Modélisation du concept capteur intelligent par une approche orientée objet : Application à un capteur intelligent de température

Damien Luttenbacher

▶ **To cite this version:**

Damien Luttenbacher. Modélisation du concept capteur intelligent par une approche orientée objet : Application à un capteur intelligent de température. Sciences de l'ingénieur [physics]. Université Henri Poincaré - Nancy 1, 1997. Français. NNT : 1997NAN10005 . tel-01748144

HAL Id: tel-01748144
https://hal.univ-lorraine.fr/tel-01748144

Submitted on 29 Mar 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ DE LORRAINE

AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite encourt une poursuite pénale.

Contact : ddoc-theses-contact@univ-lorraine.fr

LIENS

Code de la Propriété Intellectuelle. articles L 122. 4
Code de la Propriété Intellectuelle. articles L 335.2- L 335.10
http://www.cfcopies.com/V2/leg/leg_droi.php
http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

UFR Ecole Supérieure des Sciences et
Technologies de l'Ingénieur de Nancy
Ecole Doctorale IAE+M
DFD Automatique et Production automatisée

S.C.D. - U.H.P.
NANCY 1 SCIENCES
BIBLIOTHÈQUE

Thèse

Présentée pour l'obtention du titre de

Docteur de l'Université Henri Poincaré, Nancy 1

Spécialité Automatique

par **Damien LUTTENBACHER**

MODELISATION DU CONCEPT CAPTEUR INTELLIGENT PAR UNE APPROCHE ORIENTEE OBJET : APPLICATION A UN CAPTEUR INTELLIGENT DE TEMPERATURE

Soutenue publiquement le 10 février 1997 devant la commission d'examen

Membres du jury :

Président	:	G. MOREL	Professeur, Université Henri Poincaré Nancy 1
Rapporteurs	:	L. FOULLOY	Professeur, Université de Savoie
		M. BAYART	Professeur, Université des Sciences et Technologies de Lille
Examinateurs	:	A. NAPOLI	Chargé de Recherche CNRS, CRIN-INRIA
		P. PERRIN	Ingénieur, EDF-GDF
		M. ROBERT	Professeur, Université Henri Poincaré Nancy 1 (Directeur de thèse)

Centre de Recherche en Automatique de Nancy - Equipe Instrumentation Intelligente
2, rue Jean Lamour - 54500 VANDOEUVRE-lès-NANCY

Remerciements

Le travail de recherche présenté dans ce mémoire a été réalisé au Centre de Recherche en Automatique de Nancy (CRAN - CNRS URA 821) sous la direction scientifique de Monsieur Michel ROBERT, Professeur à l'Université Henri Poincaré Nancy 1. Qu'il trouve ici l'expression de ma gratitude pour sa constante disponibilité, ses nombreux conseils prodigués et surtout pour l'amitié qu'il ma toujours témoignée.

Je remercie vivement Madame Mireille BAYART, Professeur à l'Université des Sciences et Technologies de Lille, et Monsieur Laurent FOULLOY, Professeur à l'Université de Savoie qui ont accepté la lourde tâche de rapporteurs.

Mes remerciements vont également à Monsieur Gérard MOREL, Professeur à l'Université Henri Poincaré Nancy 1, Monsieur Amedeo NAPOLI, Chargé de Recherche CNRS au CRIN-INRIA et Monsieur Philippe PERRIN, Ingénieur à EDF-GDF, d'avoir accepté de participer à ce jury.

Je profite de cette occasion pour remercier tous les membres du groupe CRAN/AMI (Actionnement et Mesure Intelligents), ainsi que ceux du groupe AFCET/COOSI (Conception Orientée Objet de Systèmes d'Information), pour les nombreuses discussions fructueuses qui ont pu être mises en application dans ce travail.

Mes remerciements vont aussi à tous mes collègues de laboratoire qui m'ont témoigné leur amitié et m'ont apporté à tout moment leur contribution, en particulier :

Messieurs Jean-Marc THIRIET, Jean-Luc NOIZETTE, Jean-Michel RIVIERE, Gérard BLOCH, Fredéric HERMANN, Blaise CONRARD et Mademoiselle Sandrine ROTH.

Sommaire

Chapitre III : Modélisation et cycle(s) de développement

Chapitre IV : Démarche de modélisation selon une approche orientée objet

Chapitre V : Réutilisation : cas du modèle de référence de capteur intelligent

Introduction

Le mémoire est structuré en six chapitres. Le premier chapitre rappelle les définitions essentielles liées à la métrologie. Le concept capteur intelligent" est présenté par l'intermédiaire de divers modèles informels, inspirés des travaux du CIAME (Comité Interprofessionnel pour l'Automatisation et la MEsure).

Le second chapitre est consacré à une étude bibliographique, présentant l'évolution des méthodes de développement de systèmes, des méthodes traditionnelles aux méthodes orientées objet, ainsi que le concept d'objet et le vocabulaire associés à l'approche orientée objet qui est à la base de nos travaux pour la représentation du concept capteur intelligent.

Le troisième chapitre précise, dans un contexte cycle de vie, les notions d'intervenants et de services apportés aux différents intervenants par un capteur intelligent. Il présente également un modèle pour le développement d'un système intégrant les besoins liés à la mis en oeuvre de la réutilisation.

Dans le quatrième chapitre, la méthodologie retenue pour la modélisation du concept de capteur intelligent, en l'occurrence OMT (Object Modeling Technique), est présentée dans le cadre de l'établissement d'un modèle de référence orientée objet d'un capteur intelligent. Ce modèle décrit complètement un capteur intelligent au moyen des modèles objet, dynamique et fonctionnel.

Le cinquième chapitre revient sur un des intérêts majeurs de l'approche orientée objet, à savoir la réutilisation des composants logiciels en particulier. Certaines techniques permettant une réutilisation sont présentées dans le cadre du concept de capteur intelligent et mises en oeuvre pour la réalisation d'une bibliothèque de composants réutilisables pour la conception de l'entité "chaîne de mesure".

Dans le dernier chapitre la méthodologie basée sur l'approche orientée objet est appliquée au cas d'un capteur intelligent de température démontrant ainsi la pertinence du modèle de référence orientée objet d'un capteur intelligent en le spécialisant suivant les techniques de réutilisation développée précédemment. L'outil de simulation d'un capteur intelligent de température réalisée à partir des modèles élaborés précédemment montre l'adéquation de la méthode (OMT) et des outils (LOV/OMT pour l'édition des modèles objet, dynamique et fonctionnel; GEODE pour la simulation du modèle dynamique) pour la modélisation du concept capteur intelligent, d'une part pour l'établissement d'une boîte à outils d'aide à la conception de capteurs intelligents intégrant l'approche orientée objet, et d'autre part pour la définition d'un modèle de référence orienté objet de capteur intelligent.

Les références bibliographiques sont regroupées à la fin de chaque chapitre et sont indiquées dans le texte entre [crochets]. Les références Internet sont quant à elles indiquées dans le texte entre {accolades}. Un glossaire des principaux termes employés dans le texte est disponible à la fin du document.

Chapitre I

Mesure et capteurs

I.1 La mesure

"N'existe, au sens physique du terme, que ce qui est mesurable", Max Planck.

De façon générale, la mesure permet de passer du qualitatif (description de la perception) au quantitatif (transformation de cette perception en quantité). La science énonce des lois qui relient les différentes grandeurs pour former un ensemble cohérent de connaissances. A notre époque, science et technique interfèrent très fortement et constituent la technologie. De même que l'écriture sert de véhicule aux connaissances, la mesure constitue celui de la science et de la technologie [PRIE-95].

Il est vrai que de tout temps l'homme s'est pris comme référence. Nomade d'origine, il utilise les différentes parties de son corps comme unités de mesure, unités de mesure certe approximatives, mais universelles et toujours disponibles.

Mesurer est le propre de l'homme, au même titre que rire et le langage. Les mesures de dimension, de distance, de quantité et de temps eurent le corps humain comme étalon (par exemple : le pied équivalent à 33 cm ...).

Bien que l'industrie et la science de la mesure soient parmi les plus discrètes des activités de haute technologie, leurs progrès considérables dans la dernière décennie ont permi le développement d'une foison de concepts et de produits. L'homme, conscient des enjeux, a toujours su mettre en oeuvre les moyens en sa possession pour faire progresser ses connaissances et leurs applications. Les découvertes de la physique moderne, notamment celles du transistor et du laser , ont bouleversé les données et multiplié les espoirs. Les capteurs ne se contentent plus de transformer les données, ils deviennent "intelligents" en les interprétant "in situ". La mesure autorise également d'anticiper les pannes en analysant par exemple l'évolution du bruit d'une machine en cours de fonctionnement (machines outils, moteurs ...). Elle se diversifie et la mesure des grandeurs non électriques (pression, température, accélération, chimie, ...) présente une croissance actuelle spectaculaire.

Le terme mesure présente de nombreuses acceptions dans la langue française, il est donc préférable d'utiliser en métrologie le terme mesurage qui est l'action de quantifier ce qui n'était pas qualifié. Cela sous entend la capacité de pouvoir faire correspondre grâce à une échelle absolue ou relative un ou plusieurs nombres à une situation ou un phénomène donné. Cette correspondance doit être facilement reproductible, transmissible et, pour avoir un sens, associée à une estimation du degré de certitude que l'on peut attacher au résultat [SELI-92].

L'homme a toujours eu le désir d'attribuer une valeur aux grandeurs qui l'entourent, soit pour les échanges commerciaux, soit pour comparer, soit encore dans le but d'améliorer ses conditions de vie ou ses connaissances.

La mesure est devenue l'une des sources principales de la connaissance scientifique ; "Si vous pouvez mesurer ce dont vous parlez", observait Lord Kelvin, "et comparer par un nombre, vous savez quelque chose de votre sujet ; mais si vous ne savez pas la mesure, vos connaissances sont d'une pauvre espèce et bien peu satisfaisantes" [PRIE-95]. En fait, les sciences expérimentales ne

peuvent se développer et progresser qu'au prix d'évaluations quantitatives de plus en plus précises et soignées.

Sans doute les progrès de la métrologie suivent-ils, maintes fois, ceux des autres sciences. Mais souvent aussi, ils les précèdent et les engendrent.

On rencontre deux méthodes de mesure :

La méthode directe :

Dans le cas d'une mesure directe on évalue sans intermédiaire une grandeur. Ainsi l'utilisation d'une pipette graduée procède de cette méthode dans le cas d'un volume comme la pesée par une balance de type "Roberval". D'une manière générale se retrouvent dans cette catégorie les mesures effectuées grâce à l'utilisation d'étalons primaires ou secondaires. Ces étalons sont des dispositifs physiques ou matériels dont la valeur connue, permet de vérifier l'exactitude des résultats donnés par un système de mesure. L'intervention de grandeurs autres que celle désirée, les grandeurs d'influence, doit ici être déterminée avec précision et ce mode de mesure pourtant séduisant par sa simplicité apparente y est généralement très sensible (température, usure naturelle des étalons ...).

La méthode indirecte :

C'est le cas le plus fréquent, il consiste à faire intervenir un ou plusieurs des modes d'interaction d'une grandeur avec une autre et ainsi d'en qualifier la valeur.

Par exemple, la mesure d'une température peut ainsi utiliser :

- la dilatation d'un liquide, solide ou gaz et est alors également sensible à la pression ambiante,
- la variation des propriétés électriques d'un composé et ainsi être sujette aux phénomènes de contraintes mécaniques, de vieillissement et de corrosion.

I.1.1 Définition et caractéristiques générales

Selon la norme AFNOR (NF X 07-001), un capteur est l'élément d'un appareil de mesure ou d'une chaîne de mesure auquel est directement appliquée une grandeur à mesurer.

Une autre définition tirée du dictionnaire propose : dispositif permettant de détecter, en vue de le représenter, un phénomène physique sous la forme d'un signal (généralement électrique).

Afin de lever toute ambiguïté, il est nécessaire de préciser [NOIZ-91] :

- L'élément qui est mentionné dans la norme AFNOR (NF X 07-001) est couramment défini actuellement comme le corps d'épreuve. Celui-ci correspond à l'entité directement en liaison avec la grandeur physique à observer.
- Le dispositif qui est mentionné dans la définition issue du dictionnaire est couramment défini comme capteur. Par la suite, nous nous placerons dans ce cas de figure, correspondant à une vision selon le point de vue de l'utilisateur.

Comme nous l'avons rappelé ci-dessus le capteur "convertit" une grandeur physique observée en une grandeur selon un référentiel que le consommateur a l'habitude de manipuler (figure I.1).

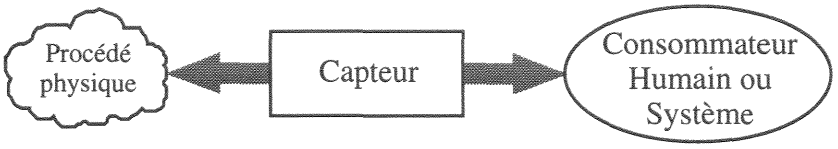


Figure I.1 : Modèle d'un capteur.

Dans un système automatisé de production (abordé plus en détail dans la suite du chapitre) la chaîne de mesure permet le recueil d'informations, sur l'état du processus, à destination du système d'information qui assure l'élaboration des signaux de commande du processus et synthétise des informations destinées aux opérateurs du système.

Seul le capteur permet d'obtenir une représentation de l'état du processus (figure I.2), cette représentation doit être la plus "vraie" possible afin qu'elle n'engendre pas d'actions erronées sur le processus. Les mesures peuvent être entachées d'erreurs dont les sources peuvent être nombreuses et variées [NOIZ-91].
La crédibilité de la mesure définit alors l'aptitude du dispositif de mesure à délivrer des informations de mesure justifiant un degré de confiance requis. On procède à leur validation, d'une part par confrontation des mesures identiques en redondance matérielle et d'autre part par confrontation des mesures différentes en redondance analytique. Avant tout, il est nécessaire de choisir le capteur en fonction de ses caractéristiques métrologiques (cf. § I.2.1) et de ses critères de sûreté de fonctionnement.

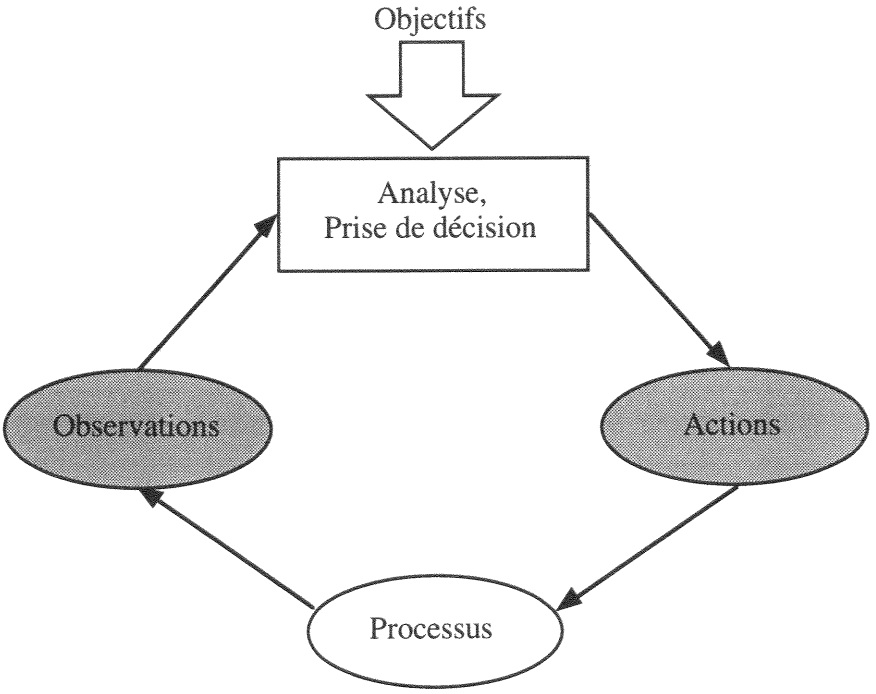


Figure I.2 : Boucle Action - Décision.

I.2 La chaîne de mesure

La chaîne de mesure est constituée de l'ensemble des dispositifs, y compris le corps d'épreuve, rendant possible dans les meilleures conditions la détermination précise de la valeur du mesurande.

Le premier élément constitutif d'un capteur est le transducteur ou corps d'épreuve. Celui-ci réalise la conversion de la grandeur physique observée en une grandeur intermédiaire (dans la plupart des cas analogique). Ce premier dispositif doit toujours être présent sans quoi le capteur est inutilisable.

On distingue habituellement deux classes de transducteurs [ASCH-91] :

- les transducteurs dits actifs,
- et les transducteurs dits passifs.

Les transducteurs actifs sont ceux pour lesquels l'effet physique assure la conversion en énergie électrique de la forme d'énergie propre au mesurande. Par exemple, grâce à l'effet piézo-électrique, l'application d'une force ou d'une contrainte mécanique à certains matériaux dits piézo-électriques (quartz) provoque une déformation et l'apparition de charges électriques égales et de signes contraires sur les faces opposées.

A l'opposé les transducteurs passifs doivent être excités par une énergie supplémentaire (le plus souvent électrique) afin de pouvoir délivrer un signal, image de l'action du phénomène physique sur le transducteur. C'est généralement l'impédance du transducteur qui est sensible aux variations du mesurande. Cette variation d'impédance peut être liée à des variations des caractéristiques géométriques (le capteur comporte alors un élément mobile ou déformable) ou des propriétés électriques des matériaux (résistivité, perméabilité magnétique, constante diélectrique) qui peuvent varier sous l'influence de certaines grandeurs physiques (température, pression, humidité). Par exemple, la résistance d'une sonde de platine est fonction de la température, mais la variation de résistance n'est mesurable que si la sonde de platine est traversée par un courant connu et constant.

Dans les capteurs analogiques, le signal porteur de l'information issue du transducteur subit une série de transformations effectuées par un conditionneur de signaux.

Le signal électrique issu du transducteur est porteur de l'information utile liée au phénomène physique mais est généralement de bas niveau et n'est pas uniquement fonction du mesurande, mais également des grandeurs d'influence qui "bruitent" le signal. Il est donc nécessaire de faire subir un certain nombre de transformation (appelé conditionnement) à ce signal de façon à extraire dans les meilleures conditions possibles une image de la valeur du mesurande. Ce conditionnement peut consister en une amplification, un filtrage, une compensation analogique des grandeurs d'influence, ...

A la sortie du conditionneur (deuxième dispositif réalisant le conditionnement) le signal électrique traité contient l'information utile relative au mesurande, plus ou moins perturbée par des bruits et dérives. Ce signal électrique est ensuite transmis à l'utilisateur ou consommateur selon certains standards (par exemples : 4-20mA ou 0-10V), à l'aide d'un transmetteur (troisième dispositif).

L'ensemble que nous venons de décrire et qui est schématisé sur la figure I.3, constitue le capteur (ou capteur analogique classique) sous sa forme la plus simple.

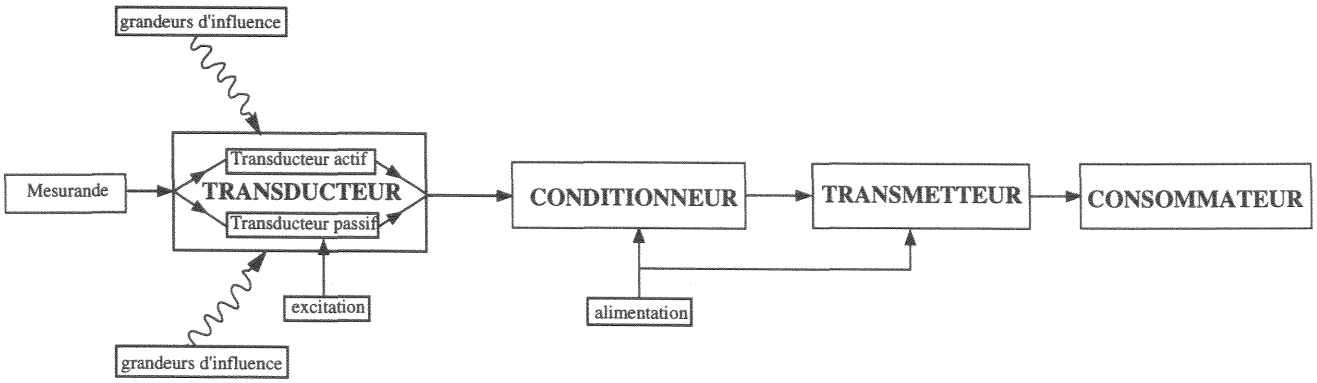


Figure I.3 : Capteur traditionnel.

I.2.1 Rappel des principales caractéristiques métrologiques

I.2.1.1 Les erreurs de mesures [ASCH-91]

Une mesure est entachée d'erreurs quand elle s'écarte de la vraie valeur du mesurande, cet écart s'appelle alors **erreur de mesure**. Les seuls mesurandes dont la valeur est parfaitement connue sont les grandeurs étalons, puisqu'elles sont fixées par convention. C'est la valeur vraie du mesurande qui détermine l'excitation du capteur. La valeur mesurée, à laquelle a accès l'utilisateur, dépend de la réponse globale de la chaîne de mesure. L'erreur de mesure, qui ne peut être connue avec exactitude mais estimée, résulte généralement des imperfections liées à la chaîne de mesure, qui dégradent l'information du signal au cours de son traitement et du non respect d'un certain nombre de précautions par l'expérimentateur lors du mesurage. On distingue deux catégories d'erreurs :

- les **erreurs systématiques**, qui sont caractérisées par un écart constant entre la valeur vraie et la valeur mesurée. Elles sont mises en évidence (après étude statistique) par un écart entre les valeurs les plus probables, tirées de deux séries de mesurages portant sur le même mesurande et effectuées par des méthodes et capteurs différents. Les erreurs systématiques ont habituellement pour causes : le décalage du zéro d'un appareil de mesure, la valeur inexacte de la tension d'alimentation d'un pont, le vieillissement d'un capteur, les perturbations apportées par le capteur, ...

- les **erreurs accidentelles**, caractérisées par une apparition, une amplitude et un signe aléatoires. Alors que certaines causes liées à l'apparition de ces erreurs peuvent être déterminées, les écarts qu'elles entraînent sur la valeur mesurée du mesurande restent inconnus. Les erreurs accidentelles résultent : des erreurs de parallaxe, des fluctuations de la tension d'alimentation (modifiant les performances des appareils), de l'influence d'un paramètre non corrigé, des perturbations électromagnétiques, ...

I.2.1.2 Fidélité, Justesse, Précision

La **précision** d'un capteur (ou encore son exactitude), c'est-à-dire son aptitude à donner des mesures proches de la valeur vraie du mesurande, dépend de ses caractéristiques de **fidélité** et de **justesse**.

La fidélité permet de juger de l'importance des erreurs accidentelles : des résultats de mesures groupés autour de la valeur moyenne attestent de la faiblesse des erreurs accidentelles.

La justesse permet de juger de l'importance des erreurs systématiques : une valeur, la plus probable du mesurande très proche de la valeur vraie, atteste de la faiblesse des erreurs systématiques.

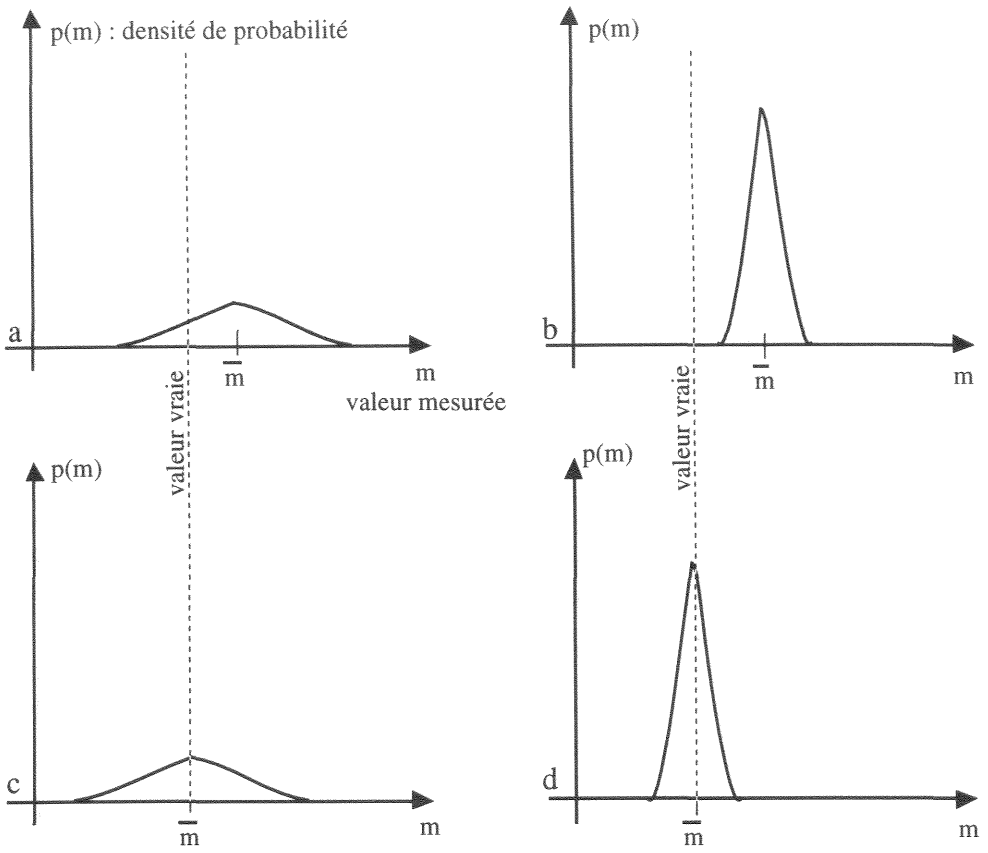


Figure I.4 : Analyse des mesures caractérisant la précision d'un capteur [ASCH-91].

a - erreurs systématiques et accidentelles importantes : capteur ni juste, ni fidèle
b - erreurs systématiques importantes, erreurs accidentelles réduites : capteur fidèle mais non juste
c - erreurs systématiques faibles, erreurs accidentelles importantes : capteur juste mais non fidèle
d - erreurs systématiques et accidentelles faibles : capteur juste et fidèle donc précis

I.2.1.3 Étalonnage, Calibration [ASCH-91]

L'étalonnage du capteur comprend un ensemble d'opérations, dans des conditions spécifiées, qui permet de définir sous forme graphique ou algébrique, la relation entre les valeurs étalons du mesurande et celles de la grandeur de sortie et ceci compte tenu de tous les paramètres additionnels susceptibles de modifier la réponse du capteur (grandeurs d'influence). Le plus souvent, l'étalonnage est effectué de façon indirecte, par comparaison avec un capteur de référence.

La calibration correspond au positionnement matériel des repères (éventuellement de certains repères principaux seulement) d'un appareil de mesure en fonction des valeurs correspondantes de la grandeur mesurée. Dans la plupart des cas, ces repères correspondent au zéro de l'appareil et de l'étendue d'échelle.

Pour que le capteur donne des résultats de mesure précis après étalonnage, il faut que la **répétabilité** des mesures soit garantie. La répétabilité témoigne alors de la capacité du capteur à délivrer des valeurs de sortie identiques, dans les limites spécifiées chaque fois que ce capteur est utilisé dans des conditions identiques : même mesurande et mêmes paramètres additionnels.

I.2.1.4 Limites d'utilisation du capteur [ASCH-91]

Les contraintes mécaniques, thermiques ou électriques auxquelles un capteur est soumis entraînent, lorsque leurs niveaux dépassent des seuils définis, une modification des caractéristiques du capteur, telles qu'elles étaient connues par étalonnage préalable ou spécifications du constructeur. Il est donc indispensable que l'utilisateur soit averti des diverses limites d'utilisation d'un capteur et des risques qu'il encourt à les dépasser.

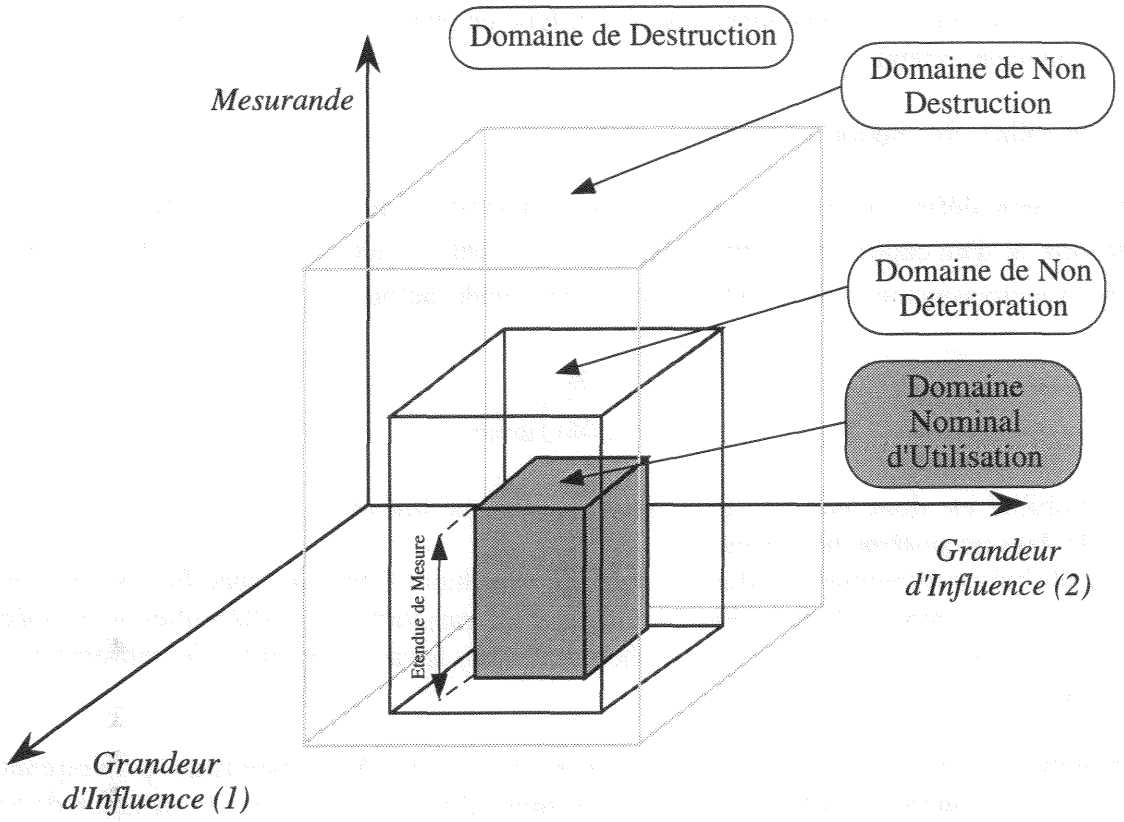


Figure I.5 : Limite d'emploi du capteur.

Domaine nominal d'utilisation (ou d'emploi) :

Il correspond aux conditions normales d'utilisation du capteur ; ses limites sont les valeurs extrêmes que peuvent atteindre de façon permanente le mesurande, les grandeurs physiques qui lui sont associées ou les grandeurs d'influence, sans que soient modifiées les diverses spécifications qui caractérisent le fonctionnement du capteur.

Domaine de non-détérioration :

Lorsque les valeurs du mesurande, des grandeurs physiques associées ou des grandeurs d'influence dépassent les limites du domaine nominal d'emploi mais restent inférieures aux bornes du domaine de non-détérioration les caractéristiques métrologiques du capteur risquent d'être modifiées ; cette altération est cependant réversible, le capteur retrouve ses caractéristiques spécifiées lorsque les conditions de fonctionnement redeviennent celles du domaine nominal d'emploi.

Domaine de non-destruction :

Lorsque les valeurs du mesurande, des grandeurs physiques associées ou des grandeurs d'influence dépassent les limites du domaine de non-détérioration tout en restant inférieures aux bornes du domaine de non-détérioration, les caractéristiques du capteur sont modifiées de façon

irréversible ; la réutilisation du capteur, dans son domaine nominal d'emploi, nécessite donc un nouvel étalonnage.

Étendue de mesure :

Elle est définie par la différence des valeurs extrêmes de la plage du mesurande dans laquelle le fonctionnement du capteur satisfait à des spécifications données. La plage du mesurande correspondant à l'étendue de mesure est souvent identique au domaine nominal d'emploi pour ce qui est du mesurande ; elle peut cependant être plus réduite ou plus étendue selon la sévérité des critères du fonctionnement retenu.

I.2.1.5 Sensibilité, Résolution [ASCH-91][PARA-86]

Généralement définie autour d'un point de fonctionnement (une valeur constante du mesurande), la **sensibilité** S_i d'un capteur est définie comme le quotient de l'accroissement Δs de la grandeur de sortie, par l'accroissement correspondant Δm du mesurande, autour de la valeur m_i.

$$S_i = \left[\frac{\Delta s}{\Delta m}\right]_{m=mi}$$

La sensibilité est donc une constante dans un système linéaire et dépend de la valeur m_i du mesurande dans un système non-linéaire.

La sensibilité est généralement donnée pour des conditions d'emplois spécifiques : en précisant l'amplitude de la tension d'alimentation dans le cas d'un transformateur différentiel ou en spécifiant la température et le coefficient de variation thermique dans le cas d'une jauge d'extensométrie semi-conductrice.

La **résolution** correspond à l'accroissement minimum du mesurande provoquant une modification de la grandeur de sortie. A titre d'exemple, il ne se produit pas de variation de tension notable au curseur d'un potentiomètre bobiné, si le déplacement du curseur est inférieur à la distance entre deux spires.

I.2.1.5.1 Ecart d'étalonnage

La **linéarité** (plus généralement **écart de linéarité**) du capteur décrit le degré de concordance entre la courbe d'étalonnage du capteur et un modèle mathématique choisi comme référence.

I.2.1.5.2 Décalage du zéro [PARA-86]

La notion de décalage de zéro s'applique essentiellement aux capteurs dont le signal de sortie est théoriquement nul pour une valeur du mesurande égale à zéro. On appelle alors décalage de zéro (ou zéro offset) la valeur du mesurande à laquelle correspond réellement le zéro de la grandeur de sortie.

I.3 Capteur intelligent

Ce paragraphe est fortement inspiré des travaux antérieurs de l'équipe instrumentation intelligente du CRAN [ROBE-93b, 93c] [RIVI-91, 93, 95] [HERM-92], des travaux du groupe de travail CIAME [CIAM-87] ainsi que des travaux de [GEHI-93] [HOSS-90].

Les possibilités technologiques de la micro-électronique ont contribué à l'apparition du concept de capteur intelligent. En dotant le transmetteur, associé aux transducteurs, d'un organe de calcul interne, les performances métrologiques et les possibilités fonctionnelles d'un capteur sont considérablement accrues.

L'architecture d'un capteur intelligent peut être décrite par la figure I.6. Ses constituants principaux sont :

- un ou plusieurs transducteurs (permettant la mesure du mesurande et des grandeurs d'influence),
- des conditionneurs spécifiques,
- l'alimentation,
- un organe de calcul interne (microprocesseur, DSP...) associé à des composants permettant des traitements locaux et l'élaboration d'un signal numérique,
- un interface de communication permettant un échange bidirectionnel d'information par l'intermédiaire d'un médium de communication.

La figure I.6 présente également la comparaison des deux types de chaînes de mesure entre le capteurs classiques et les capteurs intelligents.

La spécificité du capteur intelligent peut être résumée dans sa capacité interne de calcul, de traitement, associé aux possibilités de communication. Ces caractéristiques technologiques permettent d'améliorer la fonction "mesure" dans les systèmes d'automatisation.

Ci-dessous sont résumés les principaux gains apportés par les capteurs intelligents, tant en terme métrologique qu'en terme de fonctionnalité [SOLA-93] :

- Accroissement de l'exactitude des mesures par la prise en compte des grandeurs d'influence, la compensation locale des dérives;
- Crédibilité accrue des informations;
- Possibilité de validation et de diagnostic des mesures, de la communication;
- Elaboration des données directement exploitables aux niveaux supérieurs permet de distribuer une partie non négligeable de la puissance totale de calcul requise;

Pour l'utilisateur qui conçoit des systèmes utilisant des capteurs intelligents, le concept de capteur intelligent est synonyme de :

- Diminution des coûts par simplification des câblages (bus de terrain), réduction des transmetteurs, convertisseurs ou adaptateurs intermédiaires;
- Augmentation des performances par compensation au niveau des capteurs des grandeurs d'influence ou des non linéarités;
- Simplification de la maintenance et de la configuration (auto-calibration, diagnostic ...).

L'intérêt fonctionnel des capteurs intelligents est résumé dans le tableau ci-dessous [CIAM-87] :

Métrologie	Installation - Configuration (mise en service)	Exploitation - Maintenance
Fiabilité Précision accrue Étendues de mesure plus grandes Traitement du signal validation des mesures Corrections des grandeurs d'influence	Identification Configuration à distance Communication numérique bidirectionnelle Dialogue local (console portable)	Identification Auto-diagnostics Auto-surveillance Aide à la maintenance Étalonnage Contrôles à distance

Dans un contexte de système distribué favorisant la communication entre équipements, se pose le problème de compatibilité entre les équipements ou encore interopérabilité. Pour les équipements analogiques qui transmettent (on ne peut pas réellement parler de communication) leurs mesures par le standard 4/20 mA, la compatibilité entre équipements de divers constructeurs est assurée. En ce qui concerne les équipements intelligents qui disposent de possibilités de communication numérique bidirectionnelle, le problème est plus complexe : l'équipement avec lequel communique le capteur intelligent (ou actionneur intelligent) doit pouvoir interpréter correctement les messages échangés.

Pour les équipements que nous pouvons caractériser d'homogène (même constructeur) la solution est de type propriétaire. Dans un environnement d'équipements hétérogènes, les équipements de divers constructeurs doivent être compatibles entre eux. Il s'agit donc de disposer d'équipements capables de dialoguer entre eux et pouvant être substitués à des équipements provenant d'autres constructeurs, de même type et assurant les mêmes services (ou une partie).

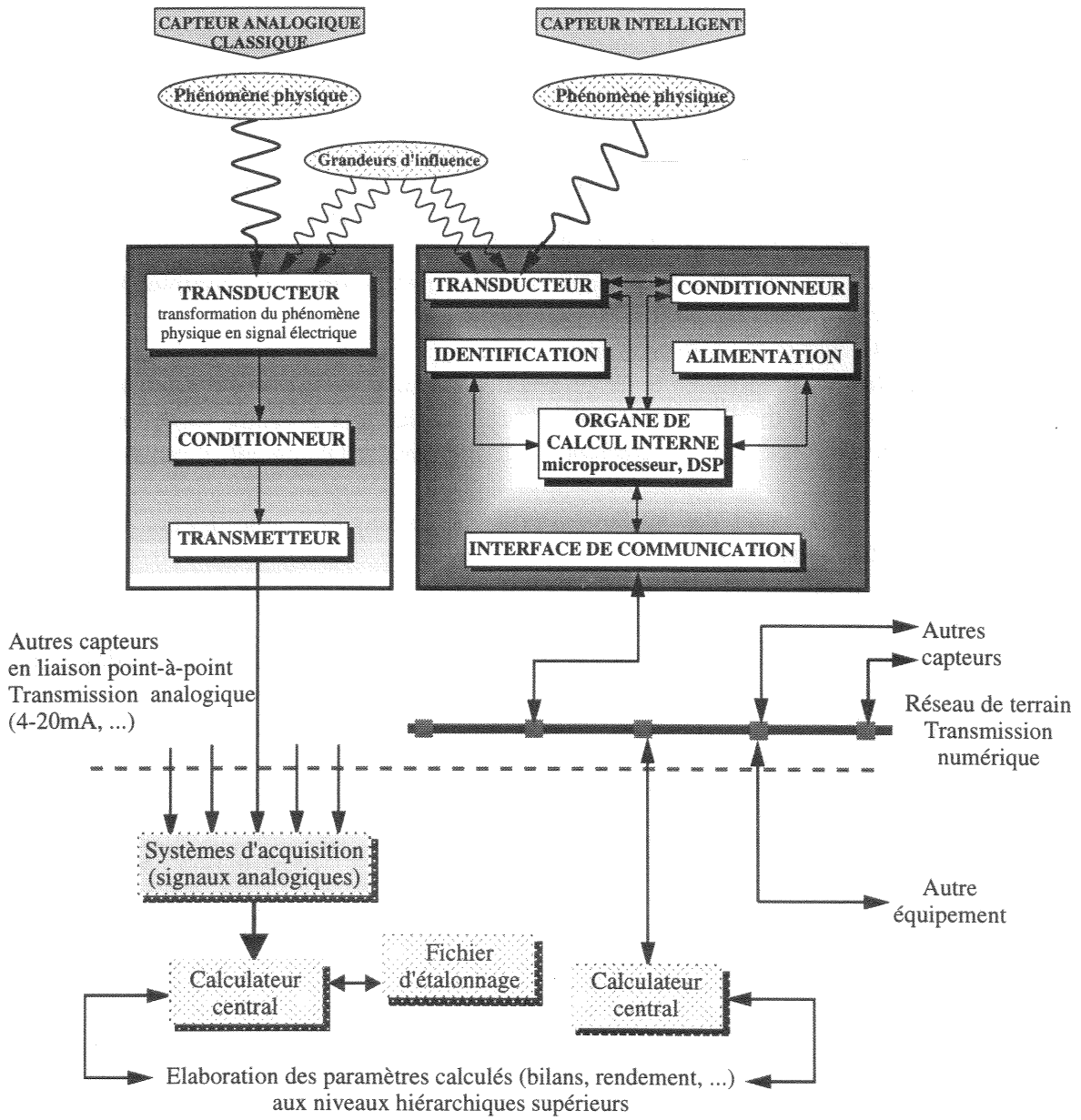


Figure I.6 : Comparaison des deux types de chaînes de mesures (analogique à gauche, numérique "intelligent" à droite)[BEAU-91].

Le passage de la mesure classique à la mesure intelligente résulte, dans un premier temps, du passage de l'instrumentation analogique à l'instrumentation numérique, et dans un second temps, des progrès faits dans le domaine de la micro-électronique et du génie logiciel. En instrumentation, l'utilisation des techniques numériques permet de numériser le signal au plus près de la source afin de diminuer les effets des perturbations en analogique [YOUS-93]. Le tableau ci-dessous résume les principaux avantages de la technique numérique.

Caractéristiques	Signaux analogiques	Signaux numériques
1. Perturbations		
- tension d'alimentation	*Influence sur la précision*	*Risque d'indisponibilité*
- vieillissement des composants	*Influence sur la précision*	*Peu d'influence sauf risque d'indisponibilités*
- environnement climatique	*Problème mieux maîtrisé*	*Plus exigeant (plus critique sur la durée de vie)*
- parasite électromagnétique	*Précautions classiques (blindage, ...)*	*Précautions classiques (blindages,..), filtrage logiciel passif*
	Détection difficile des défauts	*Détection facile des défauts (code détection)*
- défaut d'isolement galvanique	*Très grande sensibilité et dispositifs d'isolement complexes*	*Sensibilité moyenne et dispositifs d'isolement simples*
- types de pannes constatées	*Généralement dérives, erreurs, perte de précision*	*Généralement cessation de fonctionnement ou dispersion*
2. Métrologie		
- précision	*Moyenne due à la dérive des composants analogiques*	*Peut être excellente et fidèle-compromis avec la rapidité*
- bruit de quantification	*Rare*	*Existant-peut limiter la résolution et la précision*
3. Transmissions		
- multiplexage (transmission séquentielle)	*Difficile*	*Très aisée*
- transmission simultanée sur médium unique	*Difficile*	*Aisée*
- débit d'informations	*Mise à disposition en continu de toute l'information*	*Le débit peut être très élevé*
		Perte éventuelle d'information due à la fréquence d'échantillonnage
4. Divers		
- stockage d'informations	*Quasiment impossible*	*Très facile*
- consommation d'énergie	*Liée principalement au niveau d'énergie*	*Inférieur au cas de l'analogique*
		Varie avec la rapidité recherchée
- testabilité	*Tests simples*	*Tests simples, mais parfois longs*
- maintenabilité	*Moyenne*	*Peut être rendue plus aisée (auto tests, ...)*

(source CIAME - AFCET [CIAM-87])

I.3.1 Intégration du capteur intelligent dans les systèmes automatisés de production

Un système de production est constitué de l'ensemble des moyens destinés à l'élaboration de "produits" conformément à des objectifs économiques et techniques ; ces moyens sont : le processus, le système d'automatisation, et les opérateurs exploitants.

Le système d'automatisation ou SAP (Système Automatisé de Production) est composé à la fois d'éléments matériels et logiciels [ROBE-93a], à savoir :

- les dispositifs matériels et logiciels de traitement : capteurs, actionneurs, régulateurs, automates, ...
- les dispositifs matériels et logiciels de mémorisation : mémoires, disques, bandes, ...
- les dispositifs matériels et logiciels de communication : lignes, réseaux, ...

Un SAP peut être scindé, de manière fonctionnelle, en quatre sous-ensembles : système d'information, système de décision, système de communication, système d'application (figure I.7).

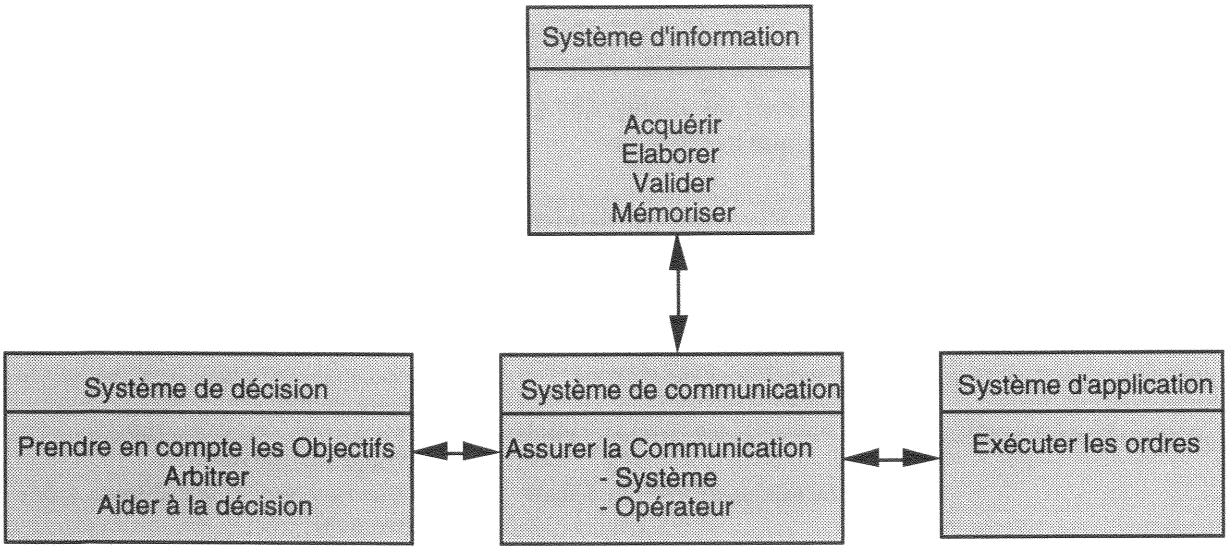


Figure I.7 : Décomposition d'un SAP en sous-systèmes.

Le système d'information acquiert, valide et mémorise l'ensemble des informations nécessaires au fonctionnement du SAP et gère la base de données validée, utilisée par le système de décision. Cette base de données contient des informations relatives aux divers constituants du SAP (base de données statique : identification des différents éléments constituants le SAP, ...) et des informations relatives au fonctionnement du SAP (constituant une base de données dynamique, représentative des variables physiques du processus et de son instrumentation ainsi que des informations concernant la disponibilité des constituants du SAP).

Le capteur est un des constituants du système d'information ; a minima, tout capteur est un producteur qui alimente une base de données, gérée par le système d'information, à partir d'informations caractérisant son environnement physique, émanant du processus et d'une source de temps.

Le système de décision décide du mode de fonctionnement du SAP et/ou seconde l'opérateur dans ce choix, en fonction de la stratégie et des objectifs émanant des niveaux supérieurs (conduite, maintenance, gestion technique, gestion de production).

Le système de communication garantit la circulation des flux d'information à l'intérieur du système ainsi que l'interfaçage avec l'extérieur ; le réseau de terrain et l'interface homme-machine sont des constituants du système de communication.

Le système d'application agit sur le procédé en fonction des consignes du système de décision.

I.3.2 La crédibilité des mesures

La crédibilité peut être définie comme la qualité qui caractérise l'aptitude d'un système à délivrer des informations, possédant un degré de confiance requis, dans ses conditions de fonctionnement. L'amélioration de la disponibilité, de la sécurité et de la rentabilité des installations industrielles passe par une crédibilité accrue des informations mises à la disposition des opérateurs dans les systèmes automatisés de production, et par l'exploitation optimale de ces données.

En ce qui concerne spécifiquement les capteurs, les signaux de mesure peuvent être affectés de divers types de défauts [ROBE-92] :

- des défauts propres au transducteur (dérive, détérioration, ...),

- des défauts dus à l'électronique associée (alimentation, dérives des composants, ...),
- des défauts liés aux conditions d'utilisation (effet des grandeurs d'influence, dépassement de l'étendue de mesure, temps de réponse, ...),
- des défauts de transmission des signaux (parasites).

Un des bénéfices apportés par les capteurs intelligents est de contribuer à l'amélioration de la crédibilité des mesures effectuées. Cette crédibilité est obtenue :

- d'une part, grâce à un ensemble de "tâches" internes visant à réaliser la validation de l'information (s'assurer que le matériel de mesure n'est pas défaillant) (figure I.8),
- et d'autre part, grâce à la mise à disposition d'informations sur l'état et l'environnement.

