématiquement les vecteurs de tests.

A supposer que l'architecture soit vérifiée, il faut faire un pas vers la réalité :

- Définir les modules du système.
- Agréger les fonctions et les répartir entre les modules.

Dans ce chapitre, nous nous sommes limités à l'agrégation des fonctions dans HiLeS et à la transformation de la représentation HiLeS à la représentation VHDL-AMS qui servira de support

aux simulations pluridisciplinaires. Les outils sont aujourd'hui opérationnels grâce aux contributions du groupe ISI-LAAS, pour l'introduction des formalismes réseaux de Petri.

L'étape terminale du processus de conception est le partitionnement qui anticipe le passage du concept aux choix technologiques. Du point de vue fonctionnel, notre proposition est de découper la représentation HiLeS en modules, en faisant appel à la connaissance experte que l'on a des technologies disponibles. Sur la base de ce découpage et des spécifications formelles qui peuvent lui être associées, on fait appel aux fournisseurs qui apportent la matière d'un véritable « prototype virtuel » du système.

Chapitre 4

Intégration, prototypage réel et validation

4.1 Introduction

Dans la représentation en « V » d'un cycle de vie d'un produit, le niveau ultime est celui de l'intégration. C'est cette étape que nous allons décrire dans ce chapitre. Nous disposons, au moins partiellement, d'une représentation VHDL-AMS du microsystème et avons la possibilité, par la simulation, de franchir des étapes ultimes d'optimisation puisque ce prototypage virtuel est sensé représenter l'exacte réalité. On peut considérer que, dans le processus de conception, l'étape de modélisation fonctionnelle est achevée. Il faut maintenant coupler ces modèles avec la représentation physique du support, c'est à dire disposer d'une **représentation structurelle** et **organique** du système. Ce couplage Fonction-Structure nous conduira à un **partitionnement** du système en modules physiquement définis (« Building blocks »).

Dans cette étape d'intégration, il est également question du choix d'une technologie d'assemblage à retenir. Des effets d'interaction peuvent apparaître entre les composants, nécessitant, le cas échéant, d'ajuster les modèles.

Les choix technologiques et les choix de fournisseurs de composants, même pour un système comme le notre de complexité limitée, peuvent conduire à des alternatives très variées et à des procédures de sélection (ce point est traité par ailleurs au LAAS et au Laboratoire d'Etude des Systèmes Informatiques et Automatiques LESIA [BRE04]). Dans ce travail, nous avons limité ces choix sur le partitionnement Matériel-Logiciel et sur les technologies d'assemblage (PCB-CMS).

Nous allons présenter ces étapes et décrire la réalisation du prototype réel avant de le valider par rapport au cahier des charges initial.

4.2 Préambule

Le lien entre la modélisation haut niveau et l'intégration physique du système est une partie délicate du processus de conception. Le modèle de haut niveau d'abstraction apporte la vision d'une décomposition du système en entités multiples, qui doivent se rapprocher du modèle physique afin que chacune d'entre elles puissent définir les spécifications d'un module à réaliser.

HiLeS est un outil qui nous a montré son intérêt dans l'aide à la description et la génération du prototype virtuel (Code VHDL-AMS). Ce point est le cœur des ambitions de HiLeS. Cependant, l'attente d'un schéma de conception complet, amenant jusqu'à la description du produit final est bien sur très forte.

Le langage VHDL-AMS a tous les attributs pour décrire les parties Analogiques et Mixtes d'un système. Cependant, nous avons repéré un point de rupture dans la chaîne de conception, lors de la réalisation des schémas du prototype réel à partir du prototype virtuel. En effet, cette étape nécessite un partitionnement des parties analogiques et mixtes qui ont été modélisées conjointement. Actuellement, le partitionnement sous VHDL-AMS n'est pas résolu.

Après avoir décrit les deux possibilités de partitionnement dans le paragraphe 3.4.2., et de part les contraintes de coût et de temps citées plus haut, nous avons fait le choix d'utiliser un microprocesseur de type PIC 16F comme plate-forme matérielle d'implémentation de la partie logicielle du modèle. De plus, la description faite de l'activité du système ne présente pas de parallélisme, ce qui est une caractéristique qui aurait induit, sinon, l'utilisation d'une plate-forme de type FPGA. Ici encore, le pont entre le VHDL-AMS et le code machine n'est pas établi. Nous avons

du passer par une étape de réécriture du code VHDL-AMS en C puis assembleur pour l'implémenter sur le microprocesseur.

Du point de vue de la schématique, la représentation physique des composants et de leur connectique électrique peut être obtenue par VHDL-AMS. Nous avons cependant utilisé le logiciel Orcad/PSpice pour l'aide qu'il apporte à la réalisation du routage de la connectique pour le dessin des PCB.

Dans le chapitre 1, nous avons présenté, avec une vue typiquement microsystème, les moyens technologiques qui sont à disposition et qui peuvent être coordonnés pour la réalisation d'un prototype final. Dans notre étude, cette réalisation finale s'est confrontée à plusieurs contraintes qui nous ont fait dévier de l'implémentation envisagée dans le chapitre 1. Nous avons du, d'une part, tenir compte du temps dont nous disposions pour réaliser le système, et d'autre part, tenir compte du coût qui peut rapidement s'accroître dans le cas d'une mise en œuvre de type microsystème. Malgré cela, la démarche générale reste la même. C'est pour ces raisons que notre réalisation finale présente un décalage par rapport au discours du chapitre 1.

Compte tenu de ces considérations, nous présenterons dans ce chapitre la réalisation du prototype, basé sur les technologies CMS et COTS qui permettent de mettre en avant l'idée de réutilisation de composants du commerce.

4.3 Modélisation structurale et organique

La modélisation structurale et organique consiste en une description du produit du point de vue « physique », pour répondre à la question : Comment se présente le produit à l'utilisateur ? Il faut donc revenir au cahier des charges (Chapitre 1) où sont décrits les exigences du maître d'ouvrage.

Le plus souvent, la structure du système se réfère à des produits déjà existants sur lesquels la maîtrise d'ouvrage apporte des aménagements personnalisés sans réellement changer ni les organes essentiels ni leurs modes d'assemblage. Ainsi, une voiture comportera : des roues, un moteur, un habitacle, des portes, etc. Ce sont des « organes standard » qui relèvent d'une « expertise métier ».

Dans un système complexe, la modélisation structurale et organique peut nécessiter des démarches de description complexes pour représenter toute l'architecture physique du produit : ses modules, ses interfaces d'assemblage, ses organes de communication, etc. Des travaux parallèles aux nôtres [RHG05] explorent cette démarche et son couplage avec la modélisation fonctionnelle pour aboutir à un « partitionnement » et obtenir des modules complets, des sous-ensembles décrits en terme de structures supports et de fonctions embarquées.

Le passage de la représentation fonctionnelle aux composants du système est illustré sur la Figure 4-1, par une vue arborescente du système à cette étape. La partie gauche de la figure concerne la vue fonctionnelle du système, avec ses blocs fonctionnels (F) et structurels (S), tandis que la partie droite présente la vue des composants (C) du système. Il s'agit d'allouer des fonctions à des composants matériels et d'en représenter leur lien. Par exemple, les fonctions F7 et F2 seront réalisées par le composant matériel C7. Les composants ainsi formés sont définis par les spécifications d'interfaces propres aux fonctions. Il faut bien comprendre que la hiérarchie fonctionnelle du système qui a été vérifiée précédemment (2.4.2) n'est pas remise en cause. Nous donnons seulement un autre point de vue du système.

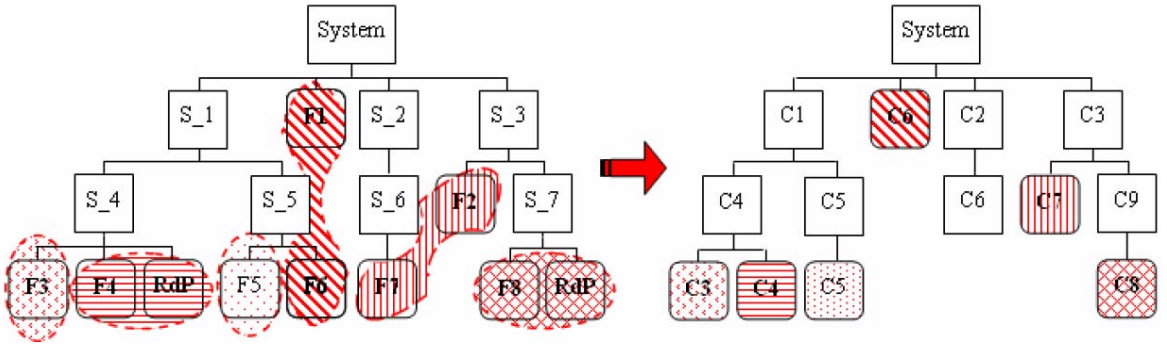


Figure 4-1 : Passage de la représentation fonctionnelle au composant.

Nous allons respecter cette logique, en l'appliquant à notre cas d'application, même si il n'y a aucune complexité apparente.

4.3.1 Description structurale et organique

Dans le Chapitre 1 présentant le cahier des charges fonctionnel, il n'y a pas d'exigences particulières sinon celle de réduire la taille du système pour tirer le meilleur parti de la démarche d'intégration.

En considérant les propositions existantes (expertises métiers), les systèmes de mesure impliquant un contact avec l'ouvrage Génie Civil à surveiller comporte un système d'ancrage en deux points et un corps de mesure. Avec une approche identique, notre système va comporter cinq modules (Figure 4-2) :

- Un module « mesure ».
- Un module « sonde mécanique ».
- Un mécanisme de transmission « renvoi d'effort ».
- Deux mécanismes d'ancrage des modules « mesure et sonde » sur le béton.

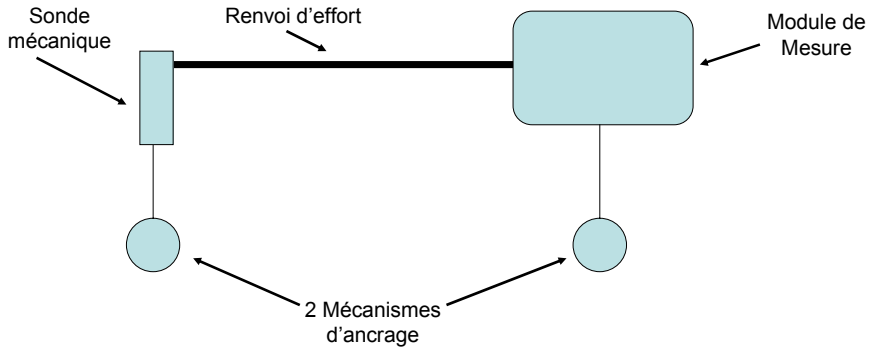


Figure 4-2 : Les 5 modules du système complet.

4.3.2 Décision de partitionnement

Nous utilisons ici le terme partitionnement, au delà de la vision électronique Hard-Soft standard, au sens de l'allocation de fonctions ou de groupements de fonctions du système à des blocs de base qui peuvent être matérialisés par des composants existant. Notre décision de partitionnement est simplement d'intégrer les fonctions, définies dans le Chapitre 1, au module de mesure pour des raisons évidentes :

- d'alimentation énergétique (un seul module électriquement alimenté),
- de facilité de montage sur site.

Cela donne les descriptions suivantes :

- le module de mesure comporte toute l'électronique et tous les capteurs et intègre une source d'énergie et une interface de communication,
- la sonde mécanique sera constituée d'un ancrage béton et des systèmes de réglages indispensables à l'étape d'installation, réglages qui doivent rester passifs et stables,
- le renvoi d'efforts comporte le renvoi proprement dit (tige rigide à faible cœfficient de dilatation thermique) et sa protection mécanique,
- les mécanismes d'ancrage.

La représentation « amont » du système global sous HiLeS (Figure 4-3) illustre notre décision de partitionnement et fait apparaître les deux familles de modules : mécaniques et mesure (SmartGec).

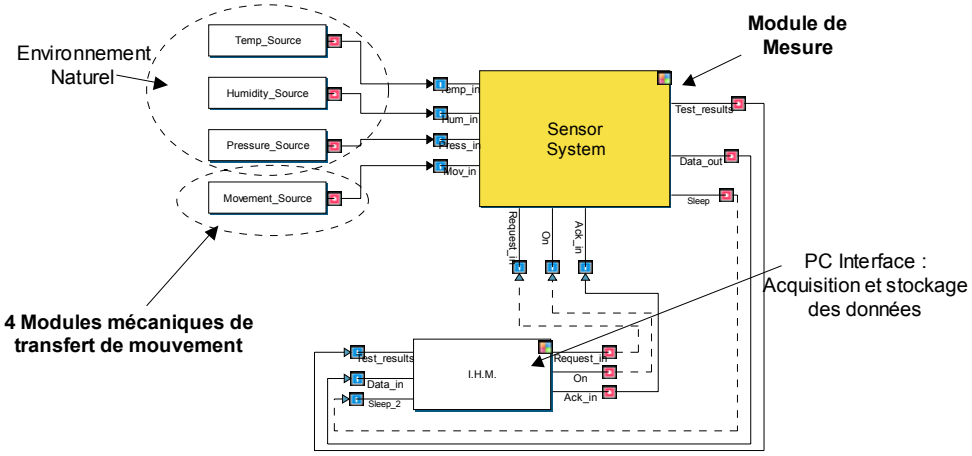


Figure 4-3 : Choix de partitionnement du système global.

En suivant la même démarche, nous proposons d'utiliser la description HiLeS pour partitionner le module mesure (« Sensor System »), afin de faire apparaître des blocs génériques, réutilisables dans le cas d'applications microsystèmes. La transformation de la représentation haut niveau vers la représentation organique est simple et directe. Elle fait apparaître 4 « building blocs » principaux (illustrés par la Figure 4-4) pour le module « Sensor System » :

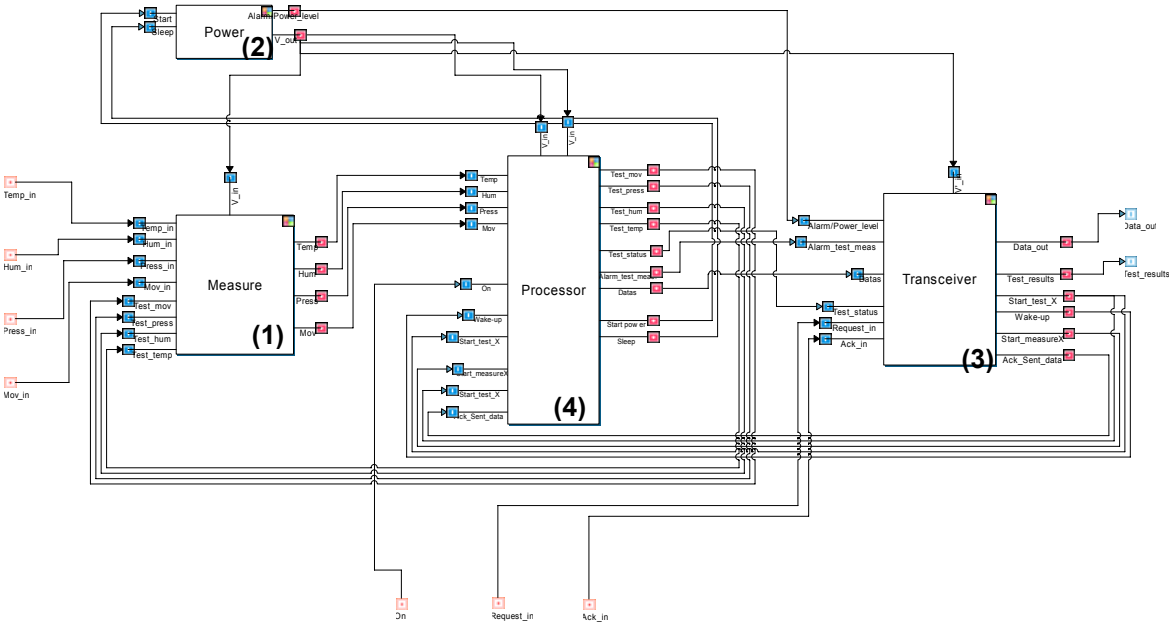


Figure 4-4 : Niveau 1 HiLeS.

- Bloc 1 : Il regroupe les microcapteurs permettant de convertir les paramètres extérieurs en signal exploitable par le système.
- Bloc 2 : Il constitue la source d'énergie du système et fournit une tension régulée.
- Bloc 3 : Il gère la communication Radio-Fréquence entre le dispositif « Sensor System » et la centrale de base distante.
- Bloc 4 : Il regroupe les procédures de traitement des signaux (analyse, stockage en mémoire, gestion d'alarmes).

Dans notre cas, la description du premier niveau dans HiLeS, suffit à faire apparaître les principaux composants constitutifs du système.

4.4 Les choix d'intégration

Dans la démarche d'intégration d'un système complexe, au stade du partitionnement en modules complètements décrits, s'ouvre l'étape du choix des technologies et des fournitures. Cette démarche peut comprendre :

- la simple consultation d'un catalogue des fournitures et des fournisseurs,
- le lancement d'un appel d'offre et de mise en concurrence de plusieurs fournisseurs,
- l'appel à des travaux supplémentaires d'ajustement des performances et même de recherche, qu'il faut éviter toutefois à ce stade d'un projet.

Cette démarche inclut, si nécessaire, un retour sur la modélisation pour tenir compte des propriétés des fournitures qui ne seraient pas exactement incluses dans les simulations déjà réalisées. Un point important est donc la **fourniture de modèles par les fournisseurs en même temps que les composants**.

Un autre point essentiel de la démarche est la sélection qu'il faut faire entre les diverses options possibles. Si l'on inventorie pour une fourniture donnée les différentes technologies possibles, les différents fournisseurs et les variantes qui s'y associent, on est face à une problématique très complexe où interviennent des critères de coût, de disponibilité, de performance, etc. Cela revient à décrire tous les chemins d'intégration possibles et à sélectionner les mieux adaptés au problème (Figure 4-5).