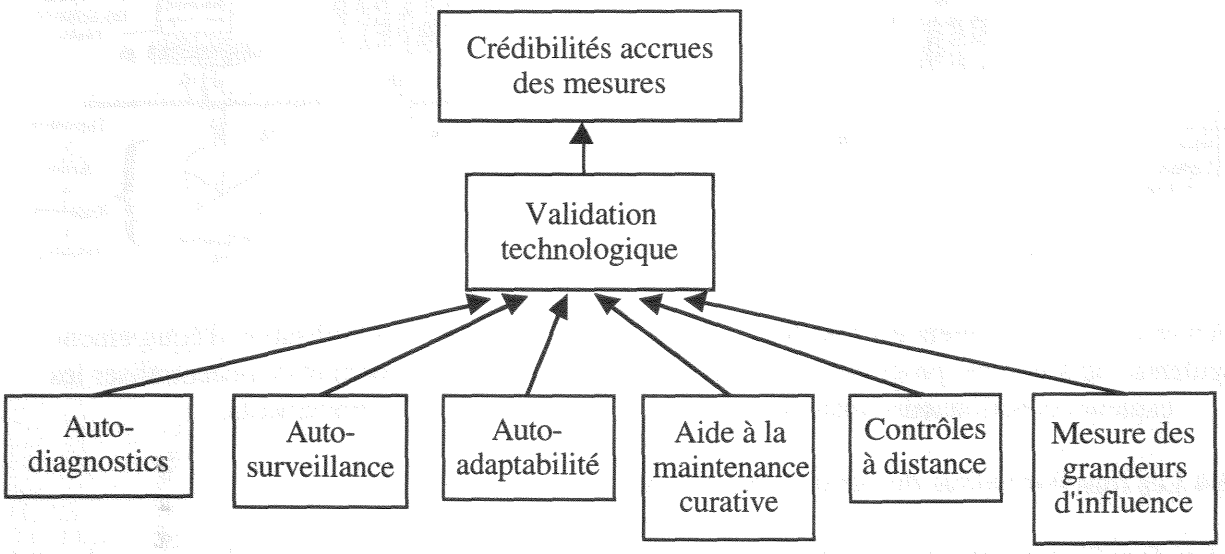


Figure I.8 : Crédibilité accrue des mesures par les nouvelles fonctionnalités d'un capteur intelligent [CIAM-87].

I.3.3 La communication

L'accroissement de l'intelligence des capteurs engendre des besoins de communication plus importants entre ces capteurs et le monde extérieur (automates, systèmes de maintenance, actionneurs, etc.)[PRAD-93].

Le besoin de communication s'accroît sur le plan qualitatif puisque les types d'informations à transmettre par un capteur intelligent peuvent être très variés (résultats de mesure, information d'états, messages de service, ...). En outre, le capteur intelligent est susceptible de recevoir des informations ou des demandes externes spécifiant tel ou tel type de fonctionnement, tel ou tel format de mesure. Le choix du système de communication à associer à ces capteurs est donc prépondérant. Les réseaux locaux industriels (plus particulièrement les réseaux de terrain) seront les outils et les supports permettant de réaliser cette fonction de communication, entre les capteurs, les actionneurs, etc.

Les figures I.9 et I.10 montrent la différence des besoins de communication entre les capteurs classiques et les capteurs intelligents [RIVI-95].

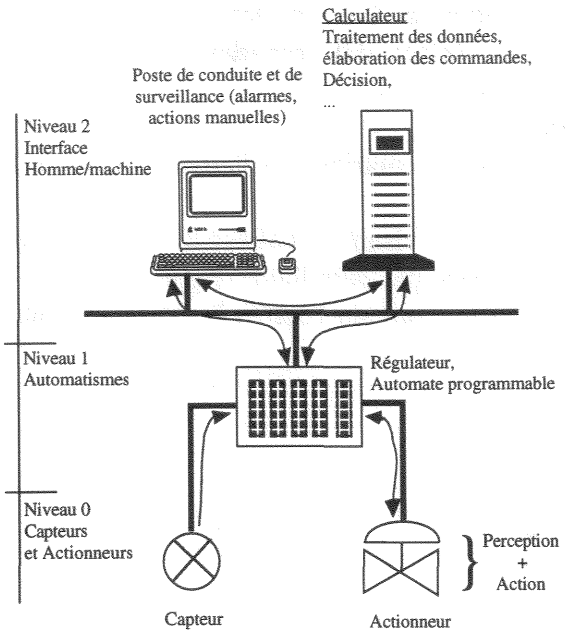


Figure I.9 : Le traitement est centralisé sur un régulateur ou automate programmable avec des capteurs et actionneurs analogiques.

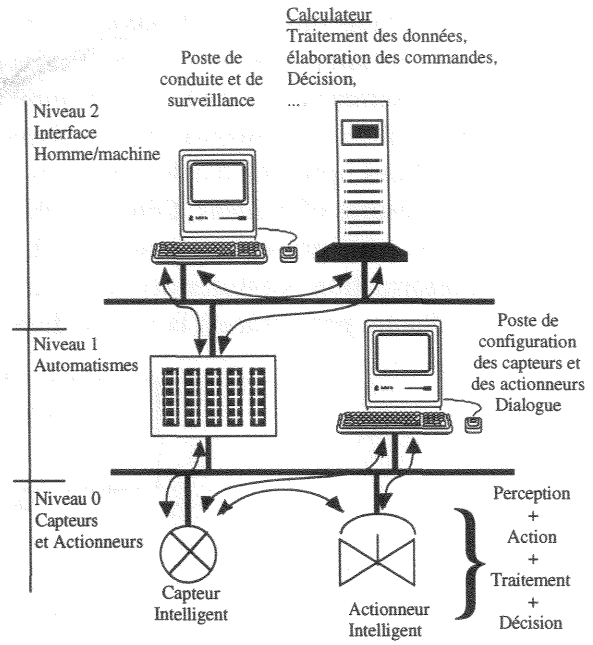


Figure I.10 : L'utilisation d'équipements intelligents permet de décentraliser les traitements.

I.3.4 Les fonctionnalités du capteur intelligent

Les fonctionnalités sont extraites du modèle CIAME provenant des travaux du groupe de travail "Fonctionnalités Capteurs" [CIAM-87]. Ce modèle s'appuie sur la décomposition initiale des fonctionnalités internes d'un capteur intelligent.

Pour l'utilisateur, le capteur intelligent peut être vu comme une machine délivrant des données métrologiques dont le contenu informationnel consiste en :

- la valeur de la mesure,
- le niveau de validité associé,
- le contenu temporel associé au phénomène physique observé et à l'information de mesure.

Toutes ces données permettent à l'utilisateur d'obtenir une mesure validée.

La figure I.11 présente les différentes fonctionnalités offertes par un capteur intelligent :

- la mesure,
- la configuration,
- la validation,
- la communication.

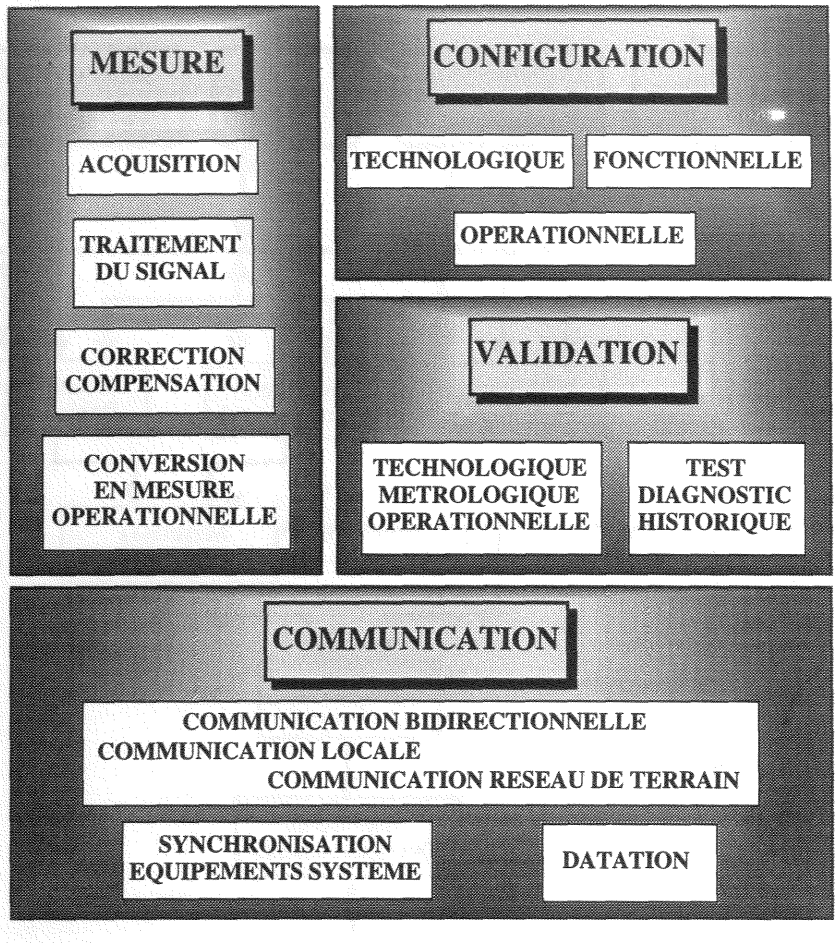


Figure I.11 : Fonctionnalités internes du capteur intelligent.

I.3.4.1 Fonctionnalité Mesure

La fonctionnalité Mesure intègre les aspects métrologique et traitement du signal liés au capteur intelligent [CHOQ-90].

Nous décrivons, figure I.12, la hiérarchie nécessaire à l'élaboration de la mesure opérationnelle, qui est une mesure validée directement exploitable par l'utilisateur.

Les grandeurs physiques sont réparties en trois classes :

- les grandeurs principales ou mesurandes,
- les grandeurs d'influence qui vont permettre de corriger la mesure en fonction des phénomènes physiques perturbants. Ces grandeurs sont mesurées, soit en interne, soit parviennent au capteur par l'intermédiaire d'un support de communication (exemples : température de l'appareil, pression ambiante),
- les variables et grandeurs d'auto-contrôle qui permettent de tester le bon fonctionnement du capteur pendant la mesure.

Toutes ces grandeurs peuvent être internes, locales ou distantes.

- Internes, c'est-à-dire produites par le capteur lui même, en particulier pour les variables d'auto-contrôle et toutes variables intermédiaires, localisées au sein du capteur;
- Locales, qui caractérisent l'environnement physique du capteur;
- Distantes, qui caractérisent l'environnement système du capteur intelligent (exemples : horloge système ainsi que toutes informations émises par tout autre capteur).

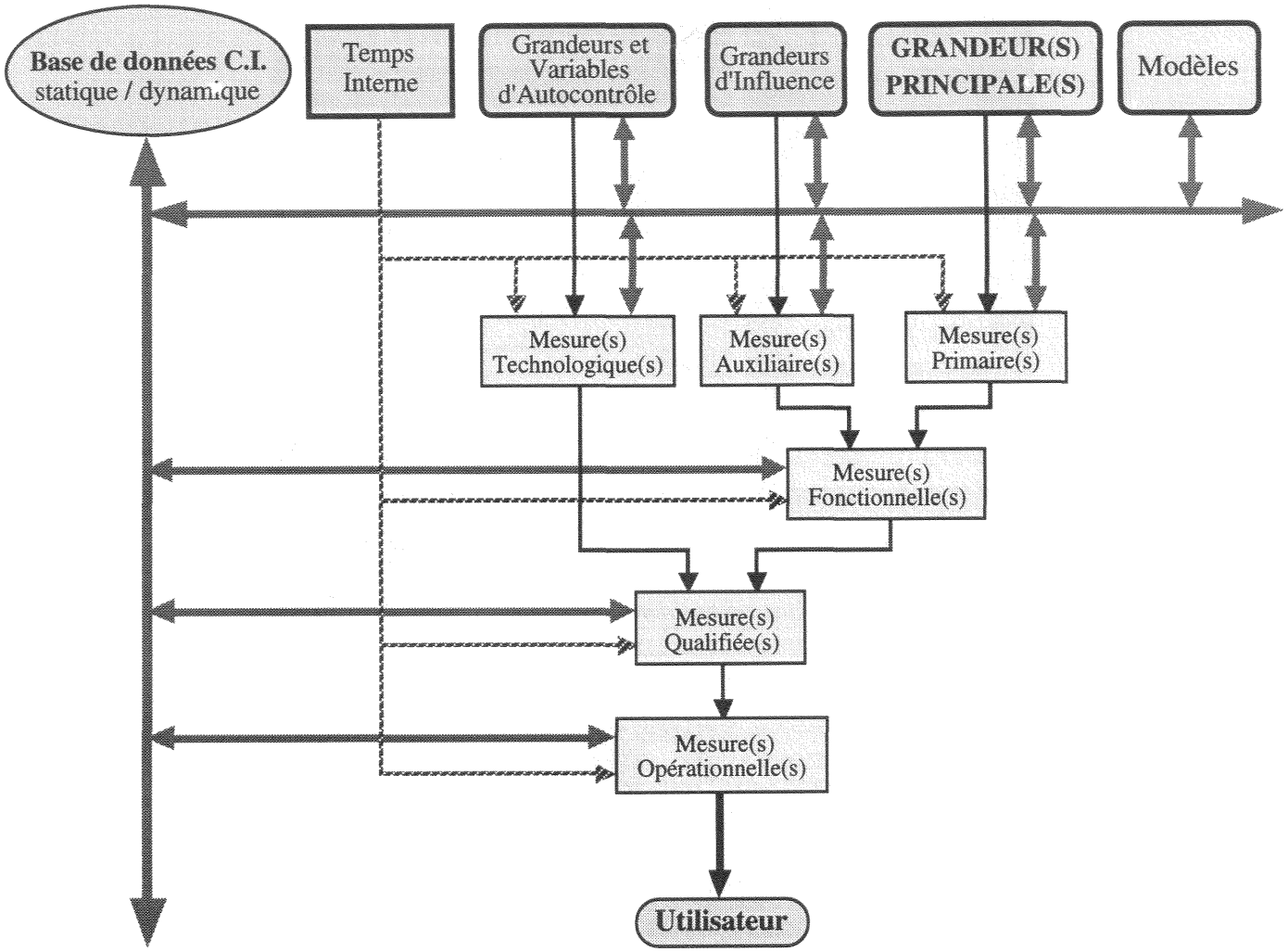


Figure I.12 : Hiérarchie conduisant à la mesure opérationnelle.

Ces informations concourent à la mise à disposition de l'utilisateur, d'une mesure opérationnelle obtenue à partir de la mesure qualifiée. Celle-ci est élaborée à partir des mesures fonctionnelles intégrant des mesures primaires représentatives des grandeurs principales, corrigées éventuellement par des mesures auxiliaires, images des grandeurs d'influence par l'intermédiaire de modèles mathématiques prenant en compte les différentes non-linéarités caractérisant les interactions des mesurandes et des corps d'épreuves.

A cette mesure fonctionnelle, vont s'ajouter des mesures technologiques qui permettent de garantir le bon fonctionnement de l'instrument lors de la prise de mesure.

Entre la mesure validée et la mesure opérationnelle s'opère une conversion qui consiste à transcrire les données du capteur dans un référentiel propre à l'usager.

Cette mesure, délivrée par le capteur intelligent par l'intermédiaire d'un support de communication, doit contenir également une information de datation, d'où la nécessité d'intégrer au capteur intelligent une gestion du temps synchronisable avec les horloges du monde extérieur.

I.3.4.2 Fonctionnalité Configuration

Cette fonctionnalité consiste à conformer le capteur aux conditions propres à une exploitation donnée : de ce point de vue le capteur intelligent bénéficiera des possibilités offertes par l'échange

d'informations avec l'opérateur. Via le système, il sera possible de fournir au capteur des données spécifiques, pour le conformer à une utilisation particulière.

La fonctionnalité configuration, activable par le constructeur, par l'installateur et l'utilisateur pendant des phases de vie différentes du capteur intelligent, inclut :

- la configuration technologique qui est le résultat d'un ensemble d'actions et leurs validations visant à intégrer le capteur dans son environnement physique. Généralement réalisée par le constructeur lors de la phase de fabrication, cette configuration est indispensable pour la mise en service du capteur intelligent,
- la configuration fonctionnelle qui est le résultat d'un ensemble d'actions et leurs validations visant à rendre le capteur mesurant et communiquant,
- la configuration opérationnelle qui est le résultat d'un ensemble d'actions et leurs validations visant à dédier le capteur à l'application.

Les configurations fonctionnelle et opérationnelle initiales peuvent être modifiées dans l'hypothèse d'une exploitation dégradée, résultant d'une défaillance détectée en interne ou peuvent également évoluer, à l'initiative de l'opérateur.

I.3.4.3 Fonctionnalité Validation

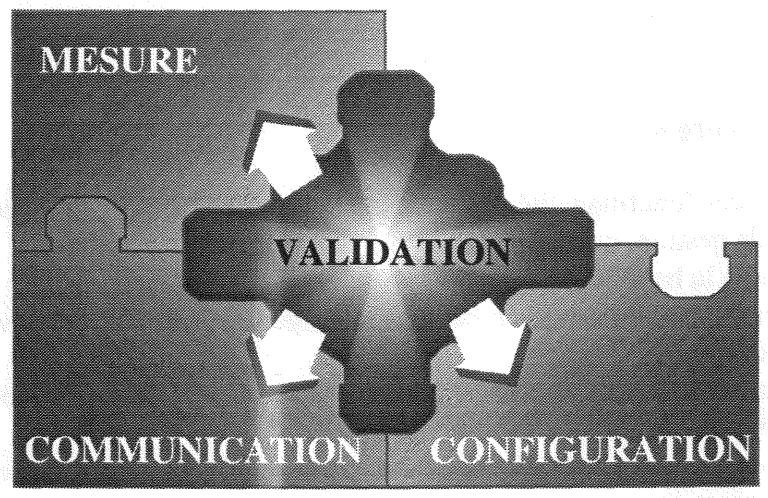


Figure I.13 : La fonctionnalité validation.

D'après [CIAM-87], cette fonctionnalité validation est indissociable des autres fonctionnalités (figure I.13), car elle est à la base du concept de capteur intelligent, puisqu'un des apports de l'intelligence est justement lié à l'accroissement de la crédibilité de la mesure, qui nécessite de s'interroger sur les conditions dans lesquelles la mesure a été réalisée.

Ainsi la mesure fonctionnelle doit être qualifiée par la prise en compte de mesures technologiques qui caractérisent le bon fonctionnement du capteur intelligent,

tant en terme de Technologie

par exemples :
- mesure de la tension d'alimentation,
- de la température de l'électronique,
- de l'intégrité de la chaîne d'acquisition,
- et vérification du bon déroulement d'un algorithme.

qu'en terme de Métrologie

par exemples :
- cohérence de la mesure vis à vis de la plage de mesure,
- respect des conditions de température du milieu ambiant.

La fonctionnalité validation peut être complétée par :

- la mise à jour systématique d'un historique stocké dans le capteur intelligent,
- le déclenchement de procédures de diagnostic, lorsqu'une défaillance a été détectée.

L'intégration de ces dispositifs de validation d'information à différents niveaux permet, bien entendu, de détecter en ligne les dysfonctionnements, d'en avertir l'utilisateur et d'envisager une exploitation du capteur intelligent en marche dégradée.

I.3.4.4 Fonctionnalité Communication

La fonctionnalité communication assure l'échange des informations entre le capteur intelligent et son environnement. Elle doit donc décoder et interpréter les ordres et messages qui parviennent au capteur intelligent, et en conséquence, déclencher les actions adéquates. Elle a également en charge la mise en forme des informations à transmettre du capteur intelligent vers le monde extérieur.

La fonction de communication pose, entre autres, la question de l'accès aux informations, notamment :

- la fréquence et le mode de communication des variables,
- la façon d'accéder aux variables en lecture et/ou en écriture,
- la façon de transmettre les variables,
- le moyen de dialogue avec l'opérateur.

I.3.4.5 Fonctionnalités complémentaires

En complément de ces fonctionnalités, il nous semble indispensable de proposer l'intégration de fonctionnalités dites de gestion, qui ont pour tâche d'assurer :

- la gestion de la base de donnée,
- la gestion du temps interne au capteur intelligent, synchronisable avec les horloges du monde extérieur,
- la gestion de la sûreté de fonctionnement qui se décline en : disponibilité, fiabilité, maintenabilité, sécurité, crédibilité, intégrité,
- la gestion de la qualité de l'équipement, par exemple, en terme de performances liées à la communication.

I.4 Conclusion

Nous avons dans ce chapitre exposé les principales différences entre capteur analogique classique et capteur intelligent. L'utilisation des techniques numériques et l'intégration d'une certaine "intelligence" au sein des capteurs permettent sans nul doute d'améliorer les caractéristiques métrologiques et de manière générale la précision ainsi que la crédibilité des mesures. Les nouvelles fonctionnalités intégrées donnent aux capteurs intelligents une place plus importante dans les systèmes automatisés de production.

L'utilisation des techniques numériques permet également d'établir entre le capteur intelligent et les autres constituants du système automatisé de production une communication bidirectionnelle facilitant ainsi la distribution des informations et des tâches au sein du système (vers les Systèmes automatisés de production à intelligence distribuée).

I.5 Références

[AFNO-88] AFNOR, "Maintenance industrielle", Recueil de normes françaises, Édition AFNOR, Paris, 1988, 3ème édition, pages 573.

[ASCH-91] Asch G.,"Les capteurs en instrumentation industrielle", Éditions DUNOD, 4ème édition, 1991.

[BEAU-91] Beaudouin F., Favennec J.M., "Les capteurs intelligents : le concept et les enjeux", SEE, les capteurs intelligents du concept aux applications industrielles, 6 et 7 novembre 1991, Annecy.

[CHOQ-90] Choquet M., Mans M. (de), Favenne C., Marchandiaux M., "Capteurs intelligents", Revue Générale de l'Électricité, N° 7, Juillet 1990, pages 26-29.

[CIAM-87] Livre Blanc "Les capteurs intelligents - Réflexions des utilisateurs", CIAME AFCET, 1987.

[DELM-93] Delmas G., "Crédibilité des systèmes de contrôle commande : concepts et outils", Revue Générale de l'Electricité, N°9, Octobre 1993, pages 23-28.

[GEHI-93] Géhin A.L., "Analyse fonctionnelle et modèle générique des capteurs intelligents - Application à la surveillance de l'anesthésie", Thèse, Université de Lille, 1993.

[HERM-92] Hermann F., "Conception d'un transmetteur intelligent de température", Rapport pratique, DEA Automatique et Traitement du Signal, Université de Nancy I, Octobre 1992.

[HOSS-90] Hossenlopp L., "Capteurs et architectures futures", Revue Générale de l'Électricité, N° 4, Avril 1990, pages 68-71.

[NOIZ-91] Noizette J.L., "Méthodologie de conception des capteurs intelligents application à un capteur granulométrique", Thèse doctorat en Automatique, Université de Nancy I, Novembre 1991.

[PARA-86] Paratte P.A, Robert P., "Systèmes de mesure", Édition Dunod, 1986, pages 370.

[PRAD-93] Pradenc H., "Des capteurs intelligents pour mieux communiquer", Revue L'Usine Nouvelle, N°2433, 18 novembre 1993, pages 87-90.

[PRIE-95] Prieur G., Nadi M., "La mesure et l'instrumentation : état de l'art et perspective", Édition MASSON, Paris, 1995, pages 726.

[RIVI-91] Rivière J.M., "Modélisation des fonctionnalités d'un capteur intelligent", Rapport pratique, DEA Métrologie, Automatique et Électrotechnique, Université de Nancy I, Juillet 1991.

[RIVI-93] Rivière J.M., Robert M., Hermann F. and Aubrun C., "Modelling of smart sensors : application to a smart temperature transmittert", Imeko TC4 Symposium Intelligent instrumentation for remote and on site measurement, May 12-13, Brussels, Belgium, pages 173-180, 1993.

[RIVI-95] Rivière J.M., "Contribution à la définition d'un modèle de référence de Capteur Intelligent : application à un Capteur Intelligent de Température", Thèse doctorat en Automatique, Université de Nancy I, Janvier 1995.

[ROBE-92] Robert M., "Contribution à l'évolution de l'instrumentation industrielle : capteurs intelligents et détection de défauts", Habilitation à diriger des recherches, Septembre 1992.

[ROBE-93a] Robert M., Marchandiaux M., Porte M., "Les capteurs intelligents et méthodologie d'évaluation", Éditions HERMES, 1993.

[ROBE-93b] Robert M., Rivière J.M., Noizette J.L., Hermann F., "Smart sensors in flexible manufacturing systems", Sensors and actuators, 1993, pages 39-46.

[ROBE-93c] Robert M., Rivière J.M., Theillol D. and Geoffroy P., "Smart or intelligent instruments in industry, past, present and future", IAR Kolloquim, November 16, Duisburg Germany, 1993.

[SELI-92] Selianov M.N.,"Définition of Metrologie, Quantity and Measurement", Traduit de Izmeritel'naya Teknica, N° 2, Février 1992, pages 11-15.

[SOLA-93] Solard G., "Des capteurs de plus en plus malins", Revue L'Usine Nouvelle, N° 2410, 13 Mai 1993.

[STAR-95] Staroswiecki M., "Algorithmes de détection de defaillances et de validation de données en ligne", Conférences capteurs : capteurs intelligents et sureté de fonctionnement, 5 octobre 1995.

[YOUS-93] Youssef W., Noizette J.L., Robert M., "Capteurs dits intelligents : gadget ou véritable besoin", Conférence CIAME - 5ème salon international des capteurs de mesure 93, 14 Janvier 1993.

Chapitre II

Les méthodes de développement de systèmes : des méthodes fonctionnelles à l'approche orientée objet

II.1 Introduction

Nous exposons dans ce chapitre les principales méthodes utilisées pour le développement de systèmes, ainsi que les différentes approches utilisées par ces méthodes (un système est un ensemble d'éléments en interaction dynamique, organisé en fonction d'un but déterminé ; nous entendons dans la suite qu'il pourra s'agir soit d'un système logiciel soit d'un système intégrant matériel et logiciel. Un système automatisé de production est un cas particulier de systèmes industriels ; Ce type de système est constitué d'un ensemble d'éléments matériels et logiciels et est en relation directe avec un processus pour en assurer la conduite). Dans ce chapitre, nous détaillerons en particulier l'approche orientée objet (cf. § II.2.5), qui est à la base de nos travaux pour la représentation du concept capteur intelligent.

Il est utile de faire la distinction entre les termes méthode et méthodologie [CALV-90]. Une méthode est un processus rigoureux permettant de générer un ensemble de modèles qui décrivent divers aspects d'un système en cours de développement, en utilisant une certaine notation bien définie. Une méthodologie est un ensemble de méthodes appliquées tout au long du cycle de développement, ces méthodes étant unifiées par une certaine approche "philosophique" générale. Le modèle est une abstraction de quelque chose de réel qui permet de comprendre avant de construire. Parce qu'il ne tient pas compte des éléments qui ne sont pas essentiels, le modèle est plus facile à manipuler que l'entité réelle originale. L'abstraction est l'une des capacités fondamentalement humaine qui permet de gérer la complexité. Pour construire des systèmes complexes, le développeur doit faire abstraction des différentes vues du système, construire des modèles en utilisant une notation précise, vérifier que les modèles satisfont aux spécifications du système, et progressivement ajouter des détails pour transformer les modèles en implémentations.

Les développements des applications en génie logiciel s'appuient traditionnellement sur une séparation entre les données et les traitements. Cette séparation a en effet le mérite de simplifier le développement des applications. Mais elle commence à trouver ses limites du fait de la taille et de la complexité croissante des logiciels [AUBE-91]. Dans le domaine du génie logiciel, les problèmes liés à la taille des logiciels sont de plusieurs ordres. Nous avons d'une part, en ce qui concerne le cycle de vie initial des logiciels, des problèmes de décomposition lors de la phase de conception et des problèmes de coordination lors de la phase de réalisation. Nous avons d'autre part, en ce qui concerne l'extension du cycle de vie des logiciels, des problèmes de maintenabilité, la maintenance représente plus de 70% du coût total d'un logiciel et des problèmes de réutilisabilité, moins de 15% des codes produits chaque année sont nouveau [MASI-90].

Les différents aspects concernant une entité sont éparpillés à travers le logiciel et leur localisation n'est pas toujours facile : en outre, la séparation données/traitements s'aggrave par la répartition des traitements. Du fait de cette séparation entre les traitements et les données, il est parfois difficile de faire évoluer les logiciels et de pouvoir réutiliser des développements. Si cette approche a prévalu jusqu'ici, c'est surtout parce que les langages de programmation existant permettaient de la mettre en oeuvre. Mais ce n'est pas pour autant qu'elle représente la meilleure adéquation entre le problème posé par l'utilisateur et sa transcription en informatique. L'utilisateur perçoit dans la pratique des

entités ayant une certaine autonomie, qui se manifestent par des informations accessibles et des comportements, c'est à dire des objets. L'objet va plus loin qu'un simple regroupement de données et de traitements, il est davantage que la donnée : il est une représentation d'une portion cohérente de la réalité.

II.2 Les différentes approches et méthodes

De nombreux ouvrages tels que [JAUL-90] [SOMM-89] [COAD-91a] et publications telles que [MALL-89] [KELL-92] [MORE-89] présentent en détail les différentes approches existantes, ainsi que les différentes méthodes de développement de systèmes. Nous ne présentons dans ce paragraphe qu'une synthèse de ces approches et ne citons que les méthodes de développement les plus répandues.

On peut classer les méthodes existantes de développement de systèmes en deux catégories : les méthodes fonctions/données et les méthodes orientées objet [JACO-92]. Par méthodes fonctions/données, on désigne les méthodes qui se concentrent sur des fonctions et/ou des données de manière plus ou moins séparée. Les méthodes orientées objet quant à elles considèrent que fonctions et données sont étroitement liées.

II.2.1 Approche orientée fonctions

II.2.1.1 La décomposition fonctionnelle

La décomposition fonctionnelle peut se définir par un ensemble de fonctions, de sous-fonctions, et d'interfaces fonctionnelles.

La démarche de base de la décomposition fonctionnelle consiste à sélectionner des étapes et sous-étapes de traitement anticipées pour un nouveau système [COAD-91a]. L'accent est mis sur la recherche des traitements requis pour le nouveau système. Le concepteur spécifie alors les traitements et les interfaces fonctionnelles. La spécification résultant d'une décomposition fonctionnelle (fonctions et sous-fonctions) correspond indirectement au domaine du problème. Rien ne fait explicitement la correspondance entre la fonctionnalité et le domaine lui-même. Cette approche permet difficilement à l'analyste et au client d'évaluer si oui ou non les besoins constituent un exposé exact de ce que le nouveau système doit faire. Avec une telle approche, la compréhension du domaine du problème n'est ni explicitement exprimée, ni vérifiée dans son exactitude; des aspects subtils du domaine du problème sont tout simplement non couverts.

Les décompositions en fonctions/sous-fonctions sont difficiles à construire (à cause de la correspondance indirecte avec le domaine du problème) et hautement volatiles (à cause du changement continuel des fonctionnalités). Pour ces raisons, nous avons le sentiment que l'approche globale de développement ne devrait pas être basée sur la décomposition fonctions/sous-fonctions; il est nécessaire d'avoir un point de vue plus "stable" de l'analyse et une organisation des spécifications.

II.2.1.2 La décomposition par flots de données

Une autre approche est la décomposition par les flots de données [COAD-91a]. Cette approche regroupe des Flots de données et de contrôle, des processus fonctionnels et de contrôle, des unités de stockage de données, des terminaisons, ainsi qu'un dictionnaire de données.

C'est une approche descendante par raffinage successifs des traitements, jusqu'à ce que tous les traitements aient été décrits en termes de flots de données.

Avec cette approche, le concepteur représente le monde réel avec des flots de données (qui transportent les données) et des traitements (qui transforment les données). Cette représentation

requiert de la part du concepteur (et de manière plus conséquente, du client) une analyse des flots de données. Toutefois, cette approche ne correspond pas à une approche intuitive. Considérer un événement et ensuite identifier les étapes de traitement qui sont mises en oeuvre en réponse à cet événement est tout à fait utile; c'est, en fait, une des démarches de l'approche orientée objet (cf. § II.2.5) pour identifier les services et les étapes dans un service. Mais ce flot décrit des étapes de traitement et non pas le flot de données et son affinement graduel est une démarche de pensée qui mène à un abstraction procédurale.

Les méthodes d'analyse dites "structurée" définies par Yourdon et DeMarco 1978-79 [DEMA-79] et celle de Gane et Sarson 1979 [GANE-79] représentent le système concerné en diagrammes de flots de données. Les problèmes avec ces démarches devinrent apparents dès que les équipes furent prises au piège, les unes après les autres, quand elles essayèrent de modéliser le système considéré, pendant que le délai, les budgets et la patience s'épuisaient. Les concepteurs ne surent tout simplement pas quand s'arrêter et dans le doute, continuèrent à enrichir le modèle. McMenamin a observé que de nombreux analystes s'enfoncèrent dans les "sables mouvants" des détails physiques, sans jamais en sortir [MCME-84].

Une autre démarche d'analyse structurée dite "moderne" est apparue dès 1982 et plus récemment développée par Yourdon 1989 [YOUR-89]. Cette démarche a été étendue, afin de couvrir la phase de conception (SASD : structured analysis and system design)[PAGE-88]. De plus, les adaptations de Ward ont permis de développer une méthode temps réel (SART) [WARD-86] [HATL-90].

L'approche flots de données met encore plus fortement l'accent sur le côté fonctionnel. Elle est sujette, de ce fait, aux mêmes faiblesses que la décomposition fonctionnelle. L'approche flots de données met très faiblement l'accent sur la structure des données. Le diagramme de flots de données donne très peu d'importance au stockage de données et cette faiblesse est justement reconnue par de nombreux auteurs qui décrivent la méthode des flots de données.

II.2.2 Approche orientée données

Ces modèles constituent incontestablement un pas en avant par rapport à la famille précédente dans la mesure où ils reconnaissent la nécessité d'expliciter des niveaux d'abstraction dans la représentation. On peut considérer qu'ils forment la base des méthodes de seconde génération dites méthodes conceptuelles [COAD-91a].

La modélisation de l'information s'est développée depuis de nombreuses années. Le principal outil de cette modélisation est le diagramme entité-association définie par Peter Chen en 1976 [CHEN-76](Objets + attributs + association ou relations).

Depuis le milieu des années 1970, la discipline de la modélisation de l'information a utilisé le terme objet d'une manière très peu standard. Avec cette démarche, la modélisation de l'information représente directement le domaine du problème en objets du modèle. Cette amélioration dans la représentation est importante, mais une représentation plus détaillée est nécessaire.

La modélisation de l'information est une méthode partielle. En effet elle ne traite pas les concepts suivants :

- service : le comportement, encapsulé et traité avec les attributs comme un tout intrinsèque.
- héritage : une représentation explicite des éléments communs pour les attributs et les services.
- structure : les liens de généralisation-spécialisation et de composé-composants comme méthodes de base humaines d'organisation, ne sont pas au centre du problème.

II.2.3 Approche orientée comportement

Bien qu'ils présentent l'avantage de tenter de représenter explicitement les contraintes de dynamique et d'évolution dans le temps, ces modèles privilégient ensuite soit l'aspect données (méthode Remora [ROLLA-88]) soit l'aspect traitements (méthode JSD : Jackson system development [JACK-83] [SUTC-88]).

II.2.4 Approches combinées

Une combinaison de plusieurs approches citées précédemment permet de prendre en compte plusieurs aspects plus ou moins dissociés : données, traitements, ou comportement.

Citons par exemple, la méthode Merise [TARD-87] qui utilise un modèle de données largement inspiré du modèle entité-association de Chen [CHEN-76] ainsi qu'un modèle de traitements. Citons également la méthode SADT (Structured analysis and design technique) qui utilise des actigrammes très largement inspirés de la représentation des flots de données de l'analyse structurée et des datagrammes [MARC-88] [IGL-88].

Ces deux méthodes proposent de prendre en compte les deux aspects données et traitements mais de façon séparée. Merise, en particulier, conseille de faire effectuer les deux études par deux équipes séparées. SADT semble accorder une place secondaire au datagramme. L'approche multiple présente incontestablement un avantage puisqu'elle permet, par référence croisée, d'assurer avec bonne probabilité que le modèle soit complet et cohérent, mais encore faut-il prendre en compte tous les aspects.

La figure ci-dessous présente les principales méthodes citées précédemment suivant les trois aspects : fonctions, données, et comportement.

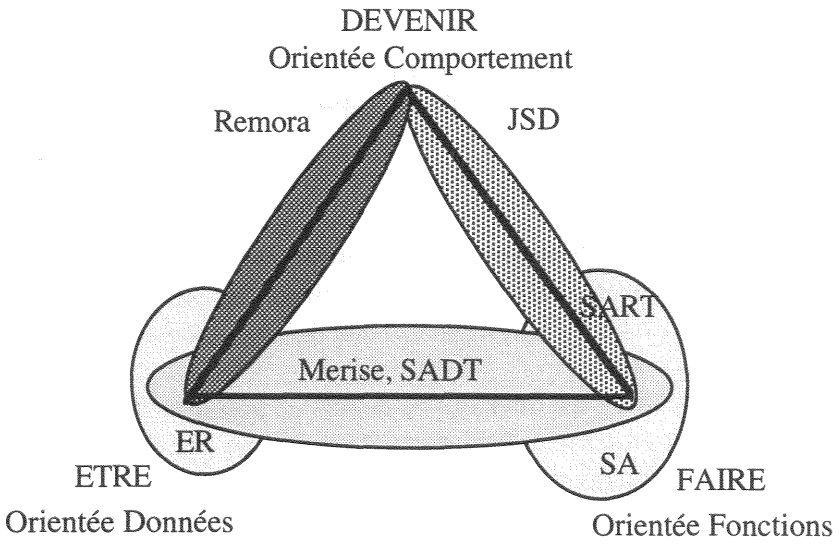


Figure II.1 : Classification des méthodes [MORA-91].

Tout en conservant à l'esprit le caractère quelque peu arbitraire d'une telle classification, il n'en reste pas moins qu'aucune des méthodes présentée n'occupe l'idéale position équilibrée au sein du triangle. Cet état de l'art méthodologique explique probablement qu'aujourd'hui, la plupart des ateliers de génie logiciel soit multi-méthodes.

Un certain retour à la systémique conduit à proposer une démarche de modélisation et de concepts réintégrant les trois aspects données, traitements et comportement dans un modèle unique du système d'information. Dans une telle approche, le paradigme orienté objets constitue à l'évidence un bon candidat.

II.2.5 Approche orientée objet

II.2.5.1 Introduction

L'orientation objet est une technique de développement qui permet de modéliser le système sous forme d'objets interagissant les uns avec les autres. Ainsi, quel que soit le type de système, on le considérera comme un ensemble d'objets reliés entre eux, d'une manière ou d'une autre.

Une approche de développement par objets encourage les développeurs à travailler et à penser en termes d'objet pendant la plus grande partie du cycle de vie (le concept d'objet est développé au paragraphe II.2.5.2). Ce n'est que lorsque les concepts inhérents à l'application sont identifiés, organisés et assimilés que les détails de structures de données et les fonctions peuvent être pris en compte efficacement.

L'intérêt pour l'approche orientée objet s'est accru lors des dernières années car elle permet :

- Une meilleure compréhension du système (la différence sémantique entre réalité et système est faible). Elle s'adapte tout à fait à la manière dont les gens perçoivent la réalité. C'est pourquoi les modèles orientés objet sont souvent faciles à comprendre;
- Une localisation des modifications apportées au modèle puisque ces modifications n'ont souvent un impact que sur un seul élément particulier du modèle, représenté par un seul objet.

L'orientation objet représente directement le domaine du problème dans un modèle. Au lieu d'une représentation indirecte du domaine du problème en fonctions/sous-fonctions ou en flots de données et traitements, la représentation du domaine du problème sur un modèle orienté objet est directe.

Cependant, dès que le projet atteint une taille importante, il devient nécessaire d'organiser les objets à plusieurs niveaux. On distingue deux composantes principales [ROSE-91] dans la relation entre objets permettant de fournir une base de structuration : la composition et la classification.

- La composition exprime le fait qu'un objet du monde réel est composé en général d'autres objets plus élémentaires (relation d'agrégation);
- La classification exprime la notion que des classes d'objets peuvent elles-mêmes faire partie de classes plus vastes. Un objet d'une sous-classe partage les propriétés de la classe parente : on dit qu'il hérite de ces propriétés (relation d'héritage).

L'ossature conceptuelle de l'approche orientée objet est le modèle objet. Celui-ci décrit la structure des objets dans un système : leur identité, leurs relations avec les autres objets, leurs attributs, et leurs opérations.

Un modèle doit utiliser les trois éléments suivant pour posséder la dénomination "orientée objet" [AUBE-91] :

- L'abstraction qui fait ressortir les caractéristiques essentielles d'un objet (qui le distinguent de tous les autres genres d'objets) et donc procure des frontières conceptuelles rigoureusement définies, relativement au point de vue de l'observateur. L'abstraction se concentre sur la vue externe d'un objet et donc sert à séparer le comportement essentiel d'un objet de son implémentation;
- L'encapsulation qui vise à la protection des données contenues dans les objets par la minimalisation et le contrôle des échanges entre objets. Par ce moyen, une réelle modularité est envisageable. L'objectif de l'encapsulation est de masquer l'implémentation des structures au programmeur qui les utilisent en fonction de leurs spécifications;
- L'héritage qui signifie que l'on peut développer une nouvelle classe d'objets en précisant simplement en quoi elle diffère d'une autre classe existante. La nouvelle classe hérite alors de la classe existante. Le principal avantage de ce mécanisme est de permettre la

réutilisation intensive des classes existantes. On place les classes les plus générales en haut de la hiérarchie d'héritage, tandis que l'on place en bas les classes les plus spécialisées.

Ces concepts seront développés plus en détail dans le paragraphe suivant.

II.2.5.2 Le concept d'objet

Le concept d'objet est apparu avec le langage Simula 67 à la fin des années 60. Il a ensuite évolué progressivement jusqu'aux différentes versions de Smalltalk. Ce dernier est a l'origine de la popularité des langages à objets [GIAM-91].

Même si dans le monde de l'orienté objet, les évolutions les plus visibles sont liées aux langages de programmation, le développement par objets n'en est pas moins un processus conceptuel indépendant des langages de programmation jusqu'aux stades ultimes. Ce type de développement est fondamentalement une nouvelle manière de penser, et non pas une technique de programmation.

Dans cette partie, nous présentons les principes fondamentaux de la technologie orientée objet inspirés des travaux de [ROSE-91] [BOOC-86] [MASI-90] [BOUC-94] [SNYD-93] [WASS-89] [REWI-95] {URL-OO1}.

L'abstraction

L'utilisation de l'abstraction pendant l'analyse signifie que l'on s'intéresse uniquement aux concepts du domaine d'application, en ne prenant pas de décision de conception ou d'implémentation avant que le problème ne soit cerné. L'abstraction est donc l'examen sélectif de certains aspects d'un problème. Son objectif est d'isoler les aspects importants et de supprimer ceux qui ne le sont pas. Son but est de limiter l'univers afin de pouvoir agir. Dans la construction de modèles, on ne cherche pas la vérité absolue, mais l'adéquation à un but donné. Il n'y a pas de modèle correct unique d'une situation, seulement des modèles adéquats ou inadéquats [RUMB-91].

Qu'est ce qu'un objet ?

Un objet est la représentation d'une entité à laquelle sont associés un comportement et des informations, seules les opérations que l'objet peut effectuer sont visibles de l'extérieur (figure II.2).

Un objet est une entité disposant d'un état (dont la représentation est cachée) et d'un ensemble donné d'opérations permettant de manipuler cet état [SOMM-89]. L'état est représenté par l'ensemble des attributs de l'objet. Les opérations associées à un objet fournissent des services aux autres objets (clients). les clients font appel à ces services lorsqu'ils doivent effectuer certaines opérations.

Les objets communiquent en s'échangeant des messages qui déclenchent des opérations. Un message est composé de deux parties :

- le nom du service auquel l'objet fait appel,
- des copies de l'information de l'objet appelant, nécessaires à l'exécution du service demandé et, le cas échéant, le nom d'une donnée pour contenir le résultat de l'opération.

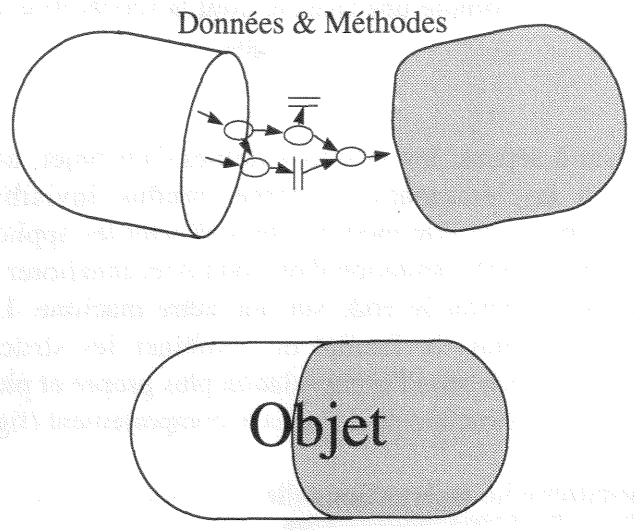


Figure II.2 : Objet encapsulé.

Chaque objet possède une interface bien définie qui spécifie quels stimuli il peut recevoir, c'est à dire quelles opérations il peut effectuer, car chaque stimulus reçu déclenche l'exécution d'une opération.

Les objets sont en relation les uns avec les autres et la dynamique d'ensemble du modèle émerge lorsque les objets commencent à interagir en envoyant des messages (figure II.3).

L'interaction entre les objets se fait à partir d'une communication par envoi de messages invoquant des méthodes (implémentation d'une opération pour une classe) déclarées dans les interfaces des objets.

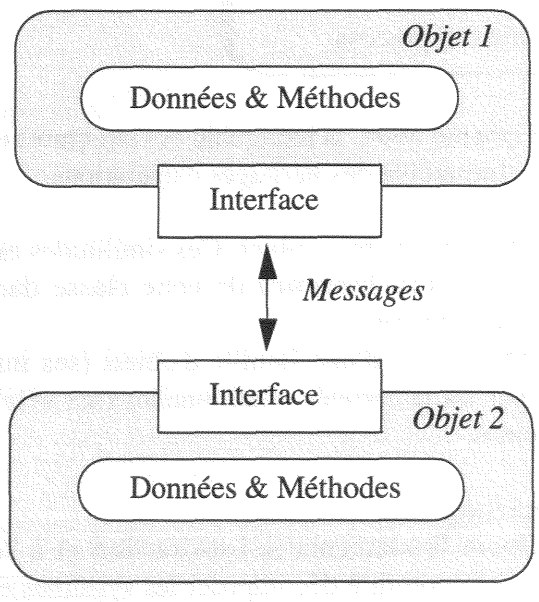


Figure II.3 : Communication entre les objets.

Un objet étant une entité indépendante dont la structure est connue de lui seul, il ne peut pas, en principe, agir directement sur un autre objet. Il doit alors utiliser une des méthodes appartenant à l'interface de cet autre objet et lui envoyer un message, qui demande l'exécution de la méthode en

question. Le message doit être vu comme une requête dont la satisfaction est à la charge de l'objet auquel elle est adressée.

L'encapsulation

L'**encapsulation** consiste à séparer les aspects externes d'un objet, accessibles par les autres objets, des détails de son implémentation interne, rendus invisibles aux autres objets. L'implémentation d'un objet peut alors être modifiée sans affecter les applications qui emploient cet objet. On peut vouloir modifier l'implémentation d'un objet pour améliorer la performance, corriger un bogue, consolider le code ou porter le code sur une autre machine. L'encapsulation n'est pas réservée aux langages à objet, mais la faculté de combiner les structures de données et le comportement dans une entité unique rend l'encapsulation plus propre et plus puissante que dans les langages traditionnels qui séparent structure de données et comportement (figure II.4).

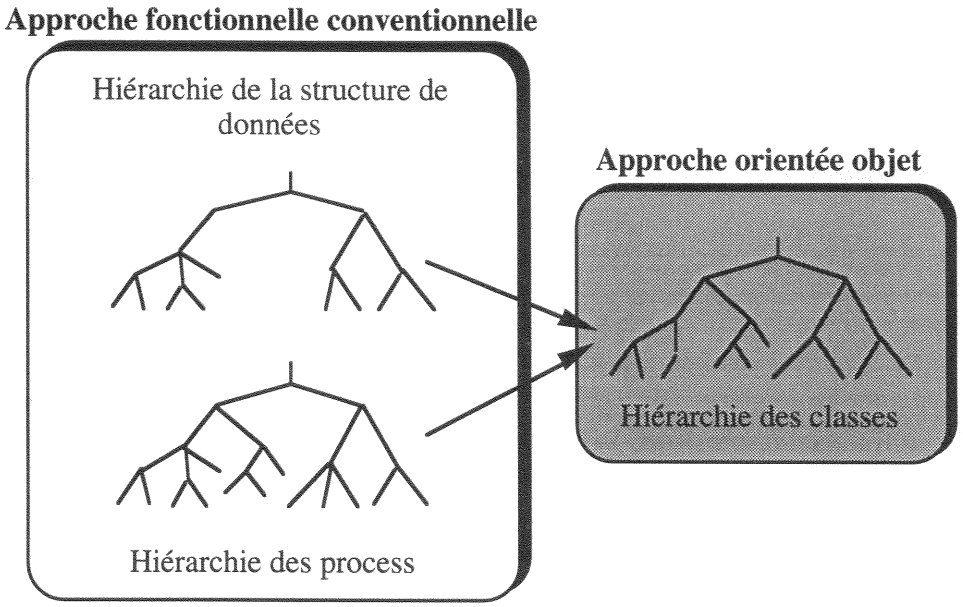


Figure II.4 : Dans un système orienté objet, la hiérarchie des structures de données est identique à la hiérarchie des héritages d'opérations.

Un certain nombre d'objets peuvent se ressembler. Ces similitudes entre objets sont décrites dans une **classe**. Chaque objet est alors une **instance** de cette classe dans laquelle sont définis son comportement et sa structure d'information.

Une classe est une entité génératrice d'une famille d'objets (ses instances), dont elle définit la structure et le comportement par les propriétés relationnelles (ses **attributs**) et fonctionnelles (ses **méthodes**), qui lui sont associées.

L'héritage et le polymorphisme

L'**héritage** est un complément fondamental à l'abstraction et à l'encapsulation. En effet , la volonté de maîtriser la complexité conduit à décomposer les systèmes en minimalisant les échanges entre leurs composants. Cette décomposition peut conduire à des redondances qui sont onéreuses aussi bien lors de l'écriture, que lors de la maintenance. C'est pourquoi, grâce à la possibilité de factorisation, l'héritage contribue à l'éco-organisation des systèmes d'information [AUBE-91].

L'héritage est le mécanisme qui définit entre les classes une relation dans laquelle une classe partage la structure ou le comportement défini dans une ou plusieurs classes, c'est à dire la réutilisation d'association de propriétés entre classes. La relation d'héritage organise hiérarchiquement

les classes en graphe d'héritage. L'héritage est dit simple si une classe ne peut hériter directement que d'une classe ancêtre. Il est dit multiple si une classe peut hériter directement d'une ou plusieurs classes ancêtres.

Le mécanisme d'héritage peut entraîner des cas de surcharge ou des cas de conflit. On parlera de surcharge si une propriété définie sur une classe est surchargée par une propriété de même nom définie sur une sous-classe. On parlera de conflit si une classe hérite de deux classes sur lesquelles deux propriétés de même nom sont définies et qu'il y a conflit pour déterminer quelle définition est à prendre en considération.