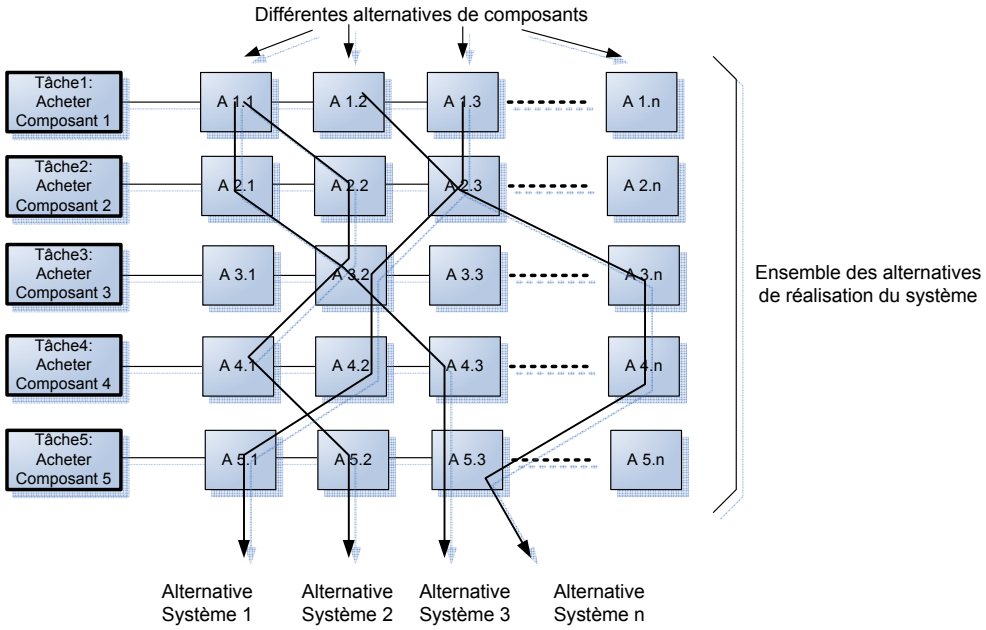


Figure 4-5 : Choix multiples des alternatives de réalisation du système.

Nous renvoyons ici le lecteur aux travaux menés en parallèles sur ce point [CBG05]. Nous avons fait des choix de bon sens compte tenu du contexte de recherche dans lequel nous avons travaillé en considérant :

- la conformité au cahier des charges,
- les fournisseurs de proximité.

Dans une phase de valorisation industrielle, ce sont des choix qui sont à reconsidérer.

4.5 Réalisation du microsystème

4.5.1 Les composants mécaniques utilisés

Nous décrirons ici les modules passifs de notre système et leurs caractéristiques propres qui présentent un intérêt pour obtenir les propriétés voulues du système. Pour cette partie du système, l'exigence particulière du cahier des charges concerne le micro-positionnement (exigence 4.6.1.3.). L'objectif fixé est celui d'être capable de déterminer la position relative entre deux ancrages avec une précision inférieure à ±5µm. Cette caractéristique est clairement la contrainte forte qui nous sert de base pour la conception des parties mécaniques. Ainsi, nous argumenterons les choix que nous avons effectué pour réaliser les 4 modules mécaniques. Comment transmettre un déplacement micrométrique d'un point d'ancrage à un autre ?

A cette échelle de dimension, l'effet de la température sur la géométrie des matériaux est le facteur déterminant qui doit être considéré pour garder la mesure la plus précise possible. Nous allons montrer comment cet effet a été pris en compte dans la conception des mécanismes de transfert de position.

Les deux modules « ancrage béton »

Nous nous sommes appuyés sur les pratiques communément suivies en laboratoire de Génie Civil pour la réalisation de ces plots d'ancrages. L'assemblage choisi pour notre module d'ancrage est celui du collage d'un plot métallique sur le béton. Nous avons choisi d'utiliser l'aluminium pour des raisons de rapidité d'usinage de pièces sur mesure dans notre atelier. Nous utiliserons cet ancrage par collage pour le module de « mesure » et pour le module « sonde ».

Le module « renvoi d'effort »

Le principe est de mesurer le déplacement relatif de deux points selon la droite passant par ces deux points. La Figure 4-6 illustre le cas de deux ancrages marqués d'une croix. L'ancrage gauche se déplace d'une position 1 à une position 2, suite à un déplacement de la paroi sur laquelle ils sont placés. Seule la composante de déplacement selon l'axe de mesure sera répercutée sur l'ancrage droit.

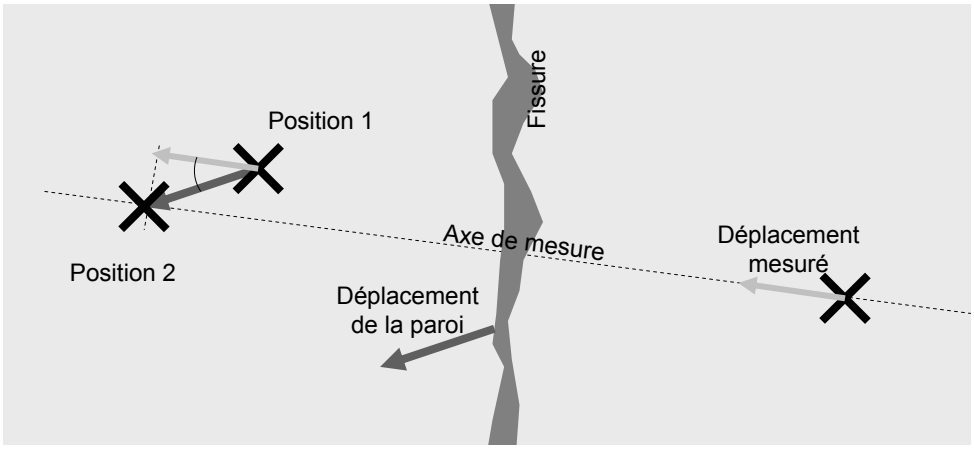


Figure 4-6 : Déplacement réel et déplacement mesuré.

Aussi, nous observons que l'orientation de l'axe de mesure par rapport à la fissure peut varier dans le temps. Il faut donc prévoir de rendre les deux points d'ancrage « libres » en rotation. Ainsi, le module de renvoi d'effort que nous avons choisi est une tige rigide d'un métal à expansion thermique contrôlée (INVAR) dont une extrémité est rendue solidaire de l'ancrage béton par une liaison rotule et l'autre par une liaison rotule et glissière pour ainsi permettre le déplacement libre selon l'axe de mesure (Figure 4-7).

Figure 4-7 : Liaisons mécaniques des ancrages.

Le module « sonde »

Nous avons conçu le module sonde de manière à ce qu'il y ait un seul point d'ancrage dans le béton et que ce point soit à l'aplomb de la rotule. Ainsi, la symétrie de l'encollage, du plot et de la rotule, autour du point d'ancrage, nous permet de considérer que l'effet de la dilatation thermique de ces parties n'influence pas le déplacement mesuré sur l'autre point d'ancrage. Ce module incorpore une vis de réglage micrométrique, pour le positionnement de la tige INVAR, indispensable pour l'étape d'installation du système sur site.

La Figure 4-8 illustre l'ancrage prévu du coté de la sonde mécanique : une gaine formée d'un tube de matériau composite carbone/époxy est collée dans la rotule par une colle cyanocrylate qui assure une liaison intime entre les deux composants. La distance d1 sépare la rotule de la pièce d'aluminium (1). Le tube carbone/époxy est serré par la pièce d'aluminium (1). La distance d2 est ajustable par la vis de réglage micrométrique. La pièce d'aluminium (2) serre la tige de renvoi d'effort. Cette tige a une longueur d3 qui correspond à l'écartement entre les deux ancrages : module « mesure » et module « sonde ». Nous avons ajouté deux ressorts, entre le plot d'ancrage

et le tube de carbone/époxy, afin de contraindre la rotule et ainsi limiter son jeu interne. En effet, le jeu de la rotule peut interférer sur la mesure dans le cas d'un changement de l'orientation du déplacement mesuré.

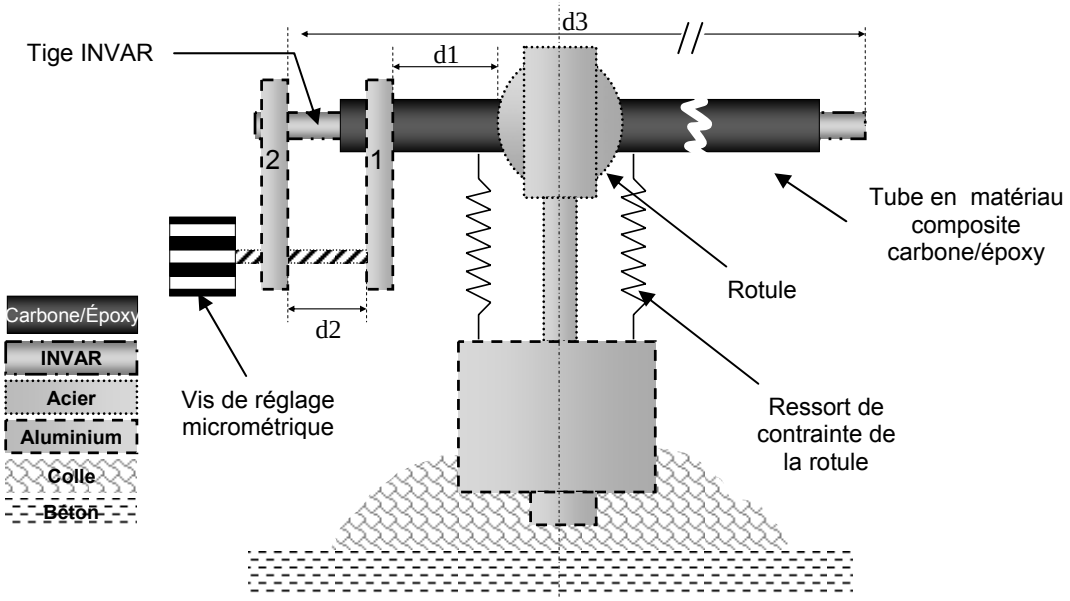


Figure 4-8 : Module « sonde » et son réglage micrométrique.

Comme la mesure d'écartement relatif des deux ancrages est faite selon l'axe de la tige, nous devons tenir compte de l'effet de la dilatation thermique des matériaux sur cet axe, sur les distances d1, d2 et d3, impliquant respectivement un matériau composite carbone/époxy (d1≈ 2mm), une vis d'acier (d2 ≈ 10mm) et une tige d'INVAR (d3 ≈ 1000mm).

Le cœfficient de dilatation thermique linéaire α est donnée en K^{-1} par : $\alpha = \frac{\Delta l}{l \cdot \Delta T}$

Où Δl est la dilatation provoquée par la variation de température ΔT sur la distance l.

Le Tableau 4-1 liste les coefficients de dilatation thermique des matériaux utilisés.

Matériau	Composite Carbone/Epoxy	Acier Fe37	Alliage à expansion contrôlée (INVAR Fe64/Ni36)	Aluminium 2017 (UNS A92017)	Colle structurale à base de Polyméthacrylate de méthyle (Plex7742/ Pleximon801)
Coefficient de dilatation thermique linéaire (α)	α_1 = 2.0x10^{-6} K^{-1}	α_2 = 12.0 x10^{-6} K^{-1}	α_3 = 2.0 x10^{-6} K^{-1} (20-90°C)	24.0 x10^{-6} K^{-1}	10-19 x10^{-6} K^{-}

Tableau 4-1 : Coefficients de dilatation thermique linéaire des matériaux utilisés.

Nous devons donc, lors de l'acquisition des mesures de distance, appliquer une correction, en soustrayant une erreur due à la dilatation des matériaux sur les trois distances décrites plus haut. La chaîne de côtes des dilatations est illustrée sur la Figure 4-9. Elle nous indique l'orientation des dilatations.

Ainsi, la valeur de la correction est la suivante :
$\Delta l_{Correction} = (\alpha_3 \cdot d_3 - \alpha_1 \cdot d_1 - \alpha_2 \cdot d_2) \cdot \Delta T$.

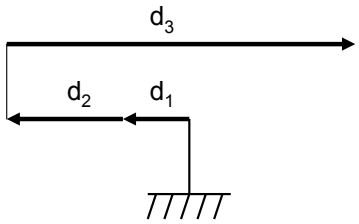


Figure 4-9 : Chaîne de côtes des dilatations considérées.

Ainsi, par exemple, pour une variation de +30°C et en appliquant l'équation précédente, nous obtenons :

$$\Delta l_{Correction} = \left(2 \cdot 10^{-6} - 2 \cdot 10^{-6} \times 2 \cdot 10^{-3} - 12 \cdot 10^{-6} \times 10 \cdot 10^{-3}\right) \cdot 30 \approx 57 \mu m$$

Or, la part des dilatations dues au matériau composite et à l'acier représentent 3,7µm. Ainsi, si nous ne corrigeons que l'effet de dilatation de la tige d'INVAR, nous obtenons une valeur mesurée entachée d'une erreur qui reste dans le cadre des exigences spécifiées (±5µm). Cependant nous prendrons en compte toute la chaîne de côte, telle que définie précédemment pour réduire l'erreur sur la mesure.

4.5.2 La sous-traitance électronique

Grâce au travail présenté au Chapitre 2, nous avons pu fournir un dossier complet de ce que nous attendions de la part des fournisseurs. Nous avons ainsi pu contacter une entreprise, AREG [15], avec laquelle nous avons collaboré pour la mise en œuvre. Nous avons fourni un schéma de principe général proposant l'architecture voulue pour le système, une liste de composants « critiques » à intégrer dans le système (Annexe 1) ainsi que l'algorithme de fonctionnement du système (Annexe 2). L'expertise de l'entreprise a permis de traiter rapidement le découpage du schéma selon le partitionnement choisi et de considérer les règles de routage pour la réalisation de PCB. Nous avons supervisé le choix terminal des composants et le suivi de la réalisation.

Le partitionnement du système fait apparaître quatre blocs : mesure, alimentation, communication et énergie. Nous avons tenu à garder une séparation physique entre ces quatre éléments sous la forme de 4 circuits PCB distincts. Ainsi, nous apportons une plus grande souplesse de modification et de réutilisation du système en donnant la possibilité de le manipuler par blocs élémentaires.

Le choix a été fait d'utiliser une technologie standard d'assemblage 3D par un empilement des quatre PCB. Ainsi, nous avons du considérer un mode d'interconnexion par un bus vertical, jouant à la fois un rôle structural de maintien et d'assemblage des niveaux et celui d'interconnexion électrique inter-niveaux. La Figure 4-10 présente les cartes électroniques qui ont été conçues pour les 4 niveaux. Le module « énergie » incorpore une pile, des régulateurs de tension et un gestionnaire d'alimentation (micro-relais). Le module « mesure » incorpore 4 capteurs. Le module « communication » intègre le module radiofréquence et son antenne. Le module « traitement » intègre un microcontrôleur, une horloge temps réel et une mémoire.

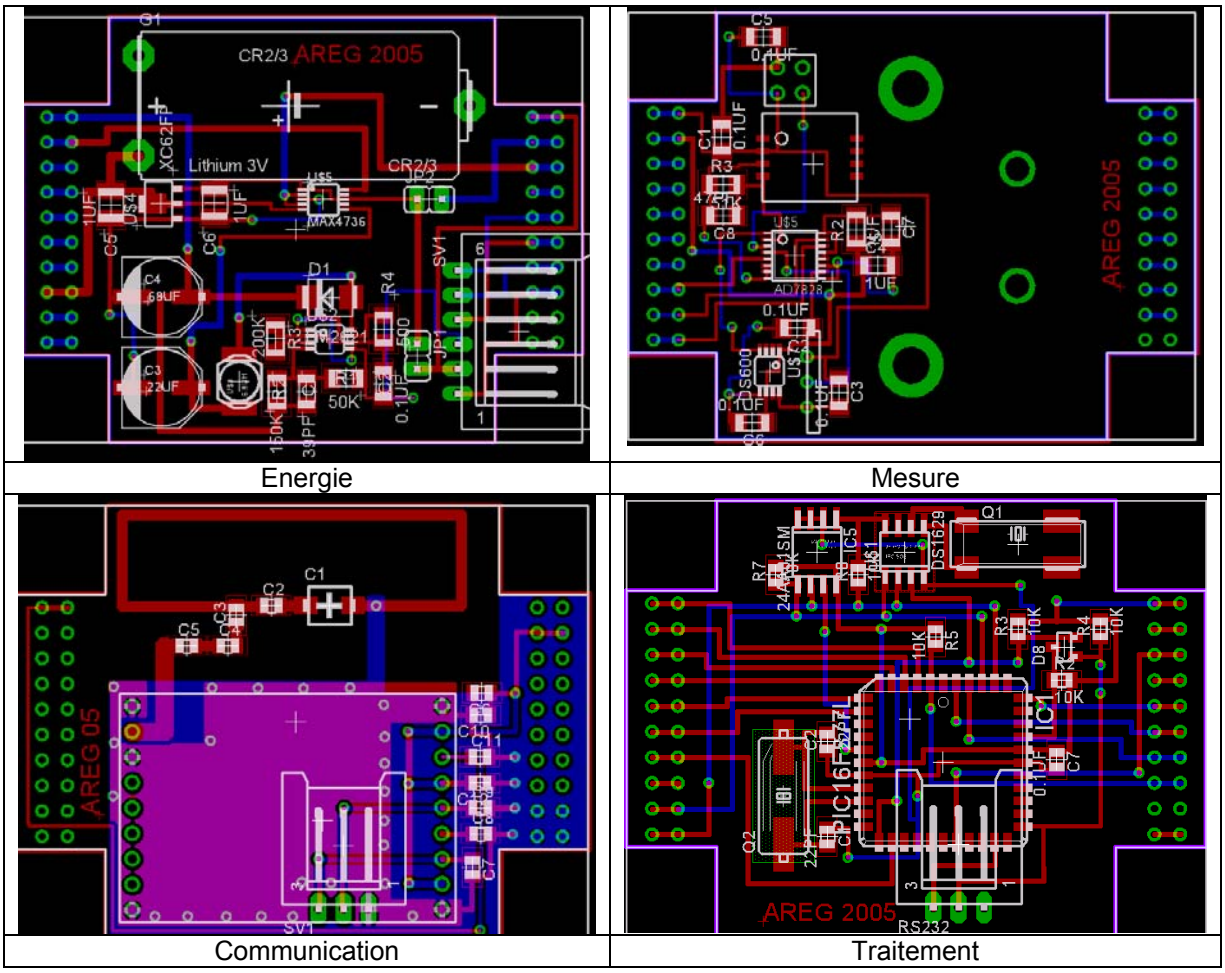


Figure 4-10 : Les quatre niveaux électroniques modulaires de SmartGec.

4.5.3 Test des fournitures

Chaque élément considéré comme étant critique, pour la précision de la mesure (microcapteurs), pour la durée de vie du système (source d'énergie), pour l'intégrité de la communication (module RF), a été testé afin d'établir une première validation de la fourniture avant son intégration dans le système. Ainsi, nous avons contribué avec EDF R&D à **valider les fournitures** telles que **les microcapteurs** [May04] et **les sources d'énergie** [Ber04] ainsi que le choix du **module de communication** radio-fréquence [Seb05]. Ces travaux ont permis de tester les fournitures et de supporter nos choix. Les réalisations mécaniques ont été faites au LAAS dans notre atelier de mécanique.