On utilise également le **polymorphisme** pour rendre plus souple la description des classes. Cela signifie que lorsque l'on définit les relations entre les objets, un objet n'a pas besoin de connaître exactement la classe de l'autre. On peut ainsi introduire de nouvelles classes sans avoir besoin de modifier les autres classes.

Le polymorphisme est donc la capacité à obtenir plusieurs réactions à une sollicitation donnée. Cela signifie qu'il est possible de définir plusieurs propriétés de même nom sur des classes différentes.

C'est une caractéristique essentielle des langages à objets par la modularité et l'extensibilité qu'elle offre. Cette caractéristique est la conséquence directe de la communication par envoi de messages. En effet, à un message donné, il pourra être associé plusieurs propriétés en fonction de l'objet qui reçoit le message. Cela signifie qu'il est possible de définir plusieurs propriétés de même nom sur des classes différentes.

Le polymorphisme permet donc d'envoyer le même message à des objets issus de différentes classes. Les objets acquièrent ainsi la capacité de répondre à une requête suivant des règles de comportement qui sont propres à leur classe, tandis qu'un message commun peut être utilisé en conjonction avec plusieurs objets de natures différentes. Cette technique porte exclusivement sur les méthodes contenues dans les classes et se comporte à leur encontre à la manière d'un sélecteur [PIGO-91].

II.3 Avantages et inconvénients de l'approche objet

II.3.1 Différences avec les méthodes fonctionnelles

La plupart des méthodes de génie logiciel traditionnelles, telles que SADT et SASD, sont des méthodes fonctions/données. Dans ces méthodes, l'accent est mis principalement sur la spécification et la décomposition des fonctionnalités du système. Une telle approche pourrait sembler la manière la plus directe d'implémenter le but souhaité, mais un système s'appuyant sur une décomposition des fonctionnalités peut nécessiter une restructuration importante lorsque les besoins évoluent. Par opposition, l'approche orientée objet se concentre d'abord sur l'identification des objets du domaine d'application, puis associe les procédures à ces objets. Bien que cela puisse sembler moins direct, un système conçu par une approche orientée objet ne nécessite que de faibles modifications localisées lorsque les besoins évoluent, puisqu'il s'appuie sur la structure sous-jacente du domaine d'application lui-même, plutôt que sur les besoins fonctionnels liés à un seul problème [RUMB-91].

En effet, un système développé à l'aide d'une méthode fonctions/données devient souvent difficile à maintenir. Un problème de fond des méthodes fonctions/données vient du fait qu'en principe, toutes les fonctions doivent connaître comment les données sont stockées, c'est à dire leur structure de données. Si l'on souhaite modifier une structure de données, il faut modifier toutes les fonctions relatives à cette structure. Les systèmes développés à l'aide de telles méthodes deviennent donc souvent "instables" (une infime modification pourra avoir de terribles conséquences)[JACO-92].

Un autre problème des méthodes fonctions/données, est lié au fait que la manière dont on structure le système n'est pas naturelle. La spécification des besoins est formulée d'habitude en langage naturel. Elle décrit souvent le QUOI : ce que le système doit effectuer, de quelles fonctionnalités il doit

disposer, et quels éléments il doit comporter. Puis, on reformule souvent cela en terme de COMMENT au moment de la division fonctionnelle. On obtient une différence sémantique importante entre les vues internes et externes du système. Alors qu'avec une méthode basée sur l'approche orienté objet, la structure d'implémentation du système sera équivalente à celle du modèle issu de la spécification des besoins. Les méthodes orientées objet structurent les systèmes sur la base des éléments qui existent dans le domaine du problème. C'est souvent une manière plus naturelle de décrire le système. Ces objets naturels issus du domaine du problème seront stables, il en résultera que la structure globale du système devrait être en générale assez stable [COAD-91a].

II.3.2 Les avantages de l'orientation objet

D'après, [COAD-91a], l'approche orientée objet permet :

- De mieux aborder les questions du domaine du problème. Pour certaines classes du système, il existe une correspondance claire entre les entités du monde réel et les objets du système qui les contrôlent. Cela permet d'améliorer la compréhensibilité du système modélisé. En effet, le développement par objet peut servir de moyen pour la spécification, l'analyse, la documentation et l'interface avec le client, autant que pour la programmation.

- D'améliorer l'interaction entre l'expert du domaine du problème et l'analyste. L'orientation objet organise l'analyse et la spécification en utilisant les méthodes d'organisation qui dominent la pensée des individus.

- D'augmenter la cohérence interne des résultats de l'analyse et de réduire la séparation entre les différentes activités d'analyse, en traitant les attributs et les services comme un tout intrinsèque.

- De représenter explicitement les éléments communs en utilisant l'héritage pour identifier et capitaliser sur les éléments communs des attributs et des services.

- De construire des spécifications résiliantes au changement. Les aspects les plus stables d'un système, ceux qui sont le moins susceptibles de changer potentiellement, sont les classes qui décrivent strictement le domaine du problème et les responsabilités du problème qu'il contient. A la base de l'orientation objet, il y a l'idée d'utiliser l'encapsulation pour cacher les éléments les plus volatiles (services et attributs) ; ceci garantit une stabilité lors des changements des besoins et lors de la construction de systèmes similaires. Le système développé devrait être plus facile à maintenir du fait que les objets sont indépendants. Ils peuvent être considérés et modifiés comme des entités à part entière. Le fait de modifier l'implémentation d'un objet ou de lui ajouter des services ne devrait pas affecter les autres objets du système [SOMM-89]. Ceci conduit à des systèmes plus résistants aux modifications.

- De réutiliser les résultats des analyses précédentes. L'orientation objet organise les résultats basés sur les constructions du domaine du problème, pour une réutilisation immédiate et future. Les objets se révèlent être des composants réutilisables appropriés, du fait de leur indépendance. On peut ainsi développer des conceptions à l'aide d'objets créés dans de précédentes conceptions, et encourager la réutilisabilité de composants.

- De fournir une représentation de base cohérente pour l'analyse et la conception. L'orientation objet définit un continuum de représentation, pour transformer

systématiquement les résultats de l'analyse en une conception spécifique réduisant ainsi les risques de développement.

Un certain nombre d'études de cas confirment ces assertions; en particulier elles montrent que l'approche orientée objet peut réduire le temps de développement et la taille du code source résultant [SCHU-86]. D'après l'expérience, les tailles des codes des systèmes orientés objet sont couramment plus petites que les implémentations non orientées objets fonctionnellement équivalentes [BOOC-91].

II.3.3 Les inconvénients de l'approche orientée objet

Deux "zones à risques" ont été localisées. Elles sont relatives :
- aux coûts de démarrage,
- et aux performances.

Pour certains projets, les coûts de démarrage associés au développement orienté objets peuvent se révéler une barrière réelle envers l'adoption d'une méthode orientée objet. L'utilisation d'une telle technologie demande la capitalisation d'outils de développement logiciels. Si l'utilisation est première, en général il n'y a aucune base de logiciels spécifiques au domaine et réutilisables. Il faut un certain temps pour acquérir l'état d'esprit adapté au développement orienté objet, et cette nouvelle façon de penser doit être adoptée par les développeurs aussi bien que par les responsables.

De plus, il est souvent malaisé d'identifier les objets appropriés. Notre compréhension naturelle de beaucoup de systèmes est plutôt fonctionnelle et il est quelquefois difficile de s'adapter à une compréhension orientée objet [SOMM-89].

L'envoi de messages d'un objet à un autre dans un langage de programmation orienté objet a sûrement un impact sur les performances. Des études indiquent qu'une invocation de méthode peut prendre entre 1,75 et 2,5 fois le temps d'un simple appel de sous-programme [SIMO-88] [PASC-86].

D'autre part, l'accès à un attribut spécifique d'un objet requiert l'ajout d'une méthode dédiée. Cette pléthore de méthodes engendre une surabondance d'invocation de méthodes. L'invocation d'une méthode à un haut niveau d'abstraction entraîne habituellement une cascade d'invocations de méthodes (en phase d'exécution). Pour les applications dans lesquelles le temps est une ressource limitée, toutes ces invocations de méthodes peuvent être inacceptables.

Le dernier risque concernant la performance des systèmes orientés objets provient de l'allocation dynamique et de la destruction des objets, qui coûte en général plus de ressources de traitement. Par exemple, Russo et Kaplan ont établi que le temps d'exécution d'un programme C est souvent inférieur à celui du programme C++ fonctionnellement équivalent [RUSS-88].

II.4 La mise en oeuvre de l'approche objet

II.4.1 Les langages objets

Il existe aujourd'hui une multitude d'outils permettant la mise en oeuvre de ces techniques et, en particulier, les langages de programmation qui offrent les mécanismes nécessaires à la définition et à l'utilisation d'objets.

Devant la panoplie étendue de la terminologie utilisée, il nous semble important de proposer une classification de ces différents langages.

Wegner introduit trois degrés permettant de qualifier les divers langages qui se parent du qualificatif de langages à objets (figure II.5)[AUBE-91].

Le premier degré sera celui des langages qui, comme Self, ne connaissent que la définition d'objets. Le second degré sera celui des langages qui, comme Clu, ajoutent à la définition d'objets la définition de classes. Le dernier degré enfin sera celui des langages à objets qui introduisent, outre la définition d'objets et de classes, la notion d'héritage. Nous pouvons citer parmi ces langages à objets les plus connus, C++, Clos, Eiffel, Simula, Smalltalk.[HEND-94].

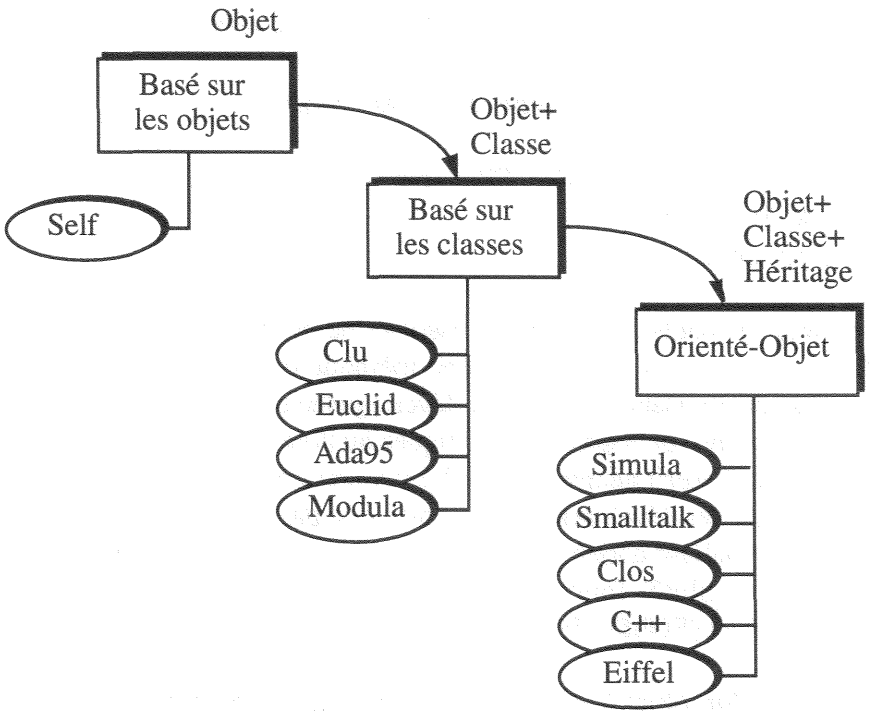


Figure II.5 : Degrés de Wegner [WEGN-88].

Seuls les langages intégrant les trois concepts (objet, classe et héritage) pourront, mériter valablement l'appellation de langages orienté objet. Un langage orienté objet doit donc au minimum étayer les concepts essentiels de l'orientation objet. Ainsi un langage orienté objet doit supporter l'encapsulation des objets, les concepts de classe et d'instance, l'héritage entre classes et le polymorphisme.

De plus, on trouve deux orientations principales dans le monde des langages à objets : celle qui privilégie la classification représentée par le langage Smalltalk et celle qui privilégie la classification, représentée par les langages à objets comme C++, Eiffel, Smalltalk et CLOS pour n'en citer que quelques uns [MEYE-90].

D'une façon générale, on peut dire que plus un langage favorise la classification et l'héritage, plus son comportement sera dynamique ; inversement, les langages qui privilégient la composition ont une approche plus statique et un typage plus fort. On peut donc répartir les différents langages à objets comme indiqué sur la figure II.6.

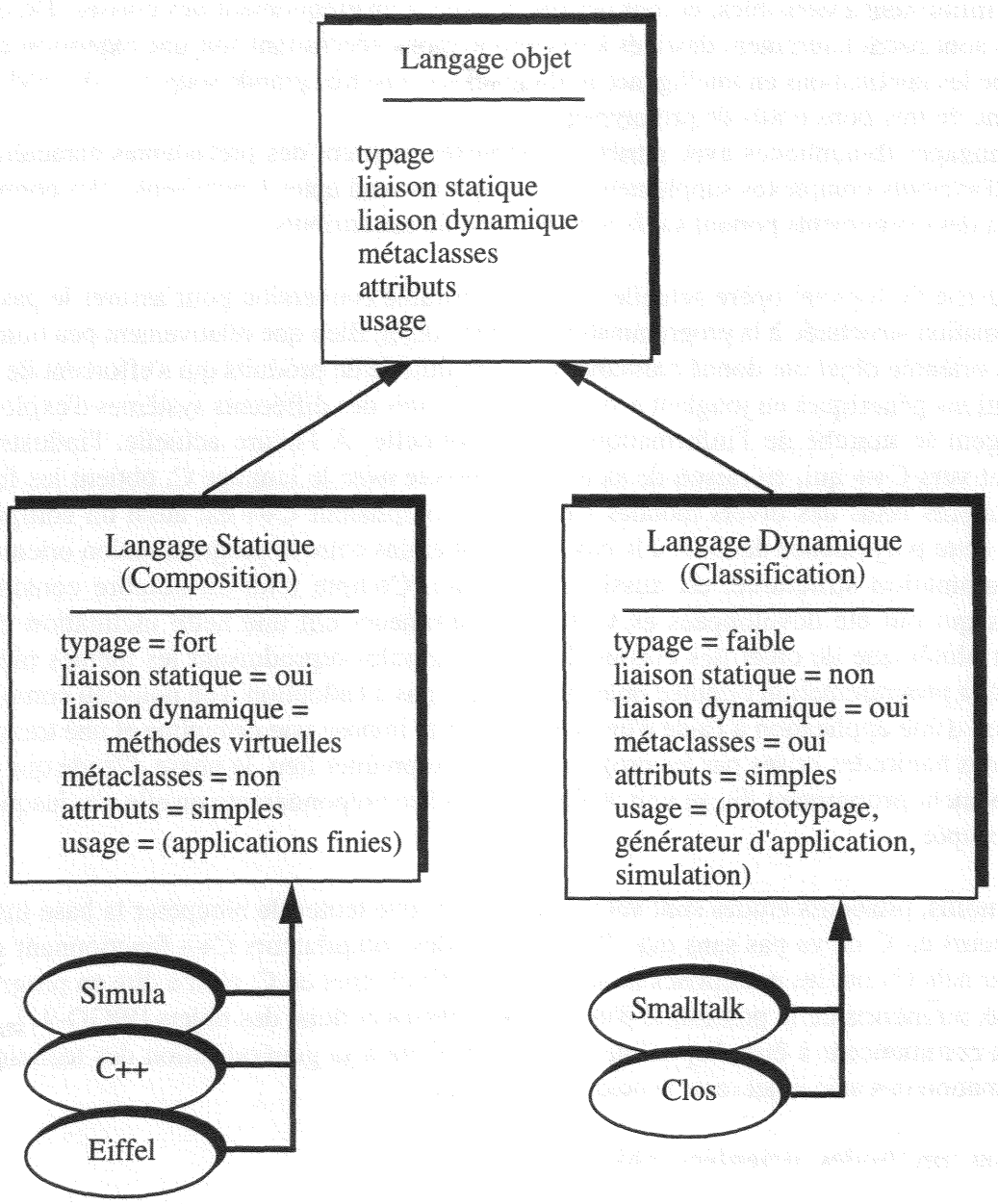


Figure II.6 : Répartition Composition-Classification [FERB-91].

Les langages orientés objets peuvent être classés selon leur usage ou selon leurs caractéristiques internes, ces deux notions ne sont pas sans relations : les caractéristiques des langages les rendent plus au moins aptes à servir telle ou telle utilisation [FERB-91].

Les langages statiques

Ces langages sont caractérisés par un typage fort, la possibilité d'avoir une liaison statique et une liaison dynamique par méthodes virtuelles. Ces langages sont essentiellement destinés à la réalisation d'applications finales et directement utilisables. Ils permettent ainsi de faire du génie logiciel dans le cadre d'une approche objet.

Les langages dynamiques

Ces langages disposent d'un typage faible, ce qui les rend plus malléables, mais aussi moins efficaces du fait de leur liaison purement dynamique. Cependant ils disposent de métaclasses, ce qui

les rend infiniment extensibles, et leur permet de créer dynamiquement des classes. De ce fait, ces langages sont particulièrement destinés à des applications nécessitant soit une extension du langage (telles que les applications en intelligence artificielle) soit une très grande souplesse de modification. Il s'agit donc de très bons outils de prototypage.

Les langages dynamiques avec attributs complexes, héritent des précédentes caractéristiques et dispose d'attributs complexes supplémentaires, ce qui les rend aptes à représenter des connaissances, et à agir à des événements portant sur la manipulation de ces attributs.

L'industrie du logiciel opère actuellement une véritable conversion pour assurer le passage de la programmation structurée à la programmation orientée objet. Bien que relativement peu nombreux, les langages orientée objet ont donné naissance à une multitude de produits qui s'efforcent de coller aux spécifications génériques en jonglant avec les particularités des différents systèmes d'exploitation qui se partagent le marché de l'informatique professionnelle. A l'heure actuelle, l'industrie penche nettement vers C++ qui, en raison de sa filiation directe avec le langage C, obtient les faveurs des programmeurs issus des divers mondes Unix (tout compilateur C++ est aussi un compilateur C). Pourtant cette polyvalence de C++, à la croisée des chemins entre la programmation orientée objet et la programmation structurée, est aussi un handicap. Compte tenu du nombre considérable de fonctions qui ont été développées en C, les programmeurs ont une nette inclination à exploiter l'existant plutôt que de créer des classes d'objets originales reproduisant les mêmes méthodes. le passage à la programmation orientée objet ne se limite pas à l'adoption d'un nouveau compilateur. La conception d'une application à l'aide d'un langage à objet impose une discipline et une transformation radicale des habitudes prises par les programmeurs. En premier lieu, la phase d'étude qui précède le développement proprement dit, se voit attribuer une place prépondérante qu'elle n'a, jusqu'à présent, jamais occupée.

Néanmoins, plusieurs études sont venues démontrer que tenter de récupérer la base installée des développeurs en C ne va pas sans mal. Près de 60% des compilateurs C++ fonctionnant sur micro-ordinateur achetés par les entreprises ne servent qu'à développer en C, c'est à dire en programmation structurée, au mépris de la possibilité d'utiliser des classes et donc des objets [PIGO-91]. Alors que les outils commencent à être disponibles, le dernier frein à la généralisation des techniques de la programmation orientée objet reste le poids des habitudes.

II.4.2 Les méthodes orientées objet

Actuellement, l'émergence des langages à objets ou orientés objet ouvre de nouvelles perpectives pour le développement logiciel, mais leur emploi déroute aussi bien les programmeurs peu expérimentés qu'expérimentés. Ainsi, il est devenu indispensable d'utiliser des méthodes, intégrant le concept d'objet qui permettent de réduire les difficultés rencontrées dans la pratique des langages objets et qui favorisent la production et la maintenance de composants logiciels de qualité.

Produire et maintenir un logiciel de qualité, en maîtrisant les coûts et les délais de développement, suppose l'utilisation d'une ou plusieurs méthodes pour les différentes phases du cycle de vie d'un système.

Cependant, l'utilisation d'une méthode nécessite de bien maîtriser ses domaines d'application, ses possibilités, mais également ses limites, ses difficultés de mise en oeuvre, etc.

Il est absolument nécessaire, si l'on veut qu'une méthode apporte les gains de productivité escomptés, de la choisir en fonction de critères tels que [JAUL-90] :

- les finalités et stratégies de l'entreprise,
- les acteurs concernés,
- le domaine d'application,
- les étapes du cycle de vie couvertes par la méthode,

- le niveau d'outillage de la méthode {URL-OO2}.

Ces méthodes aussi bien utilisées en analyse qu'en conception doivent utiliser un certain nombre de techniques qui permettent de couvrir tous les aspects de la conception d'un système.

La plupart des méthodes utilisent des techniques communes mais introduisent de nouveaux concepts de modélisation et de nouvelles notations.

Nous constatons qu'il est utile de modéliser un système à partir de trois points de vue liés mais distincts, chacun saisissant des aspects importants du système, tous nécessaires à une description complète.

Les différentes vues qu'une méthode de développement doit pouvoir décrire sont :

- la vue "informationnelle" ou structurelle d'un système qui se concentre sur la description des objets et des différentes sortes de relations statiques qui existent entre eux. Les différentes relations importantes sont les associations, l'héritage et l'agrégation;
- la vue comportementale qui décrit comment le système évolue. Cette vue est très importante pour la modélisation de systèmes temps réel;
- la vue architecturale qui décompose le système en sous-systèmes traditionnellement de façon fonctionnelle.

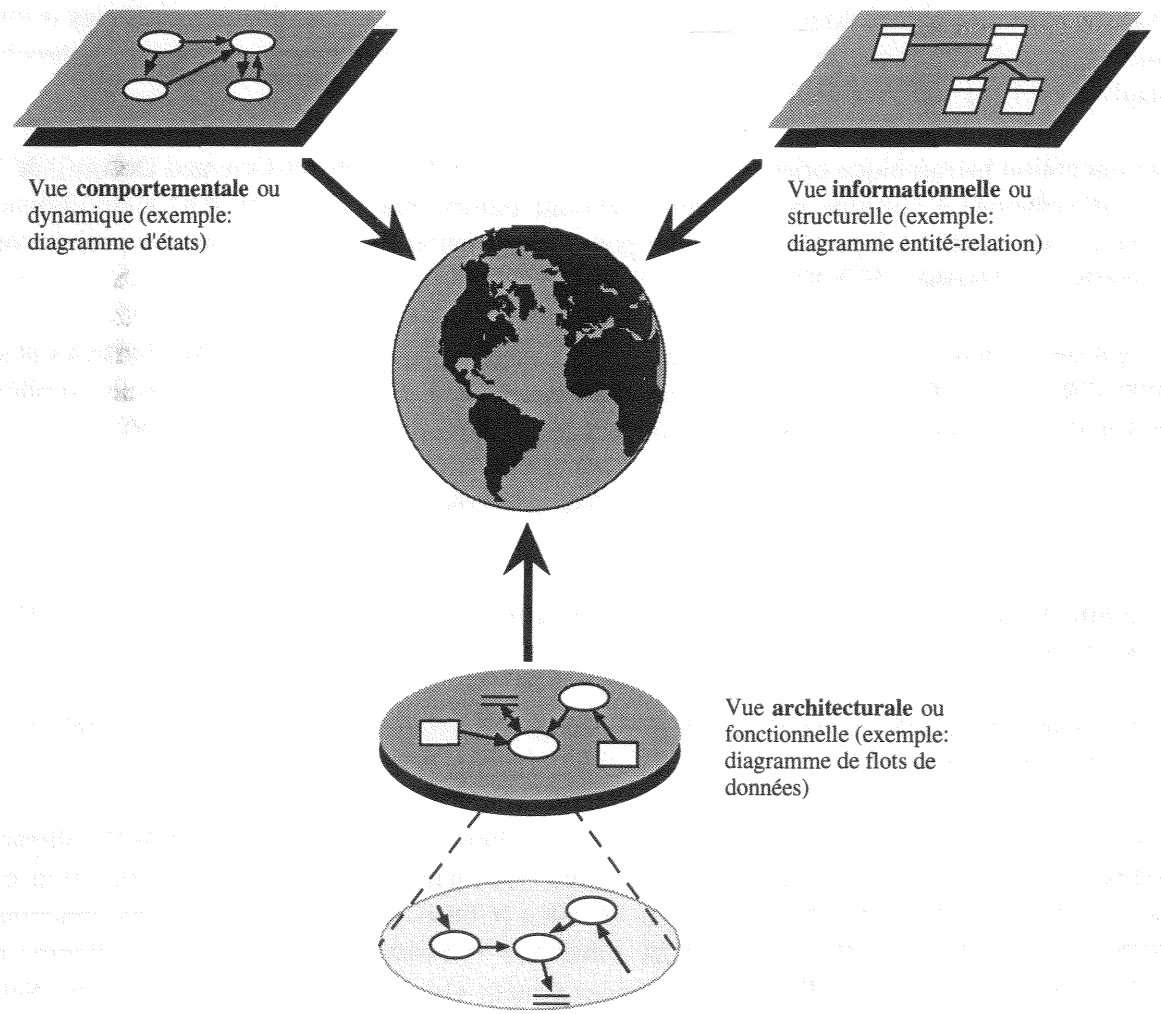


Figure II.7 : Les trois vues d'un système.

Il est naturel de s'attendre à une augmentation du nombre de méthodes du fait du fort intérêt que l'on porte actuellement aux techniques orientées objet {URL-OO3} {URL-OO4} {URL-OO5}

Actuellement, on trouve plus d'une cinquantaine de méthodes intégrant le concept d'objet (chaque année une dizaine de méthodes apparaissent) [GRAH-91] {URL-OO6} {URL-OO7}.

Les premières méthodologies utilisant la notion d'objet apparurent dès la fin des années 80, mais n'intégraient que partiellement le concept d'objet, c'est à dire qu'elles supportaient les objets mais pas l'héritage ni les classes. L'un des pionniers fut Booch dont la première méthodologie a évoluée en plusieurs étapes de 86-87 jusqu'à sa dernière version : conception orientée objet (OOD : Object-Oriented Design) [BOOC-91] [BOOC-93].

Parmi les autres méthodologies citons :

- L'analyse orientée objet des systèmes (OOSA : Object-Oriented System Analysis) de Shlaer et Mellor de 1988 [SHLA-88] [SHLA-90]. C'est essentiellement de l'analyse d'information basée sur la modélisation de données. OOSA ne réussit pas à saisir le comportement et ne contient ni héritage ni classification. Par contre elle se concentre sur la description des relations entre les objets.

- L'analyse orientée objet (OOA : Object-Oriented Analysis) de Coad et Yourdon de 1991 [COAD-91a] qui est une méthodologie incrémentale pour développer des modèles de systèmes orientées objet. Une méthodologie de conception (OOD : Object-Oriented Design) a suivi l'approche de l'analyse en 1991 [COAD-91b] [COAD-93].

- La conception hiérarchique orientée objet (HOOD : Hierarchical Object-Oriented Design), de 1989 a été développée à l'origine pour l'agence spatiale européenne (ESA). HOOD a été choisie par l'ASE comme méthodologie de conception pour la conception d'architecture sur des projets comme Colombus ou Hermès [HOOD-90].

- La conception basée sur les responsabilité (Responsability-Driven Design) de Wirfs-Broks et al. de 1990. CRC (Class Responsability Collaborator) est une méthodologie qui modélise une application en fonction des classes, de leurs responsabilités et de leurs collaborations [WIRF-90].

- La méthode de génie logiciel orientée objet OOSE simplifié de Objectory de Jacobson basée sur les cas d'utilisation [JACO-92].

- La méthode de Martin-Odell de 1992, conçue a l'origine pour les bases de données [MART-92] [MART-93].

- La technique de modélisation objet (OMT : Object Modeling Technique) de Rumbaugh de 1991 [RUMB-91] [RUMB-95].

D'autres méthodes ont vu le jour plus récemment; celles-ci intègrent ou reprennent la plupart des concepts des méthodes cités précédemment. Parmi ces méthodes de deuxième génération citons notamment la méthode FUSION de Coleman en 1994 [COLE-94] qui utilise certains concepts des méthodes comme OMT, CRC, et BOOCH, ainsi que la méthode de conception orientée objet spécialisée pour le temps réel (ROOM : Real-time Object-Oriented Modeling) proposé par Selic en 1994 [SELI-94]. Citons également la méthode unifiée de Rumbaugh, Booch ,et Jacobson de 1996 qui s'attache à définir une norme pour les méthodes orientée objet {URL-OO8}.

Des études comparatives des différentes méthodes peuvent permettre aux futurs utilisateurs de mieux connaître les méthodes existantes, en les aidant à faire un choix parmi le nombre sans cesse croissant de méthodes [FOWL-94a] [FOWL-94b] [FOWL-93] [CHAM-92] [MONA-92] [AFCE-93].

Y a-t-il une méthode meilleure que toutes les autres? Non, il n'y a pas de réponse absolue à cette question, qui n'est qu'une façon déguisée de demander : quelle est la meilleure façon de décomposer un système complexe? Autrement dit, nous trouvons un grand avantage à construire des modèles qui se concentrent sur les choses qui résident dans l'espace du problème, formant ainsi ce que nous appelons une décomposition orientée objets.

La méthode que nous avons retenue pour modéliser le concept du capteur intelligent est la méthode de Rumbaugh OMT (Object Modeling Techniques) dont les avantages sont présentés par la suite (chapitre IV).

II.5 Conclusion

Le bénéfice le plus important de l'approche objet ne provient pas d'une réduction du temps de développement ; le développement orienté objet peut en effet prendre plus de temps qu'un développement conventionnel, car il a vocation à permettre une future réutilisabilité et à réduire les erreurs en aval ainsi que l'effort de maintenance. Le temps nécessaire pour terminer le codage est probablement du même ordre que dans une approche conventionnelle, voire légèrement supérieur. Cependant, les itérations ultérieures d'un développement orienté objet sont plus simples et plus rapides qu'avec les techniques conventionnelles car les révisions sont plus localisées. Par ailleurs, en général, moins d'itérations se révèlent nécessaires car on découvre et on corrige plus de problèmes pendant le développement.

II.6 Référence

[AFCE-93] Actes des journées de synthèse "Méthodes d'analyse et de conception orientées objet des systèmes d'information", 22-23 novembre 1993, Groupe COOSI, AFCET, 450 pages.

[AUBE-91] Aubert J.P., Dixneuf P., "Conception et programmation par objets : techniques, outils et applications", Édition Masson, 1991, 172 pages.

[BOOC-86] Booch G., "Object-Oriented development", IEEE Trans. on software engineering, Vol. SE-12, N°2, pages 28-38, 1986.

[BOOC-91] Booch G., "Object-Oriented Design with Applications", Benjamin and Cummings, Redwood City, CA, 1991.

[BOOC-93] Booch G., "Object-Oriented Analysis and Design with Applications", (Second Edition), Benjamin and Cummings, Redwood City, CA, 1993.

[BOUC-94] Bouché M., "La démarche objet : concepts et outils", Édition AFNOR, 1994, 315 pages.

[CALV-90] Calvez J.P., "Spécification et conception des systèmes : une méthodologie", Édition MASON, Paris, 1990.

[CHAM-92] Champeaux D., Faure P., "A comparative study of object-oriented analysis methods", Journal of Object-Oriented Programming, Vol.5, N°1, March-April 1992.

[CHEN-76] Chen P.P.S., "The entity-relationship model : toward a unified view of data", ACM transactions on databese systems, Vol.1, N°1, March 1976, pp.9-36.

[COAD-91a] Coad P. and Yourdon E., "Object-Oriented Analysis", (Second Edition), Prentice-Hall, Englewood Cliffs, NJ, 1991, 200 pages.

[COAD-91b] Coad P. and Yourdon E., "Object-Oriented Design", Prentice-Hall, Englewood Cliffs, NJ, 1991, 200 pages.

[COAD-93] Coad P. and Nicola J., "Object-oriented programming", Prentice Hall, Englewood Cliffs, NJ, 1993.

[COLE-94] Coleman D. et al., "Object oriented development : the fusion method", Prentice-hall international editions, 1994, 314 pages.

[DEMA-79] DeMarco T., "Structured Analysis and Systems Specification", Englewood Cliffs, New Jersey : Prentice hall, 1979.

[FERB-91] Ferber J., "Conception et programmation par objets", 2nd édition, Édition Hermès, 1991, 112 pages.

[FOWL-93] Fowler M., "A comparison of object-oriented analysis and design methods", Approaches to object-oriented analysis and design, Carmichael A., Ashgate Edition, 1993.

[FOWL-94a] Fowler M., "Object-Oriented Methods : A comparative overview", Object-Oriented Analysis & Design, Finding your path, SIGS publications, 1994.

[FOWL-94b] Fowler M., "A comparison of object-oriented analysis and design methods", Tools Europe, Versaille (France), 1994.

[GANE-79] Gane C., Sarson T., "Structured Systems Analysis: Tolls and Techniques", Prentice-Hall, 1979.

[GIAM-91] Giambiasi N., Oussalah C., "Les langages à objets", Génie logiciel et systèmes experts, N°22, Mars 1991, pages 52-68.

[GRAH-91] Graham I., "Object Oriented Methods", Addison-Wesley Publishing Company, 1991.

[HATL-90] Hatley D.J., Pirbhai I.A., "Stratégies de spécification des systèmes temps réel (SA-RT)", Édition Masson, 1990, 346 pages.

[HEND-94] Henderson R., Zorn B., "A comparison of object-oriented programming in four modern languages", Software-practice and experience, Vol.24, N°11, November 1994, pages 1077-1095.

[HOOD-90] HOOD, Technical Group, "Hood Reference manual", October 1990.

[IGL-88] IGL Technology, "SADT : un langage pour communiquer", Eyrolles, 1988.

[JACK-83] Jackson, "Jackson System Development", Prentice Hall, 1983.

[JACO-92] Jacobson I., Christerson M., Jonsson P. and Overgaard G., "Object-Oriented Software Engineering : a use case driven approach", Addison-Wesley, 1992.

[JAUL-90] Jaulent P., "Génie logiciel: les méthodes SADT, SA, SA-RT, OOD, HOOD, ...", Édition Armand Colin, 1990, 288 pages.

[KELL-92] Kelly J.C., Sherif Y.S., "Comparison of four design methods for real-time software development", Information and software technology, Vol.34, N°2, February 1992, pages 74-82.

[MALL-89] Mallet R.A., "De la production à la stratégie : Le nouveau discours des methodes", Informatique hebdo, Mai 1989, pages 27-39.

[MARC-88] Marca D., McGowan C., "SADT: Structured Analysis and Design Technique", Mc Graw-Hill, 1988.

[MART-92] Martin J., Odell J., "Object-Oriented Analysis and Design", Prentice-Hall, Englewood Cliffs, NJ, 1992.

[MART-93] Martin J., "Principles of object-oriented analysis and design", Prentice Hall, Englewood Cliffs, NJ, 1993.

[MASI-90] Masini G. et al., "Les langages à objets : Langages de classes, langages de frames, langages d'acteurs", Édition InterÉditions, 1990, 594 pages.

[MCME-84] McMenamin S.M., Palmer J.F., "Essential Systems Analysis"; Prentice-Hall, 1984.

[MEYE-90] Meyer B., "Conception et programmation par objets pour du logiciel de qualité", InterÉditions, 1990.

[MONA-92] Monarchi D.E., Puhr G.I., "A research typology fot object-oriented analysis and design", Communication of the ACM, Vol.35, N°9, September 1992, pages 35-47.

[MORA-91] Morand B., "Faut-il renouveler les méthodes de conception de systèmes d'information", AFCET/INTERFACES N°103/104, MAI/JUIN, 1991, pages 59-66.

[MORE-89] Morel G., Roesch M., Iung B., "Les outils de XAO dans le cycle de vie de l'automatisation", Revue Générale d'Electricité, N°9, Octobre 1989.

[PAGE-88] Page-Jones, "The practical guide to structured systems design", 2nd édition, Prentice-Hall International Editions, 1988, 364 pages.

[PASC-86] Pascoe G., "Elements of Object-Oriented Programming", Byte, Vol.11, N°8, August 1986.

[PIGO-91] Pigot T., "Programmation Orientée Objet", INFOPC N°75, Octobre 1991, pages 213-232.

[REWI-95] El-Rewini H., Hamilton S., "Object Technology: A virtual roundtable", IEEE Computer, October 1995; pages 58-72.

[ROLLA-88] Rolland, Foucault, Benci, "Conception de systèmes d'information", Eyrolles, 1988

[ROSE-91] Rosen J.P., "Introduction au monde des objets", AFCET INTERFACES, N°103-104, Mai-Juin 1991, pages 5-95.

[RUMB-91] Rumbaugh J. et al., Object-Oriented modeling and design, Prenice-Hall International Editions, 1991, 500 pages, ISBN 0-13-630054-5.

[RUMB-95] Rumbaugh J. et al.," OMT : Modélisation et conception orientées objet", édition francaise revue et augmentée, Édition Masson et Prentice-Hall, 1995, ISBN 2-225-84684-7, 520 pages.

[RUSS-88] Russo V., Kaplan S., "A C++ Interpreter for Scheme", Proceedings of USENIX C++ Conference, Berkeley, CA: USENIX Association, 1988.

[SCHU-86] Schmuker K., "Object-Oriented Programming for the Macintosh", 1986, Hasbrouk Heights, NJ: Hayden.

[SELI-94] Selic B., Gullekson G., Ward P.T., "Real-time object-oriented modeling", Wiley professional computing, 1994, ISBN 0-471-59917-4, 525 pages.

[SHLA-88] Shlaer S. and Mellor S.J., "Object-Oriented Systems Analysis : Modeling the World in Data", Prentice Hall International Edition, 1988, 145 pages.

[SHLA-90] Shlaer S., Mellor S.J., "Object Life Cycles: Modeling the World in States", Englewood Cliffs, New Jersey: Yourdon Press, 1990.

[SIMO-88] Simonian R., Crone M., "InnovAda : True Object-Oriented Programming in Ada", Journal of Object-Oriented Programming, vol.1, N°4, November-December 1988.

[SNYD-93] Snyder A., "The Essence of Objects: Concepts and Terms", IEEE Software, January 1993, pages 31-42.

[SOMM-89] Sommerville I., "Le génie logiciel", Addison-Wesley Editions,1989.

[SUTC-88] Sutcliffe, "Jackson System development", Prentice-Hall, 1988

[TARD-87] Tardieu H. et al., "La méthode Merise tome 1 et 2," Éd. d'Organisation, 1987.

[WARD-86] Ward P. and Mellor S., "Structured Development of Real-Time Systems", Englewood Cliffs, New Jersey: Yourdon Press, 1986.

[WASS-89] Wasserman A., Pircher P., Muller R.J., "Concepts of Object -Oriented Design", Tool'89, 1989, pages 269-280.

[WEGN-88] Wegner P., "Dimensions of Object-based langage design", OOPSLA'87 Proceedings, October 4-8 1988, pages 168-182.

[WIRF-90] Wirfs-Brock R., Wilkerson B., Wiener L., "Designing object Oriented Software", Prentice Hall, Englewood Cliffs, NJ, 1990.

[YOUR-89] Yourdon E., "Modern structured Analysis", Englewood cliffs, New Jersey: Yourdon press, 1989, 672 pages.

Références URL (Uniform Resource Locator) sur l'approche orientée objet (ces références sont susceptibles d'être modifiées):

{URL-OO1} "http://www.einet.net/galaxy/Engineering-and-Technology/Computer-Technology/Object-Oriented-Systems/ricardo-devis/oo.html", The Object-Oriented Page.

{URL-OO2} "http://wwwedu.cs.utwente.nl/~misop009/misop14.htm". Tools for OMT.

{URL-OO3} "http://www.webcom.com/~hebbel/oo_links/oo_links.html", Great Object Oriented Links.

{URL-OO4} "http://cuiwww.unige.ch/OSG/OOinfo/index.html", Index to Object-Oriented Information Sources.

{URL-OO5} "http://donkey.CS.Arizona.EDU:1994/bib/Object/", Bibliographies on Object-Oriented Programming and Systems.

{URL-OO6} "http://wwwis.cs.utwente.nl:8080/dmgr/OODOC/oodoc/oo-Contents.html", Object-Oriented Analysis an Design Methods - Contents.

{URL-OO7} "http://arkhp1.kek.jp/~amako/computing_mgr/activities/OO_CollectInfor/OO_CollectInfo.html", Collection of Information on OO Approach.

{URL-OO8} "http://www.rational.com/htdocs/ot/rumbaugh_bio.html", Dr. James Rumbaugh.

Chapitre III

Modélisation et cycle(s) de développement

III.1 Les différentes modélisations du concept capteur intelligent

Il est indispensable que le développement de capteurs intelligents s'inscrive dans le cadre d'une démarche rigoureuse et méthodique, même si cela peut demander un certain investissement.

La spécification constitue la première phase de développement de tout système, elle consiste en l'expression technique du besoin auquel le produit doit répondre [ROBE-93].

La modélisation est la suite logique à la spécification ; elle fournit aux différents intervenants une représentation non ambiguë, un modèle formalisé à l'aide d'un langage particulier.

Elle permet de représenter sous la forme d'une abstraction la structure et/ou le comportement d'un système.

L'utilisation de méthodes permet de représenter un système selon trois approches différentes à savoir :

- l'approche fonctionnelle,
- l'approche informationnelle,
- l'approche comportementale.

La spécification doit permettre d'aboutir à l'établissement d'un document structuré et compréhensible, et doit pouvoir être supportée par un outil informatique adéquat.

La modélisation doit permettre :

- La réalisation d'un modèle de référence de capteur intelligent intégrant les besoins des utilisateurs et les offres des constructeurs. L'établissement d'un tel modèle doit permettre d'établir des normes de réalisation de capteurs intelligents. Ces modèles de références décrivent de la façon la plus exhaustive possible le produit et permettent de valider la cohérence des diverses étapes du cycle de conception.
- L'intégration des capteurs intelligents dans les systèmes automatisés de production à intelligence distribuée. Cette intégration nécessite une uniformisation des fonctionnalités de ces instruments pour assurer la cohérence des données circulant entre les différents "îlots" du système.

A l'heure actuelle, la représentation objet semble nécessaire pour intégrer correctement le capteur dans les SAID (Système Automatisé à Intelligence Distribuée) temps critique. De plus, les méthodes objet devraient permettre la conception de boîtes à outils d'aide à la conception et/ou à l'intégration de capteurs intelligents.

[ROBE-93] propose une nouvelle présentation du capteur intelligent en tant qu'instrument virtuel. La modélisation proposée fournit au constructeur un guide pour spécifier au mieux son instrument, en l'incitant à identifier et à distinguer des mécanismes souvent noyés dans l'ensemble de l'instrument. C'est une étape qui favorise la mise en oeuvre de l'intéropérabilité des instruments ainsi que leurs couplages par la connaissance de leurs structures.

Cette modélisation doit offrir aux concepteurs une meilleure réutilisabilité des modules logiciels et matériels et permettre aux utilisateurs potentiels de mieux décrire leurs besoins, de mieux comprendre le fonctionnement du capteur intelligent et de mieux envisager l'intégration du capteur intelligent au sein du système.

Le modèle proposé se veut un modèle de haut niveau, pour mettre en évidence des caractéristiques relativement familières des instruments, sans rentrer dans les détails.

Par ordre hiérarchique :

- Le modèle contient un ensemble de données;
- Un domaine est un ensemble de classes;
- Une classe est un ensemble d'attributs et de fonctions.

Le modèle est structuré en 6 domaines qui regroupent des fonctions ayant des caractéristiques communes :

- Acquisition de données,
- Fusion de données,
- Évaluation de situation,
- Prise de décision,
- Gestion de données,
- Restitution de données.

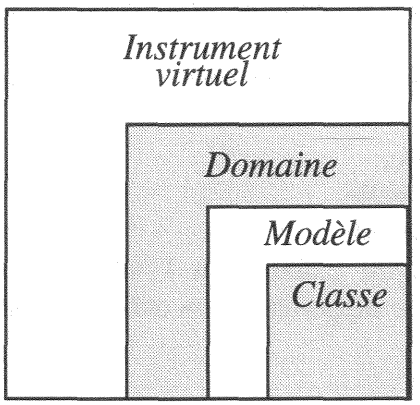


Figure III.1 : Modèle d'instrument virtuel.

[RIVI-95a] [RIVI-95b] propose la normalisation des capteurs intelligents au travers d'une modélisation basée sur le cadre de modélisation CIM-OSA (Computer Integrated Manufacturing - Open Systems Architecture) qui doit permettre la définition de constructions génériques de capteurs intelligents (La méthode SADT est ici utilisée pour formaliser et représenter le concept capteur intelligent). Ces constructions génériques sont définies à partir d'une intégration des services attendus des divers utilisateurs et du savoir-faire des constructeurs et doivent permettre selon le principe de généricité de CIM-OSA, la définition de modèles partiels puis particulier de capteurs. Ainsi les constructions génériques tiennent compte des services rendus, fonctions remplies et données manipulées par les capteurs intelligents compte tenu des divers domaines d'application (figure III.2).

La figure III.2 illustre le processus d'instanciation au sens CIM-OSA pour les divers domaines d'application des capteurs.

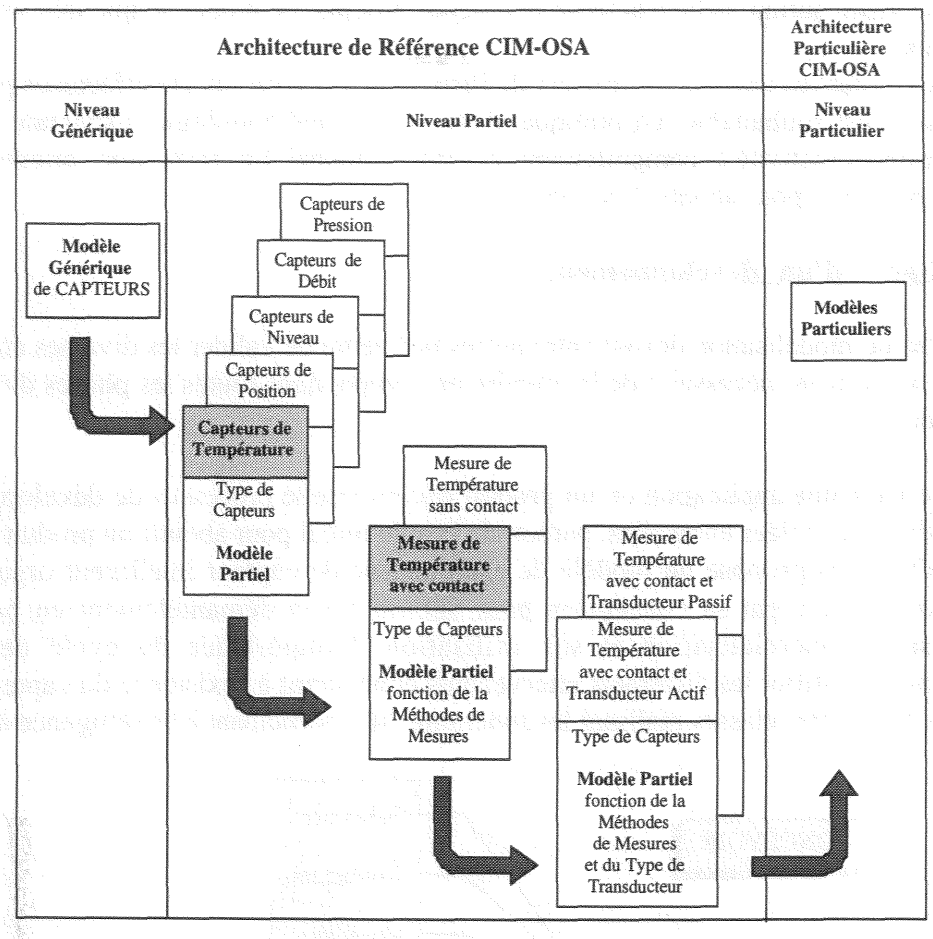


Figure III.2 : Instanciation des constructions génériques de capteurs intelligents en modèles partiels et modèles particuliers [RIVI-94].

D'autres points de vue, tels que [STAR-94] [STAR-92] [BAYA-93] {URL-LAI} proposent de décrire le capteur intelligent selon deux aspects :

- l'analyse fonctionnelle qui offre une vue interne du fonctionnement du capteur intelligent en décrivant sur quels traitements repose la réalisation d'un service,
- le modèle externe qui décrit l'ensemble des services rendus par le capteur intelligent ainsi que leur organisation. Il définit son langage (ensemble des requêtes comprises par le capteur intelligent) et son protocole de commande (conditions d'emploi des requêtes).

Le groupe de recherche AMI (Actionnement et Mesure Intelligents) propose également les bases d'une méthodologie de conception d'équipements intelligents [NEUN-95] [PANE-93] {URL-AMI}(notion plus large que celle de capteur intelligent). Il regroupe des membres appartenant à divers équipes de recherche du CRAN qui travaillent conjointement dans le domaine de l'Actionnement et la Mesure au sein des systèmes automatisés de production (SAP), non seulement d'un point de vue technologique mais aussi d'un point de vue méthodologique.

On peut citer également les travaux issus des projets européens ESPRIT II DIAS (Distributed Intelligent Actuators and Sensors) et ESPRIT III PRIAM N°6188 (Prenormative Requirements for Intelligent Actuation and Measurement) [GALA-94] [MORE-93] {URL-DIA} {URL-PRI}.

Plusieurs modélisations du concept de capteur intelligent ont déjà été effectuées, mais elles ont souvent le défaut de favoriser la vue fonctionnelle en prenant comme base la méthode SADT qui

n'utilise qu'une seule notion, celle d'activité et qui, par ailleurs, ne modélise que très imparfaitement le comportement.

Même si des tentatives utilisant les réseaux de Petri ou des Grafcet ont été effectuées pour décrire le comportement, il est souhaitable, en pratique, d'utiliser une méthodologie respectant au mieux les aspects : fonctionnel (activité et comportement) et informationnel. Les techniques orientées objet sont une des voies possibles pour aboutir à cet objectif.

III.2 Les phases d'un développement

La démarche de modélisation devant entre autres permettre de valider les diverses étapes du cycle de vie d'un produit, il est nécessaire de la prendre en compte dans toutes les phases du processus de développement.

Le cycle de vie d'une application ou un produit décompose le processus de développement selon une série d'activités couplées entre elles, partant du besoin initial pour aboutir au produit opérationnel.

Robert [ROBE-93] propose un modèle de cycle de vie de capteur intelligent organisé selon la figure III.3, qui débute par sa conception pour aboutir à son démantèlement en passant par sa production, sa commercialisation et son utilisation. L'importance du cycle de vie permet essentiellement d'identifier les différents intervenants concourant à l'existence du capteur intelligent, d'identifier les différentes classes réalisant les fonctionnalités participant à l'intelligence du capteur.