Le point le plus délicat a été de valider notre choix de détection de déplacement. Une étude détaillée du capteur a été réalisée [Mau04c] pour caractériser :

- La sensibilité.
- L'hystérésis du signal lors d'un déplacement avec changements de direction.
- L'incertitude sur la mesure de déplacement.
- La stabilité du signal dans le temps.
- La consommation.

Le microcapteur est représenté sur la Figure 4-11, accompagné de son électronique de traitement du signal. L'élément mobile est de la taille de la tête d'une épingle et a une course de 3mm. Le plongeur est muni d'un ressort de rappel. Il est donc en position d'extension complète au repos. Le corps métallique cylindrique a une longueur de 24mm pour 1,8mm de diamètre. L'électronique est contenue sur une plaque de 10X40mm.

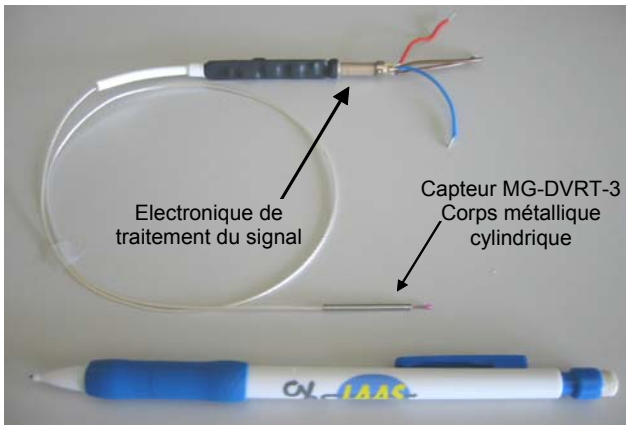


Figure 4-11 : Le microcapteur et son électronique de traitement du signal.

Le banc de test est présenté sur la Figure 4-12. Il est composé d'une platine de déplacement linéaire Newport M-UMR8.25 sur laquelle est monté un support amagnétique (téflon + polyéthylène) à l'intérieur duquel est fixé le microcapteur MG-DVRT-3 de type LVDT (Linear Variable Differencial Transducer). La platine Newport permet les déplacements linéaires millimétriques. En regard de cette platine, est monté un vérin piézoélectrique Newport AD-30 permettant des déplacements de 0 à 30µm avec une incertitude de 0,04µm.

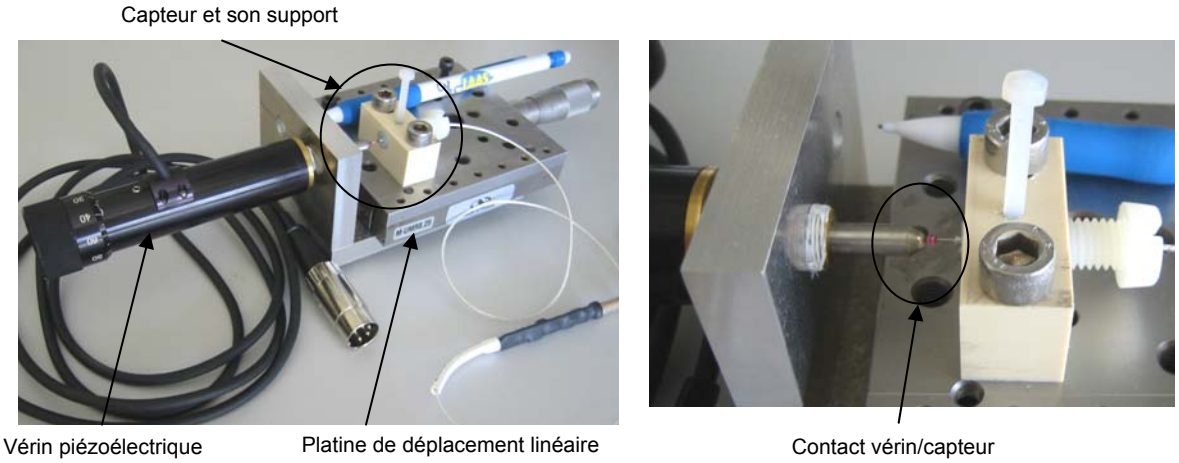


Figure 4-12 : Le banc de test de micro-déplacements linéaires.

Nous allons rappeler les principaux résultats obtenus. Tout d'abord, nous avons mesuré la sensibilité du capteur, sur toute sa gamme de mesure. Nous avons obtenu une valeur de 0,65 mV/µm, pour une alimentation du capteur de 5V. Ainsi, pour obtenir la résolution souhaitée pour notre système, il nous faudra utiliser un moyen d'acquisition du signal ayant une résolution du millivolt.

Ensuite, nous avons étudié l'évolution du signal dans le temps. Nous avons observé une variation, après la première mise sous tension, avec une stabilisation après quelques minutes (Figure 4-13).

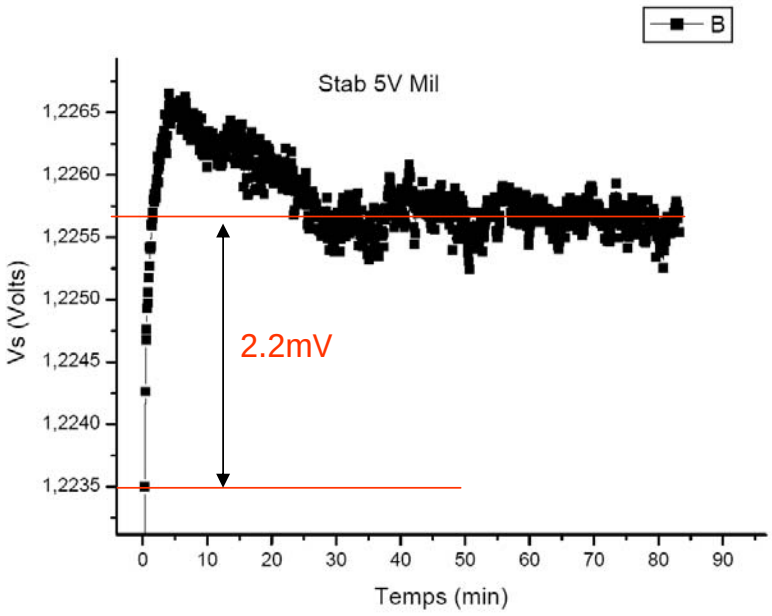


Figure 4-13 : Evolution du signal du microcapteur dans le temps pour une position fixe.

Cette évolution présente un saut de 2,2mV entre la mesure réalisée au bout de cinq secondes et la mesure stabilisée. Cette instabilité peut engendrer une erreur d'interprétation s'élevant à 1,4µm. Nous tiendrons compte de cet écart lors de la prise de mesure périodique de notre système dans les premières cinq secondes après la mise sous tension. En régime stabilisé, le capteur permet de faire des mesures avec une incertitude élargie de ±1,4µm. Il répond donc positivement à l'exigence du cahier des charges (±5µm).

4.5.4 Evaluation de la consommation

L'un des objectifs majeurs de notre application est d'être autonome en énergie pendant une durée supérieure à un an. Ainsi, nous avons voulu définir le besoin énergétique global du microsystème. Pour cela, nous avons établi une liste de composants type avec leur consommation ainsi que les phases de vie du système avec la nature des composants sollicités. Nous avons ensuite émis des hypothèses quant au choix des composants, en maximisant à chaque fois leur consommation et leur durée d'utilisation. Cela nous a ainsi amené à calculer la consommation globale mensuelle maximale du système.

Nous avons défini 5 phases de fonctionnement qui présentent des consommations distinctes, numérotées respectivement de 0 à 4 : la veille, l'acquisition automatique, l'envoi automatique d'informations, la réponse à une requête de mesure et le déclenchement d'alarme. Nous avons alors évalué la consommation sur la base de 3 scénarios types (Figure 4-14) :

- le scénario 1 présente 3 phases d'acquisition par jour et une phase d'envoi automatique d'informations par mois,
- le scénario 2 présente 3 phases d'acquisition par jour, une phase de réponse à une requête par semaine et une phase d'envoi automatique d'informations par mois,
- le scénario 3 présente 3 phases d'acquisition par jour, une phase de déclenchement d'alarme par semaine et une phase d'envoi automatique d'informations par mois.

Le motif de base de chaque scénario est présenté sur une durée d'un mois.

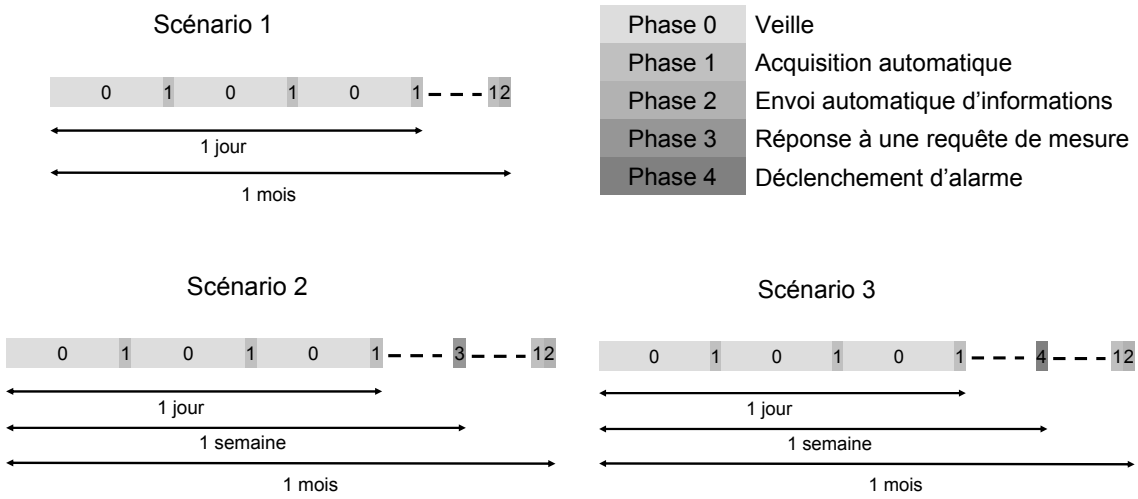


Figure 4-14 : Présentation des 3 scénarios de fonctionnement du système.

Le détail des calculs de la consommation des différentes phases est donné en Annexe 2. Nous précisons que la phase 0 peut être configurée de plusieurs manières. En effet cette phase consiste en une succession d'états de repos et d'écoute. D'une part, l'état de repos pour lequel le microprocesseur est en veille, avec aucun autre composant du système alimenté. D'autre part, l'état d'écoute pour lequel le microprocesseur est réveillé et le module radio-fréquence est en mode de réception, attentif aux signaux radio-fréquences externes. Aussi, nous avons considéré pour les calculs de consommation des phases 1 et 3 un temps minimum d'activité pour chaque capteur.

Le paramètre clef sur lequel nous pouvons jouer pour optimiser la consommation du système, outre la consommation de chaque composant, est le pourcentage du temps pour lequel la phase de veille est consacré à l'écoute. La Figure 4-15 présente l'évolution de la durée de vie du système pour différents pourcentages de temps d'écoute. Nous remarquons qu'en théorie, la durée de vie peut être doublée, dans le cas où le pourcentage de temps d'écoute passerait de 20 à 8%. Pour cette étude de consommation, nous avons fait le choix de consacrer 10% du temps de la phase 0 à l'écoute avec la répartition du temps suivant : 500ms d'écoute et 5s de repos. Le système est donc disponible pour recevoir un message toutes les 5 secondes. L'utilisateur interprétera cela comme le temps de réponse du système de 5 secondes. Toutefois, ce pourcentage reste configurable par l'utilisateur.

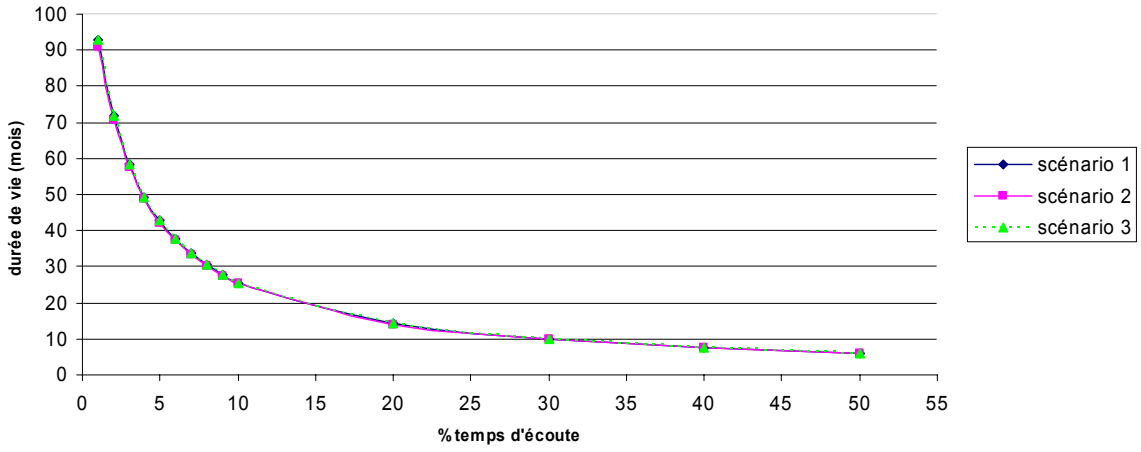


Figure 4-15 : Influence du temps d'écoute RF sur la durée de vie du système.

Après avoir choisi chaque composant sur catalogue avec une attention toute particulière pour leur caractéristique de consommation, nous avons évalué la consommation totale du système et

estimé sa durée de vie, dans le cas où le système dispose d'une alimentation de type pile 2/3 AA (3V, 2100mAh). La capacité totale d'énergie embarquée est alors de 7560mWh.

Les différences de durée de vie observées entre les 3 scénarios sont de quelques jours seulement :

- Scénario 1 : 25 mois et 14 jours.
- Scénario 2 : 25 mois et 9 jours.
- Scénario 3 : 25 mois et 14 jours.

Les 3 scénarios ont une **durée de vie** supérieure à **25 mois**. Cette évaluation de la consommation nous permet de répondre favorablement aux exigences du cahier des charges initial.

Ces estimations théoriques tiennent compte des cinq phases de fonctionnement du système (Annexe 3).

4.5.5 Intégration du microsystème

Nous avons décrit au Chapitre 1, les principales technologies accessibles aujourd'hui pour l'intégration microsystèmes.

(a) Sur le court terme, au stade du premier prototype, nous avons opté pour un assemblage 3D en technologie conventionnelle (report de composants CMS sur un substrat bicouches).
La compacité du système a été maximisée d'une part, en choisissant des composants montés en surface et d'autre part, en utilisant un PCB de faible épaisseur (700µm) de format bicouche pour augmenter la densité des connexions.

La Figure 4-11 présente les 4 niveaux modulaires de notre système.

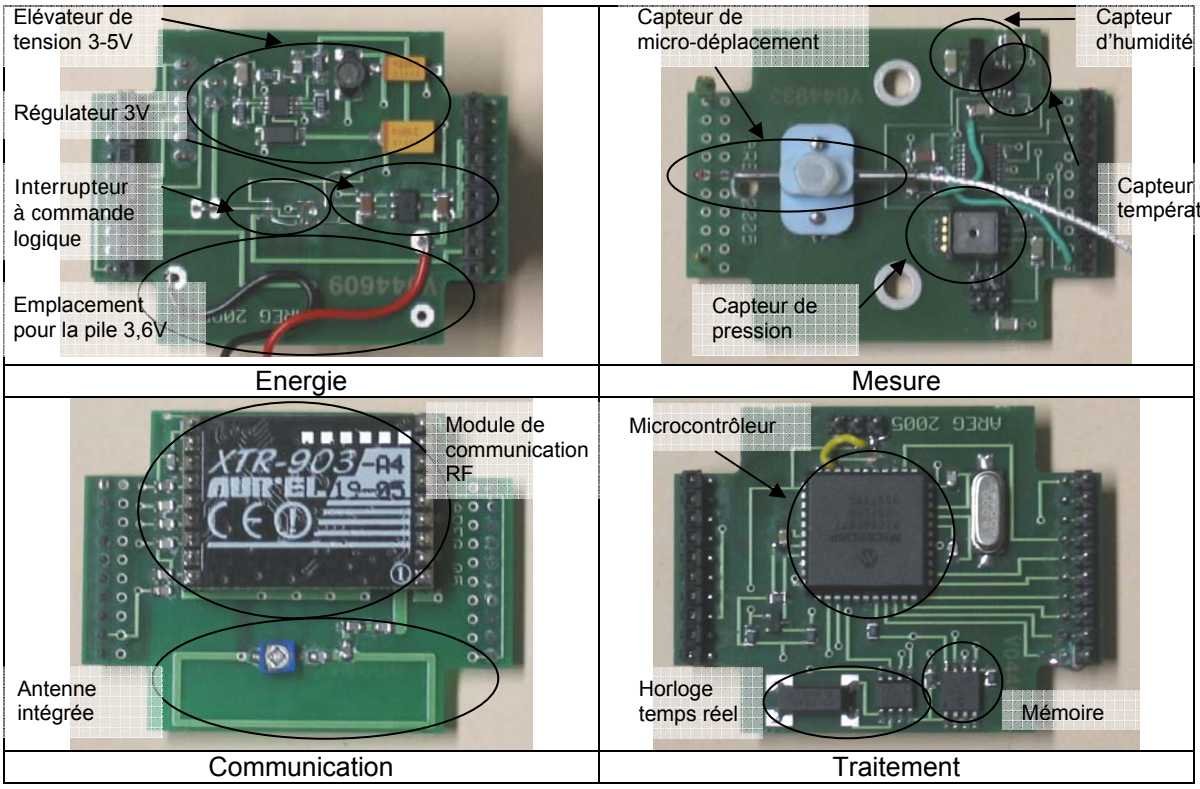


Figure 4-16 : Les quatre niveaux électroniques modulaires de SmartGec.

Les quatre étages du module de mesure assemblé sont illustrés sur la Figure 4-17. Le système SmartGec complet, associe le module de mesure et le module de renvoi d'effort (Figure 4-18).