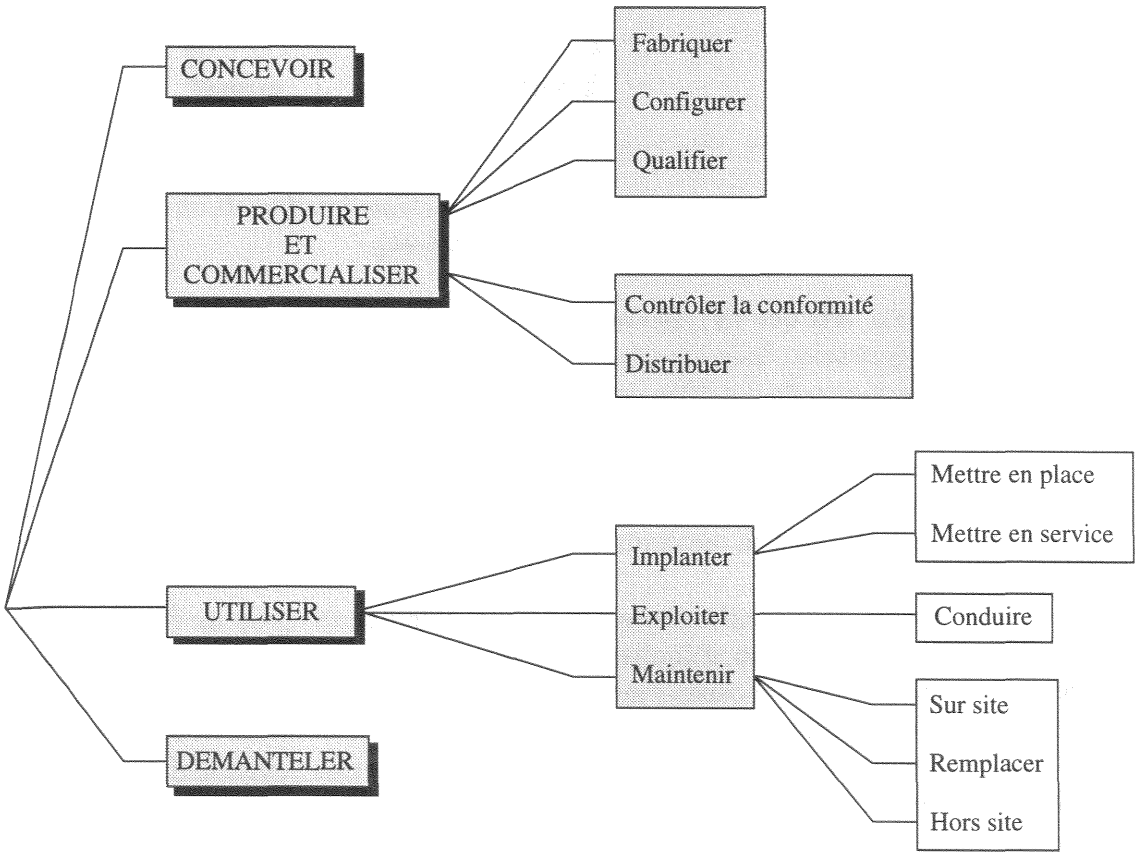


Figure III.3 : Cycle de Vie du capteur intelligent.

On donne ci-après (cf. § III.4) la signification usuelle de chacune de ces phases.

III.3 Les intervenants

A partir du cahier des charges élaboré par le client, les experts du domaine d'application du produit et concepteurs (expert du domaine de réalisation) ont à effectuer ensemble un premier travail de réflexion pour exprimer clairement l'objectif à atteindre. Intermédiaire entre la demande et la conception, une spécification ainsi élaborée sert les deux partenaires [CALV-91].

Il existe une grande diversité des partenaires côté client; pour mieux saisir la complexité du problème, il est utile de connaître, côté demandeur, les différents intervenants (un intervenant pouvant remplir plusieurs rôles) concernés par le produit. Ainsi le terme client recouvre les catégories suivantes :

- les fabricants, lorsque le système devra être reproduit dans l'organisation (ou par sous-traitance), pour une diffusion en plusieurs exemplaires.
- l'acquéreur du système : souvent négociateur et ordonnateur de la prestation. Il s'agit du responsable technique et/ou financier de l'opération. Son objectif est de satisfaire le problème au moindre coût et dans les délais impartis, et d'intégrer ce produit dans une perspective de développement de son organisation;
- les installateurs, qui eux sont préoccupés par les procédures d'installation du système sur site. Simplicité et coût réduit font partie des objectifs à satisfaire;
- les utilisateurs, qui auront à exploiter le système demandé. Ils sont concernés par les fonctionnalités, les performances, les procédures d'exploitation. Suivant la complexité du système, les utilisateurs peuvent eux-mêmes se subdiviser en catégories, par exemples les opérateurs de conduite, de gestion technique... La méthode de travail et la compétence de cette catégorie de personnel sont des éléments indispensables à considérer pour le succès du produit;
- l'équipe de maintenance, qui doit se préoccuper des qualités de maintenabilité du produit, et des moyens pour assurer la maintenance;

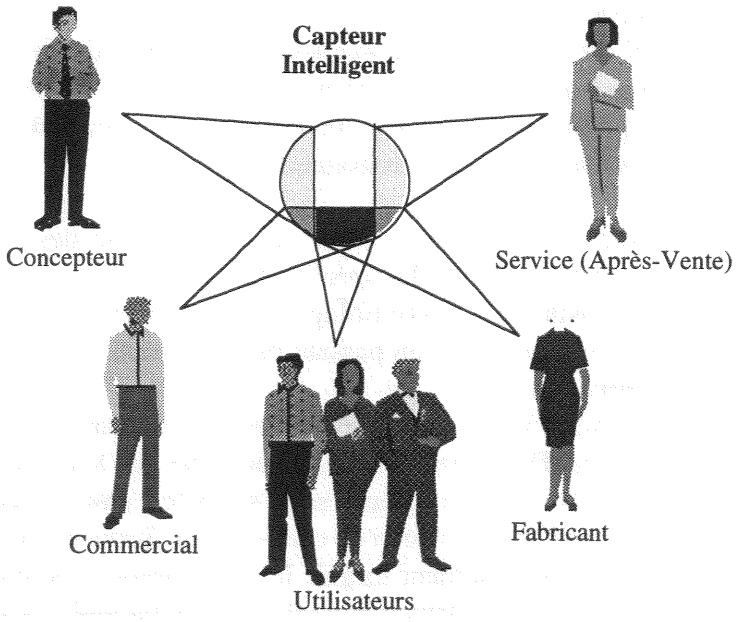


Figure III.4 : Des personnes différentes ont besoin de porter des regards différents sur le système.

Ainsi, vu la diversité des personnes concernées par le produit, chacune ayant une vue différente du produit avec des contraintes pouvant être incompatibles (figure III.4), la conception du produit se révèle souvent complexe, la modélisation étant un des outils permettant d'appréhender aux mieux cette complexité.

III.4 Les services apportés aux différents intervenants

Nous allons dans ce paragraphe recenser les services susceptibles d'être offerts aux différents intervenants par le capteur intelligent au cours de son cycle de vie.

Concevoir : cette phase est définie dans la norme AFNOR X50-127 [AFNO-88] comme l'activité créatrice qui, partant des besoins exprimés et des connaissances existantes, aboutit à la définition d'un produit satisfaisant ces besoins et industriellement réalisable. Il est primordial que tous les intervenants puissent s'exprimer à cette phase du cycle de vie du capteur.

Compte tenu de la situation de chaque partenaire concerné par le résultat, il est évident que la première tâche essentielle pour les concepteurs consiste à aboutir à une bonne compréhension du problème posé. La démarche de travail, le dialogue, la rigueur et l'objectivité dans l'analyse sont des points essentiels qui favorisent la réussite. Pour acquérir l'expertise du domaine considéré, il faut être à l'écoute de tous les acteurs côté client.

La phase de conception doit donc permettre aux différents intervenants de concevoir le capteur, en tirant parti des différents savoir-faire méthodologique et technologique, du retour d'expérience d'exploitation, tout en prenant en compte les contraintes d'ordre technologique et économique, ainsi que les nouveaux besoins, des utilisateurs par rapport aux capteurs existants, et des commerciaux.

Elle permet également la réalisation de documents de spécification, de conception, de réalisation, et de tests. Ces documents décrivent toutes les caractéristiques du produit à développer, puis la solution retenue selon une présentation descendante. On trouve en final, la description complète de la réalisation matérielle et de l'implémentation logicielle. Les procédures de tests, de vérification, de validation et les résultats obtenus sont aussi inclus dans cette partie de la documentation.

Un système n'est utilisable et maintenable (dans un contexte de qualité) que si des manuels décrivent son installation, son exploitation, et sa maintenance. Chaque type de manuel est dédié à une catégorie spécifique d'utilisateurs. Ces manuels sont :

Le manuel d'installation qui est une description explicitant la manière d'exploiter et d'administrer le système, le manuel d'utilisation qui décrit le système tel que vu par les utilisateurs, ainsi qu'un manuel ou document de maintenance servant à inventorier les demandes de changement ou d'évolution, les corrections, les adaptations à l'environnement.

Produire et Commercialiser : la phase de production se compose des activités "Fabriquer", "Configurer" et "Qualifier" le produit vis à vis des spécifications.

L'activité de fabrication consiste en l'assemblage des matières premières et des produits intermédiaires afin d'aboutir à un produit fini, en passant par des procédures de test et de validations aux différentes étapes de la fabrication.

L'activité de configuration quant à elle, consiste à effectuer des paramétrages rendant ainsi le capteur mesurant et communiquant. Des données constructeur telle que le numéro de série, la date de fabrication, voire la documentation ou les plans sont stockées dans le capteur (modifiables uniquement par le constructeur) permettant un suivi de celui-ci et un Service Après-Vente performant.

Une étape d'étalonnage intervient également lors de la configuration, permettant par la prise en compte du mesurande et des grandeurs d'influence d'affecter des paramètres spécifiques à chaque capteur (notamment ajustement des valeurs de gain et d'offset, élaboration de modèle d'étalonnage permettant la prise en compte des caractéristiques de la chaîne de mesure spécifique à chaque capteur). Ceci permet une amélioration notable des performances en terme de métrologie.

L'activité "qualifier" a pour objet la vérification des caractéristiques et des performances du capteur par rapport aux objectifs souhaités (la précision, le temps de réponse, etc.). Le contrôle est rendu plus facile grâce aux possibilités de communication et de validation du capteur intelligent (services tels que la visualisation de la configuration, les procédures d'auto-test, etc.).

La phase de commercialisation doit permettre de livrer (activité distribuer) au client un capteur conforme à ses spécifications particulières (activité contrôler la conformité). L'activité de distribution peut être réduite en une étape de stockage après production, de transport et de réception-stockage chez le client. Le capteur doit pouvoir supporter un certain nombre de conditions de stockage: température et humidité de stockage, tenue aux chocs lors du transport. La surveillance de ces conditions peut s'avérer rentable tant pour le client que pour le constructeur.

Utiliser : la phase "utiliser" se décompose en "Implanter", "Exploiter" et "Maintenir". L'implantation du capteur débute par sa mise en place sur le site d'exploitation et son intégration au sein du système automatisé de production. Ceci demande bien évidemment le respect d'un certain nombre de contraintes de montage, d'environnement d'exploitation, de raccordements mécaniques et électriques...

L'intelligence du capteur permettra de tester l'intégrité et le bon fonctionnement après déstockage, de vérifier la qualité des raccordements lors de la mise en place.

La mise en service a pour but d'intégrer de manière fonctionnelle et opérationnelle le capteur dans le système utilisateur. Cela implique notamment la définition de paramètres pour les communications bidirectionnelles (label et adresse physique du capteur), de paramètres utilisateurs (la définition de l'unité de la mesure opérationnelle, la valeur de repli, la liste des informations à archiver, les codes d'accès, la date et l'auteur de la configuration...). Tout ceci doit permettre de rendre le capteur intelligent interopérable dans la configuration de l'exploitant (les informations transmises seront interprétables par les utilisateurs).

L'activité "exploiter" est l'étape du cycle de vie où le capteur remplit son véritable rôle (délivrer une mesure la plus crédible possible). Une fois mis en service il doit répondre entièrement aux besoins des utilisateurs, ayant justement guidé le choix du capteur. Les fonctionnalités du capteur doivent permettre aux différents utilisateurs, ou aux différents équipements du système :

- de disposer de mesures opérationnelles réputées valides (du point de vue technologique et fonctionnelle) grâce notamment à la compensation des grandeurs d'influence, des auto-tests,
- d'établir une communication bidirectionnelle avec le capteur,
- d'obtenir des informations pertinentes sur l'exploitation du capteur (état de disponibilité du capteur, états des services, etc.),
- de disposer d'un historique d'exploitation (dernières mesure opérationnelle, derniers incidents survenus, etc.),
- de modifier la configuration du capteur (en ligne),
- ...

Cette activité "exploiter" permet également de recueillir des données utilisables lors d'une phase de conception d'un nouveau capteur ou d'un capteur amélioré, pouvant offrir de nouveaux services.

L'activité "maintenir" s'intéresse au diagnostic du capteur, à la localisation des défaillances et à la remise en état de bon fonctionnement du capteur. Les opérations de maintenance, qui peuvent être exécutées sur site, le sont, conformément aux documents de maintenance établis lors de la conception du capteur et peuvent également faire appel à la consultation de l'historique et à l'activation à distance des services d'auto-diagnostics du capteur dans un contexte de maintenance préventive. L'importance de certaines défaillances peut nécessiter le remplacement du capteur par un autre provenant du même fabricant ou dans l'hypothèse d'interchangeabilité d'un autre fabricant. D'autres opérations de maintenance, dites hors site, peuvent nécessiter le renvoi du capteur au S.A.V. du fabricant, par exemple, pour des opérations d'étalonnage dans le cas ou un composant de la chaîne d'acquisition défaillant est remplacé.

Démanteler : le capteur intelligent arrivant en fin de cycle de vie, il peut être procédé à la récupération d'un certain nombre de composants réutilisables. Cette dernière phase du cycle de vie influence la phase de conception et de fabrication dans le choix des matières premières, afin de pouvoir faire face aux problèmes de démantèlement du capteur, et des problèmes d'environnement par exemple.

C'est dans la phase d'utilisation que s'expriment pleinement les fonctionnalités du capteur intelligent. En plus de l'accroissement de l'exactitude et de la crédibilité des mesures dûs à la réalisation des traitements au sein des capteurs intelligents, la possibilité de communication numérique bidirectionnelle engendre une répartition des capacités de calcul ou d'intelligence à tous les niveaux du système automatisé de production.

III.5 Le cycle de vie de conception et de réalisation d'un capteur intelligent

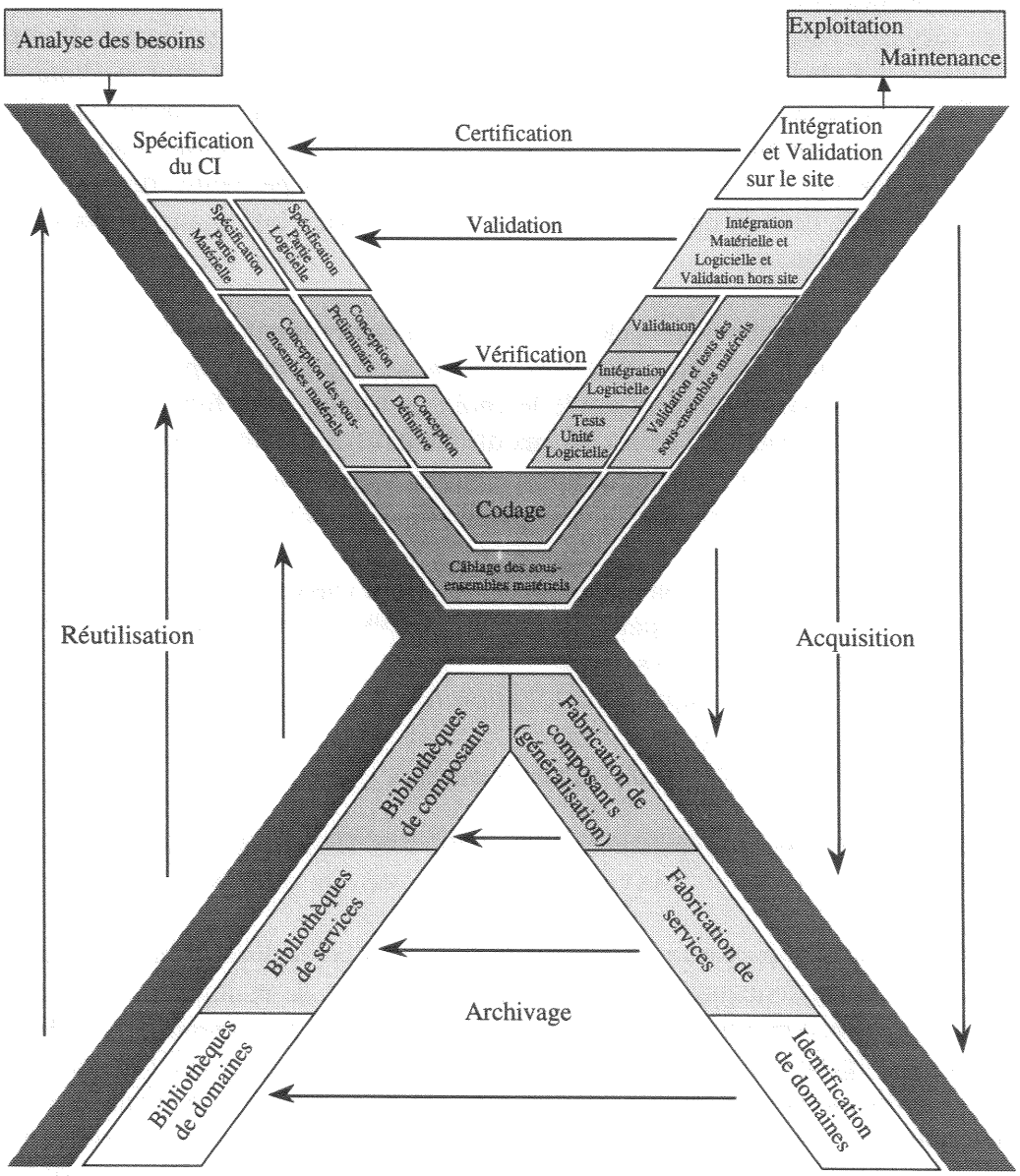


Figure III.5 : le cycle de vie en X de conception et de réalisation du capteur intelligent.

Le développement d'un système ou d'un produit passe par une description des activités permettant, en partant du besoin initial, d'aboutir à un produit opérationnel.

En plus de l'intérêt d'une structuration du travail, la décomposition du projet en étapes facilite le contrôle du développement en définissant pour chacune des phases des objectifs à atteindre planifiables et mesurables. Un modèle pour le développement sert donc de base pour la conduite de projet.

On s'appuiera sur le cycle de vie en X, (cycle de vie en V étendu à la réutilisation) pour résumer la démarche globale de développement (figure III.5) [COUL-96].

Le cycle en X a été proposé par Hodgson [HODG-91]. Il permet d'introduire la réutilisation dans le cycle de développement en V. Ce qui est proposé ici est une version adaptée du cycle de vie traditionnel du capteur intelligent destinée à introduire les besoins liés à la mise en oeuvre d'une politique de réutilisation.

Les phases hautes du cycle ont, pour un premier développement, la même signification que dans le cycle en V.

Elles expriment que la démarche de spécification-conception est globalement descendante tandis que la phase de réalisation-test est globalement ascendante car il s'agit d'assembler les constituants pour obtenir les fonctionnalités. L'axe horizontal représente les phases du projet et la durée de chacune. L'axe vertical représente le niveau d'abstraction pour l'application. Les phases de spécification et de conception conduisent à définir des niveaux de description de plus en plus détaillés.

Une première différence majeure avec le cycle en V réside dans le fait que la fin de phase de validation sur le site ne signifie pas la fin des travaux sur le développement que l'on vient de réaliser.

Il reste deux phases à traiter :

- la phase d'acquisition des composants réutilisables,
- la phase d'archivage des composants réutilisables.

La phase d'acquisition est celle durant laquelle on va identifier quelles sont les parties du projet réalisé qui sont réutilisables.

La phase d'archivage est celle pendant laquelle on stocke les composants identifiés durant la phase d'acquisition afin de les retrouver ultérieurement. Afin de prendre en compte la réutilisation, les phases de spécification, conception générale et conception détaillée du cycle sont adaptées.

La partie haute du cycle en X représente tout ce qui est spécifique à un développement, la partie basse ce qui est lié à la réutilisation.

III.5.1 La phase d'acquisition des composants réutilisables

Cette phase comporte trois parties :

- l'acquisition de composants d'analyse qui consiste à reprendre les documents de spécification et à essayer d'en tirer des informations qui semblent générales. Lors de la spécification, on a eu à identifier des informations et des traitements qui sont peut-être courants pour le type de problème étudié. Il faut extraire la partie des spécifications qui couvre ces informations et ces traitements. On obtient ainsi un dossier de spécification potentiellement réutilisable pour un domaine particulier. On le nommera composant domaine (composant d'analyse).

- l'acquisition de composants de conception qui est identique à l'acquisition de composants d'analyse. La différence vient du fait qu'elle s'applique aux documents de conception générale. Il faut isoler les descriptions de conception des différents constituants généraux

qui ont permis de traiter les problèmes du domaine concerné. On appelle ces constituants des composants services (composants de conception).

- l'acquisition de composants logiciel et matériel qui s'adresse aux conceptions détaillées. Le but est d'identifier parmi les codes ou les câblages qui ont servi à implémenter les services, ceux qui peuvent avoir un caractère de généralité et qui pourront être réutilisés dans d'autres développements soit pour implémenter le même service, soit pour en implémenter d'autres. Cette démarche est visible pour le matériel. En effet, les fabricants de composants mettent progressivement à la disposition des réalisateurs des composants de plus en plus complexes utilisables uniquement à partir de la connaissance de leurs spécifications. Les composants développés correspondent aux services nécessaires les plus couramment utiles pour les réalisations. Plus un composant est général, plus il est utilisable dans diverses applications, d'où son intérêt économique pour les fabricants.

On peut citer le projet CIAME/CAO de capteurs intelligents, dont le but est de développer un outil de conception d'instruments intelligents, dédié dans un premier temps aux capteurs. L'outil permettra à chaque constructeur de mettre en oeuvre une démarche qualité s'appuyant sur une méthodologie de conception formalisée permettant d'une part une meilleur maîtrise de leurs produits et d'autre part d'améliorer la qualité de la mesure fournie par leur capteur.

III.5.2 La phase d'archivage des composants réutilisables

La phase d'archivage est constituée de trois parties :

- l'archivage des composants d'analyse, de conception, ainsi que des composants logiciel et matériel qui consiste respectivement à les stocker dans une bibliothèque de domaines, de services, de composants afin de permettre de les retrouver plus tard et de les utiliser pour un nouveau produit ou pour implémenter une nouvelle conception générale.

III.5.3 Adaptation des phases de spécification, conception générale et conception détaillée

La réutilisation n'est possible que si les réalisateurs ont connaissance des composants disponibles. L'acquisition de cette compétence est lente et progressive et doit être entretenue en permanence.

- Phase de spécification :
  En phase de spécification, après avoir posé le problème, il est conseiller d'aller consulter les bibliothèques de domaines afin de savoir si une analyse sur le même type de problème n'a pas déjà été faite. Si c'est le cas, ce qui convient dans le nouveau problème sera réutilisé après une éventuelle adaptation.

- Phase de conception générale :
  Si une partie de la spécification a réutilisé des composants d'analyse, il est peut-être possible de réutiliser également les composants de conception associés. Au fur et à mesure de l'avancement de la conception générale, on consulte les bibliothèques de services afin de déterminer si certains constituants que l'on est en train de concevoir ne sont pas déjà connus et réutilisables.

- Phase de conception détaillée :

Pour les services réutilisés, il est possible éventuellement de réutiliser également, les composants de code ou les câblages.
Pendant la conception détaillée, le codage et le câblage, on consulte les bibliothèques de composants afin de trouver d'éventuels composants réutilisables dans le nouveau contexte.

III.5.4 Le développement incrémental

Le cycle en X peut également s'utiliser lors des phases de développement du cycle en spirale [BOEH-76] [BOEH-88]. En effet, une des critiques principales faite au cycle en V est qu'il faut attendre la fin des développements pour vérifier l'adéquation entre spécification et performance du produit. Dans la réalité, le développement d'un produit ne commence pas nécessairement à la spécification pour se terminer à l'archivage des composants. Il est plutôt habituel de considérer qu'un développement est constitué de plusieurs cycles en X imbriqués (quelques cycles peuvent avoir pour objectif la fabrication de composants logiciel ou matériel pour la réutilisation), permettant de réaliser des prototypes parfois dès la phase de spécification afin de valider certaines idées. Le développement du produit se fait alors par morceaux; certaines parties sont terminées alors que d'autre en sont encore à la phase de conception.

Les techniques orientées objet se prêtent particulièrement bien à cette approche [COCK-93] [HAYN-95] [CAPP-94]. Grâce aux techniques orientées objet telles que l'héritage, le polymorphisme et la liaison dynamique et à condition d'utiliser un langage qui supporte correctement ces techniques pour la partie logiciel. Il est possible de développer de façon incrémentale des applications en ne les remettant en cause que par parties. De plus, le processus de développement est sans rupture. Comme l'approche objet définit un ensemble d'objets liés au problème très tôt dans le projet et continue à utiliser et à étendre ces objets tout au long du cycle de développement, la séparation des phases du cycle de vie est moins tranchée. Avec la plupart des méthodologies objet (comme OMT), le modèle objet développé durant l'analyse est utilisé pour la conception et l'implémentation; le travail est canalisé dans l'effort de raffinement du modèle à des niveaux de plus en plus détaillés, plutôt que dans un effort de conversion d'une représentation à une autre comme c'est le cas le plus souvent pour les autres approches.

III.6 Cycle de vie des classes et cycle de développement

Nous avons vu précédemment que l'approche objet peut prendre en compte le haut degré de recouvrement des phases du cycle de vie ainsi que la possibilité de développement incrémental (prototypage) du système. Comme les classes du modèle objet sont identifiées dans un domaine donné, si une classe doit être réutilisée, alors elle doit être développée en dehors de l'application ou elle a été premièrement identifiée.

La figure III.6 montre que le cycle de vie de l'application et celui des classes doivent être séparés l'un de l'autre. Ceci implique que le développement des classes doit être le plus indépendant possible de l'application d'où elles ont été identifiées[MCGR-92]. Cette indépendance a un impact important sur le développement d'une classe. En effet, un développement complet d'une classe prendra plus de temps pour son développement, puisque des fonctionnalités supplémentaires non utilisées par l'application seront développées pour qu'elle soit suffisamment générique afin d'être réutilisée. Le coût de d'implémentation et de maintenance pourra être plus important que celui d'une conception originelle (dix fois).

Pour cela le modèle en "fontaine" de Henderson-Sellers [HEND-90] nous semble le plus approprié pour le développement des classes (figure III.7). Le système est vu non seulement par son cycle de vie global, mais également par plusieurs cycles de vie pour un ensemble de classes autonomes (appelé

module). Les modifications pourront alors être plus facilement faites entre le développement des classes et la spécification du système. Cela permet de ne pas figer les spécifications des besoins à une étape donnée du cycle de vie du système.

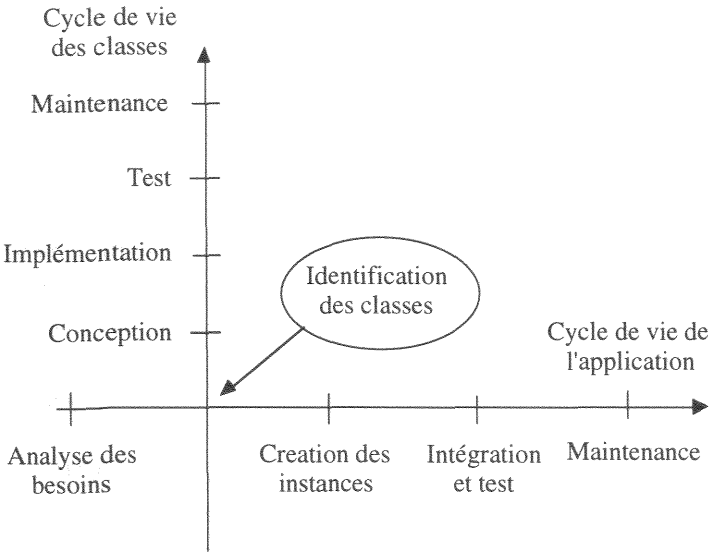


Figure III.6 : Cycles de vie d'une classe et de l'application.

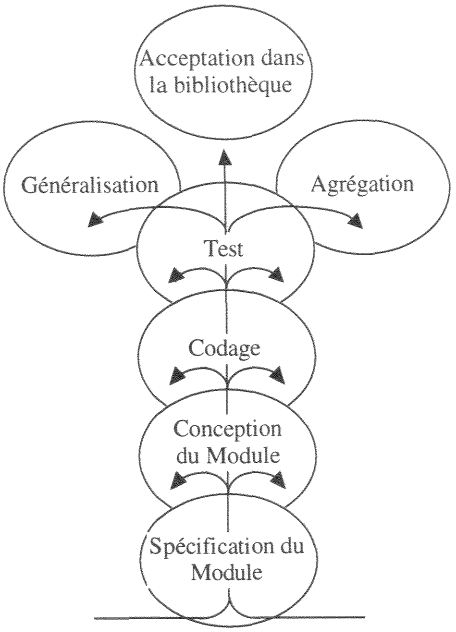


Figure III.7 : Cycle de vie pour le développement d'un module (un module peut s'apparenter à une classe individuelle, ou à un ensemble de classes).

A travers les étapes de généralisation et d'agrégation, les classes sont revues pour être suffisamment génériques afin d'être réutilisées dans un plus grand nombre d'application que celles qui ont été développées à l'origine. Ceci implique un plus grand effort à court terme par rapport à une conception-implémentation unique, mais à long terme, à l'aide des bibliothèques de composants qui auront été construites, on obtiendra une réduction significative du temps et des efforts de développement.

III.7 Conclusion

Ce chapitre a mis en évidence les nombreux apports des capteurs intelligents aux différents intervenants tout au long du cycle de vie, que se soit en phase de production, de commercialisation ou d'exploitation, grâce notamment aux services de configuration, d'auto-tests, et à la communication bidirectionnelle.

Afin d'avoir une modélisation de capteur intelligent la plus générique possible, nous avons proposé une version adaptée du cycle de vie classique de conception-réalisation en V destinée à introduire les besoins liés à la mise en oeuvre d'une politique de réutilisation.

Nous avons vu que les techniques orientées objet se prêtent particulièrement bien à la notion de réutilisation, notamment grâce aux techniques d'héritage et de polymorphisme, malgré un surcoût inévitable à cours terme lié à la fabrication des bibliothèques de composants réutilisables.

De plus, l'absence de rupture dans le développement orienté objet facilite la répétition d'étapes de développement à des niveaux de détail de plus en plus fins. Chaque itération ajoute ou clarifie les données de travail plutôt qu'elle ne modifie le travail déjà effectué. Aussi y a-t-il moins de risques d'introduire des incohérences.

La réalisation de bibliothèques de domaines, de services, et de composants doit permettre de mettre en oeuvre une boite à outils d'aide à la conception de capteurs intelligents. A cet effet, nous présentons dans le chapitre V, les règles qu'il est préconisé de suivre pour garantir que les composants seront réutilisables, comment adapter et faire évoluer ces composants réutilisables pour qu'ils deviennent utilisables dans un nouveau contexte.

III.8 Références

[AFNO-88] Norme X 50-127. "Gestion de la qualité. Recommandation pour obtenir et assurer la qualité en conception", Janvier 1988.

[BAYA-93] Bayart M., Staroswiecki M., "A generic functional model of smart instrument for distributed architecture", Imeko TC4 Symposium Intelligent instrumentation for remote and on site measurement, May 12-13 1993, Brussels, Belgium, pages 231-238.

[BOEH-76] Boehm B.W., "Software Engineering", IEEE Transactions on computers, Vol.C-25, N°12, December 1976, pages 1226-1241.

[BOEH-88] Boehm B.W., TRW defense systems group, "A spiral model of software development and enhancement", Computer, May 1988, pages 61-72.

[CALV-91] Calvez J.P., "Spécification et conception des systèmes: une méthodologie", Édition Masson, 1991, 614 pages.

[CAPP-94] Capper N.P., Colgate R.J., Hunter J.C., James M.F., "The impact of object-oriented technology on software quality: three case histories", IBM Systems Journal, Vol.33, N°1, 1994, pages 131-157.

[COCK-93] Cockburn A.A.R., "The impact of object-orientation on application development", IBM Systems Journal, Vol.32, N°3, 1993, pages 420-444.

[COUL-96] Coulange B., "Réutilisation du logiciel", Édition Masson, 1996, 324 pages.

[GALA-94] Galara D., Iung B., Morel G., Russo F., "Intelligent Actuation and Measurement system-based Modelling : The PRIAM way of working", 2nd IFAC Workshop on Computer Software Structures Integrating AI/KBS Systems in Process Control, Lund, Sweden, August 10-12, 1994.

[HAYN-95] Hayne S.C., Pendergast M., "Experiences with object-oriented group support software development", IBM systems Journal, Vol.34, N°1, 1995, pages 96-119.

[HEND-90] Henderson-Sellers B., Edwards J.M., "The object-oriented systems life cycle", Communication of the ACM, Vol.33, N°9, September 1990, pages 143-159.

[HODG-91] Hodgson R., "The X-model : a process model for object-oriented software development", proceedings of Le génie logiciel & ses applications, Toulouse 1991.

[MCGR-92] McGregor J.D., Sykes D.A., "Object-Oriented Software Development : Engineering Software for Reuse", 1992, Édition Van Nostrand Reinhold.

[MORE-93] Morel G., Lhoste P., Iung B., Petin J.F., Corbier F., Douchin O., "Discrete Event Automation Engineering : Outline of the PRIAM project", BIAS Automation 93, Workshop 2, Milan, November 23-25, pages 1105-1116, 1993.

[NEUN-95] Neunreuther E., Luttenbacher D., Suhner M.C., "Towards a partial unified reference model for intelligent equipement design", ASI'95, the annual conference of ICIMS-NOE, Cascais, Portugal, June 25-28, 1995, pages 216-222.

[PANE-93] Panetto H., Rivière J.M., Pétin J.F., "Towards a unified approach for intelligent actuators and sensors", IEEE/SMC, Le touquet, France, 17-20 octobre, 1993.

[RIVI-94] Riviere J.M., Luttenbacher D., Robert M., Jouannet J.P., "Design of smart sensors : towards an integration of design tools", Eurosensors VIII, Toulouse 25-28 Septembre 1994, Volume 2, pages 509-515.

[RIVI-95a] Riviere J.M, "Contribution a la définition d'un modèle de référence de capteur intelligent: Application a un capteur intelligent de temperature", Thèse de doctorat de l'université Henri Poincaré Nancy 1, 4 janvier 1995.

[RIVI-95b] Riviere J.M., Luttenbacher D., Robert M., "On the modeling of smart sensors : a CIM-OSA based methodology and an integration of design tools", First World Conference on Integrated Design and Process Technology, IC2 Institute - University of Texas, Austin, December 7-9, 1995.

[ROBE-93] Robert M., Marchandiaux M., Porte M. "Les capteurs intelligents et méthodologie d'évaluation", Éditions HERMÈS, 1993.

[STAR-92] Staroswiecki M., "Les systèmes et le concept d'instrumentation intelligente", Ingénieurs et scientifiques de France, Le progrès technique, N°2, pages 35-42, 1992.

[STAR-94] Staroswiecki M., Bayard M., "Actionneurs intelligents", Paris, Édition Hermès, 1994.

Références URL (Uniform Resource Locator) sur les différentes modélisations du concept capteur intelligent (ces références sont susceptibles d'être modifiées):

{URL-PRI} "http://cabernet.esprit.ec.org/esp-syn/text/6188.html", PRIAM - 6188.

{URL-DIA} "http://cabernet.esprit.ec.org/esp-syn/text/2172.html", DIAS - 2172.

{URL-LAI} "http://Lails1.ec-lille.fr/~khattabi/Lail/", Ecole centrale de Lille (LAIL).

{URL-AMI} "http://cran.esstin.u-nancy.fr/CRAN/Ami/home.htm", Actionnement et Mesure Intelligentes.

Chapitre IV

Démarche de modélisation selon une approche orientée objet

IV.1 Introduction : la méthode OMT

La méthodologie OMT (Object Modeling Technique : Technique de Modélisation Objet)[RUMB-95c] [RUMB-91a] [RUMB-91b] est une approche orientée objet de développement de système. Elle supporte en entier le cycle de vie de développement d'un système d'information à travers l'analyse, la conception du système, la conception des objets et l'implémentation. Un des bénéfices de cette méthodologie réside dans sa capacité à considérer les trois aspects d'un système : l'aspect informationnel qui introduit les structures de données, l'aspect fonctionnel qui concerne la sémantique de l'activité des objets, et l'aspect comportemental qui s'intéresse au modèle temporel d'évolution du système.

La méthodologie sera présentée comme une série d'étapes, avec des techniques et des notations associées à chaque étape. Les concepts et les notations que supporte la méthodologie seront présentés par la suite.

IV.2 Le processus de développement

La méthodologie OMT se décompose en quatre étapes (figure IV.1) :

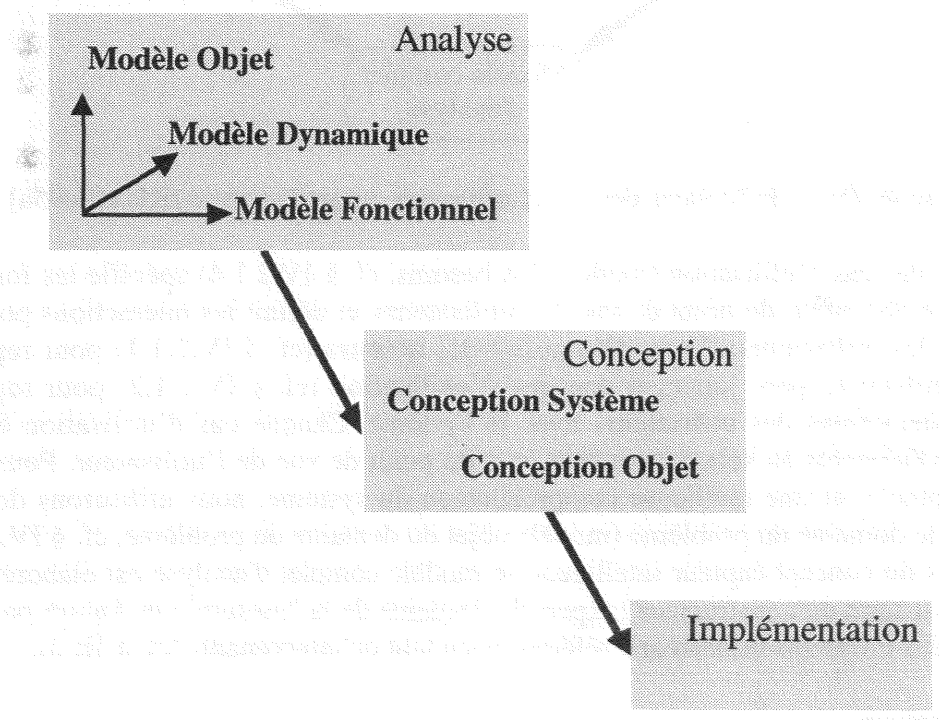


Figure IV.1: Résumé de la démarche [LARO-95].

Le cycle de vie de développement a été étendu en y ajoutant une phase de conceptualisation (formalisation de la spécification) qui précède la phase d'analyse [RUMB-94].

IV.2.1 La phase de conceptualisation (figure IV.2)

La formalisation de la spécification repose sur les **cas d'utilisation**, décrits par Ivar Jacobson. Ils définissent les fonctionnalités du système du point de vue des utilisateurs [JACO-93] [JACO-95a].

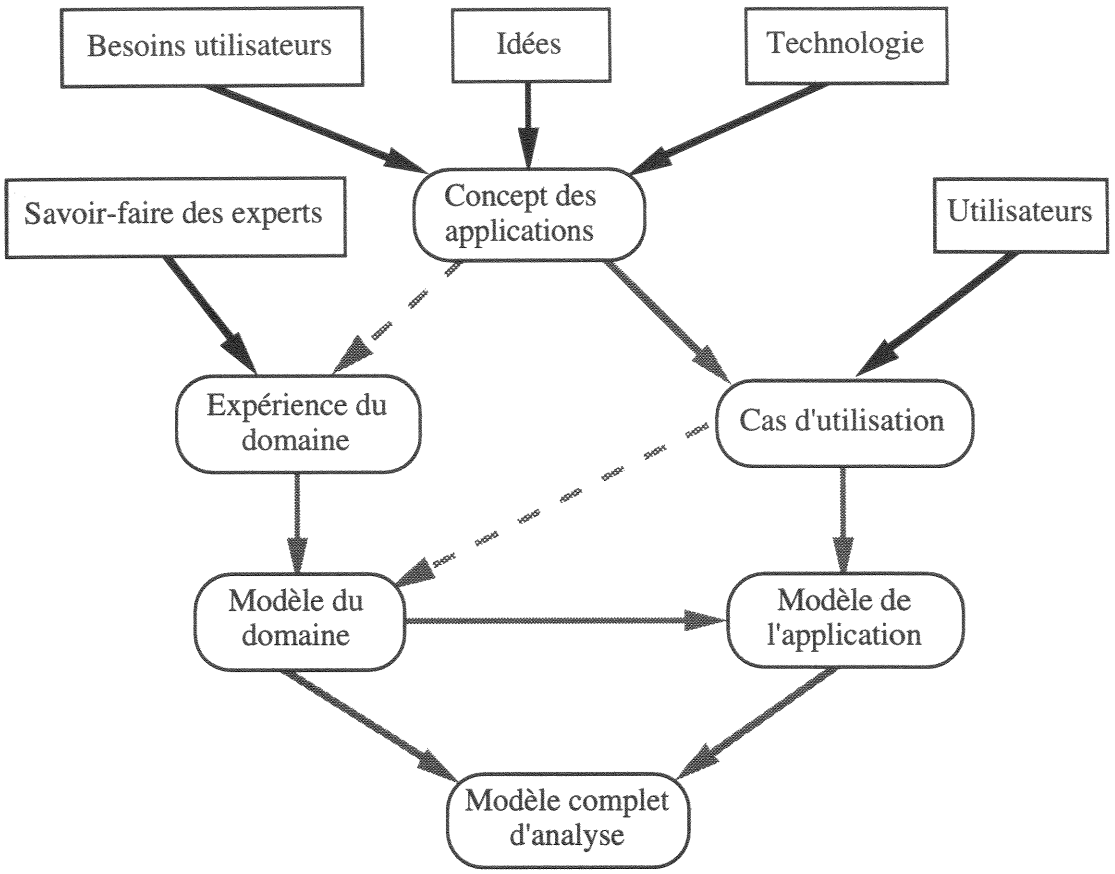


Figure IV.2 : Influences des cas d'utilisation durant l'analyse [RUMB-95a].

Le modèle des cas d'utilisation (modèle des besoins, cf. § IV.2.1.4) spécifie les fonctionnalités que le système doit offrir du point de vue des utilisateurs, et définit les interactions possibles entre le système et les utilisateurs. Ce modèle utilise des **acteurs** (cf. § IV.2.1.1) pour représenter les rôles qu'un utilisateur peut jouer, et des **cas d'utilisation** (cf. § IV.2.1.2) pour représenter les différentes interactions des utilisateurs avec le système. Chaque cas d'utilisation est une suite complète d'événements au sein du système, vue du point de vue de l'utilisateur. Pour donner une image conceptuelle et une meilleure compréhension du système, nous utiliserons des objets qui existent dans le domaine du problème (modèle objet du domaine du problème, cf. § IV.2.2.1).

Dans le cas du concept capteur intelligent, le modèle complet d'analyse est élaboré d'une part à partir du savoir faire du constructeur (expert du domaine de la "mesure") et d'autre part à partir du point de vue des différents utilisateurs, référencés en tant qu'intervenants (cf. § III.3).

IV.2.1.1 Les acteurs

Pour pouvoir identifier les cas d'utilisation à prendre en compte dans le système, on identifie tout d'abord les utilisateurs du système. Pour cela, on utilise les **acteurs**. Ceux-ci modélisent les utilisateurs potentiels qui échangent des informations avec le système. L'acteur est un type ou une catégorie d'utilisateurs (acteurs humains, mais aussi autres systèmes), et lorsqu'un utilisateur fait

quelque chose, il se comporte comme une occurrence de ce type (acteur). Une même personne peut instancier (jouer les rôles de) plusieurs acteurs différents.

IV.2.1.2 Les cas d'utilisation [JACO-95a] [JACO-94b]

Un cas d'utilisation est une manière spécifique d'utiliser le système en faisant appel à une partie de ses fonctionnalités. Chaque cas d'utilisation est constitué d'une suite d'événements invoquée par un acteur, et il spécifie les interactions entre l'acteur et le système. L'ensemble des cas d'utilisation spécifie toutes les façons d'utiliser le système.

Les cas d'utilisation sont une façon de spécifier "la fonctionnalité" du système, organisée en catégories de fonctionnalités d'un point de vue utilisateur. En énumérant les acteurs ainsi que les cas d'utilisation, il y a peu de chance d'oublier d'importantes parties du système.

IV.2.1.3 Les extensions : combinaison des cas d'utilisation

L'association d'extension est un concept puissant que l'on utilise pour structurer et pour relier les descriptions de cas d'utilisation. L'extension décrit comment on peut insérer une description de cas d'utilisation dans une autre description, en étendant ainsi cette dernière. Cela permet de décrire les extensions de cas d'utilisation de manière très simple, et les modifications ainsi que les ajouts de fonctionnalités sont particulièrement facilités.

Approfondissement du modèle des besoins :

On effectue cet affinement en identifiant et en isolant les parties semblables dans les cas d'utilisation (par exemple : même suite d'événements). Ainsi, les parties semblables ne sont décrites qu'une seule fois au lieu de l'être dans tous les cas d'utilisation présentant ce comportement. Toute modification de l'une de ces parties aura donc un impact immédiat sur tous les cas d'utilisation partageant cette partie.

Jacobson décrit deux façons de combiner les cas d'utilisation [JACO-93] :

- l'extension,
- l'utilisation.

La relation "étend" correspond à une relation d'héritage alors que la relation "utilise" correspond à une relation de type agrégation. Les relations "étend" et "utilise", peuvent être traitées comme un cas spécial d'une association "ajoute" entre un cas de base et un cas additionnel.

La figure IV.3 montre les différentes combinaisons possibles des cas d'utilisation. Les notations utilisées sont celles définies par Rumbaugh [RUMB-94] et par la méthode orientée objet unifiée [UNIF-95a] [UNIF-95b] (cette méthode est présenté brièvement en annexe II).

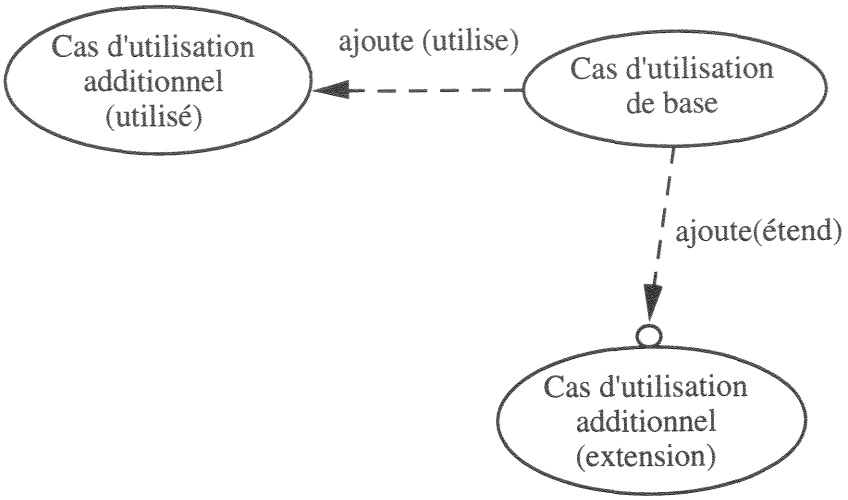


Figure IV.3 : Combinaison des cas d'utilisation.

IV.2.1.4 Le modèle des besoins : application au concept capteur intelligent

Dans le cas du concept capteur intelligent, nous pouvons identifier plusieurs cas d'utilisation (figure IV.4). On peut sommairement les décrire comme suit :

L'avantage essentiel de cette nouvelle génération de capteurs est de rendre possible un paramètrage quasi-total du système afin que l'utilisateur dispose immédiatement en sortie du capteur de l'information utile qui l'intéresse. De manière à assurer une grande disponibilité du capteur, un partage sans ambiguïté des responsabilités est nécessaire : celui-ci est obtenu par l'intermédiaire de clés et de codes d'accès qui verrouillent les modifications suivant le type de la donnée.

Le cas d'utilisation **"Configurer, paramètrer"** est utilisé pendant différentes étapes du cycle de vie du capteur intelligent. Il est d'abord initié par le constructeur lors de la phase de conception du capteur intelligent (hors site). Celui-ci s'intéresse au choix des modules matériels et logiciels (préchargés) requis pour l'exploitation du capteur intelligent pour une application particulière. La configuration du constructeur détermine un certain nombre de paramètres sans lesquels le capteur intelligent ne peut fonctionner. Elle permet entre autres de faire un suivi de production. Les paramètres ainsi définis par le constructeur sont les suivants : nom du constructeur, numéro de série, version du logiciel implanté, date de fabrication, date du dernier service après vente, ...

Le capteur dispose alors d'une configuration matérielle et logicielle minimale, nécessaire à son exploitation.

Chez l'exploitant (sur site) la configuration a pour but l'intégration du capteur intelligent dans le système d'automatisation pour une application particulière (le "responsable du capteur sur site" est l'acteur qui permet cette intégration). On peut distinguer plusieurs types de configuration :

- une configuration technologique visant à intégrer le capteur intelligent dans son environnement physique,
- une configuration fonctionnelle visant à rendre le capteur mesurant et communiquant,
- et une configuration opérationnelle visant à dédier le capteur intelligent à l'application. Dans cette dernière, les paramètres contribuant à élaborer une mesure dans le référentiel de l'utilisateur peuvent être redéfinis par les utilisateurs eux mêmes (paramétrage en ligne ne nécessitant pas l'arrêt du capteur). Pour exemple, le nombre des informations de validation associées à la mesure sera différent suivant que l'utilisateur est un opérateur de maintenance ou un opérateur de conduite. Ce dernier mettant au premier

plan la mesure, les informations de validation qui lui sont associées seront réduites, tandis que l'autre privilégiant les informations relatives au fonctionnement du capteur aura besoin d'obtenir toutes les données qui auront été élaborées par le capteur lors de l'élaboration de la mesure.