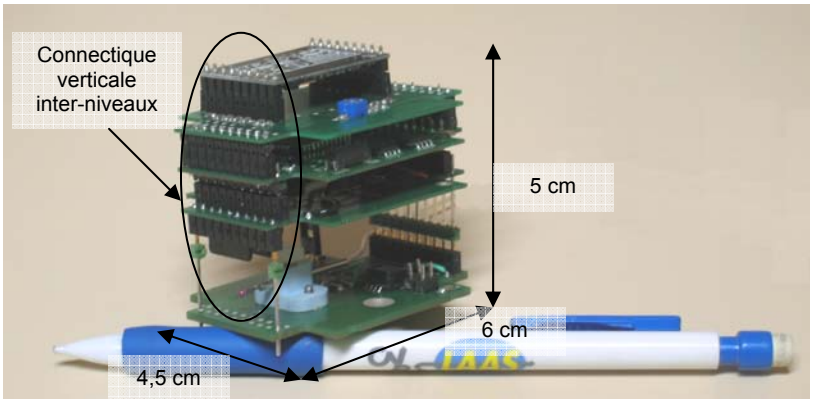


Figure 4-17 : Les quatre étages du module de mesure assemblés.

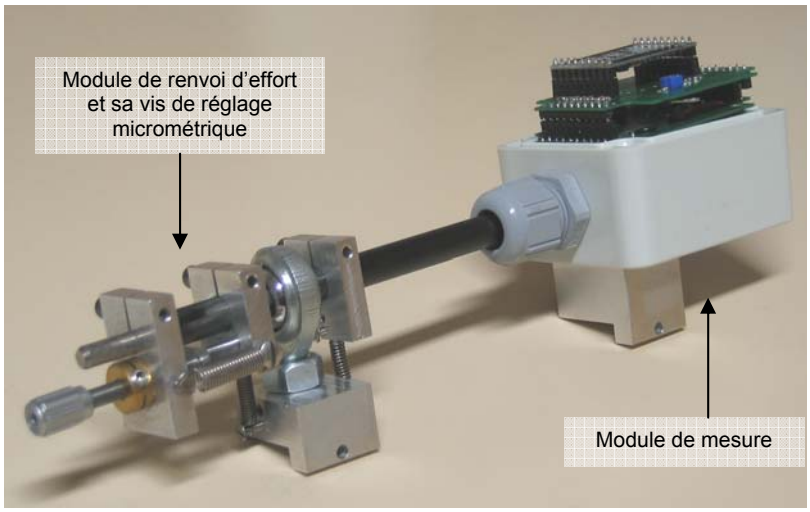


Figure 4-18 : Le système complet est constitué du module de mesure dans son boîtier associé au module de renvoi d'effort.

Le boîtier choisi est étanche à la poussière et à l'aspersion d'eau (IP66). Nous avons du modifier sa structure par deux perçages : l'un permettant de faire passer la tige d'INVAR et sa gaine de carbone, et l'autre pour avoir une ouverture perméable proche des capteurs afin de sonder l'environnement extérieur (Humidité, Pression et Température). La première ouverture est comblée par un presse-étoupe étanche (IP68) et sur la seconde nous avons utilisé un bouchon muni d'une membrane en tissu imperméable à l'eau et respirant (IP68). Ainsi, nous pouvons mesurer les variations des paramètres environnementaux extérieurs et garantir l'étanchéité du système au ruissellement.

La Figure 4-19 illustre l'assemblage du niveau capteur dans le boîtier plastique, avant l'ajout des niveaux supérieurs. Nous visualisons ainsi le point de contact entre le microcapteur de déplacement et l'extrémité de la tige INVAR.

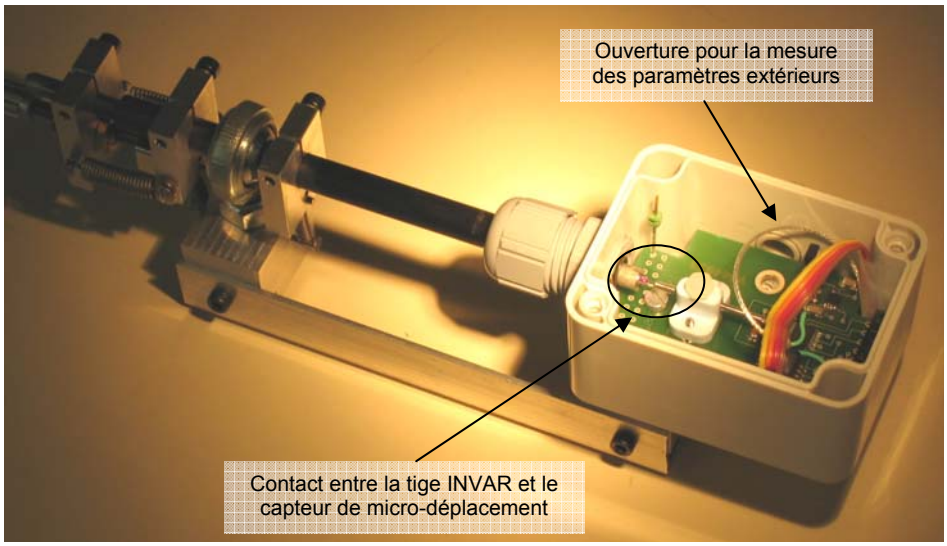


Figure 4-19 : Vue du niveau capteur intégré dans le boîtier.

Ainsi, le système SmartGec est constitué d'un module de 146 cm^3 (boîtier de mesure), et pèse moins de 200 grammes.

(b) Sur le long terme, on peut s'interroger sur l'intérêt d'aller dans le sens d'une plus grande intégration. Un argument important est que l'architecture de base que nous avons conçue est quasi-standard. Si, comme nous l'avons dit, en introduction, les systèmes de surveillance se développent, il y a une place pour rendre générique des pièces des structures de base : un microprocesseur, un système de traitement de l'information, une interface de communication, etc. Dans une telle situation, la création d'un ASIC peut être une bonne option : on y gagnerait poids et encombrement.

A coté de ce module opérationnel et dans le même esprit, il y a une place pour un véritable **gestionnaire de l'énergie** pour tous les microsystèmes conçus pour être utilisés en autonomie, sur de longues périodes.

4.6 Premières mises en œuvre et perspectives

Les premiers tests du système ont été réalisés dans le département Génie Civil de l'INSA de Toulouse, au sein du Laboratoire Matériaux et Durabilité des Constructions. Nous avons effectué des tests d'ouverture de fissure dans le cadre d'un essai de flexion trois points. Le support utilisé est une poutre de béton fibré et préfissurée, d'une section carrée de 10 cm de côté et de 60 cm de long. L'installation est constituée d'une presse qui peut être contrôlée en déplacement ou en force via un ordinateur. La mesure d'ouverture de la fissure se fait par l'intermédiaire d'un capteur LVDT relié à un ordinateur pour l'acquisition du signal.

Le but est de valider l'utilisation de la partie mécanique que nous avons développé et de vérifier ses capacités à mesurer des variations de distance micrométriques entre deux points d'ancrage.

Le positionnement des plots d'ancrage

Nous avons positionné et aligné les plots d'ancrage du système en suivant le protocole utilisé pour des essais classiques d'ouvertures de fissures. Dans un premier temps, la surface du béton est grattée de manière à supprimer la peau pulvérulente qui apparaît après démoulage de la poutre. Ensuite, les plots sont collés en surface, et maintenus alignés par des équerres (Figure 4-20). La colle utilisée est bi-composant : une poudre Plex7742 et un liquide Pleximon801 à base de Polyméthacrylate de Méthyle. Elle permet une prise rapide en quelques dizaines de secondes.

Figure 4-20 : Alignement des plots d'ancrage par des équerres.

L'assemblage par collage présente des avantages indéniables. En effet, il s'agit d'une méthode universelle à faible coût, n'altérant pas les propriétés des substrats, contrairement à certaines techniques d'assemblage (ex : rivetage, boulonnage, soudage).

L'inconvénient majeur de cet assemblage est qu'il soit indémontable. Classiquement, les plots d'ancrage sont arrachés après chaque essai puis nettoyés avant d'être remis en service. Il est donc nécessaire, pour notre système, de prévoir un dispositif pour qu'il soit démontable et repositionnable à souhait. Pour cela, nous utilisons deux plots d'ancrage « consommables » qui peuvent être abandonnés à chaque repositionnement du système ou arrachés sans porter atteinte à l'intégrité mécanique du système. Ces plots constituent la base sur laquelle nous vissons notre système (Figure 4-21).

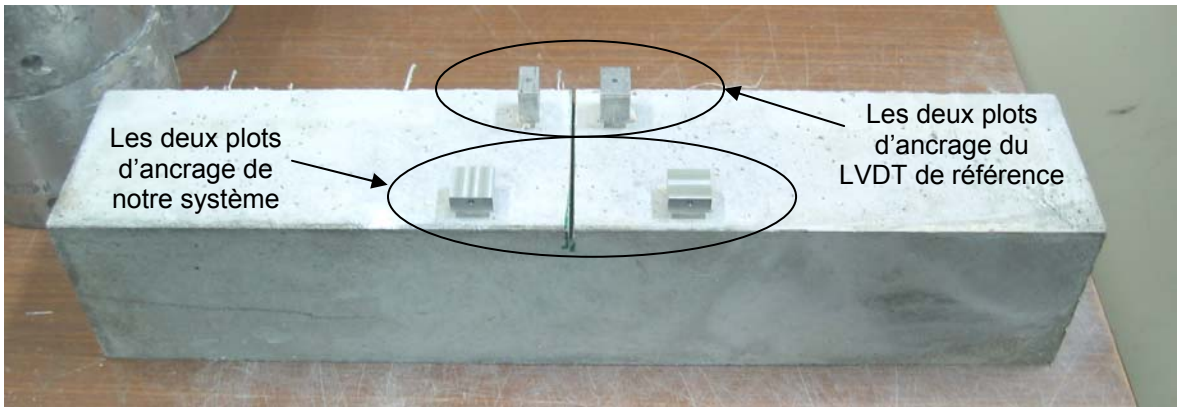


Figure 4-21 : Plots d'ancrage collés.

La Figure 4-22 présente en premier plan notre système vissé sur les plots d'ancrage. A gauche, la vis de réglage micrométrique qui nous permettra de définir le point initial de la mesure, et à droite, le module contenant les capteurs et l'électronique associée. Au second plan se détache le capteur LVDT monté dans ses plots d'ancrage.

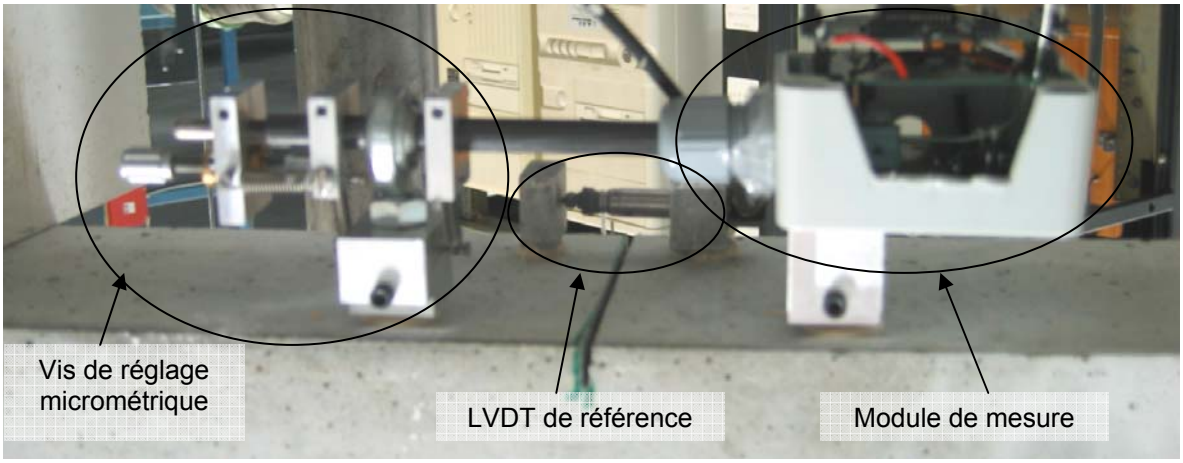


Figure 4-22 : Système et LVDT de référence positionnés sur la poutre.

La poutre d'essai est ensuite positionnée sur deux appuis, la fissure orientée vers le bas, sous la presse uniaxiale (Figure 4-23).

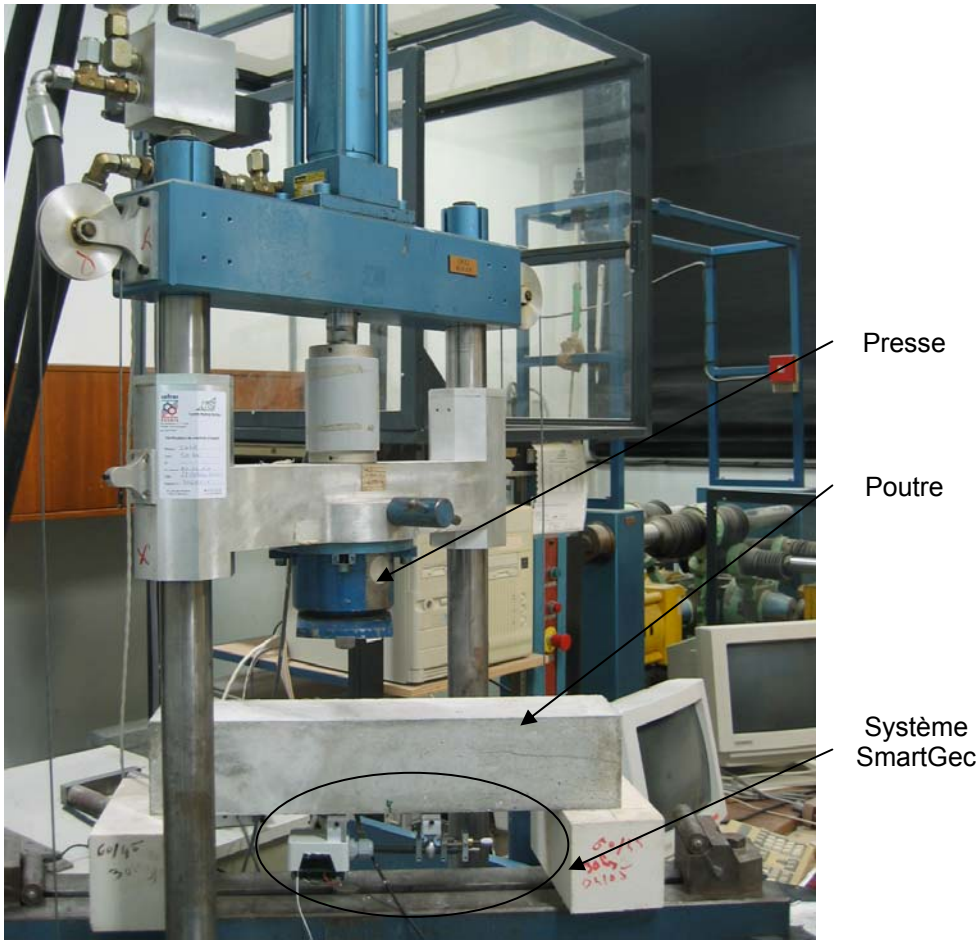


Figure 4-23 : Positionnement de la poutre fissurée sous la presse uniaxiale.

Nous avons réalisé les essais en asservissant le déplacement vertical de la presse à la mesure d'ouverture de la fissure fournie par le signal du capteur LVDT de référence. Un premier test est réalisé sans notre système, mais uniquement avec le capteur de référence, afin de vérifier que la zone élastique du matériau s'étend jusqu'à l'ouverture souhaitée de la fissure.

Une fois notre système monté sur la poutre, la vis micrométrique, située sur le module de renvoi d'effort, nous a permis de régler le point initial de mesure de notre capteur en position de mi-course. Ainsi, nous avons effectué deux séries de mesures pour une ouverture de fissure maximale de 100µm. Chaque série de mesure est réalisée avec un pas d'ouverture de 5µm.

Les résultats obtenus sont illustrés sur la courbe de la Figure 4-24.

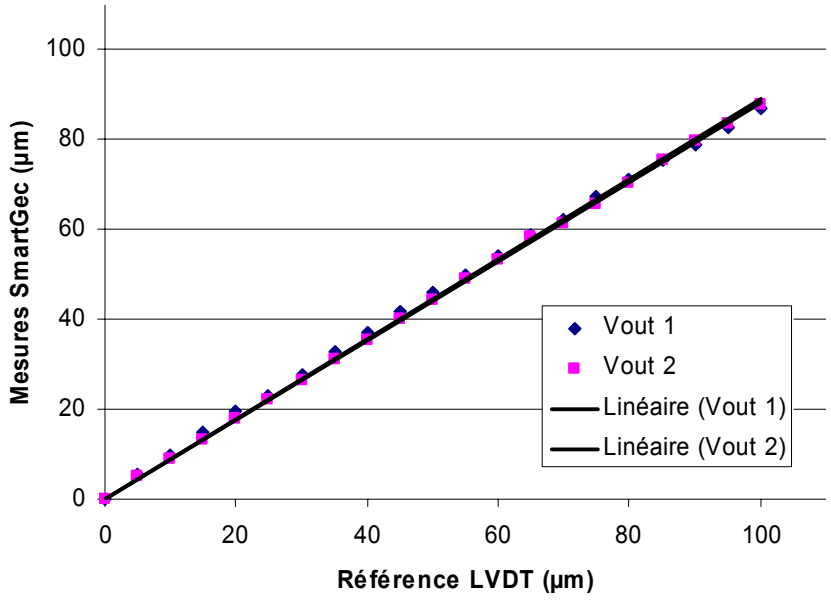


Figure 4-24 : Essais d'ouverture de fissure mesurée par notre système.

Nous observons un comportement parfaitement linéaire pour les deux essais réalisés. Ainsi, nous pouvons en déduire que notre système ne modifie pas le comportement réel du système étudié. Ce résultat nous permet de valider l'utilisation de notre système mécanique de renvoi d'effort.

Nous avons réalisé ces essais de manière à confronter les valeurs de référence aux valeurs mesurées. Pour cela, nous avons utilisé une base de mesure de 10cm entre les deux plots d'ancrage afin de se rapprocher au plus de celle utilisée pour la mesure de référence (5cm). Nous pouvons modifier cette base de mesure afin de répondre à diverses exigences, sans introduire de nouvelles incertitudes sur la mesure. Les modifications consistent à augmenter les dimensions des tiges de carbone et d'INVAR.

4.7 Validation du système par comparaison au cahier des charges

La totalité des exigences du cahier des charges a fait l'objet d'un rapport interne [Mau03b]. Le Tableau 4-2 rappelle celles qui concernent les caractéristiques principales du système.

N° Référence de l'exigence du cahier des charges	Principales caractéristiques attendues	Validation de l'exigence
4.6.1.3.	Précision de la mesure de déplacement : ±5µm	Cette exigence est validée par l'étude faite dans le paragraphe 4.5.3. La précision mesurée est de ±1,4µm.

4.6.3.1.	Puissance RF Max : 50mW	Cette exigence est validée par les données constructeurs qui annoncent une puissance maximale de l'émetteur Radio-Fréquence égale à 10mW.
4.6.3.2.	Distance de communication ≤ 60m	A ce jour, nous avons validé la communication RF en utilisant le module Aurel XTR 903 comme interface de communication entre deux ordinateurs mais nous n'avons pas encore validé la distance de communication maximale de notre module vers un ordinateur distant.
4.9.1.3.	Fiabilité par rapport à l'environnement (T° : 10-40°C ; résistance à la projection d'eau)	Nous avons veillé à choisir des composants ayant un domaine de fonctionnement pour ces domaines de température. Celles ci ne sont pas critiques et tous les composants électroniques sont conçus au regard de ces contraintes de température. La résistance à la projection d'eau est validée par l'utilisation d'un boîtier certifié IP66 (étanche à la poussière et a l'aspersion d'eau)
4.9.2.2.	Masse <250g Vol<150cm^3 Repositionnement aisé	La masse finale du système est inférieure à 200g. Le volume est du module de mesure est de 146cm^3. Le repositionnement est facilité par un assemblage vissé des points d'ancrage sur des plots collés.

Tableau 4-2 : Liste des exigences principales attendues et obtenues pour le système.

Ainsi, nous avons montré que l'assemblage final des fournitures élémentaires et les choix d'intégration contribuent à la validation des caractéristiques principales attendues du système. Ceci à la fois sur le plan fonctionnel (précision, puissance RF max, distance de communication) que sur le plan non fonctionnel (volume, masse, fiabilité par rapport à l'environnement).

4.8 Conclusion

L'intégration vers le prototype réel termine le cycle en « V » de la conception. La réalisation permet l'étape ultime de validation par comparaison des résultats d'essais avec les spécifications du cahier des charges. A partir de la modélisation fonctionnelle des Chapitres 1 et 2, nous avons étudié l'étape de partitionnement du système en modules. Bien que, dans le cas de notre application, la démarche soit simple, nous l'avons décrite pour la bonne compréhension de la démarche générale : cinq modules ont été définis, spécifiés et envoyés en fabrication.

Nous nous sommes attachés à décrire le travail relatif au module électronique confié à la société AREG et à anticiper des solutions technologiques dans le cas d'un développement à grand volume.

Nous avons abouti à un prototype de première génération et l'avons testé dans des conditions de laboratoire.

De ce point de vue, nous arrivons à la fiche technique suivante :

- Gamme de mesure de micro-déplacement : 3mm ; Précision : ±1,4µm.
- Gamme de mesure de température : de –20 à +100°C; Précision : ± 0,5°C.
- Gamme de mesure d'humidité : 0-100%Rh ; Précision : ± 2%Rh.
- Gamme de mesure de pression : 20-115kPa ; Précision : ± 67Pa.
- Puissance radio Max : 10mW.
- Fréquence radio : 433 MHz.
- Domaine de température de fonctionnement : 10-40°C.
- Etanchéité à la poussière et au ruissellement.
- Masse inférieure à 200g.
- Volume de 146cm^3.
- Repositionnement facile par points d'ancrage mécanique.

Par rapport aux objectifs du cahier des charges, cette première évaluation est encourageante.

Nous pouvons envisager de franchir de nouvelles étapes vers un produit industriel. Elles peuvent toucher les aspects d'intégration, d'ajustement et d'optimisation des caractéristiques en réponse à des besoins qui évoluent.

Aussi, nous pouvons mener de nouvelles recherches sur la réalisation d'un ASIC pour les microsystèmes de surveillance intégrant la question essentielle de la gestion d'énergie.

Le prototypage réel s'est nourri des concepts provenant de l'analyse des spécifications du système à haut niveau, faites dans le Chapitre 2. Le découpage fonctionnel a donné une vision structurée au système qui a permis de différencier, définir et représenter séparément chaque élément avant de cibler les fournitures potentielles. Cependant, le pont entre la représentation comportementale et la conception du circuit réel présente un saut important qui doit être diminué afin de bien clarifier les étapes de passage entre le haut niveau de conception et le niveau de la représentation du circuit matériel.

Nous avons traité le cas d'un système pluridisciplinaire à base d'électronique. Ainsi, nous nous sommes rapprochés de la généralité des microsystèmes qui associent un ou plusieurs organes de commande et des éléments actifs : actionneurs, valves, récepteurs, etc. Pour notre application, nous avons scindé le module de mesure en quatre parties fonctionnelles. Si un problème d'ajustement et d'optimisation se pose pour une fonctionnalité de ce module, il faudra tout d'abord cibler la partie fonctionnelle mise en cause et ensuite porter les modifications nécessaires sur cette partie uniquement, les autres parties restant inchangées.

Conclusion Générale

Notre travail se situe dans le cadre de la conception microsystème. Nous l'avons approché sous deux angles : celui des méthodes et des outils de la conception microsystème et celui de leurs applications à la conception puis à la fabrication d'un microsystème spécifié pour la surveillance en Génie Civil. L'originalité de ce travail est d'avoir réalisé ce couplage pour enrichir les méthodes et les outils de conception et pour parfaire la pratique de la conception. L'actualité et la difficulté de ce travail tiennent à son caractère pluridisciplinaire qui est intrinsèque aux microsystèmes. Compte tenu de l'évolution de la demande toujours très dynamique, la disponibilité de méthodes et d'outils performants sont une des clefs du succès permettant de répondre vite et sans faute à un cahier des charges. Nous avons pu noter dans les développements de notre thèse que la problématique microsystèmes n'était pas très éloignée de la problématique plus générale : systèmes.

Sur le plan des méthodes et des outils, notre point de départ a été la plate-forme HiLeS. Dans sa version initiale incomplète, elle était fondée sur le formalisme HiLeS fait de blocs interconnectés et de réseaux de Petri. Nous avons identifié, dans notre travail de mise en œuvre, les lacunes de l'approche dans la liaison entre les spécifications et le formalisme HiLeS : défauts d'analyse fonctionnelle et nécessité d'une description séquentielle dans les différents cas d'utilisation.

Nos premières recommandations ont permis de formuler plus clairement les besoins à cette étape très amont de la conception et d'engager avec nos collègues de l'ingénierie système [EOP05], un travail plus en profondeur sur l'usage qui pourrait être fait des langages UML et SysML. Nous avons aujourd'hui, une procédure UML complètement décrite pour traduire les spécifications du cahier des charges en une représentation HiLeS. Notre collaboration avec EDF, nous a ensuite permis d'associer à cette démarche le concept d'un « traceur de spécifications » permettant d'assurer le concepteur que telle ou telle spécification initiale est bien incluse dans la représentation HiLeS puis les représentations VHDL-AMS. C'est une contribution qui doit encore être précisée pour aboutir à une liaison complète et fiable entre le cahier des charges et la représentation HiLeS.

Une autre contribution générale de notre travail se situe davantage sur les questions de partitionnement et de vérification : HiLeS dispose d'une connexion TINA, outil de vérification mis au point au sein du groupe OLC-LAAS [BRV03]. Notre tâche a été de proposer les modes applicatifs : définition des propriétés de la représentation HiLeS à vérifier, transparence de l'évaluation et appropriation plus aisée par le concepteur des résultats de cette évaluation. Nous avons aussi identifié de nombreuses insuffisances dans l'étape du partitionnement et de son évaluation mais le travail à accomplir pour y remédier sort du cadre de ce mémoire : il s'agit des questions d'optimisation du partitionnement Hard-Soft et de la co-simulation algorithmes-processeurs-système. Notre avis toutefois est que le besoin est tel que des solutions seront rapidement proposées au concepteur.

Sur le plan de la conduite de notre projet applicatif : microsystème de surveillance en Génie Civil, l'originalité est d'avoir appliqué pas à pas la méthode HiLeS sur un exemple précis. Cela a, selon notre analyse, permis :

- L'établissement d'un cahier des charges à partir d'un travail collectif « dirigé ». Nous avons essayé dans notre document d'extraire de cette expérience de nouvelles étapes pour la recherche méthodologique (guide de spécifications, dictionnaire de fonctions) et pour la conduite de projet.

- L'établissement d'une représentation « amont » HiLeS du microsystème que nous avons appliqué à la rédaction des spécifications pour la sous-traitance de fabrication de notre microsystème : c'est un point très intéressant à développer car, avec la représentation HiLeS, on se trouve bien en situation de pouvoir proposer des spécifications formelles au fournisseur.

- La mise en œuvre d'une procédure finale de validation par comparaison du cahier des charges que nous avons élaboré collectivement et du prototype physique qui a découlé de notre travail : ici encore nous avons tenté dans notre mémoire de faire des recommandations simples pour la procédure de validation.

Au delà de ces conclusions générales, nous allons ci-dessous détailler davantage les résultats obtenus.

Dans le chapitre 1, nous avons tout d'abord présenté un état de l'art sur les technologies microsystèmes et sur les pratiques de mesure des contraintes dans les bétons. Ensuite, par une démarche structurée d'analyse fonctionnelle du besoin, nous avons pu élaborer un cahier des charges pour notre système.

Dans le chapitre 2, nous avons montré l'importance d'une conception haut niveau qui s'appuie sur les outils tels que les diagrammes UML/SysML, les réseaux de Pétri et les méthodes de vérification formelles par les réseaux de Pétri, pour la première validation du système globale. Nous avons vu l'intérêt de lier ces représentations par des outils de traçabilité d'exigences afin d'être capable de repérer, au fil du projet l'évolution et l'implémentation de ces exigences dans les différentes représentations du système.

Dans le chapitre 3, nous nous sommes intéressés à la transformation de la représentation HiLeS en une représentation VHDL-AMS qui sert de support aux modélisations pluridisciplinaires. Nous avons proposé d'agréger les fonctions et de partitionner le système HiLeS en modules pour faire apparaître des composants matériels. Ce découpage fait appel à une base de données experte que l'on a des technologies disponibles. Sur la base de ce découpage, les spécifications formelles qui lui sont associées sont regroupées avant de faire appel aux fournisseurs qui apportent la matière d'un véritable « prototype virtuel » du système.

Dans le chapitre 4, nous avons présenté la fin du cycle de conception en « V » qui se termine par la réalisation d'un prototype matériel. Nous avons obtenu un premier dispositif et l'avons testé dans des conditions de laboratoire. Par rapport aux objectifs du cahier des charges, cette première évaluation est encourageante. Elle présente une structure modulaire du système qui permet, si des exigences évoluent, de cibler uniquement les éléments du système à modifier.

Plusieurs perspectives s'ouvrent à la suite de cette étude :

- Il y a un travail de valorisation à faire sur le microsystème que nous avons conçu et réalisé : il correspond à des fonctionnalités utiles qui impliquent toutefois d'approfondir encore la technologie d'intégration. Il n'est pas interdit de penser que l'on puisse aboutir à une conception monolithique multi-capteurs en intégrant la bobine du détecteur sur le silicium.

- Il y a un travail d'approfondissement de la démarche de conception et de construction de la plate-forme. Nous avons identifié quelques pistes et fait quelques recommandations, mais à l'évidence, il s'agit d'un travail très important. Le lancement du grand projet TOPCASED conduit par Airbus sur cette thématique devrait être l'occasion de réaliser ce développement.

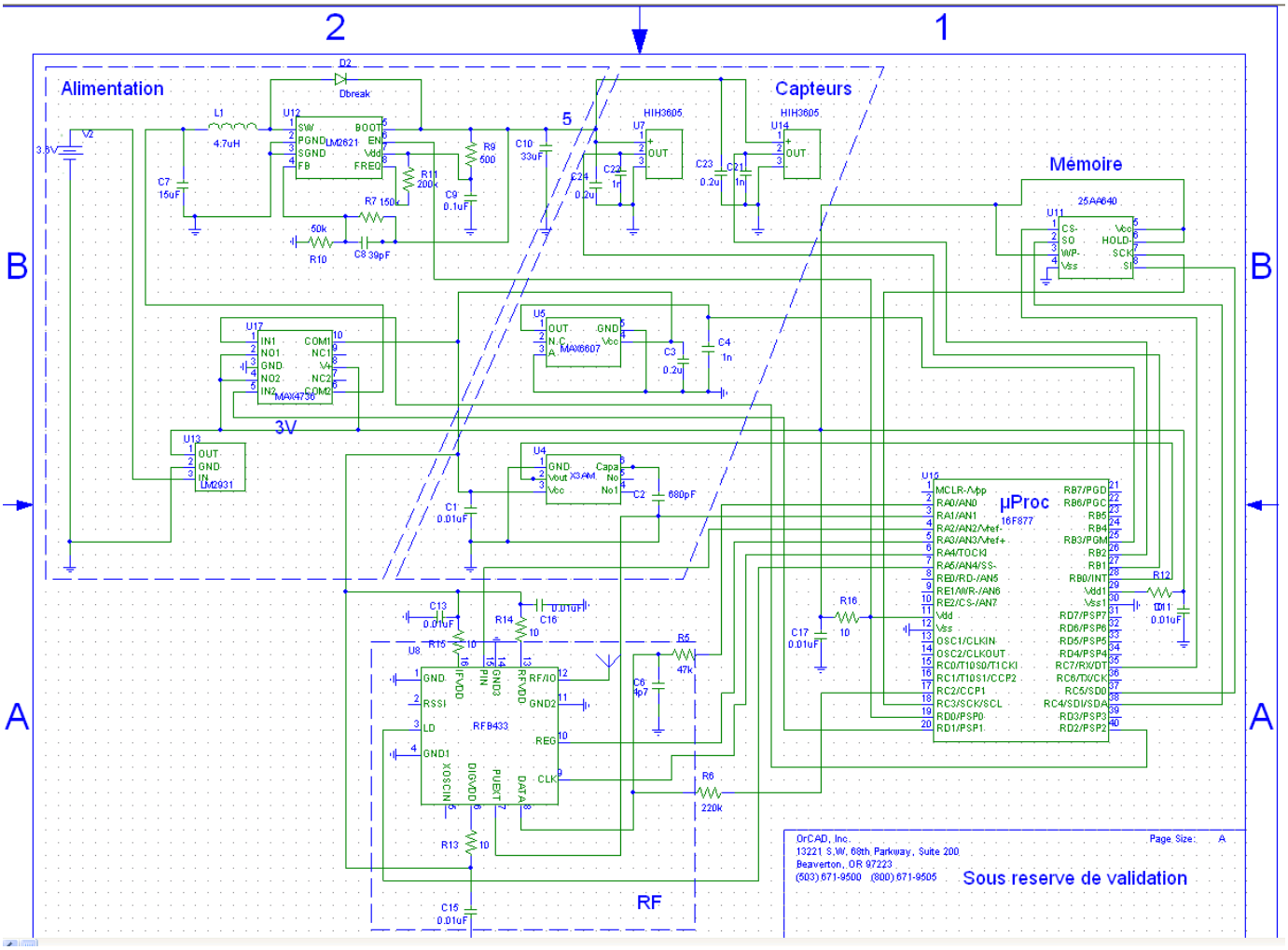
- Il y a une convergence des besoins à souligner sur les capteurs de déplacement intégrés fonctionnant à la précision du micron et les capteurs d'efforts tel qu'on a pu les voir émerger en robotique médicale. Cette convergence vers des solutions capacitives microsystèmes pourrait être également la voie d'une intégration monolithique de notre dispositif.

Annexes

Annexe 1 : Schéma de principe général et les principaux composants du système

Schéma de principe général présentant l'architecture voulue

Ce type de schématique nous a permis de générer le routage des PCB présentés dans le paragraphe 4.5.2.



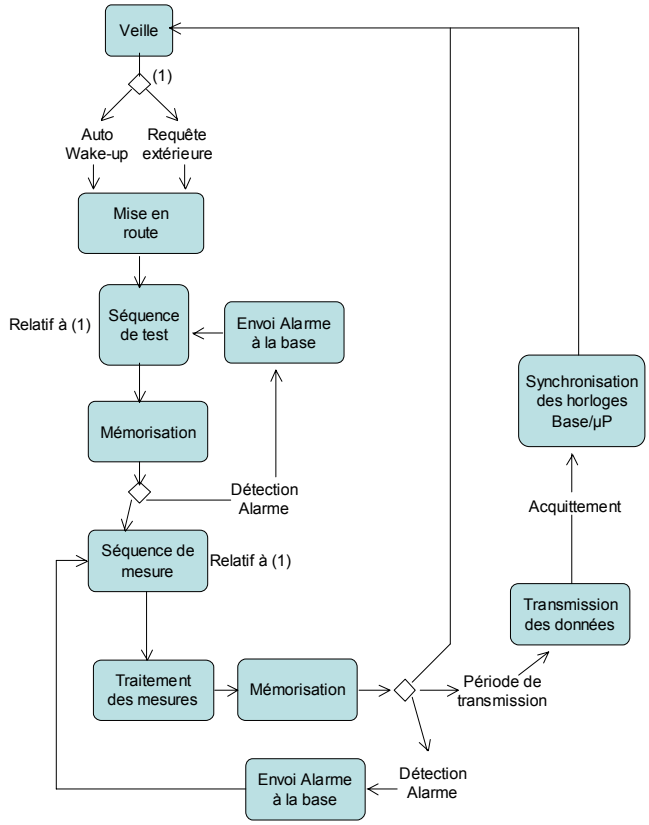
Les principaux composants du système

	Modèle	Caractéristiques	Coût	Taille	Courant Consommé
Partie Analogique Microcapteurs	Microstrain MG-DVRT-3 (Déplacement) 	±1,5µm	1000 € (capteur : 750€ électronique: 250€)	10*40mm	5V ; 8,4mA
	Motorola MPXH6115A (Pression) 	± 2,5kPa	15 €	10*8mm	5V ; 6mA
	HoneyWell HIH Series 3610 (Humidité) 	±2%RH	€ 23	4*19mm	5V ; 200µA
	Maxim Max6607 (Température) 	± 0.6°C entre +20 et 50°C ±0.7°C entre 0 et 70°C	1.5€	2*2.4mm	3V ; 8µA

	Modèle	Caractéristiques	Coût	Taille	Courant Consommé
Partie Analogique Energie	Pile 2/3 AA : Saft LS 17330 	3,6V, 2100 mAh, ,	10 €	33.4mm; Diamètre : 16.6mm	
	Interrupteur à commande logique (Analog Switch) Max4736	Alimentation 1,6 à 3,6V	1 €	3*3mm	1µA
	Régulateur élévateur de tension ajustable National Semiconductor LM2621	Tension d'entrée : de 1,3 à 14 volts Tension de sortie : de 3 à 5 Volts	5 €	3*5mm	110µA 2,5µA (veille)
	Régulateur à faible chute de tension National Semiconductor LM2985	Chute de tension de 0.6V	2 €	3*3mm	1µA

	Modèle	Caractéristiques	Coût	Taille	Courant Consommé
Partie Analogique Communicatuion	Emetteur/ Récepteur RF  Aurel XTR-903-A4	Alimentation : 3 V Taux de transfert 9.6kbps Saut de fréquences Puissance Max 10mW	27 €	25*25*3.4mm	RX: 30mA TX: 40mA Idle : 8µA
Partie Numérique Traitement de l'information	**Modèle**	**Caractéristiques**	**Coût**	**Taille**	**Courant Consommé**
	Microcontrolleur Microchip Pic16F877	3V 8 Kb de Mémoire de programme 5 ports A/N 10Bits interface SPI/I^2C	9 €	12*10 mm	~1mA Veille : 1 µA
	Mémoire Microchip 25AA640 EEPROM	3V Capacité : 64 Kbit interface SPI/I^2C	3 €	5*6*1.5mm	~1mA Veille : 1 µA
	Horloge temps réelle Maxim DS1629	3V	7 €	5*6*1,5 mm	1 µA

Annexe 2 : Algorithme de fonctionnement du système SmartGec



-Système en état de veille.