Le cas d'utilisation **"Mesurer"** est principalement utilisé en phase d'exploitation par les différents utilisateurs. Il intègre globalement la fonctionnalité *mesurer* ainsi que la fonctionnalité *valider* présentées au chapitre I. Ces deux fonctionnalités constituent l'essence même du capteur intelligent. Elles permettent d'aboutir à une mesure la plus crédible possible appelée "mesure opérationnelle" directement exploitable par les utilisateurs. Une validation technologique s'appliquant aux constituants du capteur et une validation métrologique caractérisant le bon fonctionnement du capteur en terme de métrologie font bien entendu partie intégrante des hiérarchies de mesure intervenants dans l'élaboration de cette mesure opérationnelle. Les utilisateurs de ce cas d'utilisation sont non seulement des opérateurs mais également des équipements du système d'automatisation (par exemple : un actionneur intelligent qui requiert une mesure dans un contexte "boucle de régulation").

Dans le contexte capteur intelligent, la nécessité des mesures à haut degré de crédibilité conduit à définir avant la phase d'exploitation une phase d'étalonnage rigoureuse et précise, indispensable pour l'exploitation d'un capteur intelligent.

Le cas d'utilisation **"étalonner, calibrer"** est utilisé à cet effet. Il est dans un premier temps initié par le constructeur (hors-site) lorsque le capteur dispose d'une configuration matérielle et logicielle minimale (étalonnage des différents transducteurs). Il peut être également utilisé par le responsable du capteur (sur-site) si les moyens techniques pour effectuer un étalonnage sont à sa disposition (par exemple : il pourra redéfinir ou affiner les modèles de compensation permettant ainsi de corriger les dérives éventuelles des composants dues notamment à leur vieillissement, à leur environnement, ...). L'étalonnage du capteur comprend l'ensemble des opérations qui permettent d'expliciter, sous forme graphique ou mathématique, la relation entre les valeurs du mesurande et celles de la grandeur électrique de sortie et ceci, compte tenu de tous les paramètres additionnels susceptibles de modifier la réponse du capteur [ASCH-91]. La calibration n'a plus lieu d'être dans le cas d'un capteur intelligent puisque d'une part la mesure peut s'effectuer sur toute l'étendue de mesure (correspondant à la plage d'utilisation) et d'autre part la mesure étant de type numérique, il n'y a plus besoin de recaler le signal de sortie (4-20 mA) en fonction de la plage de mesure comme c'est le cas pour les capteurs analogiques.

Le cas d'utilisation **"tester, diagnostiquer"** est principalement destiné aux opérateurs de maintenance qui peuvent dépendre soit de l'exploitant lorsque l'on est sur le site (représenté par l'acteur "opérateur de maintenance") soit appartenir au service après-vente du fournisseur lorsque le capteur est hors site (représenté par l'acteur "constructeur (hors-site)"). Il regroupe l'ensemble des fonctionnalités qui permettent, lorsque le capteur intelligent est dans un mode d'utilisation spécifique, de vérifier le bon fonctionnement des différents composants (par exemple : test de la communication, contrôle de l'alimentation générale, tests de composants matériels ou logiciels, contrôle de l'intégrité de la chaîne d'acquisition en particulier les transducteurs, ...), et le cas échéant d'aboutir à un diagnostic à partir des tests élémentaires effectués et des données archivées tout au long de l'exploitation du capteur intelligent (par exemple : les dernières erreurs datées survenues)[GEHI-94].

Le cas d'utilisation **"générer un rapport"** est initié par les différents opérateurs. Le type du rapport généré dépend du type de l'opérateur qui en fait la demande. Le principal utilisateur de ce cas d'utilisation est l'opérateur de gestion technique. Il recevra, par exemple, des rapports

journaliers lui indiquant la disponibilité du capteur intelligent (les informations contenues dans ce rapport pourront être élaborées au moyen de méthodes statistiques plus ou moins sophistiquées utilisant les informations stockées au sein du capteur intelligent comme données).

D'autres types de rapports peuvent être générés pour le responsable du capteur intelligent sur le site. Ils pourront intégrer les informations suivantes :

- le nombre de mises sous tension,
- le nombre de configurations utilisateurs,
- le temps de fonctionnement,
- le nombre d'erreurs de mesure,
- le nombre de ruptures de transducteurs,
- le nombre de mesures,
- les mesures hors seuils,
- l'extréma des mesures traitées depuis la mise en route du capteur,
- la date de dernière configuration,
- les erreurs de communication, ...

Toutes ces informations facilitent également les opérations de maintenance et de conduite. Par exemple, dans le cas des opérateurs de maintenance, il est possible de faire une synthèse des résultats de tests, en élaborant une information de validation globale ou partielle qui pourra, selon sa sophistication, constituer une première aide à la maintenance (diagnostic). Les résultats de tests pourront être mémorisés pour une durée déterminée de façon à établir un rapport plus sophistiqué comme aide à la maintenance (conditionnelle).

Comme les cas d'utilisation se concentrent souvent sur une certaine fonctionnalité du système, il est possible d'analyser les fonctionnalités de manière incrémentale. On peut ainsi développer d'abord des cas d'utilisation pour différents domaines de fonctionnalité, et regrouper plus tard les cas d'utilisation pour constituer le modèle complet des besoins.

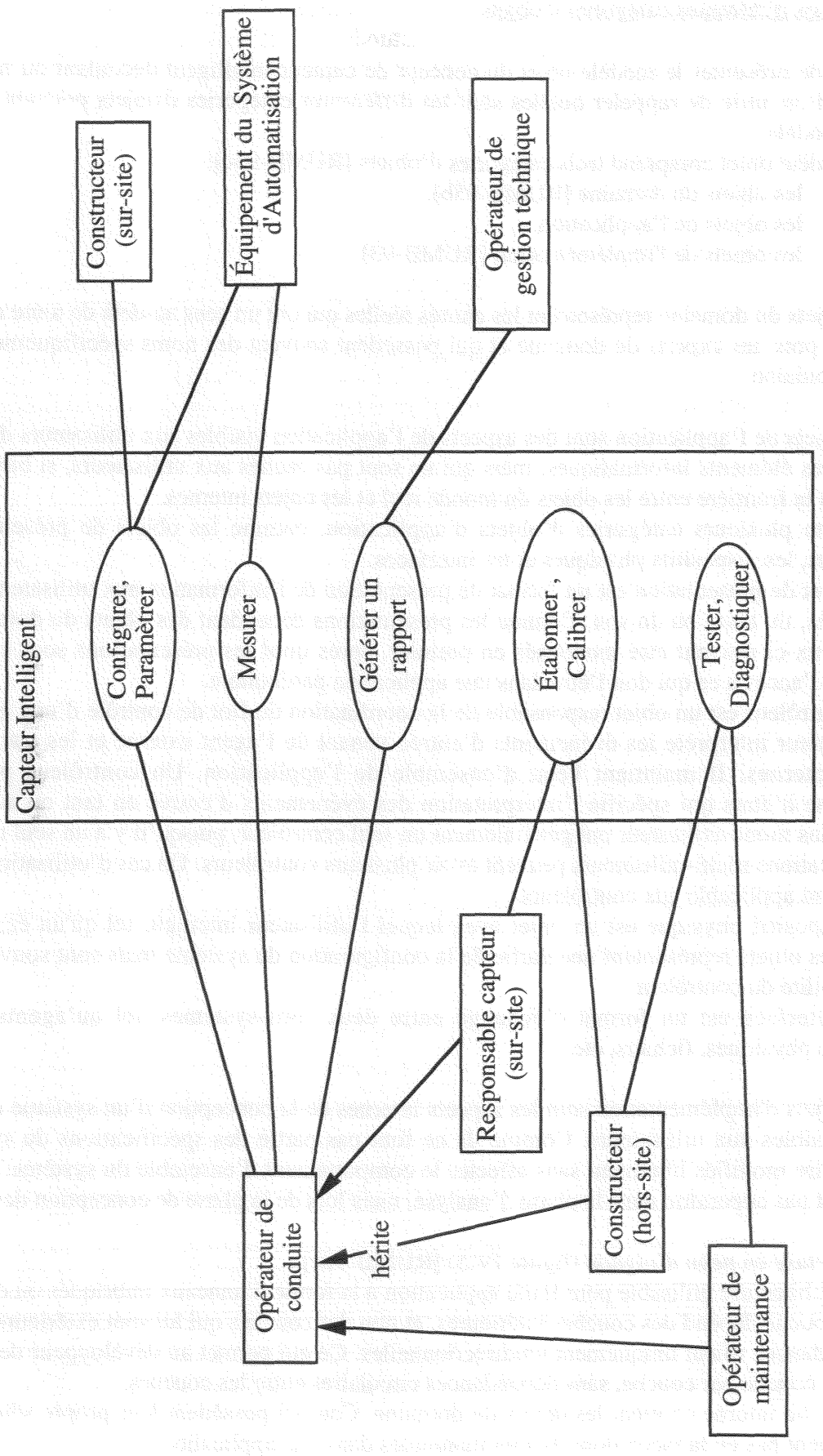


Figure IV.4 : Modèle des cas d'utilisation pour le concept capteur intelligent.

IV.2.1.5 Les différentes catégories d'objets

Avant de présenter le modèle objet du concept de capteur intelligent découlant du modèle des besoins, il est utile de rappeler quelles sont les différentes catégories d'objets pouvant intervenir dans ce modèle.

Un modèle objet comprend trois catégories d'objets [RUMB-95c]:

- les objets du domaine [RUMB-95b],
- les objets de l'application,
- les objets de l'implémentation [RUMB-93].

Les objets du domaine représentent les entités réelles qui ont un sens au-delà de toute application logicielle pour les experts du domaine et qui possèdent souvent des noms spécifiquement utilisés dans ce domaine.

Les objets de l'application sont des aspects de l'application visibles aux utilisateurs du système. Ce sont des éléments informatiques, mais qui ne sont pas cachés aux utilisateurs, si bien qu'ils se trouvent à la frontière entre les objets du monde réel et les objets internes.

Il existe plusieurs catégories d'objets d'application, comme les objets de présentation, les contrôleurs, les dispositifs physiques et les interfaces.

Un objet de présentation est un format de présentation de l'information aux utilisateurs, comme des images, du texte ou du son. Comme les présentations dépendent des objets du domaine sous-jacent, ceux-ci doivent être modélisés en premier, après quoi les présentations sont créées pour permettre l'accès à ce qui doit l'être dans une application particulière.

Un contrôleur est un objet responsable de la coordination du flot de contrôle d'une application. Le contrôleur interprète les événements d'entrée venant de l'agent externe et les transforme en actions internes. Il maintient l'état d'ensemble de l'application. Un contrôleur possède un diagramme d'états qui spécifie l'interprétation des événements d'entrée en tant qu'actions. Les applications mono-utilisateur ont généralement un seul contrôleur, puisqu'il y a un seul utilisateur ; des applications multi-utilisateurs peuvent avoir plusieurs contrôleurs. Un cas d'utilisation peut être directement applicable aux contrôleurs.

Un dispositif physique est un objet avec lequel l'utilisateur interagit, tel qu'un écran ou une souris. Ces objets représentent une partie de la configuration du système mais sont souvent sous la responsabilité du contrôleur.

Une interface est un format d'échange entre deux sous-systèmes, tel qu'agents externes, dispositifs physiques, fichiers, etc.

Les objets d'implémentation sont les aspects internes de la conception d'un système qui ne sont pas accessibles aux utilisateurs. Comme ils ne font pas partie des spécifications du système, ils peuvent être modifiés librement sans affecter le comportement d'ensemble du système. Ces objets ne doivent pas apparaître dans la phase d'analyse, mais lors de la phase de conception des objets.

L'architecture en peau d'oignon (figure IV.5) [RUMB-95c] :

Une architecture utilisable pour toute application a la forme d'anneaux imbriqués ou de couches. Chaque couche dépend des couches intérieures, et non des couches qui lui sont extérieures, afin que les dépendances soient uniquement unidirectionnelles. Ce qui permet au développeur de construire le modèle couche par couche, sans dépendances circulaires entre les couches.

La couche interne contient les objets du domaine. Ceux-ci possèdent leur propre sémantique et ne dépendent pas de la façon dont ils sont manipulés dans une application.

L'application doit être structurée de façon que les calculs propres au domaine ne dépendent pas des présentations ou du contrôle. La couche suivante contient les objets de présentation. Ces

derniers doivent être construits en tant que représentation statique des objets du domaine, les aspects dynamiques étant donnés par les contrôleurs.

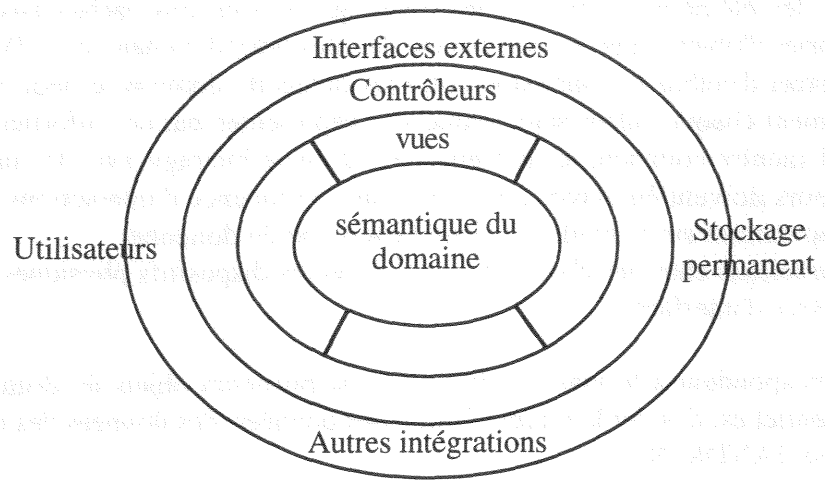


Figure IV.5 : Architecture en peau d'oignon.

Après la phase de conceptualisation intervient la phase d'analyse.

IV.2.2 L'analyse

Partant d'un problème donné, l'analyse doit construire un modèle du monde réel montrant ses propriétés les plus importantes. Le modèle d'analyse est une abstraction précise et concise de ce que doit faire le système et non pas du comment il doit le faire.

Le processus d'analyse est la stratégie globale de développement des modèles d'analyse. Il consiste d'abord à développer le modèle d'ensemble du domaine, puis y ajouter le modèle d'application.

IV.2.2.1 Le modèle du domaine (identifie les objets du domaine du problème)

Lorsque l'on travaille sur le modèle des besoins, il peut parfois s'avérer laborieux de définir la tâche du système, et notamment ses limites. C'est typiquement le cas lorsque la spécification des besoins n'existe que sous une forme très vague. Une très bonne technique consiste alors à commencer à développer une vue logique du système à l'aide des objets du domaine du problème; c'est à dire les objets qui ont un équivalent direct dans l'environnement d'application et sur lesquels le système doit raisonner [JACO-94c].

Le principal intérêt d'un tel modèle, c'est qu'il peut servir de support de communication entre les intervenants à propos du système. Les concepts évoqués seront facilement reconnus par les différents intervenants, et ce modèle pourra être utilisé pour définir ce que le système devra faire.

Le modèle d'ensemble du domaine est basé sur le domaine et l'expertise du monde réel. Il peut être développé en utilisant la définition du problème, mais il peut aussi être développé en considérant directement l'ensemble du domaine ou, mieux encore, en réutilisant des analyses antérieures du domaine. Ce modèle contient toutes les informations nécessaires pour réaliser les calculs sémantiques tels que les calculs d'ingénierie, les calculs mathématiques, sur les objets du domaine.

Ces objets représentent le savoir faire du système et constituent l'invariant fonctionnel. Ils peuvent être utilisés par plusieurs métiers différents.

IV.2.2.2 Le modèle de l'application

Ce modèle est une représentation des aspects de l'application visibles aux utilisateurs du système logiciel. Ce sont des éléments informatiques, mais qui ne sont pas cachés aux utilisateurs (les différentes catégories d'objets d'application ont été décrites précédemment, cf. § IV.2.1.5).

Ce modèle permet d'établir la connexion entre les objets du domaine et leur représentation, en déterminant comment chaque valeur sémantique sera représentée par des information visuelles (ou auditives, etc.). Il montre comment les présentations doivent interagir avec les utilisateurs. Un ou plusieurs contrôleurs doivent être créés pour contrôler la séquence d'interactions. Les commandes doivent être des opérations sur le modèle de l'application ou du domaine.

Toutes les interactions entre les classes du système et les dispositifs physiques externes doivent passer par des classes d'interfaces.

Ces objets correspondent à la vision métier d'un ou plusieurs objets du domaine du système. Leurs intérêts essentiel est d'isoler la représentation des données, des données des objets du système (objets du domaine) [ANDR-94].

Les moyens techniques sont les solutions en terme de techniques de dialogue à la représentation des objets d'application et des moyens d'interaction associés.

IV.2.2.3 Le modèle d'analyse

Le modèle d'analyse est élaboré à partir des deux modèles décrits précédemment.

Comme il a été mentionné au chapitre III, les différentes vues qu'une méthode doit pouvoir décrire sont la vue informationnelle ou structurelle d'un système, la vue comportementale et la vue architecturale.

La technique de modélisation par objet (OMT) associe ces trois vues de modélisation des systèmes.

La méthodologie OMT utilise alors trois modèles pour décrire le système :

- Le modèle objet qui représente les aspects statiques du système en définissant les objets du système, les relations entre les objets, les attributs et les opérations associés qui caractérisent chaque classe [RUMB-95b]. Ce modèle donne une vision globale de la structure du système. Les diagrammes d'objets proposent une notation graphique formelle qui permet de modéliser les objets, les classes et les relations. La notation de modélisation par objets est une des nombreuses approches venant du modèle entité-relation (ER) de [CHEN-76].
  Il existe deux types de diagrammes d'objets :
  - les diagrammes de classes qui permettent de décrire les classes d'objets,
  - les diagrammes d'instances qui décrivent comment un ensemble particulier d'objets est lié à d'autres. Ces diagrammes sont souvent utiles pour expliquer des cas de tests (et tout particulièrement des scénarios) et pour présenter des exemples.

  A un diagramme de classes donné correspond un ensemble infini de diagrammes d'instances.

- Le modèle dynamique qui représente les aspects temporels, comportementaux et de contrôle du système. Il est utilisé pour spécifier et pour implémenter le comportement temporel en terme d'événements et d'états [RUMB-95d] [ODEL-95].
  Il est représenté graphiquement par :

- des diagrammes d'états qui mettent en évidence les séquences d'états et d'événements permis dans un système pour une classe d'objets.
- des diagrammes de suivi d'événements qui montrent les événements échangés entre les objets en tenant compte de leur ordre temporel.

Les diagrammes d'états sont également en relation avec les autres modèles [COLE-92]. Les actions, dans les diagrammes d'états, correspondent aux fonctions du modèle fonctionnel ; les événements d'un diagramme d'états deviennent des opérations attachées aux objets dans le modèle objet.
La majeur partie des notations provienne du travail de David Harel [HARE-87a] [HARE-87b] [HARE-88a] [HARE-88b], qui a formalisé ses concepts dans une notation nommée "statecharts" (diagramme d'états). Le processus de Harel structure les machines à états finis afin d'éviter l'explosion combinatoire.

- Le modèle fonctionnel qui définit le processus de transformation des données en terme de valeurs et de fonctions [RUMB-95c]. Le modèle fonctionnel modélise ce que fait un système, sans s'occuper de la façon ou du moment où il le fait. Il est représenté graphiquement par des diagrammes à flots de données.
  Ces diagrammes de flots de données sont utilisés comme modèle principal dans la plupart des méthodes traditionnelles de conception de système. [YOUR-89] et [WOOD-88] présentent une méthode traditionnelle, incluant une explication complète des diagrammes de flots de données. [WARD-86] a ajouté le concept de contrôle aux notations standard des diagrammes de flots de données.

Les trois types de modèle découpent le système en vues orthogonales qui peuvent être manipulées et représentées avec une notation uniforme (Les notations de modélisation pour les trois modèles sont présentées en annexe II).

Le terme modèle possède deux dimensions : une vue système (modèle objet, modèle dynamique, modèle fonctionnel) et un stade du développement (analyse, conception ou implémentation). Chaque modèle est applicable pendant toutes les phases du développement et agrège des détails d'implémentation au fur et à mesure que le développement progresse (on peut s'appuyer sur la méthode OMT pour l'ensemble du cycle de vie).

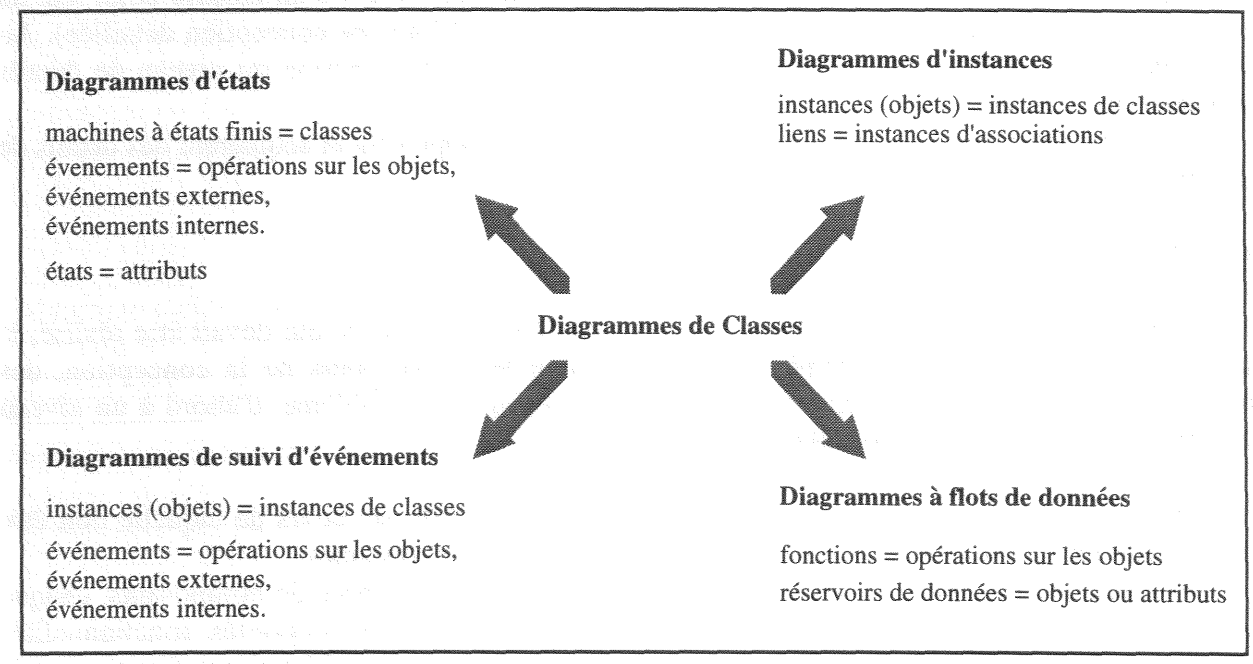


Figure IV.6 : Correspondance entres les diagrammes OMT.

Relations entre les différents modèles

Chaque modèle décrit un aspect du système et contient des références aux autres modèles. Le modèle objet décrit la structure des données que les modèles dynamique et fonctionnel utilisent. Les opérations dans le modèle objet correspondent aux événements dans le modèle dynamique et aux fonctions dans le modèle fonctionnel. Le modèle dynamique décrit le contrôle des structures des objets. Le modèle fonctionnel décrit les fonctions invoquées par les opérations dans le modèle objet et les actions dans le modèle dynamique. Il montre également les contraintes sur les valeurs des objets.

Comme l'approche objet définit un ensemble d'objets liés au problème très tôt dans le projet et continue à utiliser et à étendre ces objets tout au long du cycle de développement, la séparation des phases du cycle de vie est moins tranchée.

Dans la méthodologie OMT, le modèle objet développé dans la phase d'analyse est utilisé pour la conception et l'implémentation. Le travail est donc focalisé sur un affinement du modèle en augmentant progressivement le niveau des détails. Le processus de développement est sans rupture parce qu'il n'y a pas de discontinuité dans les notations d'une phase à l'autre.

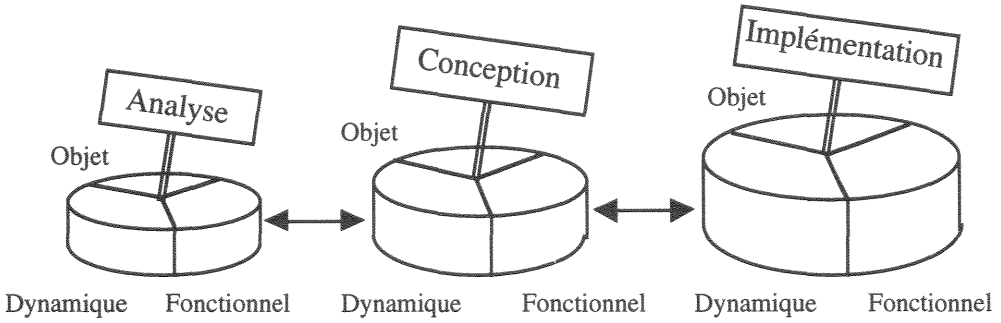


Figure IV.7 : Enrichissement des modèles.

La démarche proposée se base donc sur le principe d'enrichissement d'un modèle objet tout au long du cycle de développement (de l'étape d'analyse jusqu'à l'étape de conception détaillée). Au niveau de l'analyse, les modèles sont plus ou moins détaillés en fonction du niveau de détails souhaité par l'analyste.

Tous les modèles sont utilisables à tous les stades du développement et acquièrent des détails de mise en oeuvre au fur et à mesure du développement.

IV.2.3 La conception

Au cours de la phase d'analyse, la description s'est concentrée sur ce qui devait être réalisé, le quoi, indépendamment de la manière de faire, le comment. Au cours de la conception, des décisions, doivent être prises concernant la façon de résoudre le problème, d'abord à un niveau général, puis à des niveaux de détail de plus en plus fins.

La conception du système est la première étape de conception, au cours de laquelle doit être choisie l'architecture du système (organisation du système en sous systèmes).

Cette étape consiste donc à subdiviser le système en un petit nombre de composants. chaque sous-système englobe les aspects du système qui partagent les mêmes propriétés, fonctionnalités similaires, même emplacement physique. Un sous-système n'est pas un objet ni une fonction, mais

plutôt un paquetage de classes, d'associations, d'opérations, d'événements. Un sous-système est généralement défini par les services qu'il rend. Un service est un groupe de fonctions liées qui servent les mêmes buts, tels que le contrôle des entrée-sorties, les calculs. Un sous-système est défini de telle sorte que la majeure partie des interactions se situent à l'intérieur du sous-système plutôt qu'au travers de ses frontières, afin de limiter les dépendances avec les autres sous-systèmes.

La figure IV.8 montre l'architecture du module d'adaptation au traitement et à la communication du capteur intelligent tel que nous l'avons défini. Celui-ci comporte six sous-systèmes :

- un sous-système de communication permettant entre autres la gestion des messages provenant des opérateurs ou des automatismes du systèmes automatisé de production. Ces messages sont placés dans une file d'attente et sont traités un à un suivant leur priorité.
- un sous-système de contrôle permettant de gérer les modes d'utilisation et les services ainsi que le bon fonctionnement du capteur intelligent.
- un sous-système de mesure : dont le but principal est d'élaborer une mesure opérationnelle qui sera mise à disposition des utilisateurs.
- un sous-système d'acquisition : où s'opère une conversion des grandeurs physiques en grandeurs numériques.
- un sous système d'archivage : le stockage permanent d'informations de configuration, des modèles, de l'historique des mesure. Comme ces données doivent rester cohérentes mais que plusieurs accès peuvent y être faits de manière concurrente, elles sont maintenues dans une base de données.
- Une couche système de bas niveau permettant la gestion du temps et des temporisateurs (timers), non représentée sur la figure IV.8, mais accessibles par tous les autres sous-systèmes.

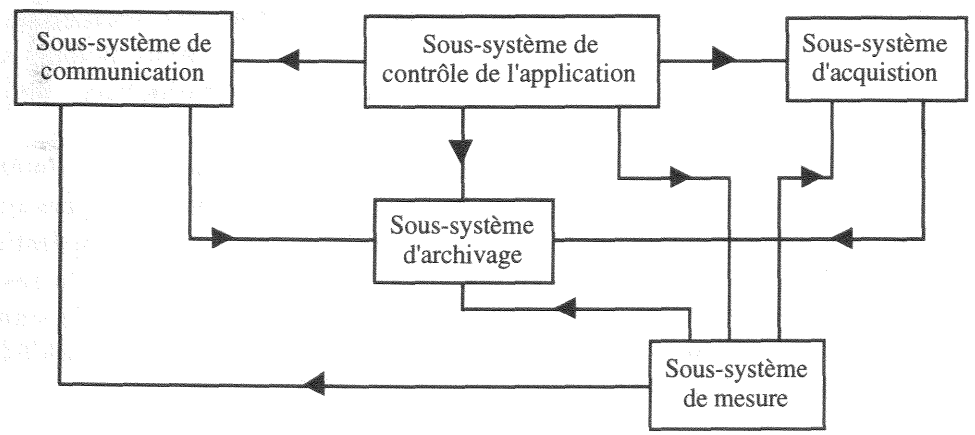


Figure IV.8 : Décomposition en sous-systèmes.

La phase de conception des objets, quant à elle, définit de façon complète les classes et associations utilisées dans l'implémentation, ainsi que les interfaces et les algorithmes des méthodes employés pour implémenter les opérations. Cette phase ajoute des objets internes d'implémentation et optimise les structures de données et les algorithmes. La conception des objets est analogue à la phase de conception préliminaire dans le cycle de vie traditionnel du logiciel.

IV.2.4 L'implémentation

Cette partie couvre la partie terminale du développement et aborde les détails pour implémenter le système en employant un langage de programmation orienté objet ou non orienté objet, ou des

systèmes de gestion de bases de données. L'écriture du code est une extension du processus de conception. Néanmoins, le codage du programme est la dernière étape de la résolution du problème, la manière dont il est écrit est donc importante pour la maintenabilité et la faculté d'extension.

Les deux derniers points (conception et implémentation) sont présentés dans le chapitre VI lors de l'application à un capteur intelligent de température. Dans le chapitre VI, nous montrons comment à partir du modèle d'analyse nous donnons corps à une ébauche d'implémentation.

Nous ne présentons alors dans la suite du chapitre IV que le modèle d'analyse élaboré.

IV.3 Le modèle de référence du capteur intelligent (modèle d'analyse)

Nous présentons ci-dessous le modèle de référence de capteur intelligent à partir du modèle d'analyse comprenant les trois modèles OMT (objet, dynamique et fonctionnel).

Nous aborderons la présentation du modèle d'analyse d'abord d'un point de vue structurel et ensuite par un examen des différents sous-systèmes réalisant la fonctionnalité "capteur intelligent".

IV.3.1 L'architecture du capteur intelligent (matériel et logiciel)

Comme le montre la figure IV.9, le capteur intelligent, désormais équipé d'une unité de traitement ("module d'adaptation au traitement") va pouvoir, en association avec une interface de communication numérique appropriée ("module de communication"), établir une communication bidirectionnelle avec tout autre équipement également connecté au réseau de communication (ou localement avec une console portable).

L'unité de traitement du capteur intelligent, permet, avant tout, d'effectuer un traitement numérique des signaux issus des éléments sensibles (corps d'épreuve, transducteurs secondaires, conditionneurs de signal, ...) composants du "module de transduction, acquisition et conditionnement".

Le module de communication ne sera pas pris en compte puisque nous ne nous intéressons qu'à la modélisation en terme de services rendus par le capteur intelligent : intégrant principalement les fonctionnalités du capteur intelligent participant à l'obtention d'une mesure opérationnelle. La description de la communication constitue en soi toute une étude impliquant la résolution des problèmes liés à l'interopérabilité des équipements. Celle-ci aura pour objectif la standardisation des protocoles régissant les échanges d'informations sur les réseaux de terrain : WorldFIP, Profibus, ISP [TELE-93] [SONG-91] [TERR-95] {URL-RT1} {URL-RT2}.

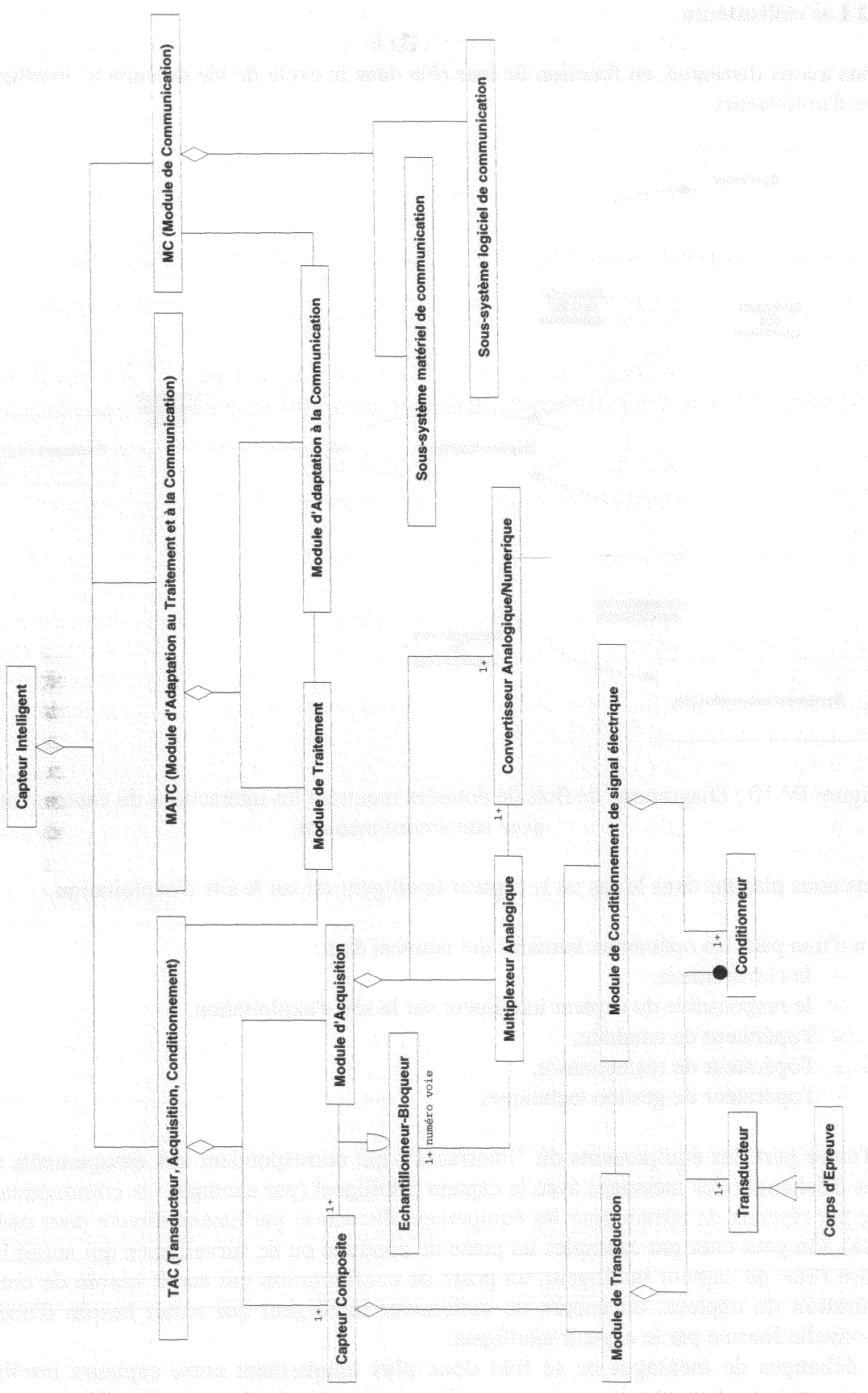


Figure IV.9 : Agrégation à niveaux multiples montrant l'architecture d'un capteur intelligent.

IV.3.2 Les utilisateurs

Nous avons distingué, en fonction de leur rôle dans le cycle de vie du capteur intelligent, deux classes d'utilisateurs.

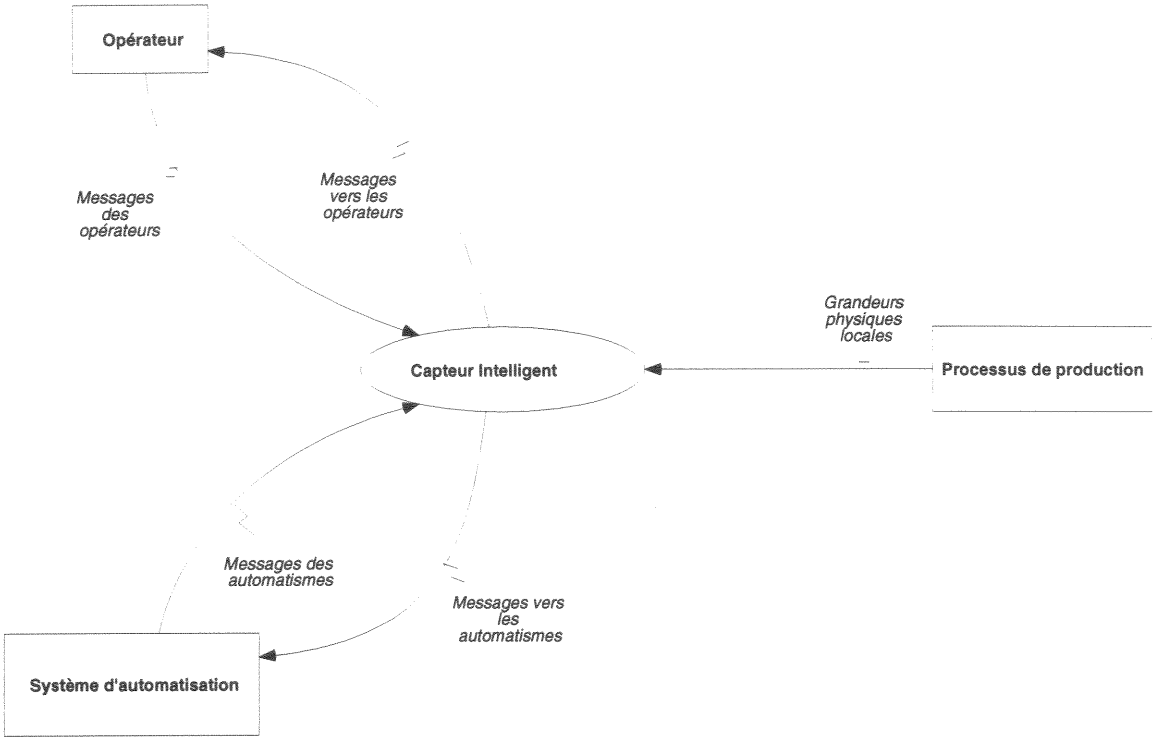


Figure IV.10 : Diagramme de flots de données montrant les interactions du capteur intelligent avec son environnement.

Nous nous plaçons dans le cas où le capteur intelligent est sur le site d'exploitation.

On a d'une part, les opérateurs humains qui peuvent être :

- le constructeur,
- le responsable du capteur intelligent sur le site d'exploitation,
- l'opérateur de conduite,
- l'opérateur de maintenance,
- l'opérateur de gestion technique,

et d'autre part, les équipements dit "interfacés", qui correspondent aux équipements ayant les moyens d'échanger des messages avec le capteur intelligent (par exemple : la communication peut se faire par réseaux de terrain pour les équipement distants et par liaison directe pour une console portable). On peut citer par exemples un poste de conduite ou de surveillance qui aurait besoin de connaître l'état du capteur intelligent, un poste de configuration qui aurait besoin de consulter la configuration du capteur, ou encore un actionneur intelligent qui aurait besoin d'une mesure opérationnelle fournie par le capteur intelligent.

Les échanges de messages ne se font donc plus uniquement entre capteurs intelligents et opérateurs, mais également entre capteurs intelligents et autres équipements intelligents.

Le constructeur est un opérateur particulier, puisqu'il a accès à tous les modes d'utilisation et services lorsque le capteur intelligent est hors site (par exemple, lors d'un service après-vente

nécessitant le renvoi du capteur chez le fournisseur pour tester le bon fonctionnement des différentes fonctionnalités). Par contre lorsque le capteur intelligent est sur site (en exploitation) une partie des modes d'utilisation et des services lui sont interdits pour des problèmes de sécurité. Il ne peut alors accéder qu'à la configuration constructeur et éventuellement changer le code d'accès de l'exploitant.

Le responsable prend en charge les étapes de configuration jusqu'à sa mise en service sur site. Il peut également, si des moyens d'étalonnage sont mis à sa disposition, configurer la chaîne de mesure en intégrantde nouveaux transducteurs en fonction des grandeurs physiques à mesurer, et étalonner le capteur.

Il est chargé de la mise en place du capteur intelligent en établissant les liaisons physiques avec le processus et le système d'automatisation. Il est également chargé de sa mise en service. Ceci inclut, par exemple :

- la définition de l'adresse physique et du libellé du capteur intelligent afin qu'il soit reconnu par les autres équipements du système d'automatisation, ainsi que les paramètres et langages de communication par défaut.
- la définition de la liste des utilisateurs autorisés à intervenir sur le capteur intelligent (mémorisation des codes d'accès des utilisateurs, des modes d'utilisation et services accessibles par les utilisateurs)
- la définition des paramètres de mesure du capteur intelligent (unité de mesure par défaut, valeur de repli par défaut, étendue de mesure par défaut, ...)
- la définition des paramètres d'archivage (liste des informations à archiver dans l'historique, ...)

Lorsque ces activités sont effectuées le capteur intelligent est intégré dans le système d'automatisation et est opérationnel pour son exploitation.

Les autres utilisateurs sont des opérateurs qui interviennent sur le système automatisé de production en phase d'exploitation (opérations de conduite, de maintenance et de gestion techniques).

La figure IV.11 montre les informations d'autorisation pour les utilisateurs d'un capteur intelligent. Les opérateurs peuvent être autorisés sur plusieurs équipements "interfacés" alors que les utilisateurs peuvent être autorisés sur plusieurs capteurs intelligents. Chaque autorisation porte des privilèges de priorité et d'accès, posés comme attributs de liens. Un utilisateur est autorisé à accéder à un ou plusieurs modes d'utilisation du capteur intelligent, mais le même mode d'utilisation peut être accessible par plusieurs utilisateurs. Chaque mode d'utilisation possède un ou plusieurs services associés. Le statut du service est posé comme attribut de lien (le statut du service permet de définir si le service est indispensable ou non pour le mode d'utilisation associé).

La liste des utilisateurs définie pour un capteur intelligent est composé d'au moins quatre utilisateurs :

deux opérateurs humains sont définis par défaut : le constructeur et le responsable du capteur intelligent sur le site d'exploitation ainsi que deux équipements "interfacés" permettant à ces deux utilisateurs la mise en service sur site.

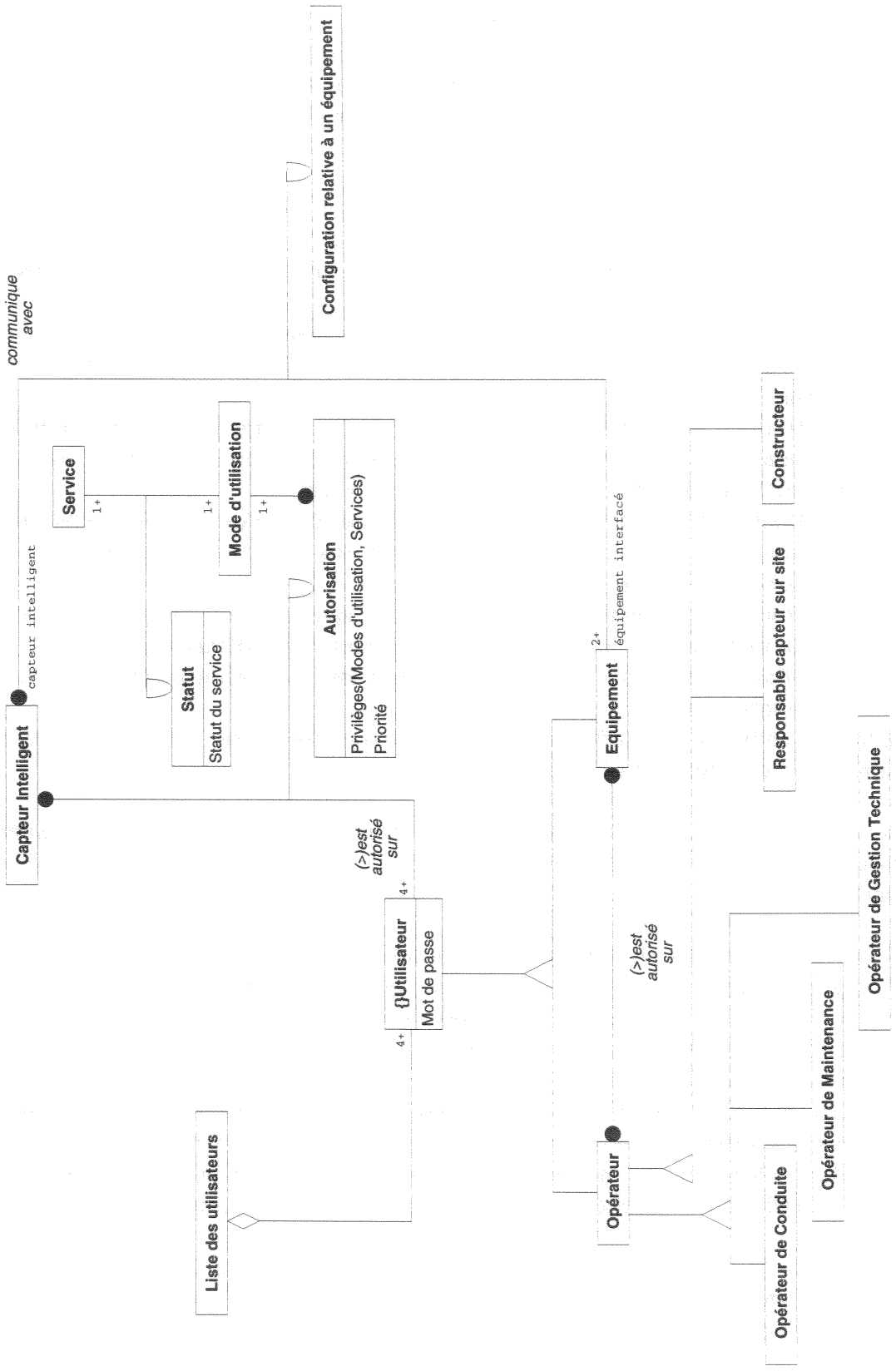


Figure IV.11 : Diagramme de classes montrant les différents utilisateurs du capteur intelligent.

IV.3.3 La configuration du capteur intelligent

Afin que le capteur intelligent puisse délivrer une mesure opérationnelle dans le référentiel de l'utilisateur, il doit être configuré, d'une part lors de sa fabrication chez le fabricant (hors site) et d'autre part lors de sa mise en place chez l'exploitant (sur site).

La configuration du capteur intelligent est composée (figure IV.12) :

- des paramètres du fabricant (numéro de série, date de fabrication, version logiciel, ...),
- des paramètres de mesure (type de transducteurs utilisé, étendue de mesure, domaine nominal d'emploi, libellé du transmetteur, unité de mesure par défaut, ...),
- de la liste des utilisateurs (intégrant entre autres les services et modes d'utilisation accessibles pour chaque utilisateur),
- des paramètres de mesure des utilisateurs (paramètres relatifs à un équipement caractérisant la mesure opérationnelle consommée : unité de mesure, période d'acquisition, valeur de repli, ...).

Les paramètres concernant le format de la mesure opérationnelle peuvent être modifiés soit par un opérateur soit par l'équipement lui-même.

Les paramètres de mesure des utilisateurs :

Ces paramètres caractérisent la manière dont est élaborée la mesure opérationnelle, ainsi que le format dans lequel elle est fournie par le capteur intelligent.

- format de la mesure :

  . la mesure est convertie dans l'unité définie par l'utilisateur,
  . un qualifieur peut être associé ou non à la mesure : le format des informations de validation associé à la mesure dépend du type de l'utilisateur,
  . datation,
  . possibilité de passage en mode de repliement : la mesure prend alors la valeur de repli spécifiée par l'utilisateur,
  . les mesures "brutes" peuvent être associées ou non à la mesure opérationnelle.

- mode de déclenchement de la mesure opérationnelle :

  . mode périodique : la mesure est élaborée à une période programmable par l'utilisateur,
  . mode apériodique : acquisition et élaboration d'une mesure dès réception d'une requête envoyée par l'utilisateur,
  . mode retardé périodique ou apériodique : acquisition et élaboration d'une mesure à partir d'une date fixée par l'utilisateur.