-Démarrage du système par une horloge interne ou par une requête externe.

-Réalisation d'une séquence de test des capteurs avec analyse et mémorisation des résultats. Possibilité d'émission d'une alarme.

-Réalisation d'une séquence de mesure des 4 capteurs, analyse et mémorisation.
Possibilité d'émission d'une alarme.

-Transmission des données mémorisées vers la centrale de base, à période d'horloge donnée.

-Synchronisation des horloges de la centrale de base et du système SmartGec.

Annexe 3 : Etude de la consommation du système

Les 4 tableaux suivants présentent la **consommation des principaux composants** de chaque partie du système : Mesure, Gestion d'alimentation, Interface de communication Radio-Fréquence et traitement de l'information.

Partie Mesure

composant	Puissance max consommée (mW)	Temps de l'exécution (secondes)	Consommation max (mW.h)	Phases de vie durant laquelle le composant est utilisé
Température : DS1820	8,25	1	2,29E-03	1.1 / 3.3
Pression : MPXH6115A	30	1	8,33E-03	1.1 / 3.3
Humidité : HIH3610	2	15	8,33E-03	1.1 / 3.3
Déplacement : MG-DVRT-3	150,6	10	4,18E-01	1.1 / 3.3

Partie Gestion Alimentation

composant	Puissance max consommée (mW)	Temps de l'exécution (secondes) 8 heures	Consommation max (mW.h)	Phase de vie
Analog Switch	0,003	28800	2,40E-02	veille
Régulateur	0,006	28800	4,80E-02	veille
Total de la veille			1,6800E-01	veille

Partie
Interface RF

composant	Puissance max consommée (mW)	Temps de l'exécution (secondes)	Consommation max (mW.h)	Phase de vie
XTR-903-A4	120	4	1,33E-01	2/3.5/4 **transmission** (19.2 kbit en 1 seconde)
	90	2	5,00E-02	3.1 **reception**
	0,024	2880	1,92E-02	écoute X %

Partie Traitement de l'information

composant	Puissance max consommée (mW)	Temps de l'exécution (secondes)	Consommation max (mW.h)	Phase de vie
µcontrolleur : PIC16F87	3,3	15	1,38E-02	1.2/2/3.4/4 Série de mesures
	3,3	2	1,83E-03	traitement
			1,56E-02	somme (série de mesure + traitement)
	0,012	28800	9,60E-02	veille
	3,3	14400	1,32E+01	écoute X %
Mémoire : 25AA640	5,5	2	3,06E-03	1.3 lecture
	27,5	2	1,53E-02	ecriture

La **durée de vie du système** est calculée par la relation suivante :

Durée de vie = capacité de la source / puissance consommée pendant 1 mois.

Exemple pour l'utilisation de 0.1mA sous 3 Volts pendant 1 mois.
Puissance utilisée = 0.3mWh.
Puissance consommée par mois = 0.3*24*30 = **216 mW.h/mois**
Capacité de la pile Saft LS 17330 : 2100mA.h sous 3,6V = **7560mW.h**

Durée de vie (mois) = 7560/216 = 35 = **2 Ans et 11 mois**.

Les résultats de calcul de la consommation du système sont donnés pour un pourcentage d'écoute de 10%, pendant lequel la radio est active.

temps veille relatif au temps de la phase 0 (s) = 8 heures	X % écoute Radio Fréquence	temps off (s)	temps écoute Radio Fréquence (s)
28800	10	23040	2880

Le tableau suivant présente la consommation totale de chaque scénario et la durée de vie associée.

consommations

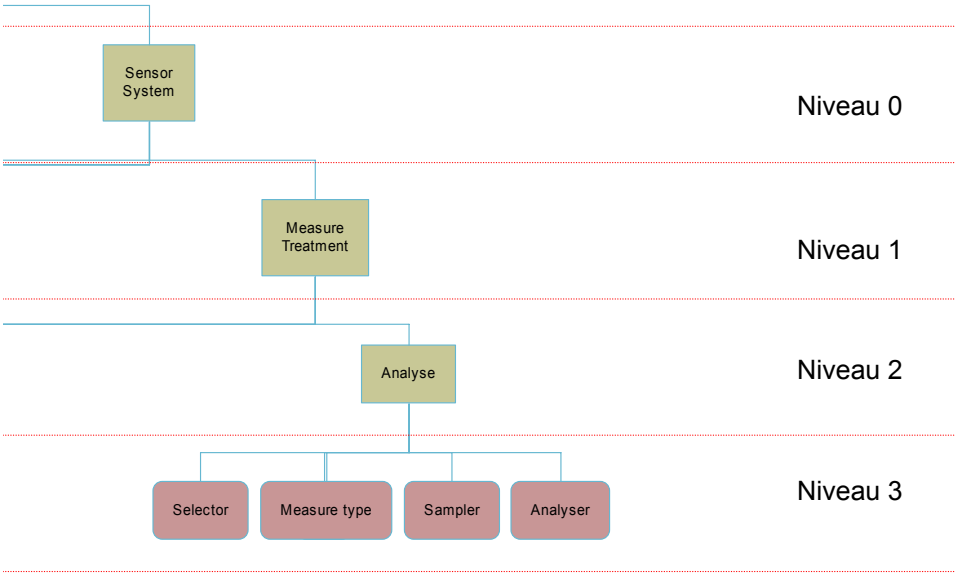
Sur une base mensuelle	phase 0 (mWh/mois)	phase 1 (mWh/mois)	phase 2 (mWh/mois)	phase 3 (mWh/mois)	phase 4 (mWh/mois)	total (mWh/mois)	Durée de vie (mois)
scénario 1 (1 mois de fonctionnement automatique + 1 envoi)	254,45	42,13	0,13			296,72	25,48
scénario 2 (1 mois de fonctionnement automatique + 1 envoi + 1 réponse à requête)	254,45	42,13	0,13	1,91		298,62	25,32
scénario 3 (1 mois de fonctionnement automatique + 1 envoi + 1 alarme)	254,45	42,13	0,13		0,13	296,85	25,47

Les résultats annoncent une durée de vie supérieure à 2 ans pour les 3 scénarios.

Le tableau ci-dessous illustre le détail des consommations relatives à chaque phase de fonctionnement.

Phase de vie	Composant utilisé	Consommation en **mW.h**		
Phase 0 (veille)	µcontrolleur / RF / Switch / Réhausseur / Régulateur	**1,35E+01**		
Phase 1 (acquisition automatique)		**Temps d'éxécution propre à chaque capteur**	**Pendant 15 sec**	**Pendant 60 sec**
1.1	capteur de température	2,29E-03	3,44E-02	1,38E-01
	capteur de pression	8,33E-03	1,25E-01	5,00E-01
	capteur d'humidité	8,33E-03	8,33E-03	3,33E-02
	capteur de déplacement	4,18E-01	6,28E-01	2,51E+00
1.2	µcontrolleur	1,56E-02	1,56E-02	6,23E-02
1.3	mémoire (écriture)	1,53E-02	1,15E-01	4,58E-01
	Total phase 1	**0,468**	**0,925**	**3,702**
Phase 2 (envoi automatique des informations)	interface RF : XTR-903-A4 en emission	**1,33E-01**		
Phase 3 (réponse à une requête de la station de base)		**Temps d'éxécution propre à chaque capteur**	**Pendant 15 sec**	**Pendant 60 sec**
3.1	interface RF en réception	5,00E-02		
3.2/3.4	µcontrolleur	1,56E-02	1,56E-02	6,23E-02
3.3	capteur de température	2,29E-03	3,44E-02	1,38E-01
	capteur de pression	8,33E-03	1,25E-01	5,00E-01
	capteur d'humidité	8,33E-03	8,33E-03	3,33E-02
	capteur de déplacement	4,18E-01	6,28E-01	2,51E+00
3.5	interface RF en émission	1,33E-01	5,00E-01	2,00E+00
	Total phase 3	**6,36E-01**	**1,31E+00**	**5,24E+00**
Phase 4 (déclenchement d'alarme)	interface RFen émission	**1,33E-01**		

Annexe 4 : Codes VHDL-AMS du bloc « Analyse » de « Measure Treatment »



Bloc Measure Treatment

niv_01_Measure_treatment.vhd

```
-------------------------------------------------------------------
-- Nom : niv_01_Measure_treatment.vhd
-- Version : 1.0
-- Auteur : R. Maurice
-- Date : 27/01/05
-- Commentaires :   Unité de traitement de signaux
-- Testé dans le projet MT_compo
-- Historique :
-------------------------------------------------------------------

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.electrical_systems.all;
--USE work.all;
--use work.variable_pack.all;
LIBRARY lib_edf;
use lib_edf.all;
use lib_edf.variable_pack.all;

ENTITY MeasureTreatment1 IS
GENERIC (FACTEURS_INFLUENCES : REAL:=25.0);
             --n : integer :=0);

PORT    (--SIGNAL init : in std_ulogic;
          TERMINAL Temp, Hum, Press, Mov : electrical; --in
         TERMINAL V_in : ELECTRICAL;
         SIGNAL Test_type : in std_ulogic_VECTOR (n downto 0);
         SIGNAL Test_OFF, ON_in, Wake_up : in std_ulogic;
         SIGNAL Start_test_X, Start_measureX  : in std_ulogic_VECTOR (n downto 0);
         SIGNAL Ack_Sent_data : in std_ulogic;
         SIGNAL Start_Power, Sleep : out std_ulogic;
         SIGNAL Alarm_test_meas : out std_ulogic_VECTOR (n downto 0);
         SIGNAL Datas : out std_ulogic_VECTOR (nbits-1 downto 0);
         SIGNAL Start_test : out std_ulogic;
```

```
        SiGNAL Test_assesment : out std_ulogic_VECTOR (n downto 0));

END;

ARCHITECTURE Struct OF MeasureTreatment1 IS

             SIGNAL Clock, Auto_wake_up, Ack_mem_datas_X : std_ulogic;
             SIGNAL MeasX_Database, TestX_Database : std_ulogic_VECTOR (ndata-1
downto 0);
             SIGNAL Idle, End_Meas : std_ulogic;
             SIGNAL X_Datas : std_ulogic_VECTOR (nbits-1 downto 0);
             SIGNAL Start_memo : std_ulogic;
             SIGNAL X_Meas_type :std_ulogic_VECTOR (n downto 0);
             SiGNAL Date_time : std_ulogic_VECTOR (Date-1 downto 0);
             SIGNAL Download,End_Process : std_ulogic;

             quantity Temp_V across Temp to electrical_ref;
             quantity Hum_V across Hum to electrical_ref;
             quantity Press_V across Press to electrical_ref;
             quantity Mov_V across Mov to electrical_ref;

      BEGIN
             Time         :      ENTITY lib_edf.Timer_rdp_1(struct)      PORT MAP
(V_in, Ack_sent_data, Auto_wake_up, Clock,  Download, Date_time);
             Mem          :      ENTITY lib_edf.Memory_rdp (petri)       PORT MAP
(V_in, Download, Date_time, Start_test_X, Start_measureX, Test_type, End_Process,
X_Datas, Start_memo, X_meas_type, Datas, Ack_mem_datas_X, MeasX_Database,
TestX_Database);
             Initial      :      ENTITY lib_edf.Initialise   (petri)     PORT MAP
(End_Process, Auto_wake_up, Download, Ack_Sent_data, ON_in, Test_OFF, Wake_up,
Start_Power, Sleep, Start_test, Idle);
             Analys       :      ENTITY lib_edf.Analyse_rdp      (petri)  PORT MAP
(Temp, Hum, Press, Mov, Clock, Auto_wake_up, Ack_mem_datas_X, MeasX_Database,
TestX_Database, Idle, Test_type, Start_measureX, End_Meas, X_Datas, Start_memo,
X_Meas_type, Alarm_test_meas, Test_assesment, End_Process);

END Struct;
```

niv_02_MT_Analyse_rdp.vhd

```
-----------------------------------------------------------------
-- Nom : niv_02_MT_Analyse_rdp.vhd
-- Version : 1.0
-- Auteur : R. Maurice
-- Date : 27/01/05
-- Commentaires :   Réseau de Petri de l'unité de traitement de signaux
--                  Testé dans le projet MT_compo
--                  architecture multiple : Struct (function “case”)
--                  et Petri(méthode CONPAR)
-- Historique :
-----------------------------------------------------------------

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.electrical_systems.all;

--USE work.all;
--use work.variable_pack.all;

LIBRARY lib_edf;
use lib_edf.all;
use lib_edf.variable_pack.all;


ENTITY Analyse_rdp IS
GENERIC (FACTEURS_INFLUENCES : REAL:=25.0);
                    --n : integer :=0);      --n=0 : test prélmiminaires

PORT   (     Terminal Temp, Hum, Press, Mov : electrical; --in
             Signal Clock, Auto_wake_up : in std_logic;
             SIGNAL Ack_mem_data_X : in std_ulogic;
             SIGNAL MeasureX_Database, TestX_Database : in std_ulogic_vector
(ndata-1 downto 0);
             Signal Idle : in std_ulogic;
```

```
              SIgnal Test_type, Start_measureX : in std_ulogic_vector (n downto
0);
              SIGNAL End_Meas : out std_ulogic ;
              Signal X_Datas : out std_ulogic_vector (nbits-1 downto 0);
              Signal Start_memo : out std_ulogic ;
              Signal X_Meas_type : inout std_ulogic_vector (n downto 0);
              signal Alarm_test_meas, Test_assesment : out std_ulogic_vector (n
downto 0);
              SIGNAL End_Process : out std_ulogic:='0');
END Analyse_rdp;



--------------------------------------------------------------------------------
-- Structural Architecture
--------------------------------------------------------------------------------


ARCHITECTURE struct OF Analyse_rdp IS
        SIGNAL place : integer := 1;
              SIGNAL X_Meas_D : std_ulogic_vector (nbits-1 downto 0);
              Signal X_Meas_type_select : std_ulogic VECTOR (n downto 0);
              SIGNAL Start_meas_select, Idle_samp, Sampling_select,
Restart_sampling_select, End_sampling_select : std_ulogic;
              SIGNAL init : std_ulogic :='1';
              quantity Temp_V across Temp to electrical_ref;
              quantity Hum_V across Hum to electrical_ref;
              quantity Press_V across Press to electrical_ref;
              quantity Mov_V across Mov to electrical_ref;
              terminal X_Signal : electrical;

       BEGIN
              X_Meas_type <= X_Meas_type_select;
              Analyser_lbl        :      ENTITY lib_edf.Analyser    (struct)
PORT MAP (MeasureX_Database, TestX_Database, X_Meas_D, Start_memo, X_Datas,
Test_assesment, Alarm_test_meas);
              Measure_type_lbl    :      ENTITY lib_edf.Measure_type2     (struct)
        PORT MAP (Test_type, Start_measureX, Auto_wake_up, Idle_samp, Idle,
End_Meas, X_Meas_type_select, Start_meas_select,End_Process);
              Selector_lbl        :      ENTITY lib_edf.Selector_analyser (struct)
        PORT MAP (X_Meas_type_select, Temp, Hum, Press, Mov, X_Signal);
              Sampler_lbl         :      ENTITY lib_edf.Sampler3          (struct)
        PORT MAP (X_Signal, Clock, Sampling_select, X_Meas_type,
Restart_sampling_select, End_sampling_select, X_Meas_D);

       --description des réseaux de Petri par la méthode AFSM de Yannick Hervé.

       Process  -- déclaration de la structure des places et des transitions
(comportement asynchrone)
       Begin
              --init<=init_val;
              Wait on place, start_meas_select ,Restart_sampling_select
,End_sampling_select, Ack_mem_data_X;
              Case place is
                     When 1 =>
                            Idle_samp <= Idle;
                            if start_meas_select = ('1')  then place <= 2; end if;
                     When 2 =>
                            Sampling_select <= '1';
                            wait for 10ms;
                            Sampling_select <='0';
                            if Restart_sampling_select = ('1') then place <= 2;
                            elsif End_sampling_select = ('1') and Ack_mem_data_X =
('1') then place <= 1; end if;
                     When others => End Case;
       End process;


       END struct;



--------------------------------------------------------------------------------
-- Petri Net Architecture (Conpar Method)
--------------------------------------------------------------------------------
```

```
          --RdP modifié par rapport à la note EDF HP-18/04/001/A   (suppression
de la boucle récursive (t2 / restart_sampling)
          -- intégration de cette boucle au bloc Sampler
ARCHITECTURE petri OF Analyse rdp IS
             SIGNAL X_Meas_D : std_ulogic_vector (nbits-1 downto 0) :="00000000";

             SIGNAL X Meas type select : std ulogic VECTOR (n downto 0):="00";
             SIGNAL Start_meas_select, Idle_samp, Sampling_select,
Restart_sampling_select, End_sampling_select: std_ulogic :='0';
             terminal X_Signal : electrical;


             --Petri signals
             SIGNAL Np1 : std_ulogic:='0';
             SIGNAL p2, Np2 : std_ulogic:='0';
             SIGNAL p3, Np3 : std_ulogic:='0';
             -- SIGNAL p3a, Np3a : std_ulogic:='0';
             SIGNAL p4, Np4 : std_ulogic:='0';
             --initialisation signals
             SIGNAL p1,pi1,pi2  : std_ulogic:='1';
             SIGNAL Npi1, Npi2 : std_ulogic:='0';
             -- Transition signals
             SIGNAL ti1, ti2, t1, t3, t4 : std_ulogic:='0';--t3a,t2,
             SIGNAL init : std_ulogic:='0';

      BEGIN
             X_Meas_type <= X_Meas_type_select;

             Analyser_lbl :      ENTITY lib_edf.Analyser   (struct)      PORT MAP
(MeasureX_Database, TestX_Database, X_Meas_D, Start_memo, X_Datas,
Test_assesment, Alarm_test_meas);
             Measure_type_lbl    :      ENTITY lib_edf.Measure_type2     (struct)
       PORT MAP (Test_type, Start_measureX, Auto_wake_up, Idle_samp, Idle,
End_Meas, X_Meas_type_select, Start_meas_select, End_Process);
             Selector_lbl        :      ENTITY lib_edf.Selector_analyser (struct)
      PORT MAP (X Meas type select, Temp, Hum, Press, Mov, X Signal);
             Sampler_lbl         :      ENTITY lib_edf.Sampler3          (struct)
       PORT MAP (X_Signal, Clock, Sampling_select, X_Meas_type,
Restart_sampling_select, End_sampling_select, X_Meas_D);


process begin
             wait on Start_meas_select,clock , idle, Restart_sampling_select,
End_sampling_select, Ack_mem_data_X; -- ajout du signal init
             if init ='1' then
                    --pi1 <= '1';
                    pi2 <= '1';
                    p1 <= '0'; -- Idle
                    p2 <= '0'; -- Start Meas
                    p3 <= '0'; -- Sampling
                    --p3a <= '0'; -- artefact nécessaire pour vider la place p3
et ainsi provoquer un changement de signal sur Sampling_select
                    p4 <= '0'; -- Sampling_ended
                    wait for 0 ns;
                    init<='0';
             else
                    --pi1 <= Npi1;
                    pi2 <= Npi2;
                    p1 <= Np1; -- Nextplace marking
                    p2 <= Np2;
                    p3 <= Np3;
                    --p3a <= Np3a;
                    p4 <= Np4;
                    end if;
      end process;


      ti2 <= pi2 and Start_meas_select and not p2;
      -- descritpion des transitions
      t1 <= p1 and p2 and not p3;-- correspond aux critères à valider pour
sensibiliser la transition
      t3 <= p3 and End_sampling_select and not p4;
      t4 <= p4 and Ack_mem_data_X;-- and not pi1;
      -- description du flux de données pour le marquage des place : pas
d'intervention des signaux externes
      Npi2 <= t4 or (pi2 and not ti2);
```

```
      Np1 <= t4 or (p1 and not t1); -- or ti1 or correspond aux critères à
valider pour sensibiliser la place
      Np2 <= ti2 or (p2 and not t1);
      Np3 <= t1 or (p3 and not t3);
      Np4 <= t3 or (p4 and not t4);
      -- signaux de sortie
      Idle_samp <= t4;--pi1;
      Sampling_select <= t1;--p3; -- pb de timing si on utilise t1

      -- transitions en conflit
      ASSERT NOT (t1='1' and t4='1')--and t2='1'
            report "Petri Net may be deadlocked"
            Severity error;
      ASSERT NOT (t1='1' and t3='1')
            report "Petri Net may have a sampling error"
            Severity error;


END petri;


  --ASSERT NOT (Idle='1' and ti2='0')
      --     report "Le signal d'initialisation n'est pas pris en compte"
      --     Severity warning;
      --ASSERT NOT (p4='1' and pi1='1')
      --     report "Conflit entre t4 et pi1"
      --     Severity warning;
      --ASSERT NOT (pi1='1' and p1='1')
      --     report "Pi1 n'est pas consommé lors de la transition ti1"
      --     Severity warning;
```

Bloc structurel Analyse (dans le bloc Measure Treatement)

niv_03_mt_analyse_selector.vhd

```
-------------------------------------------------------------
-- Nom : niv_03_MT_Analyse_selector.vhd
-- Version : 1.0
-- Auteur : R. Maurice
-- Date : 27/01/05
-- Commentaires :   Selecteur de signaux analogiques en fonction d'un signal
numérique (vecteur de bit).
-- Testé dans le projet MT_compo
-- Historique :
------------------------------------------------------------- --

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.electrical_systems.all;
--USE work.all;
--use work.variable_pack.all;
LIBRARY lib_edf;
use lib_edf.variable_pack.all;


ENTITY Selector_analyser IS
GENERIC (FACTEURS_INFLUENCES : REAL:=25.0);


PORT   (     SGNAL X_Meas_type : in std_ulogic_vector (n downto 0);
             --QUANTITY Temp, Hum, Press, Mov : in voltage;
             --QUANTITY X_Signal : out voltage);
             Terminal Temp, Hum, Press, Mov, X_Signal : electrical);
             --Terminal X_Signal : electrical);
END;


ARCHITECTURE Struct OF Selector analyser IS
                quantity Temp_V across Temp to electrical_ref;
                quantity Hum_V across  Hum to electrical_ref;
                quantity Press_V across  Press to electrical_ref;
                quantity Mov_V across  Mov to electrical_ref;
                quantity X_V_Signal across i5 through X_Signal to electrical ref;
```

```
      BEGIN

      break on X_Meas_type;
      if X_Meas_type = ('0','0') use
      X_V_Signal==Temp_V;


      elsif X_Meas_type = ('0','1') use
      Hum_V==X_V_Signal;

      elsif X_Meas_type = ('1','0') use
      X_V_Signal==Press_V;

      elsif X_Meas_type = ('1','1') use
      X_V_Signal==Mov_V;

      else
      X_V_Signal==0.0; --electrical_ref

      end use;

      END Struct;
```

niv_03_mt_analyse_meastype.vhd

```
-------------------------------------------------------------
-- Nom : niv_03_MT_Analyse_measuretype.vhd
-- Version : 1.0
-- Auteur : R. Maurice
-- Date : 27/01/05
-- Commentaires :   Routines de séquencement des mesures.
-- Testé dans le projet MT_compo
-- Historique :
------------------------------------------------------------- --

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.electrical_systems.all;
--USE work.all;
--use work.variable_pack.all;
LIBRARY lib_edf;
use lib_edf.variable_pack.all;


ENTITY Measure_type2 IS
GENERIC (FACTEURS_INFLUENCES : REAL:=25.0);


PORT   (     SIGNAL Test_type, Start_measureX : in std_ulogic_vector (n downto
0);
             SIGNAL Auto_wake_up, Idle_samp, Idle : in std_ulogic;
             SIGNAL End_Meas : out std_ulogic :='0';
             SIGNAL X_Meas_type : out std_ulogic_VECTOR (n downto 0):="UU";
             SIGNAL Start_meas_select : out std_ulogic:='0';
             SIGNAL End_Process : out std_ulogic:='0');
END;


ARCHITECTURE Struct OF Measure_type2 IS
      signal routine : integer := 0;
      signal routine_s : bit :='0';

BEGIN

  -------------------------------------------------------------
  -- Process de selection de routine
  -------------------------------------------------------------
      process
            begin
                  wait on Start_measureX, Test_type, routine_s, Auto_wake_up;
                  if Test_type'event and Test_type="UU" then routine <= 1 ;
                  --Auto_wake_up = ('1')
                  elsif Test_type'event and Test_type/="UU" then routine <= 2;
                  --  'event valeurs ('0','1') prises pour test
                  elsif Start_measureX/="UU" then routine <= 3;
                  -- valeurs ('1','0') prises pour test
```

```
                  elsif routine_s'event then wait for 1ms;
                  -- introduction d'un delais necessaire pour réaliser 2
            process recursifs
                  routine <= 0;
                  else end if;
      end process;

  ------------------------------------------------------------
  -- Process de séquencement des mesures pour les différentes routines
  ------------------------------------------------------------


      Process
            variable i : integer := 1;
            begin
            wait on routine;    --auto_wake_up,  Idle_samp,

            If routine = 1 then
            -- cas d'une succession de mesure sur les 4 capteurs
             --    wait on Idle;
            for i in 1 to 4 loop
                  if     i = 1 then X_Meas_type <= ('0','0');
                  -- mesure de température
                        Wait for 100us;--Délais d'établissment du signal à
mesurer (t, h, p, d) et l'echantillonnage
                        Start_meas_select <= ('1');
                        Wait for 100us;
                        Start_meas_select <= ('0');
                        Wait on Idle_samp;
                        --Wait for 20ms; --si l'on utilise le signal de
controle idle_sample, les mesures ne se declenchent pas
                  elsif i = 2 then X_Meas_type <= ('0','1');    -- mesure
d'humidité 01
                        Wait for 100us;--Délais d'établissment du signal à
mesurer (t, h, p, d) et l'echantillonnage
                        Start_meas_select <= ('1');
                        Wait for 100us;
                        Start_meas_select <= ('0');
                        Wait on Idle_samp;
                  elsif i = 3 then X_Meas_type <= ('1','0');    -- mesure de
pression 10
                        Wait for 100us;--Délais d'établissment du signal à
mesurer (t, h, p, d) et l'echantillonnage
                        Start_meas_select <= ('1');
                        Wait for 100us;
                        Start_meas_select <= ('0');
                        Wait on Idle_samp;
                  elsif i = 4 then X_Meas_type <= ('1','1');    -- mesure de
déplacement 11
                        Wait for 100us;--Délais d'établissment du signal à
mesurer (t, h, p, d) et l'echantillonnage
                        Start_meas_select <= ('1');
                        Wait for 100us;
                        Start_meas_select <= ('0');
                        Wait on Idle_samp;
                  End_Process <=      ('1');
                  Wait for 100us;
                  End_Process <=      ('0');
                  end if;
                  End_Meas <=  ('1');
                  Wait for 100us;
                  End_Meas <=  ('0');
            end loop;


            elsif routine = 2 then -- la déclaration du type de signal est
contenue dans test_type
      -- cas d'un test sur un capteur
                  --wait on Idle;
                  --wait on Test_type;
                  X_Meas_type <= Test_type;
                  Wait for 100us;
                  Start_meas_select <= ('1');
                  Wait for 100us;
                  Start_meas_select <= ('0');
                  Wait on idle_samp;
                  End_Meas <=  ('1');
```

```
                    Wait for 100us;
                    End_Meas <=  ('0');
                    End_Process <=      ('1');
                    Wait for 100us;
                    End_Process <=      ('0');

              elsif routine = 3 then
       -- cas d'une demande explicite d'une mesure sur un capteur X
                    --wait on Idle;
                    X_Meas_type <= Start_measureX;
                    Wait for t_logic;
                    Start_meas_select <= ('1');
                    Wait for t_logic;
                    Start_meas_select <= ('0');
                    Wait until idle_samp = ('1');
                    End_Meas <=  ('1');
                    Wait for t_logic;
                    End_Meas <=  ('0');
                    End_Process <=      ('1');
                    Wait for 100us;
                    End_Process <=      ('0');

              elsif routine = 0 then
       -- état d'initialisation
                    Start_meas_select <= ('0');
                    X_Meas_type <= ('X','X');-- necessité de 5 types
temp,hum,press,mov et "idle" ('X','X') pour measure type
                    End_meas <= ('0'); --electrical_ref
                    End_Process <=      ('0');
              else
                    Wait for 1 ms;
              end if;

              X_Meas_type <= ('0','X');-- reinitialisation des signaux de sortie
              Start_meas_select <= ('0');
              End_Meas <=  ('0');
              End_Process <=      ('0');
              routine_s <= not routine_s;
       end process;

END Struct;
```

niv_03_mt_analyse_sampler3.vhd

```
-------------------------------------------------------------------
-- Nom : niv_03_MT_Analyse_sampler3.vhd
-- Version : 1.0
-- Auteur : R. Maurice
-- Date : 27/01/05
-- Commentaires :   Convertisseur Analogique Numérique avec une gamme
d'echantillonnage variable.
-- Testé dans le projet MT_compo
-- Historique :
------------------------------------------------------------------- --

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.electrical_systems.all;
--USE work.all;
--use work.variable_pack.all;
LIBRARY lib_edf;
use lib_edf.variable_pack.all;

ENTITY Sampler3 IS
GENERIC (FACTEURS_INFLUENCES : REAL:=25.0);

PORT   (     terminal X_Signal1 : electrical;
             -- signal analogique provenant d'un des 4 capteurs
             Signal Clock : in std_logic;
             SIGNAL Sampling_select : in std_ulogic;
             SIGNAL X_Meas_type : in std_ulogic_VECTOR (n downto 0);
             SIGNAL Restart_sampling_select, End_sampling_select : out
std_ulogic:='0';--restart_sampling_select n'est plus utilisé
             signal X_Meas_D : out std_ulogic_vector (nbits-1 downto
0):="11110000");

END;
```

```
ARCHITECTURE Struct OF Sampler3 IS

      quantity X_Signal1_V across X_Signal1 to electrical_ref;--X_Siganl1_I
through
      signal max_val : real := 3.0; -- tension analogique maximum en sortie de
      capteur

BEGIN
  -------------------------------------------------------------
  -- Process de selection de la tension max d'échantillonnage
  -------------------------------------------------------------
      process
      begin
            wait on Sampling_select;
            if X_Meas_type =("00") or X_Meas_type =("10") then
                  -- cas d'une mesure de temp ou press
            max_val <= 3.0;
            elsif X_Meas_type =("01") or X_Meas_type =("11") then
                  -- cas d'une mesure d'hum ou dis
            max_val <= 5.0;
            else end if;
      end process;

  -------------------------------------------------------------
  -- Process de conversion A/N
  -------------------------------------------------------------
      process
            constant num_levels : real := 2.0**nbits;
            -- expression correcte : 2**(nbits-1)
            variable X_D :  std_ulogic_vector (nbits-1 downto 0);
            variable status,x : integer :=1; -- 1=input state ; 2=convert state

            variable sample :integer;
      begin
            wait on Sampling_select until Sampling_select='1';
            --attente du signal Sampling_select pour commencer le sample

            for i in n_sample downto 1 loop
            -- n_sample :nombre d'échantillonnage par signaux le nombre dépend
            du temp de présence du signal mesuré, ici le timing fait que 4
            echantillons représente une valeur adéquate

            wait on Clock until Clock ='1';
            sample := integer(X_Signal1_V*num_levels/max_val);
                  for i in (nbits-1) downto 0 loop
                        -- convertisseur analogique/numérique sur
                        if integer(sample/(2**i)) >=1 then
                        X_D(i):= '1';
                        sample:=sample-(2**i);
                        else X_D (i):= '0';
                        end if;
                        wait for 5us;
                  end loop;
                  X_Meas_D<=X_D;
                  wait for 50us; -- temps de présence du signal
            for i in (nbits-1) downto 0 loop -- réinitialisation des sorties
                  X_Meas_D(i)<='X';
            end loop;
            end loop;
                  wait for 10us;
                  End_sampling_select<=('1');
                  wait for 100us; --temps de reaction
                  End_sampling_select<=('0');
      end process;

END Struct;
```

niv_03_mt_analyse_analyser_2304.vhd 06/12/04

```
-------------------------------------------------------------
-- Nom : niv_03_MT_Analyse_analyser.vhd
-- Version : 1.0
-- Auteur : R. Maurice
-- Date : 27/01/05
-- Commentaires :   Détection d'alarmes et calcul de la moyenne sur un
échantillon de n_sample (variable_pack).
```

```
--                                Testé dans le projet MT_compo
-- Historique :
------------------------------------------------------------------- --

LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
USE ieee.electrical_systems.all;
LIBRARY arithmetic;
USE arithmetic.std_logic_arith.all;
--USE work.all;
--use work.variable_pack.all;
LIBRARY lib_edf;
use lib_edf.variable_pack.all;


ENTITY Analyser IS
GENERIC (FACTEURS_INFLUENCES : REAL:=25.0);

PORT   (     SIGNAL MeasureX_Database : in std_ulogic_vector (ndata-1 downto 0);
             SIGNAL TestX_Database : in std_ulogic_vector (ndata-1 downto 0);
              -- attention à l'ordre de déclaration des ports
             SIGNAL X_Meas_D : in std_ulogic_vector (nbits-1 downto 0);
                        -- ne doit pas changer
             SIGNAL Start_memo : out std_ulogic:='0';
             SIGNAL X_Datas : out std_ulogic_vector (nbits-1 downto
0):="XXXXXXXX";
             SIGNAL Test_assesment, Alarm_test_meas : out std_ulogic_vector (n
downto 0):="00");

END;


ARCHITECTURE Struct OF Analyser IS

      --type Memo_sample is array (n_sample downto 1) of std_ulogic_vector
(nbits-1 downto 0);

      --comparaison des valeures recues dans X_Meas_D
      -- avec les valeurs de la base de donnée MeasX_Database ou TestX_Database

      Signal Data_moy :std_ulogic_vector (nbits-1 downto 0):="00000000";
      --signal relais entre les deux process
      Signal St_memo : std_ulogic :='0';

BEGIN

  ------------------------------------------------------------
  -- Process de traitement du signal (moyenne)
  ------------------------------------------------------------

process is

      variable Sample_data : Memo_sample :=("00000000", others=> "00000000");
      variable  Datas: std_ulogic_vector (nbits-1 downto 0):="00000000";
      variable x,val2, val_moy : integer:=0;
      variable val  : real;
begin

      --wait on MeasureX_Database, TestX_Database;

      --Boucle de stockage des n_samples mesurés dans le tableau Sample_data(i)
            for i in n_sample downto 1 loop -- n_sample
                  Wait on X_Meas_D until X_Meas_D /= "XXXXXXXX";
                  -- X_Meas_D /= "XXXXXXXX" then
                  Sample_data(i) := X_Meas_D;
                  --sample_data(i) = tableau des valeurs echantillonnées
                  val2:=conv_integer(X_Meas_D)+val2;
                  --Boucle de sommation des mesures mémorisées
            end loop;
            wait for 100us;
            val_moy:=val2/n_sample;
            Datas:=To_StdulogicVector(val_moy,nbits);
      X_Datas<=Datas;-- X_data doit contenir les infos : moyenne, min, max
      Data_moy<=Datas;
      Wait for 1ms;
      Start_memo <= ('1');
```

```
	St_memo <= ('1');
	Wait for 1ms;
	Start_memo <= ('0');
	St_memo<= ('0');
	Wait for 1ms;
	X_Datas<= "XXXXXXXX"; --pas de signal sur la ligne
	val2:=0;
	for i in (nbits-1) downto 0 loop -- réinitialisation des sorties
		Datas(i):='0';
	end loop;
end process;

  -------------------------------------------------------------
  -- Process de détection d'alarmes pour le test et la mesure
  -------------------------------------------------------------
process
begin
	wait on MeasureX_Database, TestX_Database, St_memo;

	if MeasureX_Database /= "UUUUUUUU" then
		wait on St_memo until St_memo='1';
		if Data_moy>MeasureX_Database then -- détection de seuil
			Alarm_test_meas <="11";--faire apparaitre le type de mesure
			wait for 20 ms;
			Alarm_test_meas <="00";
		else end if;
	elsif TestX_Database /= "UUUUUUUU" then
		wait on St_memo until St_memo='1';
		if Data_moy=TestX_Database then -- détection de test positif
			Test_assesment <="11";
			wait for 20 ms;
			Test_assesment <="00";
			else Alarm_test_meas <="11";
			wait for 20 ms;
			Alarm_test_meas <="00";
		end if;
	else Test_assesment <="11";
		wait for 20 ms;
		Test_assesment <="00";
	end if;
end process;


END Struct;
```

Références

Les références à des sites Internet sont données par des chiffres entre crochets.