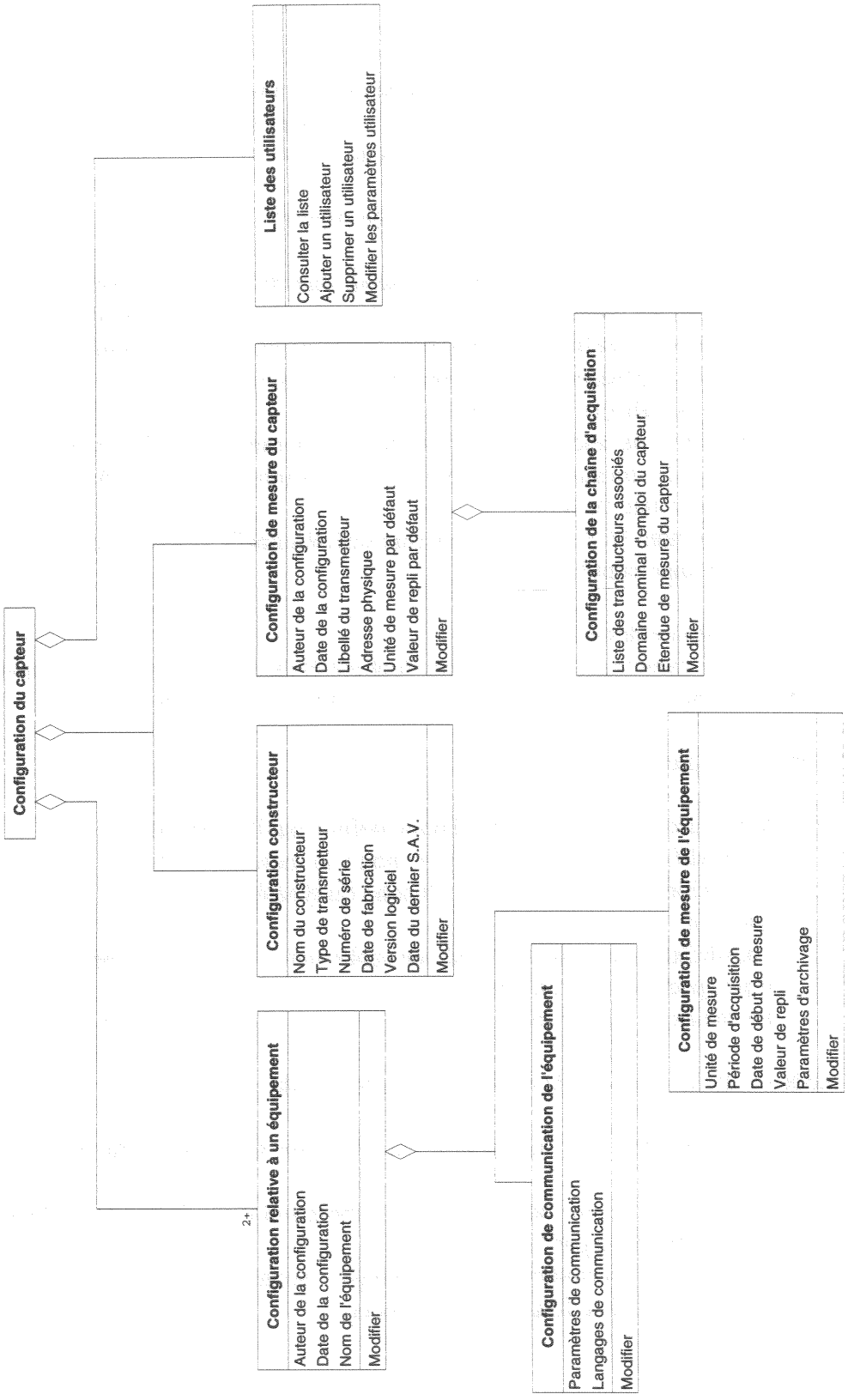


Figure IV.12 : Diagramme de classes montrant la configuration du capteur intelligent.

IV.3.4 Le mesurage

Comme nous l'avons mentionné précédemment la fonctionnalité de mesure constitue l'essence même du capteur intelligent. Le diagramme de classes de la figure IV.13 montre comment est réalisée cette fonctionnalité [LUTT-95]. Les hiérarchies de mesure correspondent aux hiérarchies d'obtention d'une mesure opérationnelle. La classe représentant ces hiérarchies est en fait une agrégation qui peut être vue de deux niveaux différents :

- une vue générale (figure IV.13) montrant les relations avec les autres classes l'utilisant. On peut citer les mesures opérationnelles (relatives aux différents utilisateurs), qui sollicitent les différentes hiérarchies de mesure afin d'obtenir une mesure validée qui sera ensuite traduite dans le référentiel de l'utilisateur.
- et une vue détaillée (figure IV.15) montrant la structure interne. Cette structure s'appuie sur les travaux du CIAME concernant le modélisation informelle du concept de capteur intelligent (chapitre I). La structure interne de chaque hiérarchie de mesure dépend du type de transducteur utilisé pour effectuer la mesure ainsi que des grandeurs physiques mises en jeu.

Une hiérarchie de mesures se décompose de la façon suivante (figure IV.14) :

Plusieurs types de mesures ont été identifiées :
- la mesure primaire : image d'une grandeur principale (mesurande)
- la mesure auxiliaire :
  - mesure auxiliaire interne : image d'une grandeur d'influence
  - mesure auxiliaire externe : mesure provenant d'un autre capteur par le réseau de communication
  - mesure interne opérationnelle : mesure connectée à une sous-hiérarchie de mesure. Ceci permet d'avoir une réentrance au niveau des hiérarchies de mesure. Par exemple, lorsque une mesure auxiliaire est élaborée à partir d'autres mesures.
- la mesure fonctionnelle : élaborée à partir de mesures primaires et auxiliaires
- la mesure technologique : image d'une grandeur ou d'une variable d'autocontrôle
- la mesure validée : élaborée à partir des mesures fonctionnelles et technologiques

Chaque classe de type "mesure" possède les opérations : élaborer, acquérir, calculer, et valider. Seule l'opération "élaborer" est accessible par les autres objets.

La propagation des opérations est l'application automatique d'une opération à un réseau d'objets quand l'opération est appliquée à un objet quelconque de départ. L'opération élaborer se propage de la mesure opérationnelle (objet de départ) aux objets constituant la hiérarchie de mesure.

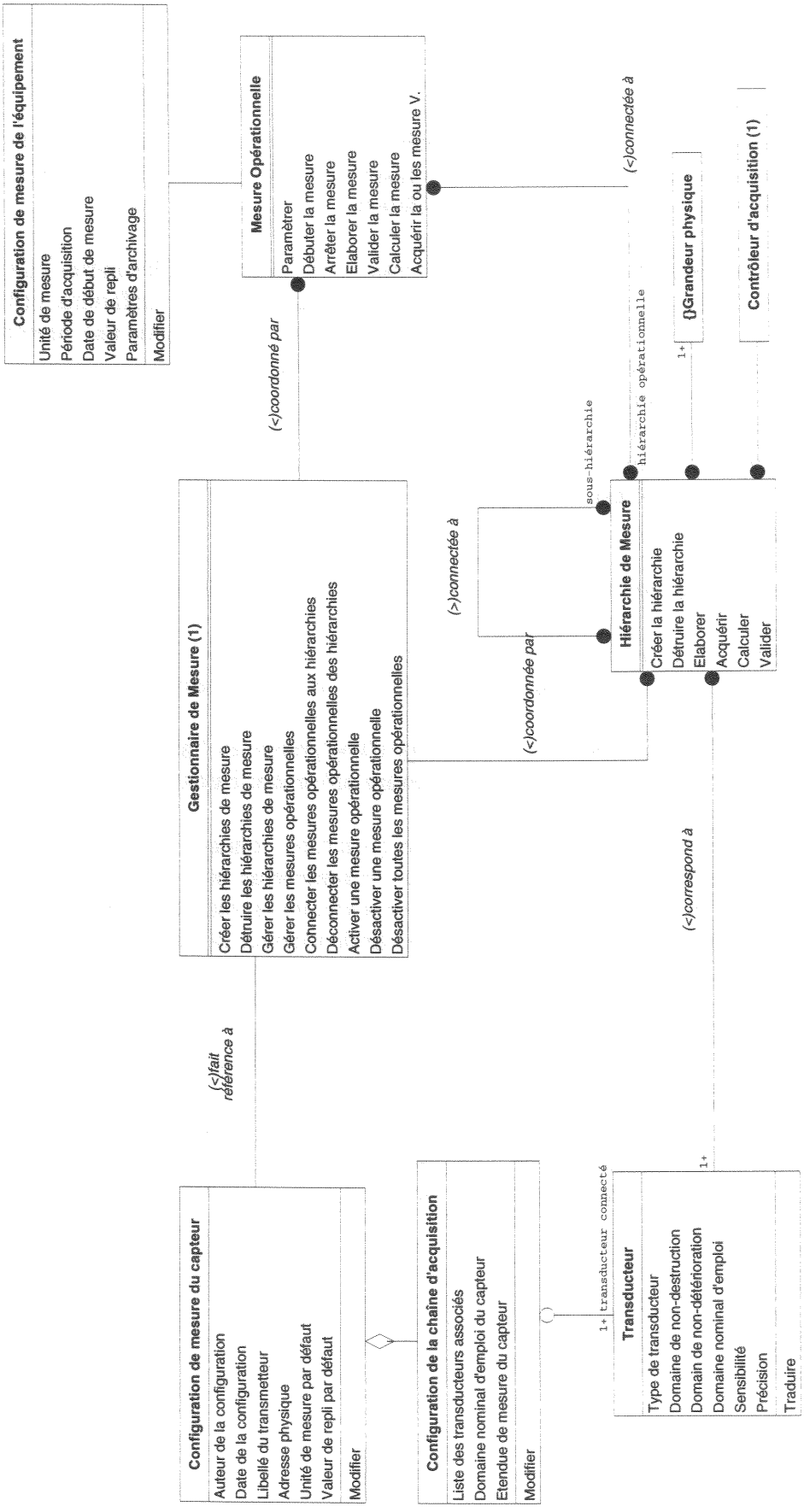


Figure IV.13 : Relations de la classe "hiérarchie de mesure" avec les autres classes.

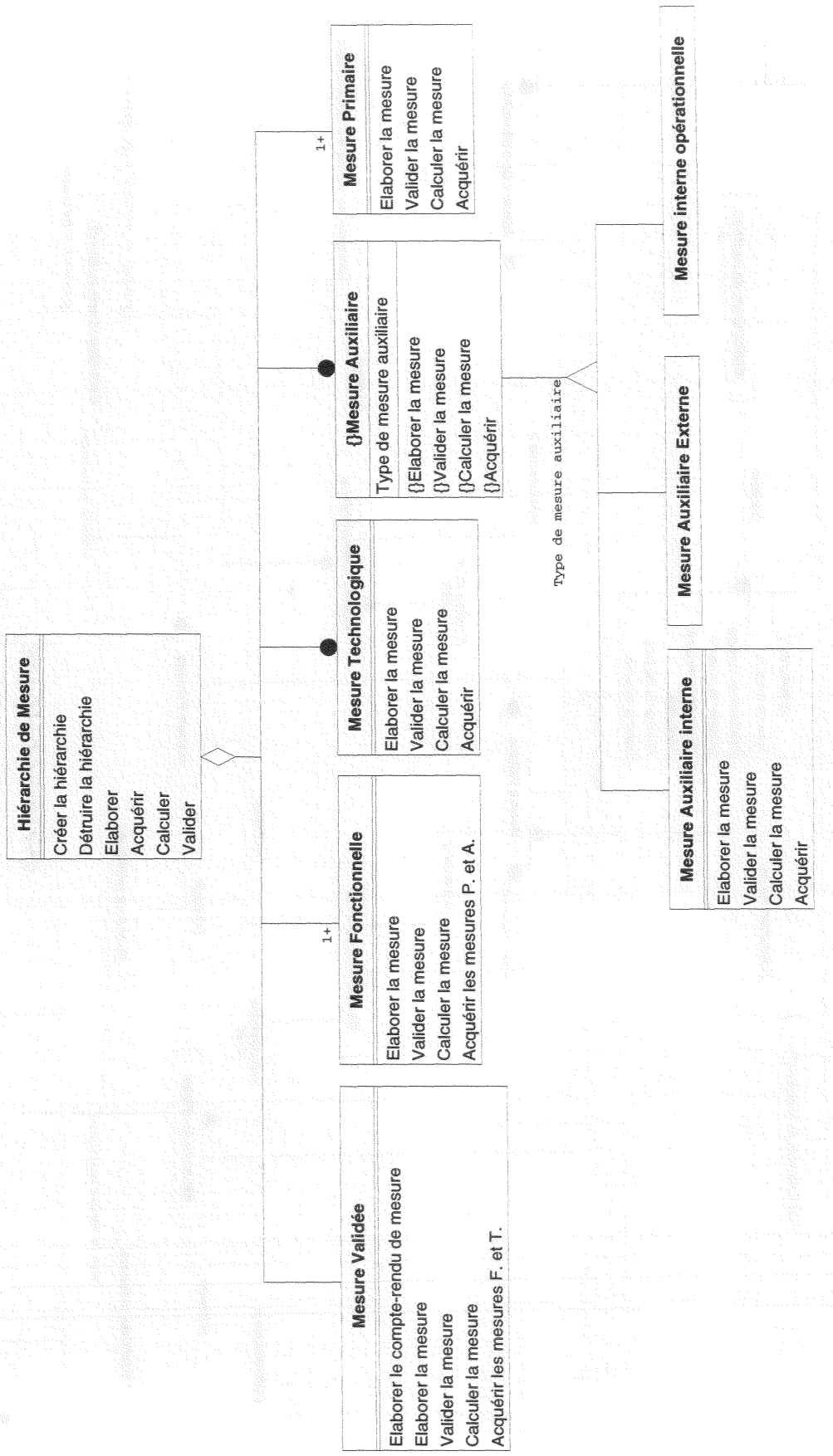


Figure IV.14 : Agrégation montrant les éléments formant la classe "hiérarchie de mesure".

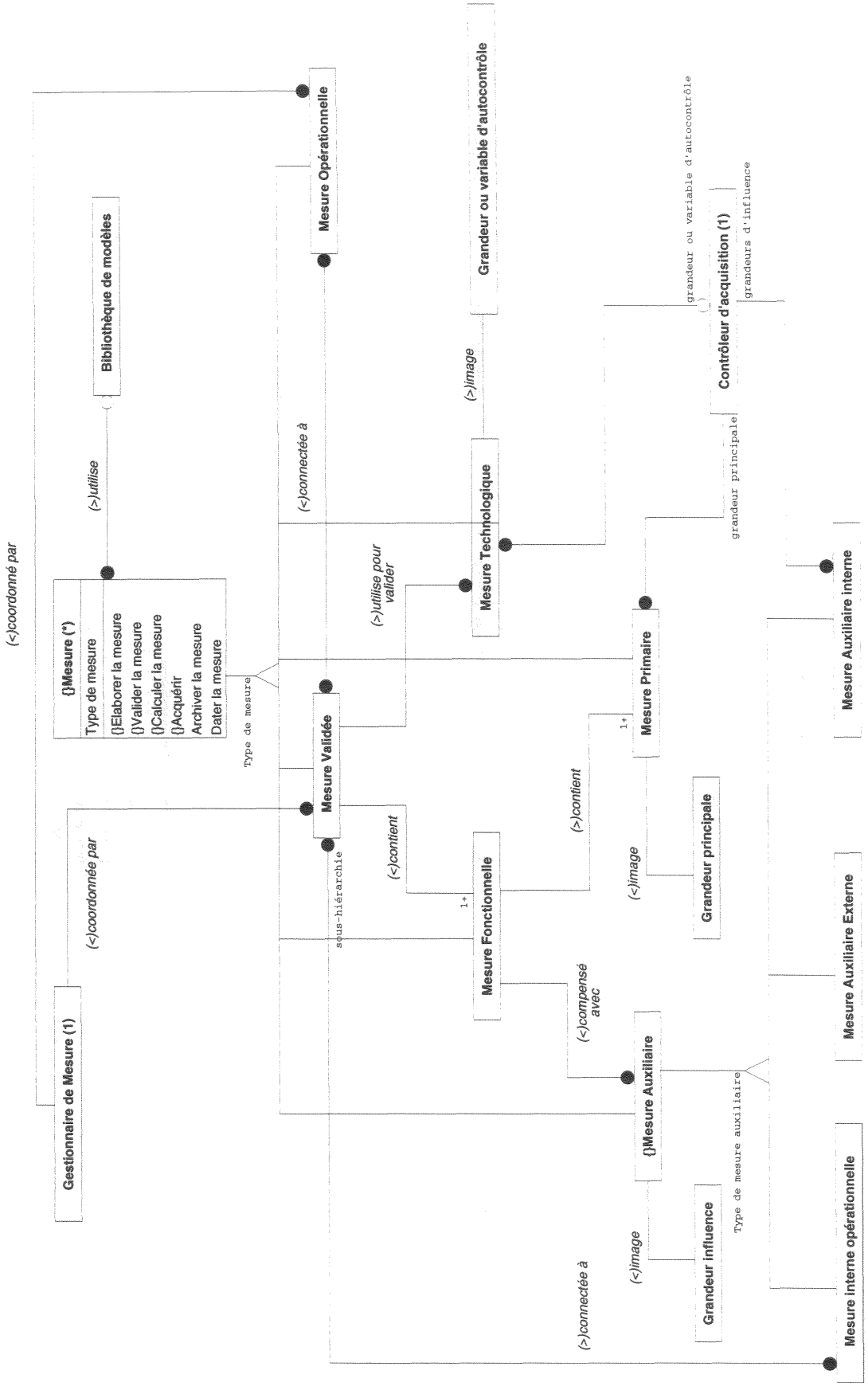


Figure IV.15 : Vue détaillée de la classe "hiérarchie de mesure".

A chaque niveau de la hiérarchie de mesure, des modèles mathématiques sont utilisés permettant ainsi l'élaboration d'une mesure opérationnelle. Ces modèles peuvent être du type polynomial, neuronal ou sous forme de table, ... Dans la plupart des cas, ils représentent les composants de la chaîne de mesure ; on peut citer par exemple, les transducteurs et les conditionneurs (amplificateur à gain programmable).

Un exemple de diagramme d'instances de la classe "hiérarchie de mesure" est donné au chapitre VI lorsque le modèle de référence est instancié à un cas particulier de capteur intelligent (capteur intelligent de température).

Les hiérarchies de mesures seront bien entendu liées au système d'acquisition afin d'avoir une image des grandeurs physiques, et seront également liées au système de communication par l'intermédiaire des mesures auxiliaires externes provenant d'autres capteurs (intelligent ou non).

IV.3.5 Les modes d'utilisation

Tout au long de son cycle de vie, le capteur fait intervenir des gens de métiers différents (en particulier les exploitants : opérateur de conduite, opérateur de maintenance, opérateur de gestion technique). La majeure partie de ces personnes peut bénéficier des services rendus par le capteur intelligent.

L'ensemble des services caractérise du point de vue externe, le fonctionnement du capteur intelligent. Cependant il est facile d'admettre par exemple, que cet équipement ne peut pas être en même temps en maintenance et en exploitation, c'est pourquoi il convient d'organiser l'ensemble des services. Il est nécessaire d'organiser l'ensemble des services en sous-ensembles cohérents, qui correspondent à des modes d'utilisation [STAR-94] [GEHI-94].

Un mode d'utilisation est un sous-ensemble de l'ensemble des services du système. A chaque mode d'utilisation est associé un ensemble (au moins un) de services (par exemple : l'ensemble des services permettant de configurer entièrement le capteur intelligent), et chaque service appartient à au moins un mode d'utilisation. Ainsi, l'ensemble des modes d'utilisation est un recouvrement de l'ensemble des services.

Chaque mode d'utilisation est relatif à une utilisation par un ou plusieurs opérateurs donnés, dans une situation donnée (phase du cycle de vie).

Un mode d'utilisation peut bien sûr être accessible par plusieurs utilisateurs. Cependant, les utilisateurs ayant accès à un même mode d'utilisation n'ont pas forcément accès au même sous-ensembles de services de ce mode.

A l'évidence, la liste des services de tout mode d'utilisation doit comporter un service spécifique de changement de mode, faute de quoi il deviendrait impossible de le quitter. La demande de changement de mode doit indiquer le mode destination. Cependant, on ne peut pas rejoindre n'importe quel mode à partir de n'importe quel autre, pour des raisons de sécurité et de cohérence de fonctionnement. Ainsi, les services de changement de mode possèdent comme tous les services des conditions d'activation exprimant entre autres les conditions logiques qui permettent le passage d'un mode à l'autre. La donnée de l'ensemble des modes et des services de passage définit entièrement la gestion des modes d'utilisation du capteur intelligent.

Selon la phase de vie dans laquelle il se situe et en fonction de ce qu'en attend l'utilisateur, le capteur intelligent peut être dans l'un des modes d'utilisation suivants lorsqu'il est en service : Configuration, Exploitation (ou Opérationnel) et Maintenance

Le tableau AIII.2 en annexe III définit quels sont les services accessibles aux utilisateurs en fonction du mode d'utilisation.

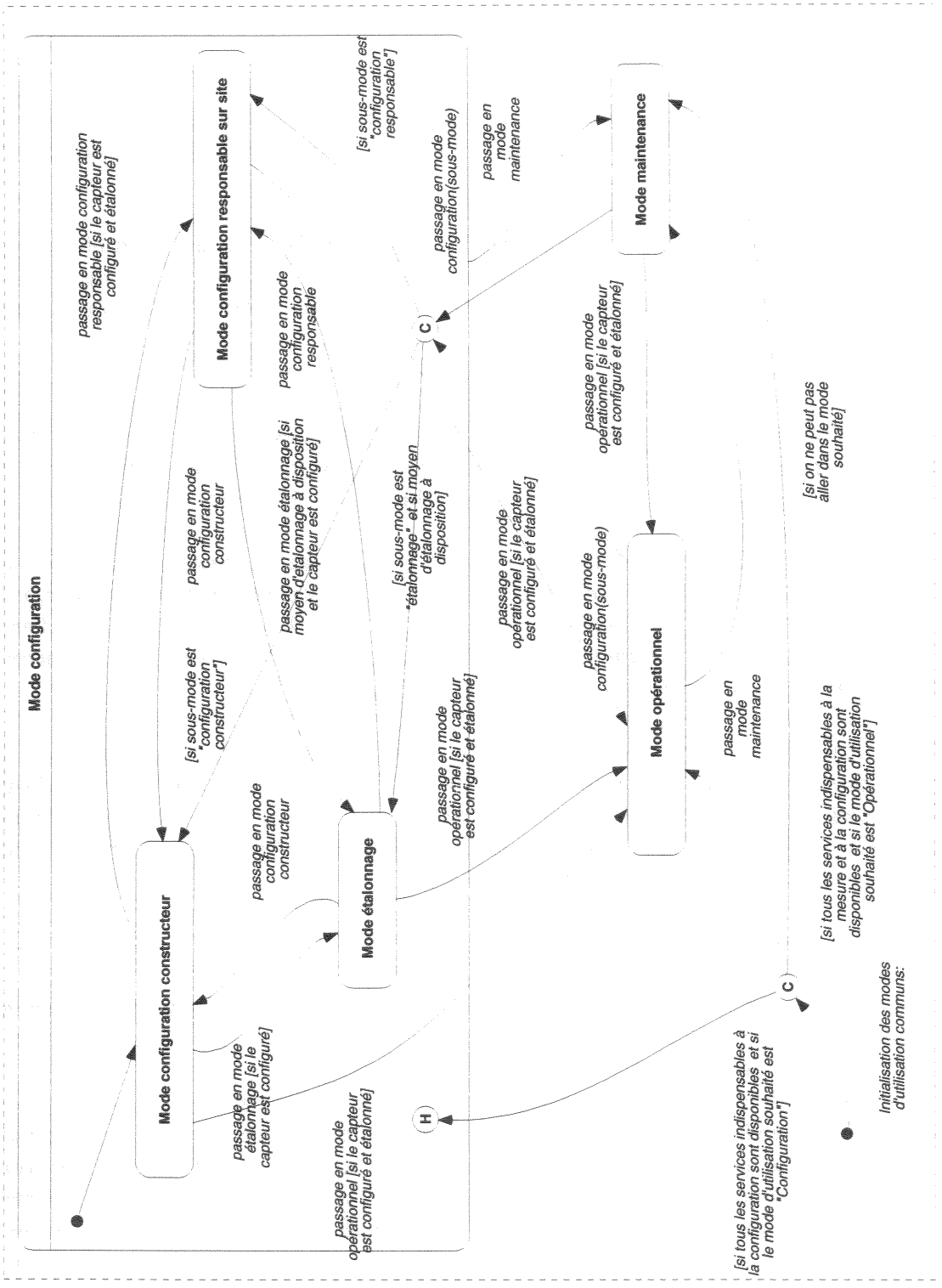


Figure IV.16 : Graphe d'enchaînement des modes d'utilisation (correspond au diagramme d'états de la classe "contrôleur").

Un service met en oeuvre un ensemble de traitements exécutés en vue d'obtenir un résultat déterminé. L'exécution de ce service est tributaire de l'état des ressources nécessaires aux traitements appartenant à ce service. Pour des raisons de robustesse face aux défaillances du capteur intelligent et/ou de son environnement, le constructeur peut avoir prévu des traitements de substitution. Dans ce cas, pour un même service, il existe plusieurs versions de traitements, chacune d'elle étant caractérisée par l'ensemble des ressources qu'elle nécessite. Un service est disponible si au moins une des versions peut être exécutée. Dans le cas contraire, le service est indisponible.

Par ailleurs, cette autorisation d'accès peut être complétée en précisant par quel(s) moyen(s) de transmission, l'utilisateur peut accéder à ce service.

L'organisation des services en modes d'utilisation constitue un précieux moyen de structuration du fonctionnement d'un capteur intelligent. A un moment donné, le capteur intelligent n'accepte que les requêtes de services cohérents avec son mode d'utilisation, émanant d'un utilisateur autorisé, ce qui accroît sa sûreté de fonctionnement [BAYA-94].

Le mode d'utilisation "configuration" permet à certains utilisateurs (le constructeur et le responsable du capteur intelligent sur le site) de définir les paramètres (paramètres constructeur, paramètres de mesure, liste des utilisateurs, ...) nécessaires à la mise en service du capteur intelligent sur le site d'exploitation.

Le mode d'utilisation "maintenance" est appelé lorsque les opérateurs de maintenance veulent effectuer des tests logiciels ou matériels qui nécessitent la plupart du temps l'arrêt de la mesure voir l'arrêt du capteur. Il peut servir également de mode de repli, par exemple : lorsque le capteur intelligent ne peut plus être configuré ou paramétré.

Le mode d'utilisation "opérationnel" correspond à une exploitation, pour laquelle le rôle du capteur intelligent est de produire et de mettre à disposition de l'exploitant :

- une mesure opérationnelle, validée, datée, compensée des grandeurs d'influence et exprimée dans le référentiel de l'utilisateur.
- des données pour la gestion technique, en particulier : historique, résultats d'auto-tests, ...

et pour laquelle une reconfiguration en ligne des paramètres de mesure du capteur est possible.

Comme nous l'avons mentionné précédemment, chaque équipement du système d'automatisation qui communique avec le capteur intelligent peut paramétrer la mesure opérationnelle qui lui est fournie. Ces paramètres (période d'acquisition, unité de mesure, ...) peuvent être modifiés "en ligne".

Ceci implique, que lorsque le capteur intelligent fournit une mesure opérationnelle à un équipement, il peut être en même temps paramétré par un autre équipement. Le mode d'utilisation "opérationnel" a été décomposé en deux sous modes d'utilisation (sous mode "configuration de la mesure opérationnelle" et sous mode "mesurage") pour chaque équipement du système d'automatisation qui communique avec le capteur intelligent.

La gestion des modes d'utilisation et des services est représenté par les figures IV.16 et IV.17. La classe "coordinateur" est chargée de gérer les requêtes (gestion des priorités) provenant des différents utilisateurs du capteur intelligent.

Les contrôles suivants sont effectués pour chaque requête des utilisateurs :

- contrôle du code d'accès,
- contrôle de l'appartenance du code commande (service) du message aux codes reconnus par le capteur intelligent,
- contrôle d'accès du service pour l'utilisateur,
- contrôle de la disponibilité du service demandé;
  A ce niveau, un superviseur (classe "gestionnaire des services") permet de contrôler le déroulement de toutes les opérations internes grâce notamment à des auto-tests, des auto-diagnostics, et aux compte-rendus d'erreurs élaborés par les autres objets du système. Il mettra à jour en "temps réel" une liste des états des services et des ressources indiquant la disponibilité de ceux-ci.
- contrôle de cohérence avec le mode d'utilisation courant,
- contrôle des données associées au service (par exemple : vérification de la taille des fichiers de paramètres pour un service de configuration).

Lorsque toutes ces vérification sont effectuées, le "coordinateur" transmet la requête "validée" a un "contrôleur utilisateur" spécifique qui exécutera le service demandé par l'utilisateur (une instance de la classe "contrôleur utilisateur" est créée pour chaque utilisateur du capteur intelligent). Les services des utilisateurs pourront alors être traités de façon parallèle sous certaines conditions (partage des ressources matérielles, temps de traitements, ...). Seuls les services relatifs à un changement de mode d'utilisation seront traités séquentiellement afin de respecter une certaine cohérence dans la gestion des modes d'utilisation et des services.

Nous avons mis à profit le mécanisme d'héritage pour la gestion des modes d'utilisation et des services :

- d'une part, en spécialisant les objets de type "contrôleur utilisateur" qui traitent les services demandés par les utilisateurs pour chaque type d'utilisateur (opérateur de conduite, de maintenance, de gestion technique et équipement "interfacé"). On a alors un "contrôleur utilisateur" spécialisé pour chaque type d'utilisateur qui traitera uniquement les services accessibles pour ce type d'utilisateur. Ceci permet de n'implémenter dans un "contrôleur utilisateur" qui a été spécialisé pour un type d'utilisateur que les services qui lui sont accessibles au lieu d'implémenter tous les services dans un seul type de "contrôleur utilisateur" pour lequel certains services pourraient ne jamais être utilisés par un type d'utilisateur.

- d'autre part, en faisant hériter le modèle dynamique d'une classe à ses sous-classes. Les sous-classes héritent à la fois des états et des transitions de leur ancêtre.

Si un diagramme d'états de super-classe et un diagramme d'états de sous-classe traitent des ensembles disjoints d'attributs, il n'y a pas de problème. La sous-classe possède alors un état composé de diagrammes d'états concurrents. Si, toutefois, le diagramme d'états de sous-classe englobe certains attributs du diagramme d'états d'une super-classe, un conflit potentiel apparaît. Le diagramme d'états d'une sous-classe doit être un affinement du diagramme d'états de la super-classe. Bien que l'affinement des diagrammes d'états hérités soit possible, le diagramme d'états d'une sous-classe doit généralement être un ajout indépendant, orthogonal et concurrent au diagramme d'états hérité de la super-classe, défini sur des ensembles différents d'attributs.

Nous avons donc fait hériter le diagramme d'état gérant les modes d'utilisations (ajout indépendant) au "coordinateur" ainsi qu'à tous les "contrôleurs utilisateurs". Ceci permet à chaque "contrôleur" ("coordinateur" et "contrôleurs utilisateurs") de connaître à tout moment le mode d'utilisation courant.

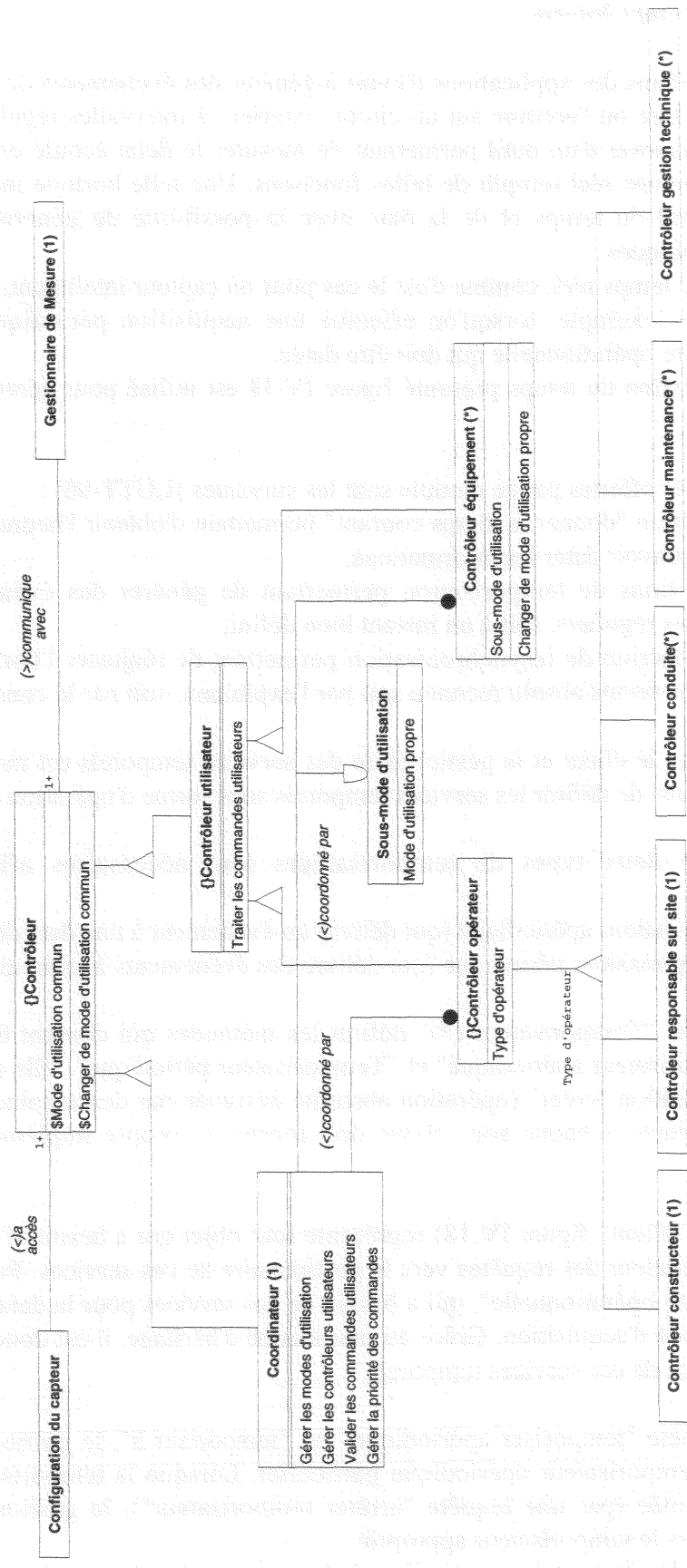


Figure IV.17 : Diagramme de classes montrant la gestion des requêtes utilisateurs.

IV.3.6 Gestion du temps interne

Il arrive souvent dans des applications d'avoir à générer des événements de manière périodique ou d'effectuer la lecture ou l'écriture sur un circuit extérieur à intervalles réguliers. Dans ce cas, il est nécessaire de disposer d'un outil permettant de mesurer le délai écoulé entre chaque mesure. Seule une horloge temps réel remplit de telles fonctions. Une telle horloge incorpore toujours un mécanisme de calcul du temps et de la date avec la possibilité de générations d'événements (interruptions) périodiques.

Dans un contexte temps réel, comme c'est le cas pour un capteur intelligent, la gestion du temps est primordiale, par exemple, lorsqu'on effectue une acquisition périodique de données afin d'élaborer une mesure opérationnelle qui doit être datée.

Le module de gestion du temps présenté figure IV.18 est utilisé pour gérer de telles fonctions relatives au temps.

Les fonctionnalités offertes par ce module sont les suivantes [LUTT-96] :

- une fonction "donner le temps courant" permettant d'obtenir l'heure et la date complète afin de pouvoir dater les informations,
- des fonctions de temporisation permettant de générer des événements, soit à des intervalles réguliers, soit à un instant bien défini,
- et une fonction de resynchronisation permettant de réajuster l'horloge temps réel par rapport au temps absolu reconnu soit par l'exploitant, soit par le constructeur.

L'association entre le client et le gestionnaire des services temporels est modélisée en tant que classe, permettant ainsi de définir les services temporels sous forme d'opération de l'association.

Il apparaît, que deux types de temporisateurs sont nécessaires afin de réaliser ces fonctionnalités :

- le temporisateur apériodique (qui délivre un événement à une date définie),
- et le temporisateur périodique (qui délivre des événements à intervalles réguliers).

La classe abstraite "Temporisateur (*)" définit les méthodes qui doivent être héritées par les sous-classes "Temporisateur apériodique" et "Temporisateur périodique". Elle définit également le protocole pour l'opération "créer" (opération abstraite désignée par des accolades) sans fournir de méthode correspondante. Chaque sous-classe doit fournir sa propre implémentation pour cette opération.

Le client (classe "client" figure IV.18) représente tout objet qui a besoin d'utiliser les services temporels; il est l'initiateur des requêtes vers le gestionnaire de ces services. Par exemple, on peut citer la classe "mesure opérationnelle", qui a besoin de ces services pour la datation des mesures et le respect de la période d'acquisition. Grâce au mécanisme d'héritage, il est donc très facile de faire hériter d'autres classes de ces services temporels.

Pour chaque requête "temporiser apériodique" ou "temporiser à", le gestionnaire des services temporels crée un temporisateur apériodique particulier. Lorsque la temporisation a expirée, ou lorsque elle est annulée (par une requête "arrêter temporisateur"), le gestionnaire des services temporels détruit alors le temporisateur approprié.

Pour chaque requête "temporiser périodique", le gestionnaire des services temporels crée un temporisateur périodique particulier. Lorsque la temporisation expire, le temporisateur est réinitialisé afin de recommencer une nouvelle temporisation. Ce type de temporisateur est très utile, puisqu'il peut être utilisé pour respecter la période d'échantillonnage de la mesure opérationnelle.

Lorsqu'il y a expiration d'un temporisateur, qu'il soit périodique ou apériodique, l'événement de fin de temporisation produit par le temporisateur est relayé par le gestionnaire des services temporels vers le client qui a fait la requête.

A chaque requête "temporiser", un identificateur correspondant au temporisateur créé est retourné au client, lui permettant par la suite, s'il le veut, d'arrêter la temporisation.

Le diagramme d'états (figure IV.19) du gestionnaire des services temporels montre comment sont gérés les différents services temporels. On peut remarquer que l'événement de fin de temporisation qui est relayé vers le client peut être redéfini par celui-ci. Ceci permet à un client qui utilise plusieurs temporisateurs, de recevoir des événements de type "fin de temporisation" ayant des noms différents. On remarque également que le "temps courant" est directement accessible par le client; ne passant pas par le gestionnaire des services temporels pour avoir l'heure et la date, les requêtes vers celui-ci sont donc réduites.

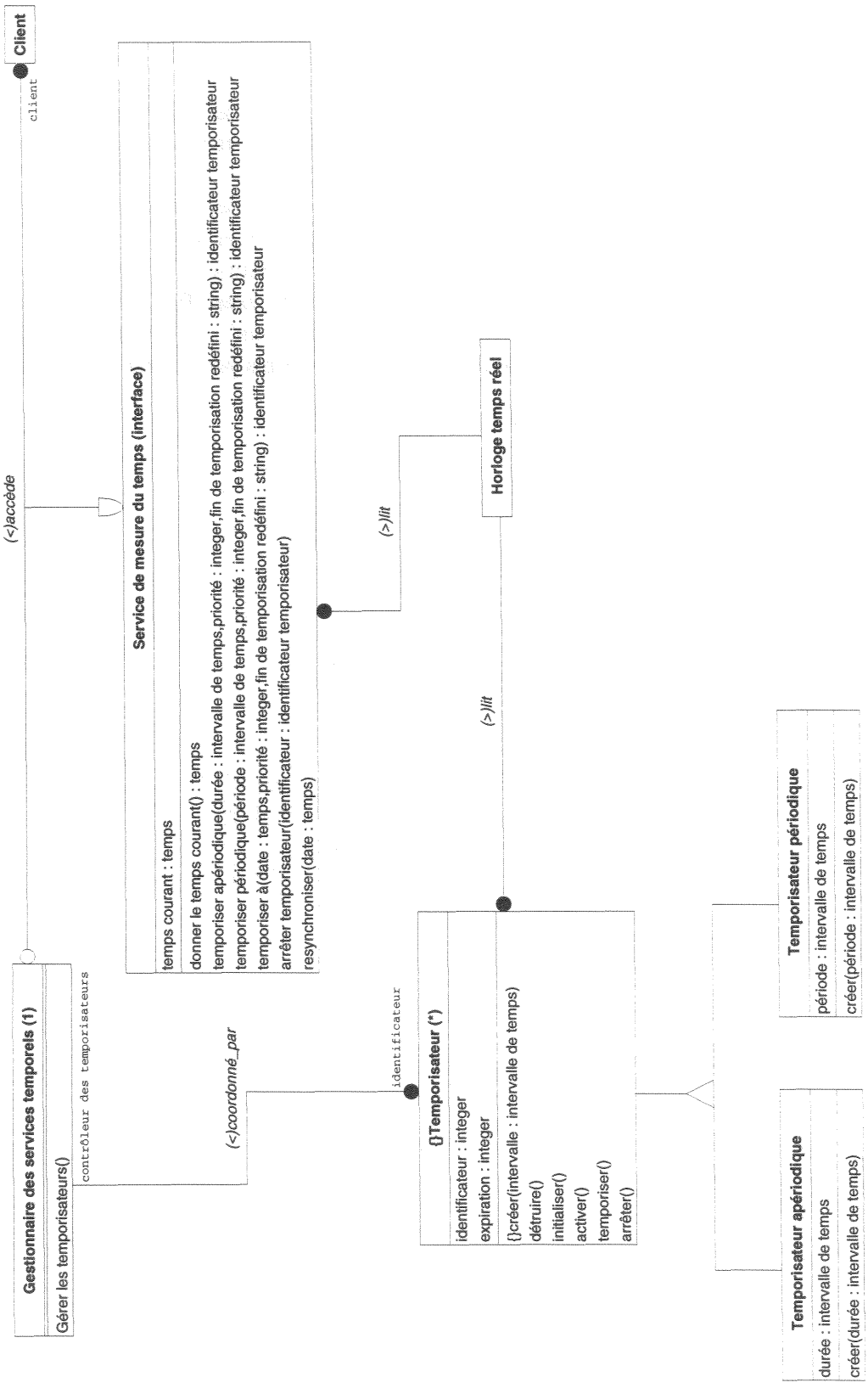


Figure IV.18 : Diagramme de classes montrant les services offerts par le module de gestion du temps.

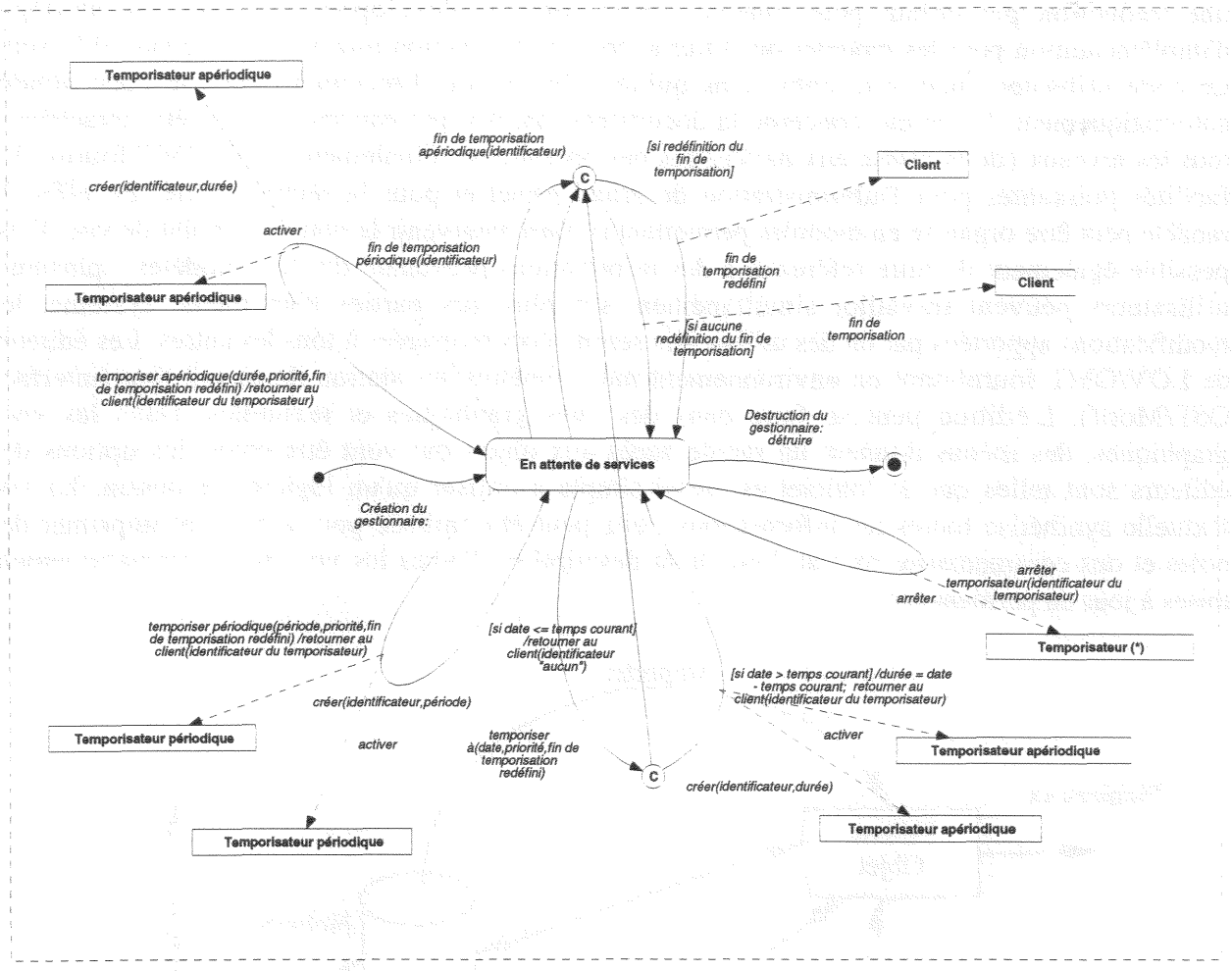


Figure IV.19 : Diagramme d'états du gestionnaire des services temporels.

Nous avons présenté dans ce paragraphe les parties du modèle les plus importantes. Nous ne donnerons pas une description complète de tous les diagrammes, en particulier pour les diagrammes d'états des différentes classes. Ces diagrammes se trouvent en annexe.

Le paragraphe suivant décrit quelques fonctionnalités du logiciel utilisé pour l'illustration des modèles OMT du capteur intelligent.

IV.4 Logiciel LOV/OMT

Le logiciel LOV/OMT est un outil commercialisé par la société VERILOG spécialement dédié à l'analyse et à la conception orientée objet. Basé sur la technique de modélisation objet OMT, LOV/OMT permet aux utilisateurs de décrire leurs systèmes avec des diagrammes de classes et des diagrammes d'instances (Éditeur objet), des diagrammes d'état et des diagrammes de flots d'événements (Éditeur dynamique) et des diagrammes de flots de données (éditeur fonctionnel). LOV fournit plus que l'addition de trois éditeurs, en effet, chaque vue se réfère aux autres vues. Les processus et les puits dans un diagramme de flots de données peuvent être des opérations et des classes du diagramme de classe, les diagrammes d'états peuvent manipuler les attributs et exécuter les opérations d'une classe. Les éditeurs de LOV/OMT permettent facilement d'exprimer la relation des points de vues structurel, comportemental et fonctionnel. L'éditeur objet de LOV permet de générer du code C++ automatiquement à partir des différents diagrammes. L'utilisateur peut choisir

une traduction par défaut pour chaque construction ou développer une nouvelle stratégie d'implémentation pour les modèles ou entrer le code de la fonction manuellement quand il le veut. Le code utilisateur peut être entré sans quitter l'éditeur ou directement dans le code généré automatiquement. En ce qui concerne la documentation, des annotations peuvent être attachées à tous les niveaux (de la classe aux paramètres des opérations). Finalement, LOV/OMT fournit des facilités puissantes pour l'administration de grand projet et pour la réutilisabilité. En effet, le modèle peut être organisé en modules permettant de faire intervenir la notion de point de vue. Il est possible également de faire référence à des informations provenant d'autres modèles : plusieurs utilisateurs peuvent travailler simultanément sur plusieurs parties d'un même système, les modifications apportées par un des utilisateurs seront alors propagées à tous les autres. Les éditeurs de LOV/OMT fournissent un environnement multi-fenêtres sur station de travail Sun (interface OSF/Motif). L'édition peut se faire dans des vues graphiques et textuelles. Dans les vues graphiques, des menus donnent un rapide accès aux objets qui vont être crées, les options des éditeurs sont telles que le logiciel est aussi simple à utiliser qu'un logiciel de dessin. La vue textuelle synthétise toutes les informations : elle peut être utilisée pour ajouter et imprimer des notes et des commentaires tout au long de la description. Toutes les vues sont automatiquement mises à jour en permanence.

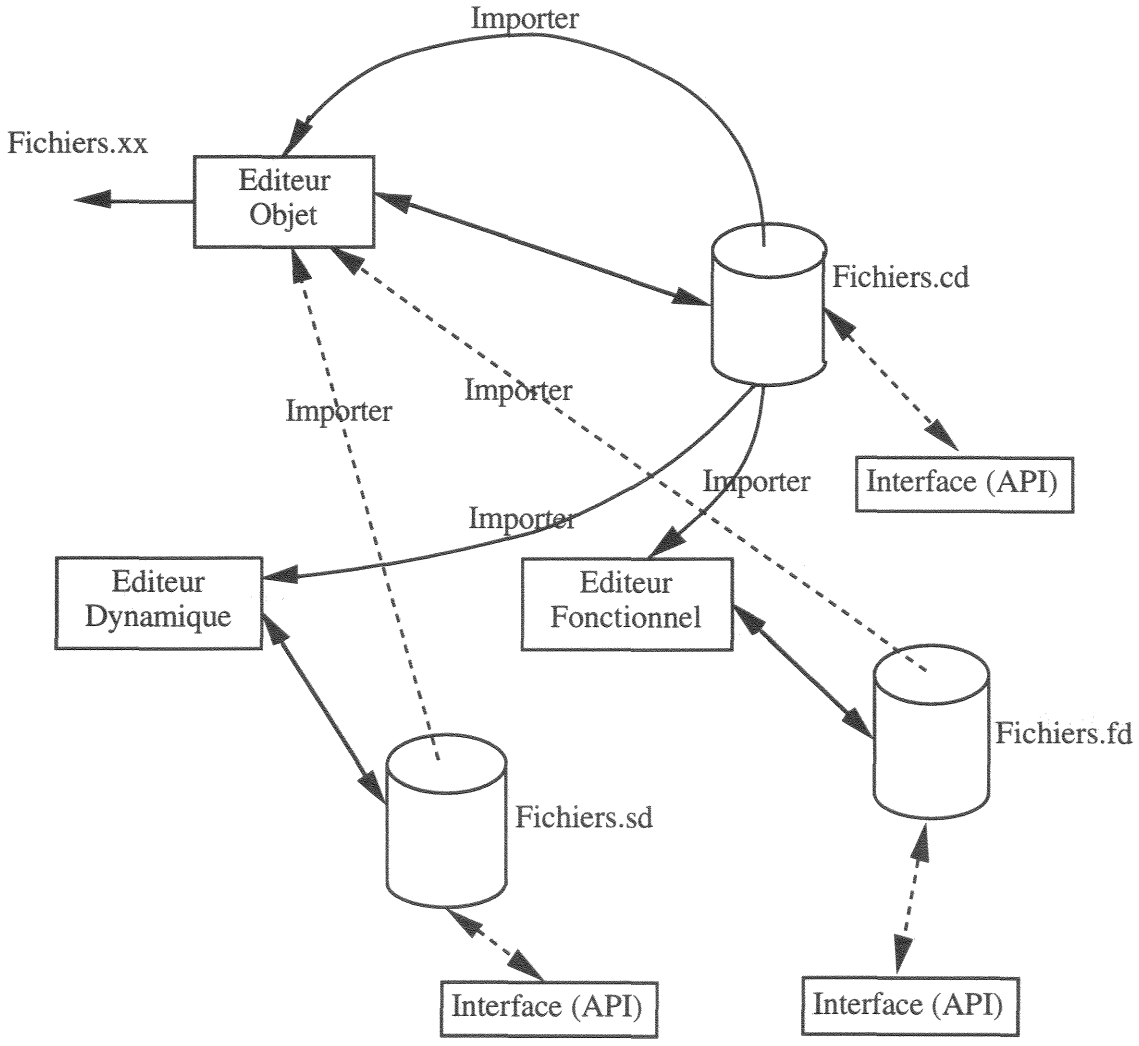


Figure IV.20 : Organisation de l'outil LOV/OMT.