[1] http://www.microbonding.com/fr/cob_fr.htm

[2] http://www.amkor.com/products/AdvPackageGlossary/AdvancedPackgeGlossary.pdf

[3] http://www.amkor.com/Products/all_products/fcCSP.cfm

[4] http://www.maineci.com/realisations.php3

[5] http://www.signetics.com/st_productc01.html

[6] http://www.roctest.com ; http://www.soil.co.uk ; http://www.singer-instruments.com/products/lvdts.html.

[7] http://sitessa.bao.stockho.com/Images/Upload/doc_pj/jou_sitescom6.pdf

[8] http://www.laas.fr/tina/

[9] http://www.afis.fr/upload/SDD/RECHERCHE/JPM_050414_0002.pdf

[10] http://www.telelogic.com/products/doorsers/doors

[11] http://www-306.ibm.com/software/awdtools/reqpro/.

[12] http://www.irqaonline.com/irqa/Default.htm.

[13] http://www.ansoft.com/products/em/simplorer.

[14] http://www.mentor.com/products/sm/systemvision/index.cfm.

[15] http://perso.wanadoo.fr/areg/

Les références bibliographiques d'articles ou d'ouvrages sont notées par les initiales majuscules du nom des principaux auteurs, suivies de l'année de la publication. Dans le cas d'un auteur unique, la référence est notée par les trois premières lettres du nom dont la première est en majuscule et les suivantes en minuscules, et ceci suivi de la date de la publication.

[ABG04] D. Andreu, N. Bruchon, T. Gil, « Du modèle à l'exécution :Traduction automatique de réseau de Petri interprété en Langage VHDL ». Rapport de recherche LIRMM n°04008, Juillet 2004.

[Afn96] "Analyse fonctionnelle : Caractéristiques fondamentales", Norme Afnor X50-100, 1996.

[Alb05] V. Albert, « Traduction d'un modèle de système hybride basé réseaux de Petri en VHDL-AMS », Master de conception en architecture de machines et systèmes informatiques, Université Paul Sabatier, LAAS-CNRS, Septembre 2005.

[ATU04] P. Acker, J.-M. Torrenti, F.-J. Ulm, "Comportement du Béton au Jeune Age" Traité MIM-Mécanique et Ingénierie des matériaux, Hermès Science, Lavoisier, 2004.

[BBB04] J. Burrell, T. Brooke, and R. Beckwith, "Vineyard Computing: Sensor Networks in Agricultural Production", PERVASIVE computing, IEEE, Volume 3, Issue 1, Jan-Mar 2004, Pages: 38-45.

[Ber04] P. Berckmans, « Description de sources d'énergies électrochimiques pour équipements « nomades » », rapport EDF R&D, HM-29/04/062/A, novembre 2004.

[BG03] M. Beigl, H. Gellersen, "Smart-Its: An Embeded Plateform for Smart Objects", Smart Objects Conference, Symposium Objets communicants, May 15-17, 2003 - Grenoble France.

[Boo93] G. Booch, "Object-Oriented Design with Applications". The Benjamin/Cummings Publishing Company, Inc., 2 ème édition, 1993.

[BRE04] C. Baron, S. Rochet, D. Estève, "GESOS : a multi-objective genetic tool for a project management considering technical and non-technical constraints", 1st IFIP International Conference on Artificial Intelligence Applications and Innovations (AIAI'2004), Toulouse (France), 22-27 Août 2004, Kluwer Academic Publishers, ISBN 1-4020-8150-2, Vol.6, pp.329-342.

[BRJ00] G. Booch, J. Rumbraugh, I. Jacobson, "UML : le guide de l'utillisateur", collection technologies objets, Eyrolles, 552 pages.

[BRV03] B. Berthomieu, P.-O. Ribet, F. Vernadat, « L'outil TINA -- Construction d'espaces d'états abstraits pour les réseaux de Petri et réseaux Temporels », Modélisation des Systèmes Réactifs, MSR'2003, Hermes, 2003.

[CCL02] M. Chan, E. Campo, E. Laval, D. Estève, "Validation of a remote monitoring system for the elderly: Application to mobility measurement", Technology and Health Care, vol. 10, n°5, pp. 391-399, Octobre 2002.

[Cha05] N. Chamseddine, « Mémoire de Stage de Fin d'étude » OLC-LAAS-CNRS, Septembre 2005.

[CGK03] M.S. Corquodale, F.H. Gebara, K.L. Kraver, D.M. Marsman, R.M. Senger, R.B. Brown, "A Top Down Microsystems Design Methodology and Associated Challenges", Proceedings of the Design, Automation and Test in Europe Conference and Exhibtion, 2003, IEEE Computer Society.

[CSE03] E. Campo, J.P. Scotto Di Rinaldi, D. Estève, N. Bailly, F. Benard, "Développement d'une nouvelle génération de gestionnaire d'énergie auto-configurable pour l'habitat : le concept ERGDOM", Annales du Bâtiment et des Travaux Publics, ISSN: 1270-9840, n°2, pp. 43-49, août 2003.

[EJT02] D. Estève, B. Jammes, A. Titli, M. Gonzales-Mendoza, A. Santana-Diaz, "Hypovigilance diagnosis module : developments and experiments", Contrat AWAKE N° IST-2000-28062, Septembre 2002, 16p.

[EOP05] P. Esteban, A. Ouardani, M. Paludetto, J.C. Pascal, "A Component Based Approach for System Design and Virtual Prototyping" , 12th Annual European Concurrent Engineering Conference (ECEC'2005), Toulouse (France), 11-13 Avril 2005, pp.85-90.

[Est97] D. Estève, “Basic Research for Microsystems Integration”, BARMINT, 1997, Ed. Cépaduès.

[FTM89] Y.-C. Tai and R.S. Muller, "IC-processed electrostatic micromotors," Sensors and Actuators, 20, 41-8, (November 15, 1989).

[GBG05] C. Guitierrez, C. Baron, L. Geneste, P. Clermont, D. Estève, S. Rochet, “How to interconnect Product Design and Project Management including Experience Feedback and Reusability Requirements”, The 2005 IEEE International Conference on Information Reuse and Integration. IEEE IRI-2005, 15-17 Août 2005.

[GL93] J.A. Goguen, C. Linde; Requirements Engineering, 1993., Proceedings of IEEE International Symposium on 4-6 Jan. 1993 Page(s):152 – 164.

[GMW04] Gaynor, M.; Moulton, S.L.; Welsh, M.; LaCombe, E.; Rowan, A.; Wynne, J.; “Integrating wireless sensor networks with the grid”, Internet Computing, IEEE, Volume 8, Issue 4, July-Aug. 2004 Pages:32-39.

[Gui03] D. Guihal. « Mémoire de Stage de Fin d’étude. Codesign HW-SW : Etude d’une interface entre outil de conception système et VHDL-AMS ». ENSPS Strasbourg. Airbus France, Avionics and simulation products 2003.

[Ham05] J.C. Hamon, « Méthodes et Outils de la Conception Amont pour les Systèmes et les Microsystèmes », Thèse de doctorat, Institut National Polytechnique de Toulouse, Février 2005.

[Har00] N. Harchani, « Etude d’une Méthodologie de Conception Descendante des Microsystèmes: Conception d’un Micro Système pour la surveillance des Contraintes Mécaniques en Aéronautique », Thèse de doctorat, Institut National Polytechnique de Toulouse, 2000.

[HC02] J.L. Hill, D.E. Culler, ”Mica: a wireless platform for deeply embedded networks”, Micro, IEEE Vol. 22, Issue 6, Nov.-Dec. 2002. Pages:12-24.

[Her02] Y. Hervé, « VHDL-AMS, Applications et enjeux industriels », Dunod, 2002.

[HRD01] J.M. Hender, T.L. Rodgers,. D.L. Dean, J.F. Nunamaker, “Improving group creativity: brainstorming versus non-brainstorming techniques in a GSS environment”, System Sciences, 2001. 34th Annual Hawaii International Conference on, 3-6 Jan. 2001.

[HSW00] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, K. Pister, “System Architecture Directions for Networked Sensors”, 9th International Conference on Architectural Support for Programming Languages and Operating Systems, Nov. 2000. Pages : 93-104.

[Jac94] I. Jacobson, “Object-Oriented Software Engineering : A Use Case Driven Approach”, Addiso-Wesley Object Technology Series, Mars 1994.

[Jim00] F. Jimenez, « Spécification et Conception de Micro-systèmes Basés sur des Circuits Asynchrones », Thèse de doctorat, Laboratoire d’Analyse et d’Architecture des Systèmes du CNRS, 2000.

[KMM03] Y. Kawahara, M. Minami, H. Morikawa, T. Aoyama, „Design and implementation of a sensor network node for ubiquitous computing environment”, Vehicular Technology Conference, 2003. IEEE 58th, Vol. 5, 6-9 Oct. 2003 Pages: 3005-3009.

[Kun00] K. Kundert, « Top-Down Design of Mixed-Signal Circuits”, Advances in Analog Circuits Design”, April 2000.

[Mac98] R.J. Machado et Al., An object oriented model for rapid prototyping of data path / control systems, the 9th symposium on information control in manufacturing, INCOM 1998.

[Mau03a] R.Maurice, « Etat actuel des réalisations des composants et des technologies d'assemblage utilisables pour la conception d'un microsystème multicapteur autonome interrogeable à distance », LAAS-CNRS / EDF R&D, Mars 2003.

[Mau03b] R. Maurice, « Cahier des charges fonctionnel pour la conception d'un microsystème multi-capteurs autonome interrogeable à distance », note EDF R&D, HP-18/03/047A, Juillet 2003.

[Mau04a] R. Maurice, « Description sous HiLeS du Dispositif Multicapteur Communicant EDF/LAAS », note EDF R&D, HP-18/04/001/A, Janvier 2004.

[Mau04b] R.Maurice, « Méthodologie de conception pour la réalisation d'un microsystème multicapteurs autonome communicant ». Article et poster, 7èmes Journées Nationales du Réseau Doctoral de Microélectronique (JNRDM'2004), Marseille, 4-6 Mai 2004, pp.166-168.

[Mau04c] R. Maurice, « Etude du microcapteur de déplacement Microstrain MG-DVRT-3 », Compte rendu d'essais EDF R&D, CR P18-2004-079, Août 2004.

[Mau05a] R. Maurice, « Simulation VHDL-AMS pour la validation comportementale d'un système intégré multicapteurs communicant ». Article et poster, Journée annuelle de l'école doctorale GEET, Toulouse, 17 Mai 2005, 3p.

[Mau05b] R.Maurice, D. Bouchet, E.Campo, D.Estève, « From requirements analysis to virtual prototyping: top-down design methodology for a reusable industrial microsystem ». Présentation orale et article, 18th International Conference on Software and Systems Engineering and their Applications (ICSSEA2005), Paris, 29, 30 Novembre et 1er Décembre 2005, 11p.

[May04] A. Maynial, « Adaptation matérielle et logicielle d'un banc d'étalonnage pour une acquisition numérique de microcapteurs et de capteurs intelligents », rapport EDF R&D, HP-18/04/055/A, Août 2004.

[MBS01] R. Min, M. Bhardwaj, S.-H. Cho, E. Shih, A. Sinha, A. Wang, A. Chandrakasan, "Low-power wireless sensor networks", VLSI Design, 2001. 14th Int. Conf. on, 3-7 Jan. 2001 Pages: 205 – 210.

[Mil56] G.A. Miller, « The magic number seven, plus or minus two : some limits on our capacity for processing information", Psychological Review 63: 81-97 (1956).

[MPS02] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, J. Anderson, "Wireless Sensor Network for Habitat Monitoring", 1st ACM international workshop on Wireless Sensor Networks and Applications, Pages: 88-97.

[PDB05] M. Paludetto, J. Delatour, A. Benzina, « UML et réseaux de Petri », Technique et Science Informatiques, Vol.23, N°4, pp.543-567, 2004.

[PTB98] S. Pinel, J. Tasselli, J.P. Bailbé, A. Marty, P. Puech and D. Estève, "Mechanical lapping, handling and transfer of ultra-thin wafers", 1998 J. Micromech. Microeng. Volume8, 338-342.

[RBP91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W.Lorensen. "Object-Oriented Modeling andDesign", Prentice-Hall, Inc., 1991.

[RHG05] S. Rochet, J.C. Hamon, C. Gutierrez, C. Baron, D. Estève, « Vers une démarche de conception cohérente : Proposition d'une méthode et d'outils pour la gestion des alternatives de conception », 6e Congrès international de génie industriel, 7-10 juin 2005.

[Seb05] J. Sebaa, « Rapport de Stage ENSEEIHT », 18 p, Juillet-Août 2005.

[SNP04] P.Schmitt, J.M.Nicot, F.Pressecq, X.Lafontan, C.Oudea, D.Estève, J.Y.Fourniols, H.Camon, « Space microsystems and reliability : the contribution of behavioral modelling", CANEUS'04. Conference on Micro-Nano-Technologies for Aerospace Applications, Monterey (USA), 1-5 Novembre 2004, pp.95-100.

[SRS04] K.I. Shinotani, P.M. Raj, M. Seo, S. Bansal, H. Sakurai, S.K. Bhattacharya, R. Tummala, "Evaluation of Alternative Materials for System-on-Package (SOP) Substrates", IEEE Transactions on Components and Packaging Technologies, Vol. 27, N°. 4, Décembre 2004.

[Sze94] S.M. Sze Semiconductor sensors, J.Wiley, cop. 1994.

[TM99] R.R. Tummala, V.K. Madisetti, "System on Chip or System on Package?",Design and Test of Computers, IEEE, Apr-Jun 99, Pages:48-56.

[TP91] T. Tse, L. Pong, "An Examination of Requirements Specification Languages", The Computer Journal, vol. 34, no. 2, 1991, Pages: 143 – 152.

[Vac03] Vachoux, « Modélisation de systèmes analogiques et mixtes : Introduction à VHDL-AMS », http://lsmwww.epfl.ch/Education/Documents/modelmix03.pdf, « VHDL-AMS Essentiel », http://lsmwww.epfl.ch/Education/Documents/VHDLAMS_instant.pdf.

[VD05] Y. Vanderperren, W. Dehaene, "UML2 and SysML: an Approach to Deal with Complexity in SoC/NoC Design", Proceedings of the Design, Automation and Test in Europe Conference and Exhibtion, 2005, IEEE Computer Society.

[WLL01] B. Warneke, M. Last, B. Liebowitz, K.S.J. Pister, "Smart Dust: Communicating with a Cubic-millimeter Computer", Computer, Vol. 34, Issue 1, Jan. 2001 Pages:44-51.

[WN91] K.D. Wise, N. Najafi, "The Coming Opportunities in Microsensor Systems", Solid-State Sensors and Actuators, 1991. Digest of Technical Papers, TRANSDUCERS '91.,24-27 June 1991 Pages:2-7.

[YOD02] X. Yang, K. G. Ong, W. R. Dreschel, K. Zeng, C. S. Mungle and C. A. Grimes, "Design of a Wireless Sensor Network for Long-term, In-Situ Monitoring of an Aqueous Environment" Sensors 2002, Vol 2, Issue 11, Pages: 455-472.

Résumé

Ce travail de thèse porte sur la problématique de conception et de réalisation d'un microsystème multicapteur communicant pour une application Génie Civil. Ces travaux ont associé le LAAS-CNRS et EDF R&D. Il présente et définit la problématique de recherche et de développement de nouvelles générations d'outils pour la conception système et traite conjointement un exemple d'application proposant notamment une solution de mesure de micro-déplacement sur site.

Ce manuscrit fait d'abord le point sur la dynamique de développement microsystème et rappelle les méthodes de conception et d'intégration de systèmes à base de composants COTS. Le projet qui nous concerne est ensuite présenté et analysé par la méthode de l'analyse du besoin pour établir un cahier des charges, porte d'entrée de la démarche générale de conception.

Il montre, sur l'exemple proposé, les étapes que nous avons conduites pour atteindre le stade des spécifications validées. La méthodologie de conception amont proposée associe la démarche Top-Down et SysML pour réduire au maximum le lien entre le cahier des charges et la première représentation modélisée du système. Cette modélisation amont, indépendante de l'implémentation, est effectuée sous l'outil HiLeS et ouvre la voie de la première vérification du système, par le biais de la logique temporelle.

Ensuite, les choix d'agrégation, de sélection et d'implémentation des composants sont décrits et permettent d'aboutir à la modélisation fonctionnelle sous VHDL-AMS et au prototypage virtuel.

Enfin, ce travail présente les étapes d'intégration et les choix de composants qui mènent au prototype réel. Une première validation de ce prototype réalisé est effectuée par des mesures sur site.

Mots clefs : Prototypage virtuel, conception microsystèmes, analyse des exigences, vérification haut niveau, VHDL-AMS, système communicant.

Abstract

The study described in this thesis focuses on the design and realisation of a wireless multisensor microsystem for Civil Engineering. It defines the problem associated with the realisation of a new generation of system design tools. A possible solution is presented and used for the design of an autonomous micro-displacement measuring system. This microsystem has been developed in a joint collaboration between the LAAS-CNRS and EDF R&D.

Initially, this manuscript examines the trends in microsystem development and points out the methods of design and integration of systems containing COTS components.

The aim of this project is then presented and analysed using the requirement analysis method in order to establish the specifications of the micro-displacement measuring system. These specifications are then utilised in the general design methodology.

The proposed design methodology associates Top-Down and SysML approaches to fill the gap between the system specifications and the first system model. This high level model, which is independent of the implementation, is made using the HiLeS software tool and opens the way for the first system model verification, via temporal logic. Then, aggregation, selection, and implementation choices for components enable the functional modelling under VHDL-AMS and virtual prototyping. Finally, this work presents the component choices and the integration steps that have lead to the realisation of a prototype which has been validated by laboratory tests.

Keywords : Virtual prototyping, microsystem design, requirement analysis, high level verification, VHDL-AMS, autonomous system.