API : Application Program Interface

IV.5 Conclusion

Une modélisation formelle du concept de capteur intelligent basée sur les cas d'utilisation intégrant les besoins utilisateurs et le savoir-faire des constructeurs doit permettre la réalisation d'un modèle de référence de capteur intelligents.

Ce chapitre présente la démarche de modélisation basée sur le concept de l'objet mis en oeuvre par l'outil LOV/OMT support de la méthode OMT.

La démarche de modélisation est appliquée à la modélisation d'un capteur intelligent de température au chapitre VI.

IV.6 Références

[ANDR-94] André J. et Brisson G., "Modélisation objet d'un poste de travail", Colloque ERGO'IA, Ergonomie et Infirmatique Avancée, 26-28 octobre 1994, Biarritz, France.

[BAYA-94] Bayart M., "Modes de marche d'un équipement de système automatisé", Journées d'étude S3 : sûreté - surveillance - supervision, Stratégie de conduite en présence de défaillances, Paris, 8 décembre 1994.

[CHEN-76] Chen Peter Pin-Shan, "The entity-relationship model : toward a unified view of data", ACM transactions on databese systems, Vol.1, N°1, March 1976, pages 9-36.

[COLE-92] Coleman D., Hayes F., Bear S., "Introducing objectcharts or how to use statecharts in object-oriented design", IEEE transactions on software engineering, Vol.18, N°1, January 1992, pages 9-18.

[GEHI-94] Gehin A.L., "Analyse fonctionnelle et modèle générique des capteurs intelligents : application à la surveillance de l'anesthésie", thèse de l'université des sciences et technologies de Lille, 26 janvier 1994.

[HARE-87a] Harel D., Pnueli A., Schmidt J.P., Sherman R., "On the formal semantics of statecharts", Proceedings of the 2nd IEE symposium on logic in computer science, Tthaca, N.Y., June 22-24 1987, pages 54-64.

[HARE-87b] Harel David, "Statecharts : a visual formalism for complex systems", Science of computer programming, Vol.8, 1987, pages 231-274.

[HARE-88a] Harel D., "On visual formalisms", Communications of the ACM, Vol.31, N°5, May 1988, pages 514-531.

[HARE-88b] Harel D. et al., "Statemate : a working environment fot the development of complex reactive systems", pages 396-406, Proceedings of the 10th IEEE international conference on software engineering, April 1988.

[JACO-93] Jacobson I., "Le genie logiciel orienté objet : une approche fondée sur les cas d'utilisation", Éditions Addison-Wesley France, Septembre 1993, ISBN 2-87908-42-8, 522 pages.

[JACO-94a] Jacobson I., "Basic Use-Case Modeling", Report on Object Analysis and Design, Vol.1, N°2, July-August 1994.

[JACO-94b] Jacobson I.," Basic Use-Case Modeling (continued)", Report on Object Analysis and Design, Vol.1, N°3, September-October 1994.

[JACO-94c] Jacobson I., "Use Cases and Objects", Report on Object Analysis and Design, Vol.1, N°4, November-December 1994.

[JACO-95a] Jacobson I., Sten Jacobson, "Beyond methods and Case : The software engineering process with its integral support environment", Object magazine, January 1995.

[JACO-95b] Jacobson I., "Use Cases in Large-Scale Systems", Report on Object Analysis and Design, Vol.1, N°6, March-April 1995.

[LARO-95] Laroque P., Perrin P., "Présentation de la métode Object Modeling Technique (OMT)", AFCET Groupe de travail COOSI, 1995.

[LUTT-95] Luttenbacher D. et al., "Intelligent sensor : object approach", Control Eng. Practice, Vol.3, N°6, pages 805-812, 1995.

[LUTT-96] Luttenbacher D. et al., "Application of object-oriented technology to intelligent sensor development", Measurement, Vol.17, N°3, pages 201-216, 1996.

[ODEL-95] Odell James, "Approaches to finite-state machine modeling", Journal of Object-Oriented Programming, Vol.7, N°8, January 1995.

[RUMB-91a] Rumbaugh J. et al., "Object-Oriented modeling and design", Prenice-Hall International Editions, 1991, 500 pages.

[RUMB-91b] Rumbaugh J. et al., "Solutions Manuel : Object oriented modeling and design", Prentice hall, 1991, ISBN 0-13-629858-3, 265 pages.

[RUMB-93] Rumbaugh J., "Controlling code. How to implement dynamic models", Journal of Object-Oriented Programming, Vol.6, N°2, May 93.

[RUMB-94] Rumbaugh J., "Getting started : Using use cases to capture requirements", Journal of Object-Oriented Programming, Vol.7, N°5, September 1994.

[RUMB-95a] Rumbaugh J., "OMT : The development process", Journal of Object-Oriented Programming, Vol.8, N°2, May 1995.

[RUMB-95b] Rumbaugh J., "OMT : The object model", Journal of Object-Oriented Programming, Vol.7, N°8, January 95.

[RUMB-95c] Rumbaugh J. et al., "OMT : Modélisation et conception orientées objet", édition francaise revue et augmentée, Édition Masson et Prentice-Hall, 1995, ISBN 2-225-84684-7, 520 pages.

[RUMB-95d] Rumbaugh J., "OMT : The dynamic model", Journal of Object-Oriented Programming, Vol.7, N°9, February 1995.

[RUMB-95e] Rumbaugh J., "OMT : The functional model", Journal of Object-Oriented Programming, Vol.8, N°1, March-April 1995.

[SONG-91] Song Y.Q., "Etude de performances de FIP, aide au dimensionnement d'applications", Thèse de doctorat de l'INPL, juillet 1991.

[STAR-94] Staroswiecki M., Bayard M., "Actionneurs intelligents", série automatique, Édition Hermès, Paris, 1994, 218 pages.

[TELE-93] "Réseaux industriels : le grand bond des performances", Télecoms magazine, N°20, mars 1993, pages 37-53.

[TERR-95] "Réseaux de cellule, de terrain, de capteurs-actionneurs, ... ou sont les frontières ?", CiMax : Edition Terrain, N°4, Avril-Mai-Juin, 1995.

[UNIF-95a] Unified Method for Object-Oriented Development, Document Set Version 0.8, Grady Booch, James Rumbaugh, Rational Software Corporation, 1995.

[UNIF-95b] Introduction to the Unified Method, Unifying the Booch & OMT Methods, James Rumbaugh, Grady Booch, Rational Software Corporation, 1995.

[WARD-86] Ward Paul T., "The transformation schema : an extension of the data flow diagram to represent control an timing", IEEE transactions on software engineering, vol. SE-12, N°2, February 1986, pages 198-210.

[WOOD-88] Woodman M., "Yourdon dataflow diagrams : a tool for disciplined requirements analysis", Information and software technology, Vol.30, N°9, November 1988, pages 515-533.

[YOUR-89] Yourdon E., "Modern structured Analysis", Englewood cliffs, New Jersey: Yourdon press, 1989, 672 pages.

Références URL (Uniform Resource Locator) sur les réseaux de terrain (ces références sont susceptibles d'être modifiées):

{URL-RT1} "http://cran.esstin.u-nancy.fr/CRAN/Cran/ESSTIN/FieldBus.html", FieldBus.

{URL-RT2} "http://www.industry.net/isa96/ffield.htm", Fieldbus on Final Approach.

Chapitre V

Réutilisation : cas du modèle de référence de capteur intelligent

V.1 Introduction

Bien que l'on en parle depuis la décennie 1960, la réutilisabilité (logicielle et/ou matérielle) est rarement pratiquée dans les faits [COAD-91].

La réutilisation à l'intérieur d'un projet demande un certain investissement et doit être institutionnalisée. Cela signifie que les opportunités de réutilisation doivent être recherchées et récompensées. Une telle activité implique l'identification d'opportunités pour parties communes, la découverte d'habitude par des revues d'architecture et d'implémentation et l'exploitation de ces opportunités, d'habitude par production de nouveaux composants ou adaptation de ceux qui existent. La réutilisation de ces composants tend alors à suggérer des améliorations futures et des généralisations [BOOC-86].

La planification d'une réutilisation future demande plus de prévoyance et représente un certain investissement.

Le premier avantage de la réutilisation est une productivité plus élevée. Cependant, les gains sont rarement aussi importants, car bénéficier de la réutilisabilité demande :

- Tout d'abord, un investissement en capital pour créer des composants réutilisables qui est amorti par le nombre de systèmes nouveaux qui pourront faire usage de ces composants;
- Un investissement pour assurer un plus haut niveau d'assurance qualité que celui qui serait normalement attendu dans un composant logiciel à usage unique;
- Un investissement pour maintenir les bibliothèques, et autres outils pour que les ingénieurs logiciels trouvent le composant au moment où ils en ont besoin. Typiquement, cela peut être atteint au mieux en déléguant l'activité de réutilisation à certaines personnes. Celles-ci seront alors responsables de la gestion des bibliothèques de composants. Elles devront chercher activement des occasions pour utiliser des parties communes et faire en sorte qu'elles soient exploitées ; souvent ils produisent et adaptent des composants pour l'usage général à l'intérieur du projet ou de toute l'organisation [BOOC-93].

Une autre raison de mettre l'accent sur la réutilisabilité est d'accroître la qualité. Un composant réutilisable requiert plus d'assurance qualité que son homologue non réutilisable. Les composants ayant un taux de réutilisation fort auront une qualité supérieure à celle des composants ordinaires; les bogues sont évacués plus rapidement et plus complètement.

V.2 Niveau de réutilisabilité

Comme nous l'avons mentionné dans le chapitre III, le cycle de vie de conception et de réalisation du capteur intelligent doit prendre en compte la réutilisation tout au long du développement. Ci-dessous sont présentés les différents niveaux de réutilisabilité lors du développement correspondant aux phases d'acquisition et d'archivage des composants réutilisables [COAD-91].

Réutilisation du code

Le terme réutiliser le code est d'ordinaire interprété comme un appel à un module de bibliothèque, mais cela peut également prendre une des diverses formes listées ci-dessous.

Héritage : Les langages de programmation orientés objet disposent de l'héritage simple ou multiple. Ceci procure un mécanisme pour réutiliser par extension.
On pense parfois que l'utilisation de l'héritage va accroître la réutilisation de code dans un programme, alors qu'une véritable relation super-classe/sous-classe n'existe pas. Pour ne pas utiliser cet héritage, on emploie plutôt la délégation [RUMB-95]. La délégation offre le mécanisme approprié pour assurer la réutilisation désirée du code. L'opération est prise dans la classe considérée et envoyée dans une autre classe pour l'exécution véritable. Les noms des opérations dans la classe d'accueil peuvent différer de ceux de la classe procurant les opérations. Chaque classe doit choisir les noms les plus adaptés à ses objectifs.

Liens binaires : Avec la plupart des langages de programmation, la compilation est suivie par une phase d'édition de liens au cours de laquelle tous les modules requis pour le code objet sont liés. Ceci peut inclure des modules de bibliothèque préalablement compilés que le programmeur d'application utilise dans son programme.

Il est fréquent de vouloir du code qui a été développé pour une application ayant des conventions d'interface différentes. Plutôt que d'insérer un appel direct à du code externe, il est plus sûr d'encapsuler son comportement dans une opération ou une classe. De cette manière, le code de la procédure ou du paquetage externe peut être modifié, et votre code ne devra être modifié qu'à un seul endroit [RUMB-95].

Réutilisation des résultats de la conception

Plutôt que de réutiliser le code pourquoi ne pas réutiliser le modèle de la conception du programme. La raison la plus évidente pour se placer à ce niveau de réutilisabilité est de faciliter le portage d'une application sur une plate-forme matérielle et ou logicielle entièrement différente.

Réutilisation des résultats de l'analyse

Un niveau encore plus élevé de réutilisation est celui de la réutilisation des spécifications, consistant en un modèle d'analyse orienté objet.

Ce niveau de réutilisation serait adéquat si l'équipe projet souhaitait convertir un système d'une technologie matérielle ancienne vers une technologie plus récente et plus puissante. Les besoins utilisateurs resteraient les mêmes, mais l'architecture interne du système, telle que documentée dans les diagrammes de conception orientée objet pourrait être entièrement nouvelle.

D'autre exemples de réutilisation.

Les bibliothèques de composants orientés objet : Un des avantages les plus significatifs de disposer d'une bibliothèque de classes réutilisables est que la réutilisation peut être accomplie non pas en modifiant le code existant, mais plutôt en étendant et en spécialisant les classes trouvées dans la bibliothèque, au travers des mécanismes d'héritage.

Comme il a été mentionné dans le chapitre III, la première phase liée à la réutilisation est la phase d'acquisition durant laquelle on identifie les parties du projet réalisées qui sont réutilisables. Étant donné qu'il n'existe pas une façon universelle de faire des composants réutilisables, mais des techniques adaptées au type de composants à obtenir, nous allons seulement présenter quelques-unes de ces techniques qui permettront de constituer des composants réutilisables selon les différents types de formalismes que l'on utilise (modèle objet et modèle dynamique).

Grâce à l'utilisation de l'héritage, les techniques orientées objets permettent très facilement de définir un modèle de référence pour des capteurs intelligents particuliers. La figure V.1, représente une hiérarchie d'héritage pour les capteurs intelligents de température en tenant compte des diverses méthodes de mesure.

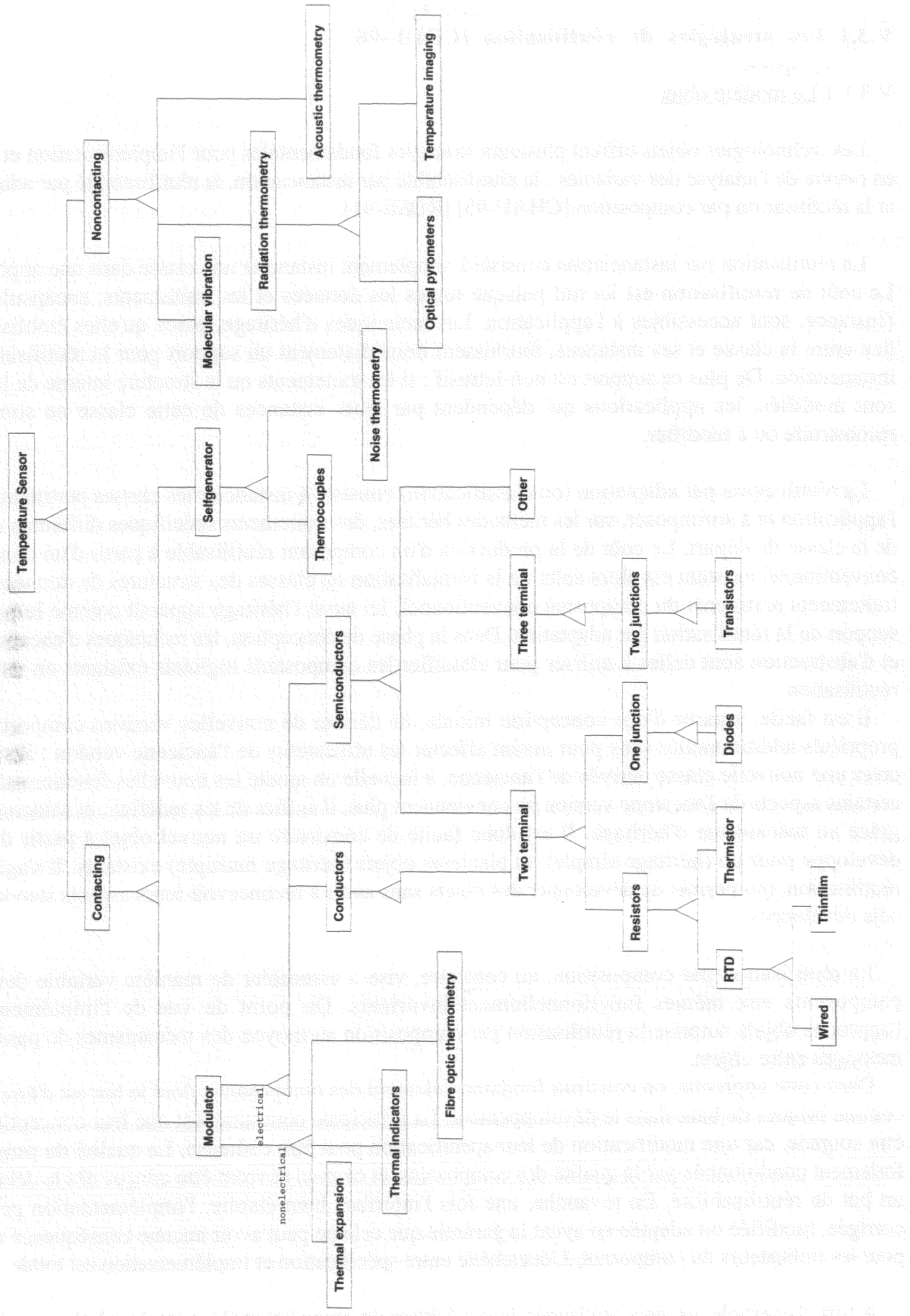


Figure V.1 : Taxinomie des capteurs de température inspirée de [HEND-93].

V.3 Fabriquer les composants réutilisables

V.3.1 Les stratégies de réutilisation [COUL-96]

V.3.1.1 Le modèle objet

Les technologies objets offrent plusieurs stratégies fondamentales pour l'implémentation et la mise en oeuvre de l'analyse des variantes : la réutilisabilité par instanciation, la réutilisabilité par adaptation, et la réutilisation par composition [CHAU-95] [ROSE-91].

La réutilisation par instanciation consiste à simplement instancier une classe dans une application. Le coût de réutilisation est ici nul puisque toutes les données et les traitements, encapsulés dans l'instance, sont accessibles à l'application. Les techniques d'héritage, parce qu'elles établissent un lien entre la classe et ses instances, fournissent immédiatement un support pour la réutilisation par instanciation. De plus ce support est non-intrusif : si les traitements ou la structure interne de la classe sont modifiés, les applications qui dépendent par leurs instances de cette classe ne sont pas à reconstruire ou à modifier.

La réutilisation par adaptation (ou classification) consiste à instancier les classes pertinentes pour l'application et à surimposer, sur les méthodes héritées, des traitements spécifiques différents de ceux de la classe de départ. Le coût de la production d'un composant réutilisable à partir d'un composant conventionnel existant est alors celui de la formalisation en classes des structures de données et des traitements pertinents du composant conventionnel. Ici aussi l'héritage apparaît comme le principal support de la réutilisation par adaptation. Dans la phase de conception, les techniques d'encapsulation et d'abstraction sont celles à utiliser pour classifier les composants logiciels existants en vue d'une réutilisation.

Il est facile, à partir d'une conception initiale, de dériver de nouvelles versions comportant des propriétés additionnelles sans pour autant affecter les utilisateurs de l'ancienne version : il suffit de créer une nouvelle classe dérivée de l'ancienne, à laquelle on ajoute les nouvelles fonctionnalités. Si certains aspects de l'ancienne version ne conviennent plus, il suffira de les redéfinir, et seulement eux grâce au mécanisme d'héritage. Il est donc facile de construire un nouvel objet à partir du code développé pour un (héritage simple) ou plusieurs objets (héritage multiple) existants. Il s'agit d'une réutilisation, qui permet de développer des objets sans avoir à reconcevoir leurs aspects standards ou déjà développés.

La réutilisation par composition, au contraire, vise à assembler de manière variable des petits composants eux mêmes fonctionnellement invariants. Du point de vue de l'implémentation, l'approche objets autorise la réutilisation par composition au moyen des mécanismes de passage de messages entre objets.

Dans cette approche, on construit fondamentalement des composants, dont le but est d'être utilisé comme briques de base dans le développement. La principale contrainte est que leur conception doit être soignée, car une modification de leur spécification peut être coûteuse. La qualité du projet sera fortement conditionnée par la qualité des composants, et ceux-ci devront être conçus dès le début dans un but de réutilisabilité. En revanche, une fois l'interface bien établie, l'implémentation peut être corrigée, modifiée ou adaptée en ayant la garantie que cela ne peut avoir aucune conséquence néfaste pour les utilisateurs du composant. L'étanchéité entre spécification et implémentation est totale.

A titre d'exemple, on peut envisager de synthétiser un ensemble d'objets et de relations en un seul objet (objet composite) dont les attributs et méthodes représentent les principaux attributs, méthodes et

relations des objets du composant [RUMB-94]. Ceci permet de simplifier la compréhension du composant.

Dans la phase de réutilisation, on utilisera cette vue générale pour montrer les objets qui utilisent le composant. Dans une vue plus détaillée, on montrera quels sous-objets du composant sont utilisés.

Lors de la fabrication de ce type de composant, il faut examiner toutes les factorisations possibles entre objets par héritage. Une factorisation peut être le point de départ de la découverte d'un nouveau composant bien plus général que ce que l'on imaginait. Il faut vérifier également que les différents objets qui constituent le composant ne sont pas déjà connus. Si c'est le cas, le nouveau composant utilise un autre composant. Il faut l'indiquer.

L'objet composite est une agrégation qui peut être vue de deux niveaux différents, l'un détaillé et l'autre comme un simple objet abstrait. Les objets composites ne sont pas des sous-systèmes et par conséquent n'encapsulent pas les classes qui le composent. La même classe peut apparaître dans plusieurs objets composites différents, mais une instance donnée ne peut jamais faire partie de plus d'un objet composite.

Dans la plupart des cas, il est utile d'avoir un objet dans l'objet composite qui gère les informations communes du composite. Celui-ci est appelé objet dominant. Un objet dominant doit avoir une multiplicité de un avec son composite.

V.3.1.1.1 Application des stratégies de réutilisation au cas du modèle objet du capteur intelligent

Les trois diagrammes de classes présentées ci-dessous sont issus du modèle objet (cf. chapitre IV § IV.3). Ils montrent comment est utilisé le concept d'objet composite pour la mise en oeuvre de composants réutilisables permettant de réaliser la fonctionnalité de mesure du capteur intelligent. Cette portion du modèle objet devra être par la suite adaptée lorsque l'on spécialisera le modèle de référence à un cas particulier de capteur intelligent. Quelques techniques permettant d'adapter ces composants sont proposées dans le prochain paragraphe.

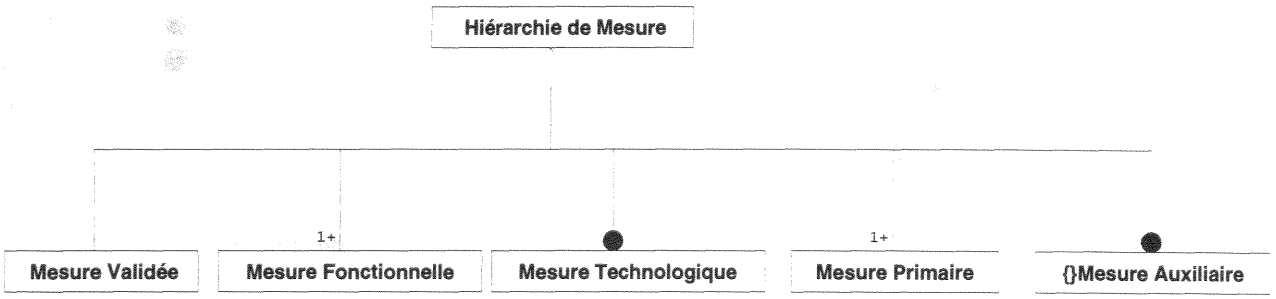


Figure V.2 : Agrégation montrant quelles sont les sous-classes de la classe composite "hiérarchie de Mesure".

L'objet dominant qui gère les informations communes de l'objet composite (instance de la classe "Hiérarchie de mesure") est une instance de la classe "Mesure Validée". Ces informations sont essentiellement des données relatives aux types de transducteurs utilisés, ainsi qu'aux grandeurs physiques mises en jeux pour faire la mesure.

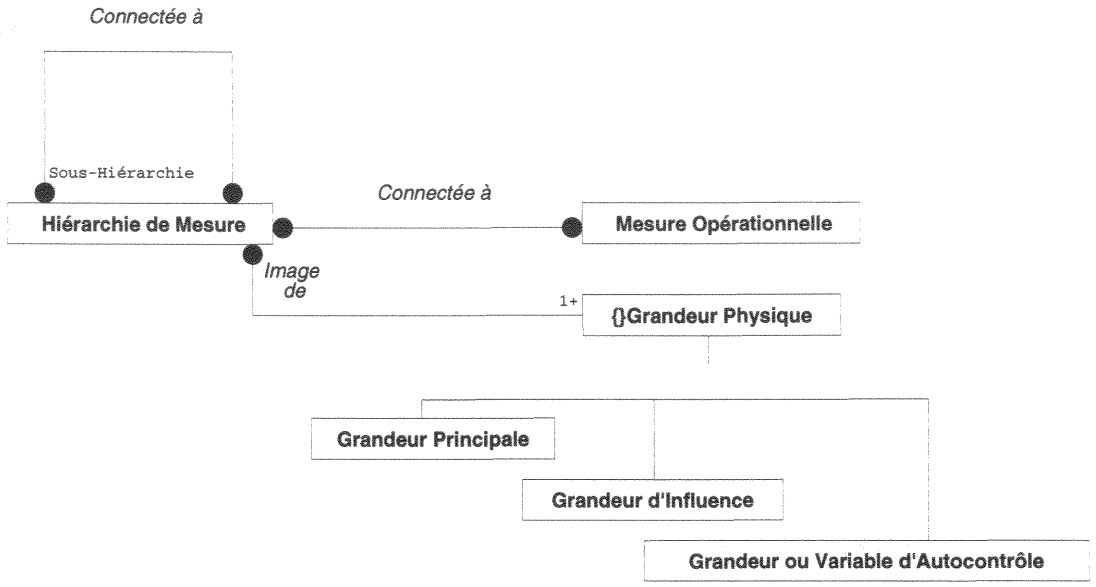


Figure V.3 : Vue générale montrant les relations de la classe composite "Hiérarchie de mesure" avec les autres classes l'utilisant.

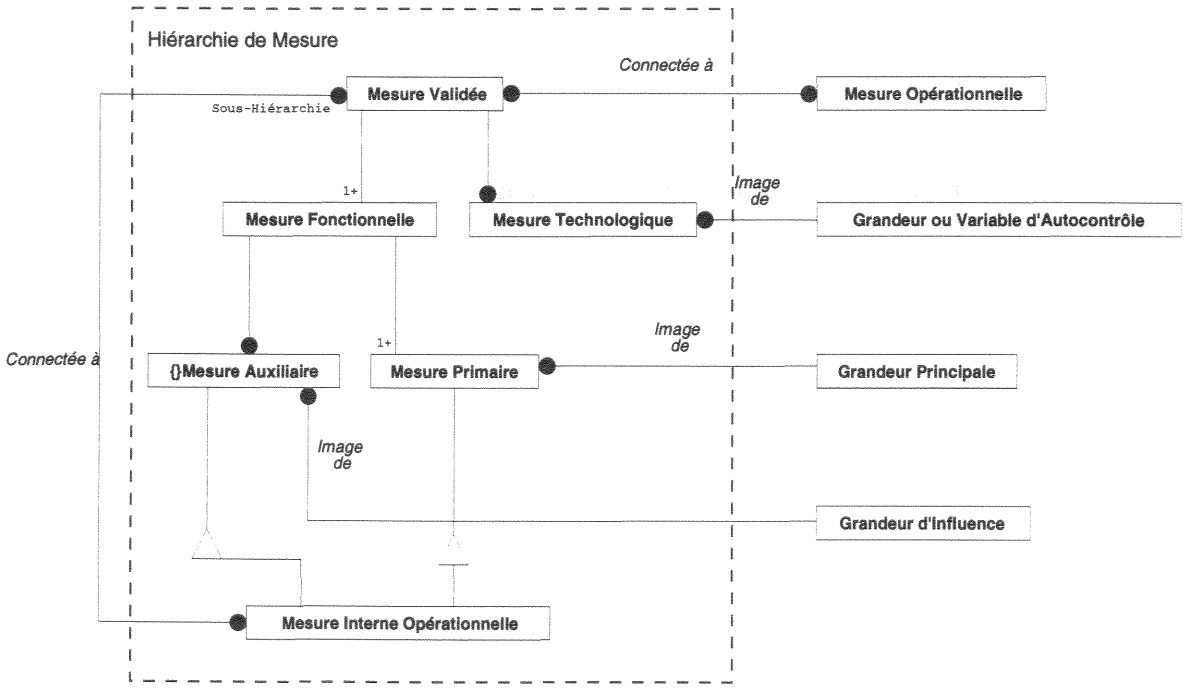


Figure V.4 : vue détaillée de la classe composite "Hiérarchie de mesure" montrant sa structure interne.

La vue détaillée permet de représenter la structure interne de la classe composite "Hiérarchie de Mesure", ainsi que les relations de ces sous-classes avec les autres classes l'utilisant (citons par exemple : la classe "Mesure Opérationnelle").

V.3.1.2 Le modèle dynamique

Des stratégies de réutilisation peuvent également être utilisées pour les diagrammes d'états des classes.

Trouver des sous-automates réutilisables dans un automate consiste, en fait, à identifier des sous-ensembles stables d'états. Un tel ensemble est constitué d'états et d'événements tels que, quel que soit l'événement de l'ensemble, son traitement fait rester dans un des états de l'ensemble. Afin que ce sous-automate soit utilisable, il faut le compléter par l'indication d'événements qui permettent d'en sortir.

Un composant automate est donc constitué d'un ensemble d'états dans lequel on reste après le traitement d'un certain nombre d'événements.

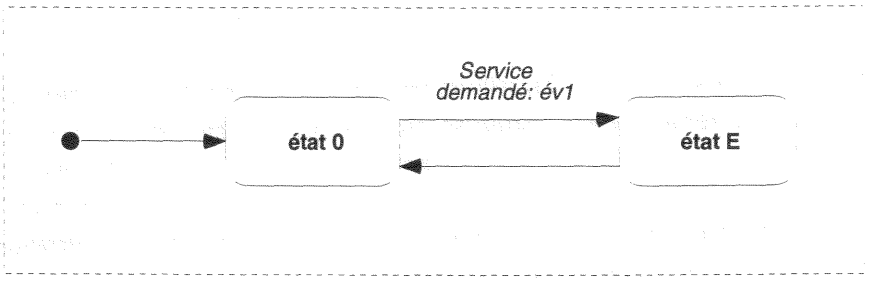


Figure V.5 : diagramme d'états montrant les relations du composant automate "état E" avec les autres états.

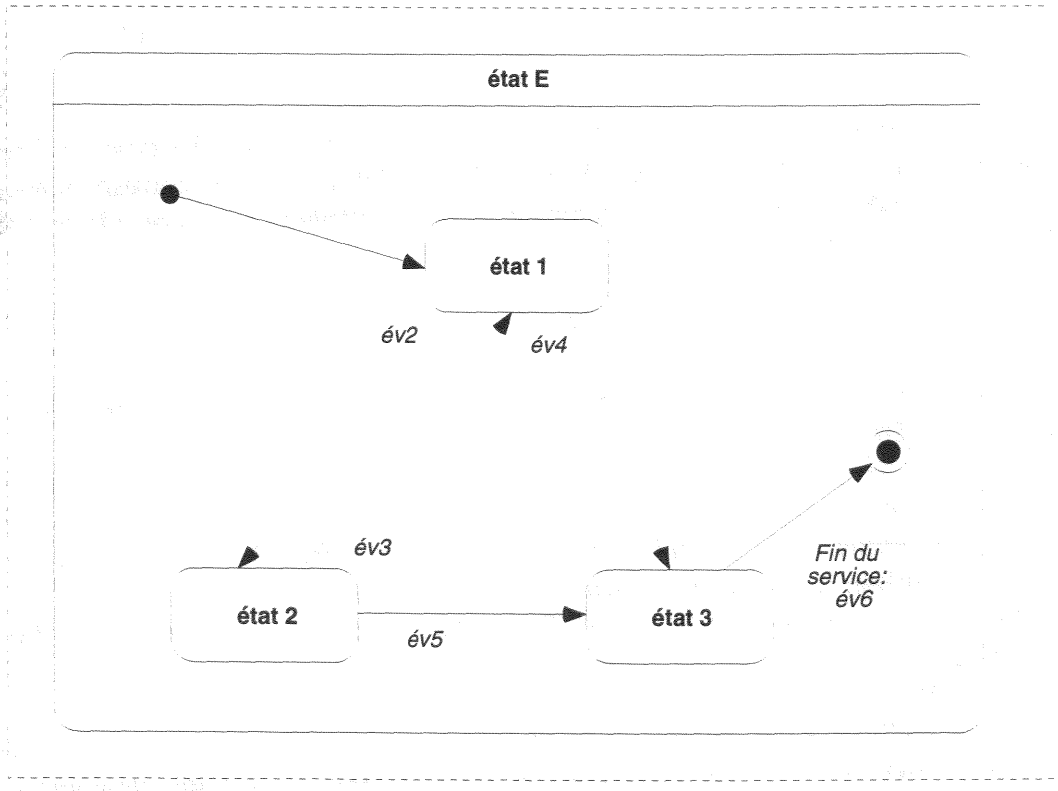


Figure V.6 : Vue détaillée du composant automate "état E"

Ces composants automates sont très utiles lors de la description des services dans les différents "contrôleurs utilisateur" puisque certains services peuvent être invoqués par plusieurs utilisateurs (cf. chapitre IV § IV.3.4).

V.4 Adapter les composants

Dans ce qui suit, nous allons détailler un ensemble de techniques qui permettent de réutiliser un composant qui ne correspond pas exactement à ce que l'on voulait. Deux cas peuvent se présenter [COUL-96] :

Cas où les attributs de la classe sont différents :

Les différences entre les attributs de l'objet composant que l'on propose et les attributs que l'on a identifiés peuvent être diverses :

- Si l'objet composant a plus d'attributs que ce que l'on souhaite, il faut regarder si ce n'est pas un oubli à réparer dans l'analyse que l'on est en train de faire ou si ce n'est pas un défaut dans le composant qui est trop spécialisé. Dans ce dernier cas, il est certainement préférable de réutiliser un ancêtre de l'objet proposé. Si celui-ci n'existe pas, on peut essayer de généraliser l'objet composant; sinon, si la possibilité d'attribuer une valeur par défaut existe, alors le composant est utilisable.
- Si l'objet composant a moins d'attributs que ce que l'on veut, on est dans un cas normal de spécialisation. On réutilise l'objet proposé en créant un héritier qui ajoute les attributs manquants.
- Si les attributs que l'on a identifiés sont répartis dans plusieurs objets composants, alors on utilise l'héritage multiple pour fabriquer l'objet que l'on souhaite à partir des composants trouvés.

De plus, en conception et codage, un ou plusieurs attributs peuvent avoir des types différents de ce que l'on voulait, il faut donc chercher le type le plus général. S'il n'y a pas de correspondance de type possible, il est préférable de nommer différemment le ou les attributs et de considérer que l'on est dans un cas de différence décrit ci-dessus.

Cas où les méthodes de la classe sont différentes :

Les différences entre les méthodes de l'objet composant que l'on propose et les méthodes que l'on a identifiées peuvent être diverses :

- Si l'objet composant a plus de méthodes que ce que l'on souhaite, ce peut être un oubli à réparer dans l'analyse que l'on est en train de faire ou le composant est trop spécialisé. Dans ce dernier cas, il est préférable de réutiliser un ancêtre de l'objet proposé, ou on peut aussi réutiliser le composant en ignorant la méthode en trop.
- Si l'objet composant a moins de méthodes que ce que l'on souhaite, on est dans un cas normal de spécialisation. On réutilise l'objet composant en créant un héritier qui ajoute les méthodes manquantes.
- Si une méthode a des paramètres ou un type en retour différents, alors il faut considérer que la méthode n'est pas celle que l'on cherche (sauf si certains paramètres peuvent prendre des valeurs par défaut, ou si le type en retour est un type ancêtre de type que l'on souhaitait).

De même que pour les attributs, si les méthodes que l'on a identifiées sont réparties dans plusieurs objets composants, alors on utilise l'héritage multiple pour fabriquer l'objet que l'on souhaite à partir des composants trouvés.

En ce qui concerne les relations entre les objets, dans le cas où il manque des relations par rapport à ce que l'on voulait, il est possible de les ajouter dans un héritier.

La réutilisation de composants est séduisante car elle améliore la productivité. Elle peut amener à fabriquer des exécutables trop gros si, chaque fois que l'on réutilise un composant, on récupère au passage un ensemble de traitements et données que l'on n'utilise pas.

Dans la mesure du possible, il faut essayer de toujours réutiliser le composant qui correspond le plus exactement au besoin. Un bon réflexe dans le concept objet consiste, si on souhaite utiliser une classe et que cette classe a des ancêtres, à vérifier si un des ancêtres ne convient pas.

L'héritage est une technique précieuse pour adapter les composants. Elle a des limites qu'il faut prendre en compte surtout si on réutilise en analyse ou en conception.

L'héritage simple pose peu de problèmes. Par contre l'héritage multiple pose de nombreux problèmes. Par exemple, on peut hériter de classes qui possèdent des méthodes de même nom. Il faudra pour distinguer les méthodes avoir la possibilité de les renommer (les langages comme Eiffel ou C++ propose des techniques de renommage).

Un autre problème majeur de l'héritage multiple est liés à la présence d'héritage répété. C'est à dire si une classe hérite plusieurs fois d'une même classe, il y a risque de conflit pour les attributs et les méthodes héritées.

Les techniques orientées objet utilisées avec un langage supportant l'héritage répété sont les meilleures pour adapter les composants dans un grand nombre de cas.

V.5 Maintenir et faire évoluer

Après avoir vu comment fabriquer et comment adapter les composants, il est nécessaire de s'intéresser à leur maintenance et leur évolution.

Les composants ne sont jamais figés. Au début de leur existence, ils évoluent beaucoup car il faut corriger leurs anomalies. Plus ils sont utilisés, plus il y a d'anomalies découvertes. Il faut aussi leur ajouter de nouvelles possibilités car celles que l'on a imaginées lors de leur création sont toujours insuffisantes. La phase suivante de leur existence est celle de l'évolution des technologies. Les composants ont été conçus à partir de certaines habitudes, d'un certain savoir-faire et les techniques ont évolué. La question se pose alors de leur évolution ou de leur remplacement par des composants plus au goût du jour. Toutes les modifications doivent être faites sans perturber les applications qui utilisent les composants.

Dans le cas de l'utilisation de techniques orientées objet certains utilisateurs corrigent l'anomalie en créant un héritier du composant qui a des problèmes et en redéfinissant ce qui ne fonctionne pas.

L'identification d'évolutions dans un composant peut également avoir plusieurs origines [COUL-96] :

- la veille technologique peut mettre en évidence l'existence de nouvelles technologies qui remettent en cause des composants existants,
- une analyse de l'existant peut permettre de découvrir qu'il existe de nombreuses variantes pour un composant donné. Cela signifie que le composant est mal conçu et que ceux qui l'utilisent sont obligés de le modifier. Il est sans doute nécessaire de le faire évoluer en prenant en compte le contenu de toutes ses variantes.

Quand on modifie un composant, il faut examiner les composants qui lui sont associés dans le cycle de vie. Si on change un composant d'analyse, il faut vérifier si le changement n'est pas à répercuter dans les composants de conceptions associés et si on change un composant de conception, il faut regarder si les composants de code associés ne sont pas également à modifier.

V.6 Bibliothèque de composants réutilisables pour la conception de l'entité "chaîne de mesure"

Ci dessous est présenté un modèle objet (figure V.7) permettant la réalisation d'une bibliothèque de composant pouvant être intégrée dans un outil d'aide à la conception de capteur intelligent et utilisée lors du développement de la chaîne de mesure.

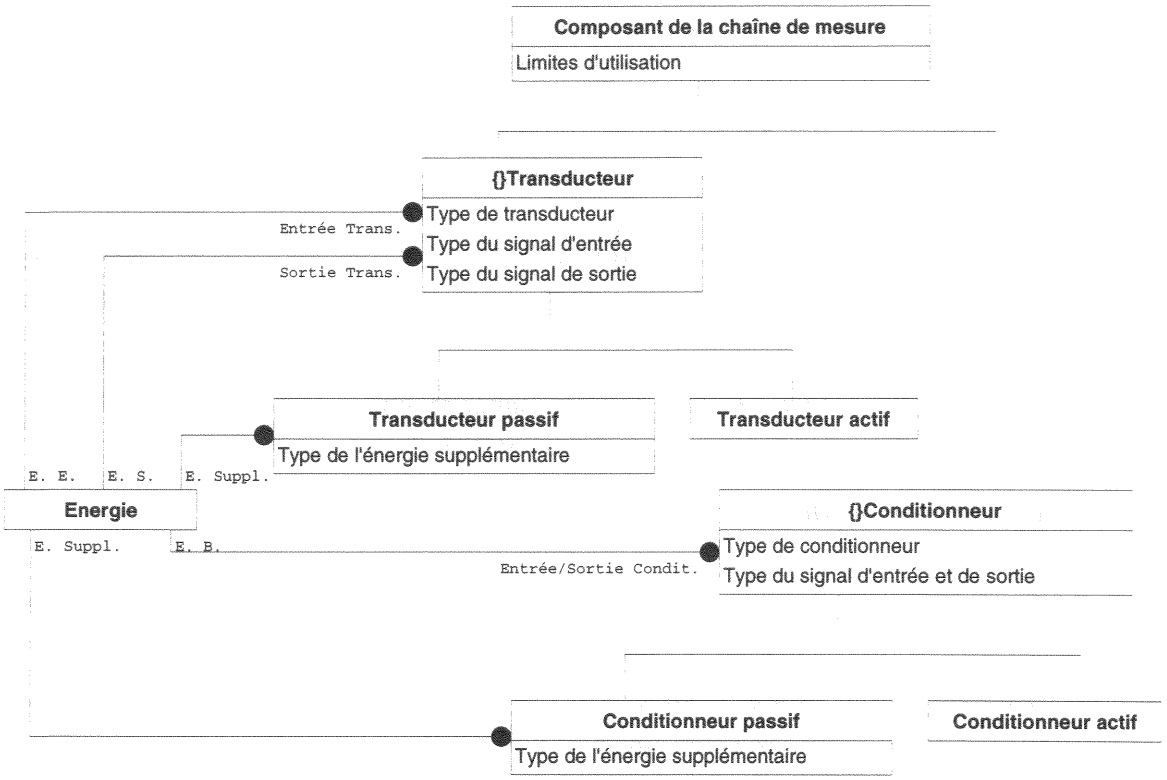


Figure V.7 : Modèle objet pour les composants de la chaîne de mesure en fonction des domaines d'énergie.

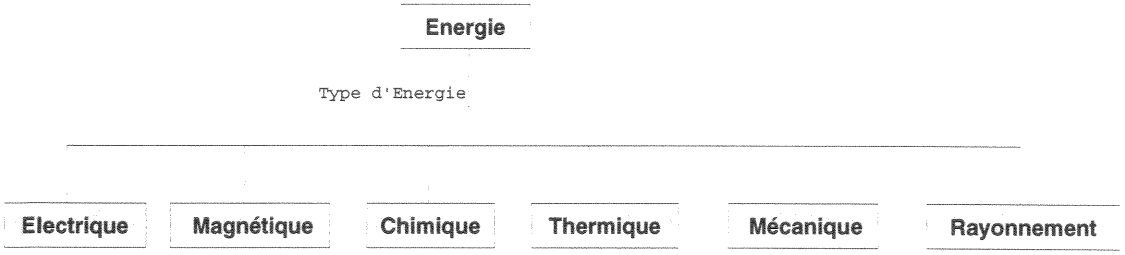


Figure V.8 : Les différent domaine d'énergie utilisable en instrumentation [LION-69].

La classe "Énergie" est aussi une classe concrète puisque tous les types d'énergie n'ont pas pu être précisés. Ceci indique que des sous-classes additionnelles existent, mais ont été omises, peut-être par manque d'espace, ou parce qu'elles n'étaient pas pertinentes dans le contexte présent.

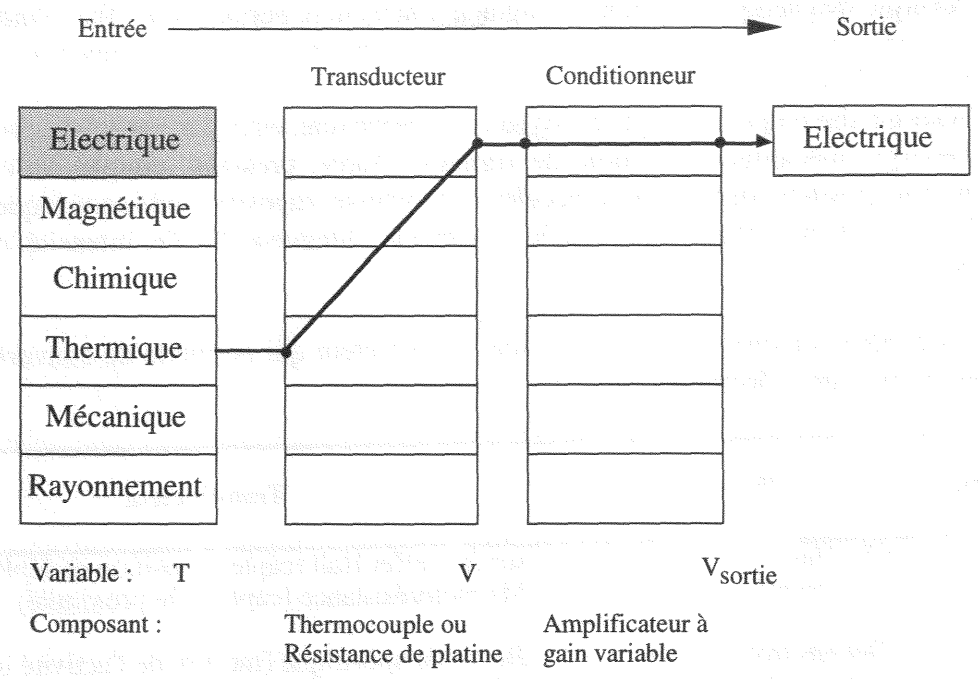


Figure V.9 : Exemple d'une chaîne de mesure d'un capteur de température utilisant un thermocouple ou une résistance de platine.

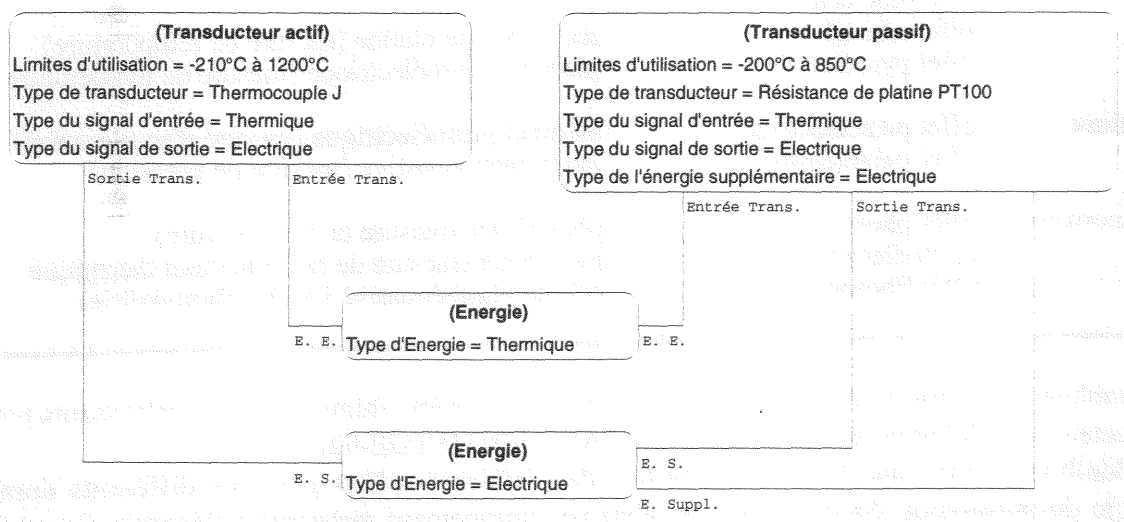


Figure V.10 : Diagramme d'instance d'un thermocouple et d'une résistance de platine

Comme le transport des informations relatives au mesurande est inconcevable sans le transport d'énergie, les transducteurs sont en fait des entités qui convertissent l'énergie d'une forme en une autre. En conséquence, pour comprendre quelles sont les classes de transducteur existantes, il est nécessaire de différencier les différentes formes d'énergie [SYDE-92] :

énergie de rayonnement, énergie gravitationnelle, énergie mécanique, énergie électrique, énergie thermique, énergie moléculaire, énergie de masse, énergie nucléaire, énergie atomique.

Cependant, pour l'instrumentation, il n'est pas nécessaire de considérer toutes ces formes d'énergie. Dans ce cas on utilise le groupement défini par Lion [LION-69] qui distingue six groupes d'énergie :

- l'énergie électrique (exemples de signaux : tension, courant, résistance, inductance, capacité, constante diélectrique, fréquence, polarisation électrique),

- l'énergie magnétique (exemple de signaux : induction, perméabilité, flux d'induction),
- l'énergie chimique (exemple de signaux : concentration, vitesse de réaction, toxicité, ph, tension d'oxydoréduction),
- l'énergie thermique (exemple de signaux : température, entropie, capacité calorifique),
- l'énergie mécanique (exemple de signaux : force, pression, couple, volume, masse, niveau, position, déplacement, accélération, vitesse, rugosité, onde acoustique),
- l'énergie de rayonnement (exemple de signaux : longueur d'onde, intensité, polarisation, phase).

Dans le cas général, on cherche a avoir un seul transducteur qui permette de convertir un signal non-électrique en un signal électrique.

Energie d'entrée/sortie (électrique)	Effet utilisé	Transducteur
Magnétique	effet Hall	sonde a effet Hall (capteur position et déplacement)
	effet magnétorésistif	Magnétorésistance (capteur de proximité)
	...	
Chimique	effet électrochimique	électrode spécifique (mesure de l'activité d'ion)
	conductivité électrolytique	cellule de conductimetrie (mesure des conductivités)
	effet résistif	Hygromètre résistif (mesure d'humidité)
	...	
Thermique	effet Seebeck (thermoélectrique)	thermocouple (mesure de température)
	effet résistif	résistance de platine (mesure de température)
	effet pyroélectrique	plaquette pyroélectrique (mesure de température
	...	
Mécanique	effet piézoélectrique	élément piezoélectrique (mesure d'accélération)
	effet piézorésistif	jauge piézorésistive (mesure de pression)
	...	
Rayonnement	effet photovoltaïque	photodiode (mesure de température)
	photoélectromagnétique	pyromètre (mesure de rayonnement thermique)
	effet Photoémissif	cellule photoémissive à vide (photométrie)
	...	

Le tableau ci-dessus montre quelques effets physiques et/ou chimiques sont intéressants pour les transducteurs qui délivrent un signal électrique [ASCH-91] [SYDE-92].

Middelhoek donne une liste très détaillée des différents effets pour les différents domaines d'énergie du mesurande. Sa classification n'est pas uniquement restreinte à l'énergie électrique en sortie mais est étendue à tous les autres domaines d'énergie cités précédemment.

Dans le cas d'un capteur intelligent, la sortie de la chaîne d'acquisition devra être un signal électrique. On utilisera alors que cinq des six domaines d'énergie pour le mesurande (énergie magnétique, chimique, thermique, mécanique, et rayonnement).

En considérant les différents transducteurs et leurs effets, il apparaît que deux classes de transducteurs existent :

Les transducteurs qui assurent la conversion en énergie électrique, l'énergie propre au mesurande sans source d'énergie supplémentaire, comme un thermocouple ; ces transducteur fonctionnant en générateur sont appelés transducteurs actifs (self-generating transducers).

Les autres transducteurs, tels que les résistances de platine, ne peuvent convertir l'une des cinq énergies d'entrée en énergie électrique qu'avec une source d'énergie supplémentaire; ce type de transducteur sont appelés transducteurs passifs (modulating transducers). Lorsque la source d'énergie

supplémentaire est électrique, on peut distinguer cinq types différents de transducteurs. Ces transducteurs sont basés sur les effets de photoconductance, piézorésistance, thermorésistance, magnétorésistance et conductance électrolytique.

Dans un conditionneur, le signal analogique peut être amplifié, filtré, retardé, modulé, comparé avec un signal connu.

Le choix d'un conditionneur est une étape importante dans la réalisation d'un ensemble de mesure. C'est en effet l'association transducteur-conditionneur qui détermine le signal électrique utile; de la constitution du conditionneur dépend un certain nombre de performances de l'ensemble de mesure : sensibilité, linéarité, insensibilité à certaines grandeurs d'influence [ASCH-91].

De même que pour les transducteurs , les conditionneurs peuvent être actifs ou passifs.

Pour des raisons de coût ou de facilité d'exploitation on peut être amené à utiliser un transducteur non pas sensible au mesurande mais à l'un de ces effets. Le corps d'épreuve est le dispositif qui, soumis au mesurande en assure une première traduction en une autre grandeur physique non-électrique, le mesurande secondaire, qu'un transducteur adéquat traduit alors en grandeur électrique.

Notons qu'il existe des conditionneurs pour lesquels le domaine d'énergie du signal n'est pas électrique, par exemple, dans le domaine de l'énergie mécanique, une cellule de charge convertit un déplacement en une contrainte. D'autres exemples sont donnés par Middelhoek [MIDD-82a] [MIDD-82b] dans le domaine de l'énergie de rayonnement, de l'énergie thermique, de l'énergie magnétique.

On peut donc distinguer deux classes de composants pour la chaîne de mesure :

- les transducteurs,
- et les conditionneurs.

Ces composants peuvent être représentés de la même façon, à savoir modélisés suivant le domaine d'énergie des grandeurs d'entrée et de sortie ainsi que de la source d'énergie supplémentaire si elle existe.

On appellera transducteurs les composants qui modifient le domaine d'énergie de la grandeur d'entrée en un autre domaine d'énergie pour la grandeur de sortie (exemple : thermique en électrique).

On appellera conditionneurs les composants qui ne modifient pas le domaine d'énergie (exemple : électrique en électrique).

De plus on distinguera deux classes de composants :

- les composants actifs qui n'ont pas besoin de source d'énergie supplémentaire,
- les composants passifs qui ont besoin d'une source d'énergie supplémentaire.

A ces composants, des modèles mathématiques pourront être associés permettant ainsi la simulation de la chaîne de mesure [ABDU-85]. Dans la plupart des cas, on peut représenter sous forme d'équations mathématiques le comportement de ces transducteurs et de ces conditionneurs. Mais dans certains cas, par exemple en cas de fortes non-linéarités, on aura recours à des modèles de type boîte noire pour représenter ces composants (par exemple : les modèles neuronaux).

Cette représentation suivant les domaines d'énergie doit permettre au concepteur de la chaîne de mesure, de réaliser et de simuler celle-ci en procédant en un assemblage de transducteurs et de conditionneurs en respectant les domaines d'énergie ainsi que les grandeurs d'entrée et de sortie de chacun de ces composants [ABDU-93]. D'autres critères de sélection pourront également entrer en jeu pour la réalisation de la chaîne de mesure à savoir les limites d'utilisations, la précision, le temps de réponse, les grandeurs d'influence des composants. Une base de données réalisée à partir du modèle objet pourra servir de base pour la réalisation d'un outil d'aide à la conception de chaîne de mesure.

V.7 Conclusion

En ne réutilisant pas, on se condamne à assurer un effort de maintenance de plus en plus important. A terme, on peut être réduit à ne plus faire que de la maintenance et à ne plus progresser. En réutilisant, la maintenance est partagée et donc prend moins de temps.

Réutiliser, c'est une nouvelle façon d'aborder les problèmes, c'est automatiser une partie des tâches que l'on faisait manuellement avant. Il est difficilement concevable de mettre en oeuvre de la réutilisation sans disposer d'outils (outils pour l'archivage et la recherche de composants dans des librairies, outils de documentation et de gestion de configurations permettant de gérer les évolutions des composants).

La réutilisation, comme on l'a vu apporte beaucoup d'avantages [COUL-96] :

- meilleure productivité,
- meilleure maintenabilité,
- meilleure évolutivité.

Il n'y a pas de réutilisation sans efforts : on ne se met pas à réutiliser en passant à l'utilisation de l'approche orientée objet ; il faut apprendre à utiliser certaines techniques et mettre en place une organisation pour favoriser la réutilisation.

L'orienté objet est un bon choix pour la réutilisation à condition d'en respecter les règles (encapsulation en particulier).

V.8 Références

[ABDU-85] Abdullah F., "Mathematical modelling aids transducer design", C&I, October 1985, pages 75-76.

[ABDU-93] Abdullah F., Finkelstein L., Khan S.H., Hill W.J., "Modelling in Measurement and Instrumentation", State and Advances of Measurement and Instrumentation science, IMEKO TC1 and TC7 Colloquium, september 1993, pages 230-246.

[ASCH-91] Asch G., "Les capteurs en instrumentation industrielle", Editions DUNOD, 4ème édition, 1991.

[BOOC-86] Booch G., "Object-Oriented development", IEEE Trans. on software engineering, Vol. SE-12, N°2, pages 28-38, 1986.

[BOOC-93] Booch G., "Développement logiciel orienté objet : la méthode Booch processus et pragmatique", Génie logiciel & systèmes experts, N°30, Mars 1993, pages 4-13.

[CHAU-95] Chauvet J.M., "Une analyse économique de la réutilisabilité : l'exemple de macDonnell Douglas", AFCET/INTERFACES, N°103/104, Mai/Juin 1991, pages 26-33.

[COAD-91] Coad P. and Yourdon E., "Object-Oriented Design", Prentice-Hall, Englewood Cliffs, NJ, 1991, 200 pages.

[COUL-96] Bernard Coulange, "Réutilisation du logiciel", Édition Masson, 1996, ISBN 2-225-85089-5, 324 pages.

[HEND-93] Henderson I.A., McGhee J., "Classical taxonomy: An holistic perspective of temperature measuring systems and instruments", IEEE Proceedings-A, Vol.140, N°4, July 1993, pages 263-268.

[LION-69] Lion K.S., "Transducers : Problems and Prospects", IEEE Trans. Indust. Electron. Contr. Instrum., IECI-16, 1969, pages 2-5.

[MIDD-82a] Middelhoek S., Noorlag D.J.W., "Three-dimensional representation of input and output transducers", Sensors and actuators, Vol.2, 1982, pages 29-41.

[MIDD-82b] Middelhoek S., Noorlag D.J.W., "Signal conversion in solid-state transducers", Sensors and actuators, Vol.2, 1982, pages 211-228.

[ROSE-91] Rosen J.P., "Introduction au monde des objets", AFCET INTERFACES, N°103-104, Mai-Juin 1991, pages 5-95.

[RUMB-94] Rumbaugh J., "Building boxes: Composite objects", Journal of Object-Oriented Programming, Vol.7, N°7, November-December 1994.

[RUMB-95] James Rumbaugh et al., "OMT : Modélisation et conception orientées objet", Edition française revue et augmentée, Édition Masson et Prentice-Hall, 1995, 520 pages.

[SYDE-92] Sydenham P.H., Hancock N.H., Thorn R., "Introduction to Measurement Science and Engineering", Édition Wiley, 1992, 327 pages, ISBN 0471935719.

Chapitre VI

Application à un capteur intelligent de température

VI.1 Introduction

La mesure des températures est, avec celle des pressions, la plus répandue dans la plupart des processus industriels. Le choix du capteur intelligent de température, comme support de démonstration des concepts de modélisation exposés dans les chapitres précédents, s'est fait dans le cadre d'un transfert technologique pour la modélisation d'un transmetteur intelligent de température [KHOU-95] [GEOF-94].

Le modèle de référence a été spécialisé (grâce notamment au mécanisme d'héritage) puis instancié au cas d'un capteur intelligent de température avec contact.

Le but essentiel de cette instanciation est la vérification de la faisabilité de réalisation d'un modèle de référence de capteurs intelligents à partir de la méthodologie et des outils décrits dans le chapitre IV, ainsi que des techniques permettant une réutilisation du modèle de référence décrites dans le chapitre précédent.

Les parties du modèle de référence concernées par la spécialisation sont très restreintes. En effet, seules les classes "capteur composite", "hiérarchie de mesure" et "grandeur physique" présentées dans le chapitre IV doivent être spécialisées. Une fois ces classes redéfinies pour un cas particulier de capteur intelligent de température avec contact, le modèle de référence du capteur intelligent pourra alors être instancié.

Nous allons dans ce chapitre développer les parties du modèle de référence de capteur intelligent qui font l'objet d'une spécialisation.

Afin de réaliser une simulation la plus pertinente possible il est nécessaire à ce stade de choisir des solutions technologiques concernant par exemple la structure de la chaîne d'acquisition. Ces choix technologiques influeront sans aucun doute sur les classes permettant de réaliser une validation technologique de la mesure.

VI.2 Spécialisation de la chaîne d'acquisition

Le diagramme d'objets (figure VI.1) montre la structure de la chaîne de mesure que nous avons adoptée pour la simulation. Cette structure a été conçue à partir des remarques de Khoury [KOUR-95] et Rivière [RIVI-95] concernant les améliorations à apporter à la chaîne de mesure pour améliorer sa fiabilité ainsi qu'à l'amélioration de la crédibilité de la mesure. On peut citer, l'utilisation d'une redondance de la partie matérielle qui réalise la conversion des signaux analogiques en numériques (plusieurs convertisseurs analogique/numérique en parallèle), ou encore l'adoption d'un amplificateur à gain programmable et d'un bloqueur pour chaque entrée tension du multiplexeur analogique permettant ainsi de choisir un gain différent pour chaque voie dans un souci de conserver la plus grande exactitude possible lors de l'acquisition (pour une utilisation pleine échelle du convertisseur analogique/numérique), ou encore l'utilisation de plusieurs chaînes d'acquisition en parallèles permettant de réaliser la simultanéité de l'acquisition et donc de dater au même instant les différentes mesures. Le choix des transducteurs contribue directement à l'amélioration de la crédibilité des mesures, en tenant compte par exemples des limites de température, de l'environnement de fonctionnement, des nouvelles technologies et techniques de fabrication, Rivière [RIVI-95] expose quelques aspects de réalisations contribuant à l'amélioration technologique des thermocouples et des résistances de platine.

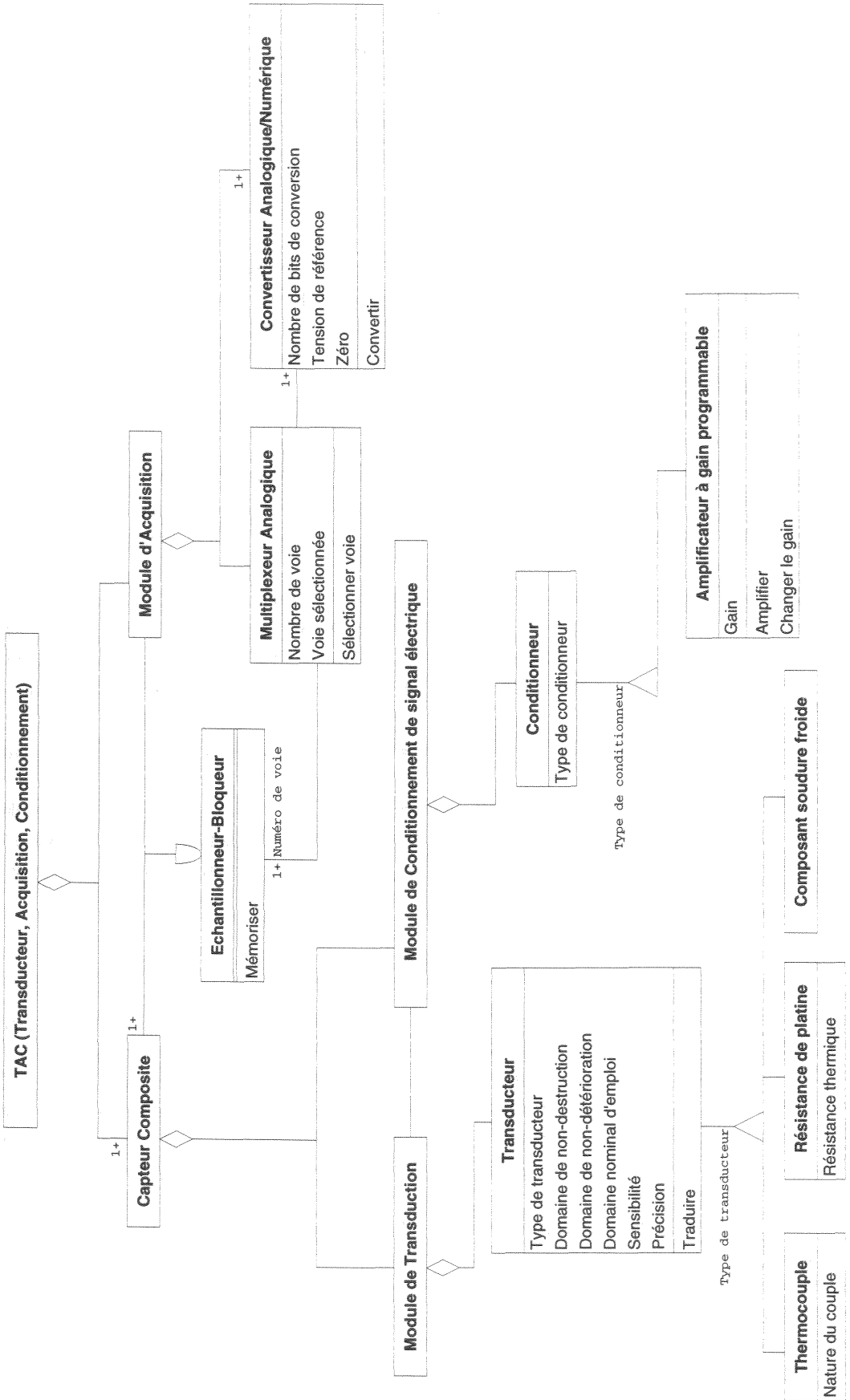


Figure VI.1 : Agrégation à niveaux multiples montrant la décomposition matérielle de la chaîne de mesure appliquée à un capteur intelligent de température avec contact.

La partie "Capteur Composite" a pour rôle de traduire l'énergie propre à une ou plusieurs grandeurs physiques en une énergie électrique facilement exploitable.

Les transducteurs, permettant de traduire la grandeur principale (la température), utilisés pour la simulation sont de deux types: les thermocouples et les résistances de platine dont le fonctionnement nécessite un apport d'énergie pour la traduction des grandeurs physiques. Le fonctionnement de ces deux types de transducteurs est développé par la suite.

La chaîne de mesure (classe composite "TAC (Transducteur, Acquisition, Conditionnement)") est donc spécialisée et instanciée afin d'obtenir pour les thermocouples une f.é.m. en fonction d'une température (température jonction chaude) par rapport à une température de référence (température jonction froide) et pour les résistances de platine une tension en fonction de la température et d'un courant les traversant.

On a également utilisé deux types supplémentaires de transducteurs qui permettent la traduction des grandeurs d'influence, à savoir :

- pour la mesure de la température avec un thermocouple, un composant électronique (classe "Composant soudure froide") délivrant une tension en fonction de la température ambiante dans laquelle est placé le capteur intelligent (correspondant à la température de jonction froide du thermocouple),
- et pour la mesure de la température avec une résistance de platine, une résistance calibrée et de précision permettant de mesurer le courant traversant la résistance de platine.

Pour compenser et corriger les dérives des convertisseurs analogique/numérique nous effectuons la mesure du zéro permettant de corriger l'erreur d'offset et la mesure de la tension de référence permettant de compenser le gain dû aux dérives de température [YICK-94][BRIG-94].

VI.2.1 Les différents types de transducteurs utilisés

Nous avons choisi deux types différents de transducteurs, l'un actif comme les thermocouples et l'autre passif comme les résistances de platine afin de mettre en évidence les différents types de grandeurs d'influence pouvant intervenir lors de la mesure, notamment l'influence d'une énergie supplémentaire pour les transducteurs passifs (cf. chapitre IV).

Ci dessous nous présentons très brièvement la théorie de la mesure de température avec des thermocouples et des résistances de platine [ASCH-91] [MILL-84] afin de montrer comment seront spécialisées les classes de la hiérarchie de mesure.

VI.2.1.1 Thermométrie par thermocouples

Le fonctionnement des thermocouples, ou couples thermoélectriques repose sur l'effet thermoélectrique : lorsque deux fils en métaux différents sont connectés à leurs extrémités, un courant continu circule dans la boucle ainsi formée, s'il y a une différence de température entre les deux jonctions.

On distingue donc la jonction chaude à la température $\{T_c\}$ et la jonction froide à la température $\{T_{ref}\}$.

La f.é.m. générée par le thermocouple dépend de la différence de température entre les deux jonctions. Le coefficient de Seebeck définit la sensibilité thermique d'un couple ou pouvoir thermoélectrique. Il dépend de la nature des deux métaux constituant le thermocouple et de la température.

$$E_{A/B}^{T_c 0°C} = E_{A/B}^{T_c T_{ref}} + E_{A/B}^{T_{ref} 0°C} \qquad (E.1)$$

La connaissance de la température de la jonction froide (appelé jonction de référence ou soudure froide) $\{T_{ref}\}$ permet de calculer $\left\{E_{A/B}^{T_{ref}\,0°C}\right\}$; la mesure de la f.é.m du thermocouple fournit une valeur correspondant à : $\left\{E_{A/B}^{T_c\,T_{ref}}\right\}$; on en déduit la f.é.m dont le thermocouple serait le siège si la température de référence était 0°C d'après l'équation (E.1); On peut dès lors déterminer la température de jonction chaude (appelée point chaud thermocouple) $\{T_c\}$.

Les modèles $\left\{E_{A/B}^{T_c\,0°C}\right\}$ sont fournis par la norme NF C 42-321 sous la forme de table ou d'expressions polynomiales.

En général on utilise plutôt une expression mathématique (du type polynomiale, ...) qui traduit algébriquement la relation entre la f.é.m. et la température.

VI.2.1.2 Les sondes à thermorésistances

Le principe de fonctionnement est basé sur la variation de la résistance des métaux en fonction de la température.

La variation de température peut avoir deux origines : l'élévation de la température ambiante et l'élévation de la température produite par le courant qui traverse la sonde.

La recherche d'une bonne sensibilité de mesure conduit à faire traverser la résistance de platine par un courant relativement important. Cependant, celui-ci risque alors de provoquer par effet Joule un échauffement du capteur qui peut cesser d'être négligeable.

L'échauffement du capteur par le courant i de mesure est défini par :

$$\Delta T_c = T_{ci} - T_c \qquad \text{(E.2)}$$

où $\{T_{ci}\}$ est la température du capteur parcouru par le courant $\{i\}$, et $\{T_c\}$ la température inconnue qu'aurait le capteur en l'absence de courant.

La tension mesurée dépend de la résistance de la sonde. On peut, dés lors, à l'aide d'une table ou d'une expression polynomiale, déterminer la température $\{T_{ci}\}$. L'une des normes les plus couramment suivies est la norme DIN 43760 qui donne une résistance en fonction de la température pour une résistance de platine.

L'échauffement dû au courant de mesure peut être calculé par la relation suivante :

$$\Delta T_c = R_{\vartheta xc} R_{ci} i^2 \qquad \text{(E.3)}$$

où $\{R_{\vartheta xc}\}$ est la résistance thermique et $\{R_{ci}\}$ est la résistance pour une valeur déterminée de $\{i\}$.

La connaissance de la température $\{T_{ci}\}$ et de l'échauffement, nous permet alors de connaître la température $\{T_c\}$ correspondant à la température mesurée (E.2).

En pratique la nécessité d'une correction de l'autoéchauffement de la sonde dépend de la précision de mesure recherchée. Lorsque l'on connaît une valeur maximale de la résistance thermique dans les conditions de mesure, on peut calculer le courant maximum en dessous duquel la correction devient nettement inférieure à l'erreur de la précision admise.

Les échanges thermiques entre la sonde et le fluide sont très dépendants des caractéristiques du fluide : viscosité, conduction thermique et vitesse. Il en est donc de même du temps de réponse de la sonde, de l'écart à l'équilibre entre sa température et celle du fluide, et en particulier dans le cas des sondes résistives de leur autoéchauffement par le courant de mesure.

Le tableau ci-dessous indique, à titre d'illustration, pour quelques sondes à résistance de platine (fabricant : Rosemount) placées dans divers fluides à différentes vitesses, les résistances thermiques correspondantes.

Influence de la nature du fluide et de sa vitesse sur la résistance thermique.

Modèles		134 MA	150 MA	152 T	177 MA
Fluide	Vitesse m/s	Résistance thermique °C/W			
Eau	1	1	2.5		13.3
Huile	1	3.3	5	1.7	15.3
Oxygène Liquide	1.5 7.5 15	1.9 1.43 1.35	4.5 3.3 3.2	0.67 0.34 0.24	185 186 186
Oxygène	1.5	83	167	31	500
Gazeux	7.5 15	34 22	71 48	19 14	333 250

VI.3 Les grandeurs physiques

Les deux diagrammes objet ci-dessous montrent quelles sont les grandeurs physiques mises en jeu lors de la mesure de température par un thermocouple ou par une résistance de platine.

Comme nous l'avons vu précédemment, d'une part pour un thermocouple il est nécessaire d'avoir la température de jonction froide pour élaborer la vraie température de jonction chaude dans le cas où l'on fait une compensation par logiciel. Cette température de jonction froide sera donc une grandeur d'influence ainsi qu'une grandeur d'autocontrôle pour vérifier que la température de fonctionnement du capteur intelligent est respectée.

D'autre part, pour une résistance de platine il est nécessaire de connaître le courant d'excitation lorsque l'autoéchauffement résultant est non négligeable. Le courant d'excitation sera donc une grandeur d'influence ainsi qu'une grandeur d'autocontrôle permettant la surveillance des dérives éventuelles de la source d'énergie supplémentaire (générateur de courant). Pour des raisons pratiques, on assimilera la température de jonction froide du thermocouple à la température de fonctionnement permettant ainsi de ne pas rajouter un composant spécifique pour la mesure de la température de fonctionnement dans le cas d'une résistance de platine. La température de jonction froide sera alors une grandeur d'autocontrôle dans le cas d'une résistance de platine.

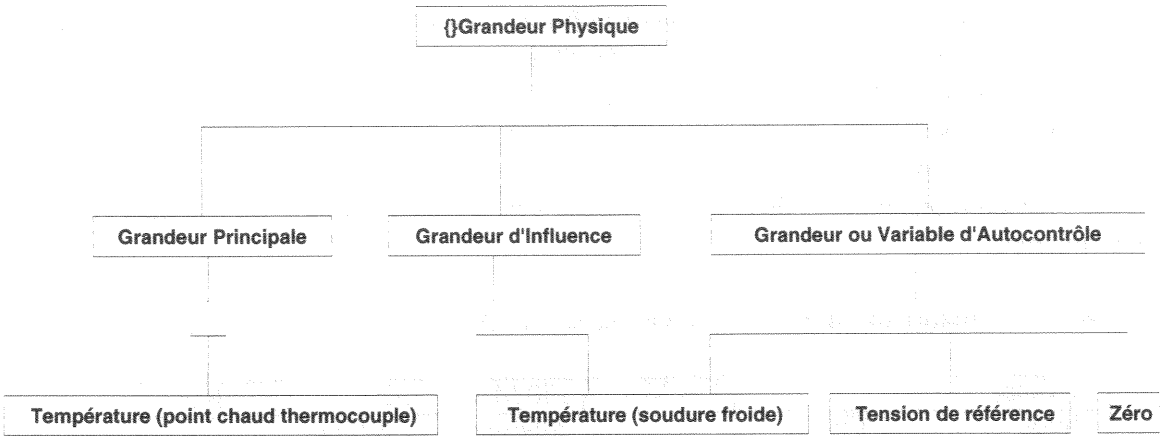


Figure VI.2 : Diagramme d'objets comportant un héritage multiple montrant les différentes grandeurs physiques intervenant pour la mesure de température avec un thermocouple.

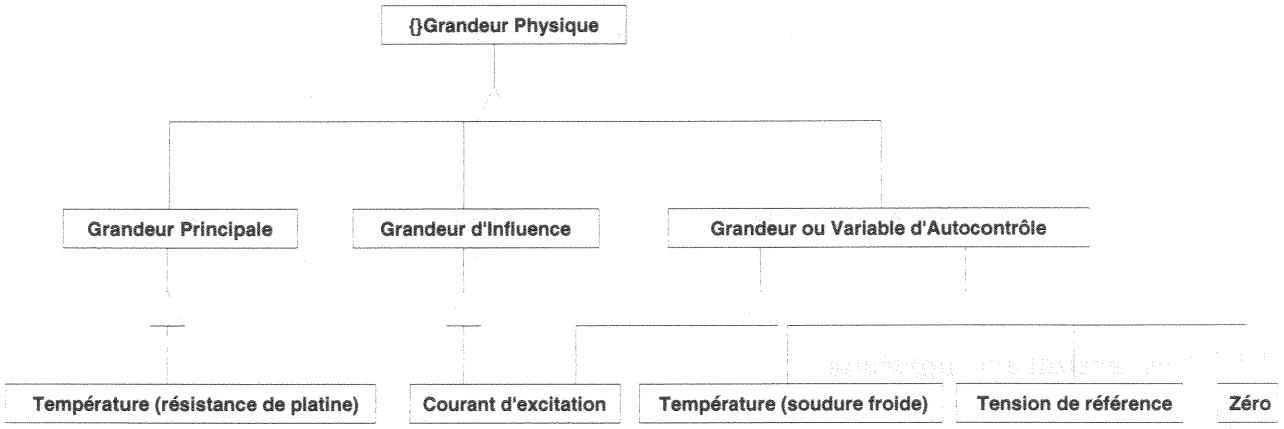


Figure VI.3 : Diagramme d'objets comportant un héritage multiple montrant les différentes grandeurs physiques intervenant pour la mesure de température avec une résistance de platine.

On remarque que pour les thermocouples et les résistances de platine la tension de référence et celle du zéro sont considérées comme des grandeurs d'autocontrôles. Ces deux grandeurs nous permettent de faire un suivi des convertisseurs analogique/numérique.

VI.4 Modèle d'obtention de la mesure opérationnelle

Les modèles d'obtention de la mesure opérationnelle sont réalisés en s'appuyant sur les travaux du CIAME, concernant la modélisation informelle du concept de capteur intelligent.

Les hiérarchies d'obtention des différentes grandeurs intermédiaire sont représentées suivant l'axe vertical. De plus, ce même axe vertical représente le temps sous certaines hypothèses. Ci-dessous sont représentées les deux hiérarchies d'obtention d'une mesure opérationnelle respectivement pour un thermocouple et une résistance de platine.

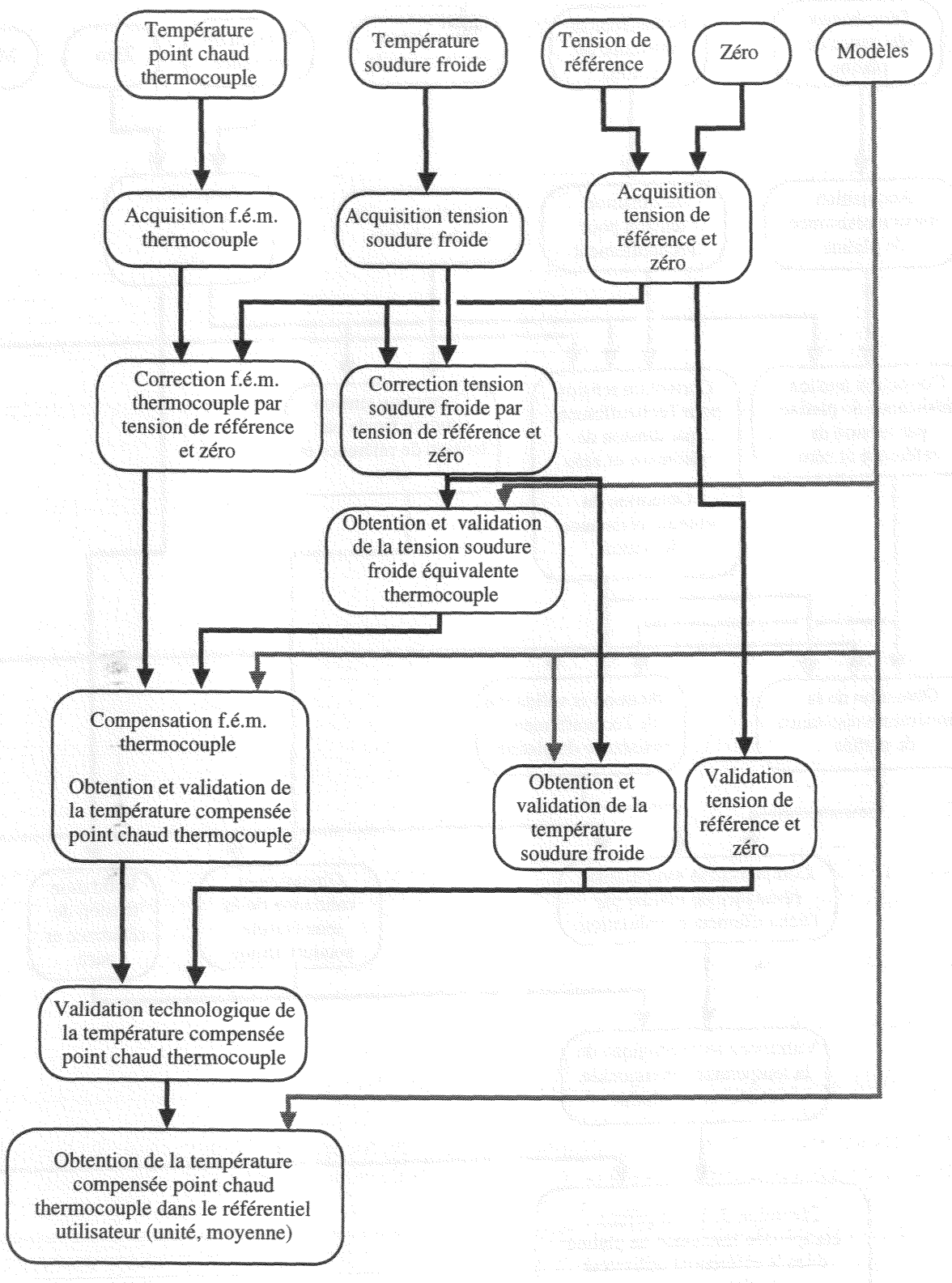


Figure VI.4 : Hiérarchie d'obtention d'un mesure opérationnelle pour un thermocouple.

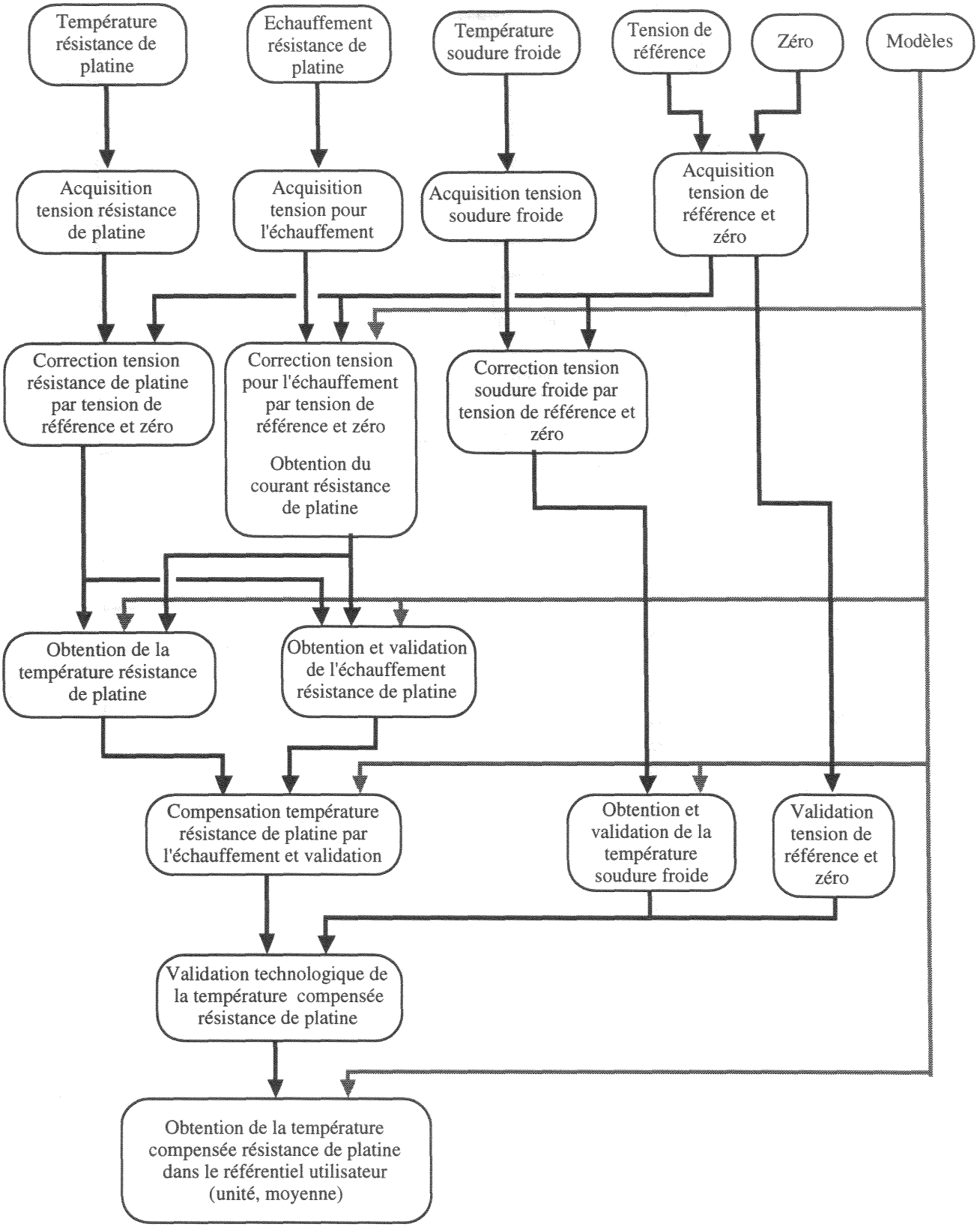


Figure VI.5 : Hiérarchie d'obtention d'un mesure opérationnelle pour une résistance de platine.

Il est possible, lors de la simulation, de simuler des défauts, qui sont susceptibles d'intervenir dans la chaîne de mesure lors de son fonctionnement, dont les effets doivent être détectés et/ou signalés et corrigés si possible par le capteur intelligent. La liste de ces défauts n'est pas exhaustive, mais l'on

peut causer par exemple : une dérive de la tension de référence, un décalage d'offset (zéro), un défaut d'une ou plusieurs voies d'acquisition (causé par exemple par la défaillance d'un ou plusieurs bloqueurs ou par le multiplexeur), ...

A partir de ces deux hiérarchies d'obtention d'une mesure opérationnelle, nous avons spécialisé au moyen du mécanisme d'héritage, la classe "hiérarchie de mesure" pour des transducteurs du type thermocouple et résistance de platine.

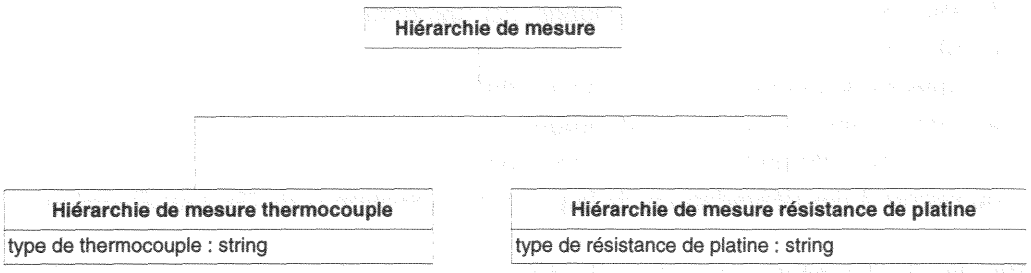


Figure VI.6 : Héritage montrant deux types de hiérarchie de mesure utilisées pour la mesure de température.

Comme nous l'avons montré dans le chapitre précédent, ces classes sont des classes composites, c'est à dire, composées d'autres classes par une relation d'agrégation.

Le diagramme d'instance ci-dessous montre deux hiérarchies de mesure correspondant à deux thermocouples différents, l'un de type J et l'autre de type R, mesurant la même grandeur principale (température jonction chaude). Chaque mesure opérationnelle se connecte de façon dynamique aux différentes hiérarchies de mesure, c'est à dire que la mesure opérationnelle peut se connecter soit à l'une des deux hiérarchies de mesure soit aux deux pour demander l'élaboration d'une mesure validée. Le choix de la ou des hiérarchies de mesure connectées évolue dans le temps suivant l'évolution de la température mesurée.

L'un des intérêts majeurs de l'utilisation de hiérarchies multiples pour l'élaboration d'une mesure opérationnelle est lié au fait que le capteur intelligent sélectionne lui même le transducteur approprié en fonction de la valeur du mesurande à mesurer. Il se basera par exemple dans le cas de la mesure de température, sur les caractéristiques (plage de mesure, ...) de chaque thermocouple associé aux hiérarchies de mesure.

En effet, la plage de température à l'intérieur de laquelle un thermocouple est utilisable est limité :

- aux basses températures, par la décroissance de son pouvoir thermoélectrique;
- aux températures élevées, par les risques de contamination par l'atmosphère ambiante, par l'évaporation de l'un des constituants d'un alliage du couple, ou enfin par la fusion de l'un des conducteurs.

Le diagramme d'instances indique également pour les deux types de couples les températures limites d'utilisation normale, ainsi que la précision.

Par exemple, on pourra utiliser le thermocouple de type J pour les basses température puisque son pouvoir thermoélectrique élevé lui permet d'avoir une grande sensibilité, ce qui n'est pas le cas du thermocouple de type R qui ne peut mesurer des températures inférieur à zéro degré celsius. Par contre on utilisera le thermocouple de type R pour les hautes températures puisque celui-ci à une meilleure précision. Toutefois, il faut noter que le domaine de non-destruction du capteur lorsque l'on utilise plusieurs thermocouples en redondance pour mesurer la même grandeur physique doit être l'intersection de leurs domaines de non-destruction. Par exemple, dans le cas étudié, la borne supérieure du domaine de non-destruction du capteur sera celle du thermocouple de type J.

Un autre intérêt de cette approche réside dans le fait que la redondance de ces deux hiérarchies de mesure correspondant à deux transducteurs différents, permet lorsqu'il y a détection d'une dégradation pour l'un des deux transducteurs de changer de hiérarchie de mesure pour effectuer la mesure.

Pour les thermocouples les dégradations les plus souvent rencontrées sont :

- la détérioration de l'isolant entre la gaine de protection et les conducteurs,
- l'inhomogénéité de section des conducteurs, due aux attaques chimiques de la gaine de protection et de l'isolant atteignant les conducteurs,
- la détérioration de la jonction chaude, par une rupture totale ou partielle de la soudure des conducteurs,

pour les résistances de platine des dégradations dues :

- aux vibrations et aux chocs mécaniques,
- aux attaques chimiques de l'environnement,
- ou encore à des dégradations de la gaine de protection et de l'isolant électrique.

Des méthodes de diagnostic permettant la détection de ces dégradations pourront être implantés dans les classes "hiérarchie de mesure". On peut citer par exemple, la mesure de la résistance de l'isolant ou encore une méthode mettant à profit l'effet joule [RIVI-95].

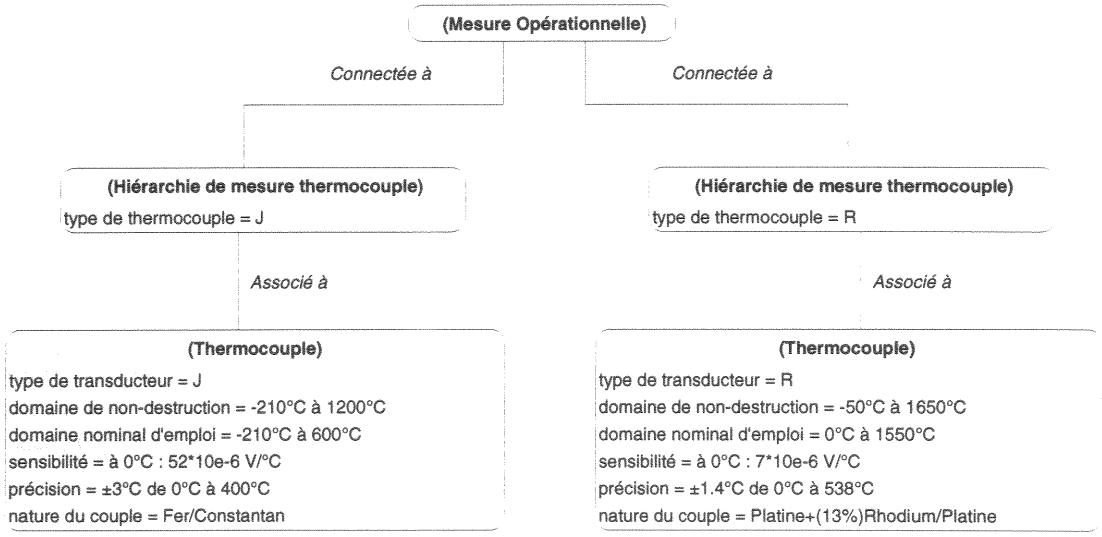


Figure VI.7 : Diagramme d'instances partiel pour une mesure opérationnelle.

La partie du modèle objet réalisant l'acquisition à été détaillée dans le chapitre IV. Nous avons redéfini certaines méthodes de la classe "contrôleur d'acquisition" qui a pour mission de coordonner l'acquisition et le conditionnement, afin de prendre en compte les choix technologiques qui ont été faits pour la chaîne de mesure (celle-ci est représentée par la classe "TAC (Transduction, Acquisition, Conditionnement) "; elle réalise la conversion des grandeurs physiques à mesurer en grandeurs électriques qui seront ensuite numérisées).

Toutes les valeurs acquises sont corrigées par la tension de référence et le zéro, ainsi que par les divers modèles des différents composants de la chaîne de mesure, si ils existent. On peut citer pour exemples, les modèles permettant de compenser les non-linéarités des amplificateurs à gain programmable.

Les mesures issues de l'acquisition sont ainsi corrigées et compensées en fonction des divers composants de la chaîne de mesure qui ont servi à leur élaboration. Ceci permet d'avoir une

indépendance entre les hiérarchies de mesure et les choix technologiques qui ont été faits pour l'acquisition et le conditionnement. Une première validation portant sur la mesure de la tension de référence et le zéro de la chaîne d'acquisition permet de déterminer l'intégrité de cette dernière.

On peut à ce niveau, lors de la simulation, provoquer un défaut de ces mesures corrigées et compensées avant leurs traitements par les hiérarchies de mesure afin de mettre en évidence la validation des mesures qui est faite à tous les niveaux de la hiérarchie.

Les figures VI.8,9,10,11,12 illustrent la vue détaillée de l'objet composite "Hiérarchie de Mesure" spécialisé au cas d'un capteur intelligent de température avec contact utilisant comme transducteur principal un thermocouple ou une résistance de platine.

La classe "Hiérarchie de mesure" regroupe l'ensemble des classes ("mesure primaire", "mesure auxiliaire", "mesure fonctionnelle", "mesure technologique", "mesure validée") qui permettent à partir des mesures corrigées numérisées d'aboutir à une mesure validée. Cette mesure validée sera ensuite transcrite dans le référentiel propre à l'utilisateur (changement d'unités, calcul d'une moyenne, ...) au moyen de la classe "mesure opérationnelle" définie dans le chapitre IV.

Les transformations subies par les mesures primaires et les mesures auxiliaires sont de type informationnel :

mesure(s) primaire(s) et auxiliaire(s)	en	mesure fonctionnelle
mesure(s) fonctionnelle et technologique(s)	en	mesure validée (qualifiée)
mesure validée	en	mesure(s) opérationnelle(s)

Toutes les classes de la hiérarchie de mesure ne possèdent qu'une seule opération accessible par les autres classes (cette opération est dite "publique" puisqu'elle est visible de l'extérieur): l'opération "élaborer" qui retourne un compte-rendu de mesure comprenant la valeur de la mesure élaborée, ainsi que les informations de validation qui ont été élaborées.

Les opérations "acquérir", "calculer" et "valider" sont des opérations internes aux classes de la hiérarchie de mesure et contribuent à l'implémentation de l'opération publique "élaborer".

Puisque les méthodes privées dépendent de décisions internes d'implémentation, il est très facile au concepteur de modifier le nombre et le type des arguments si l'implémentation change. Ces méthodes privées accroissent la modularité. Les détails internes de la méthode affectent seulement les méthodes de la classe, et non les autres.

La classe "mesure primaire" à donc été spécialisée (héritage) en deux classes, une classe "F.é.m thermocouple" pour la hiérarchie de mesure relative à un thermocouple, et une classe "Température résistance de platine" pour la hiérarchie de mesure relative à une résistance de platine. Dans cette dernière, on calcule la température de la résistance de platine traversée par le courant d'excitation. Une validation portant sur les mesures corrigées en sortie de la chaîne d'acquisition permet de déterminer la validité de ces mesures par rapport à leur domaine de fonctionnement (f.é.m thermocouple corrigée, tension résistance de platine corrigée et courant résistance de platine corrigée). Cette première validation au niveau de la hiérarchie de mesure peut permettre de détecter une rupture ou une absence de transducteurs.

De même, la classe "mesure auxiliaire" a été spécialisée en deux classes conformément aux hiérarchies d'obtention d'une mesure opérationnelle pour un thermocouple et une résistance de platine, en une classe "tension soudure froide équivalente thermocouple" et une classe "échauffement résistance de platine". Une validation est également faite à ce niveau permettant par exemple, dans le cas d'une résistance de platine, de vérifier à partir du calcul de l'autoéchauffement s'il n'y a pas une dégradation de celle-ci.

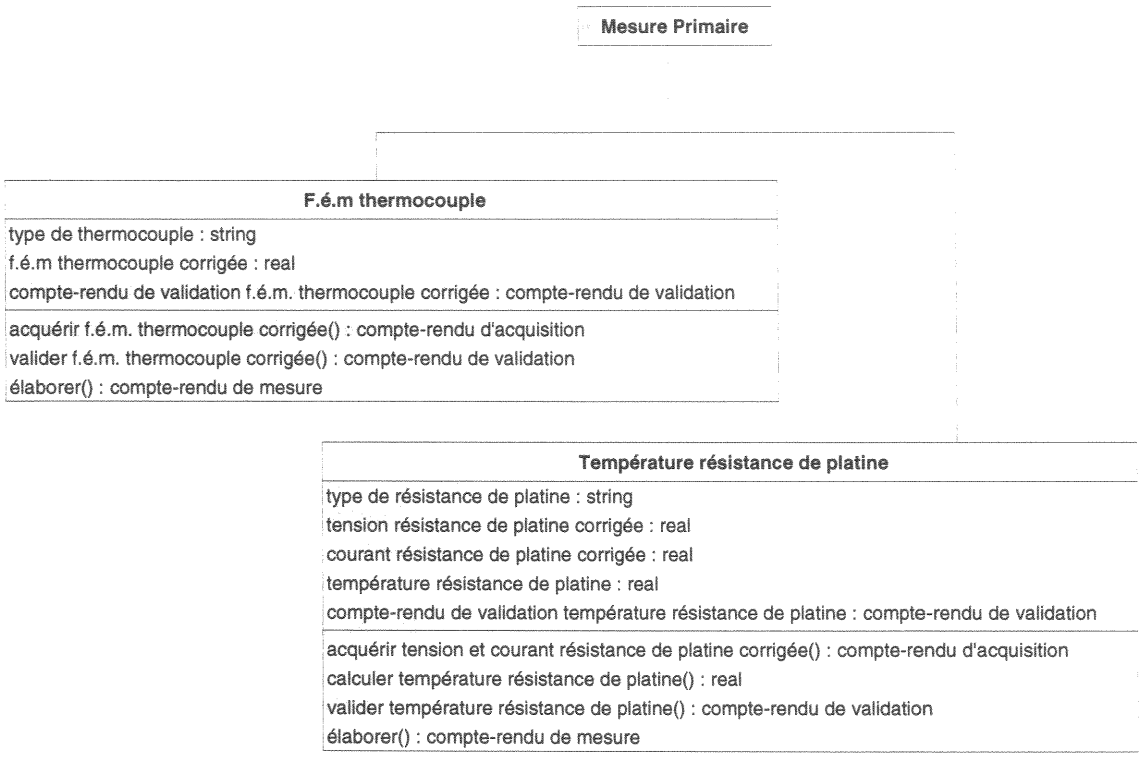


Figure VI.8 : Portion du diagramme d'objet de la hiérarchie de mesure montrant la hiérarchie d'héritage pour la classe "Mesure Primaire".

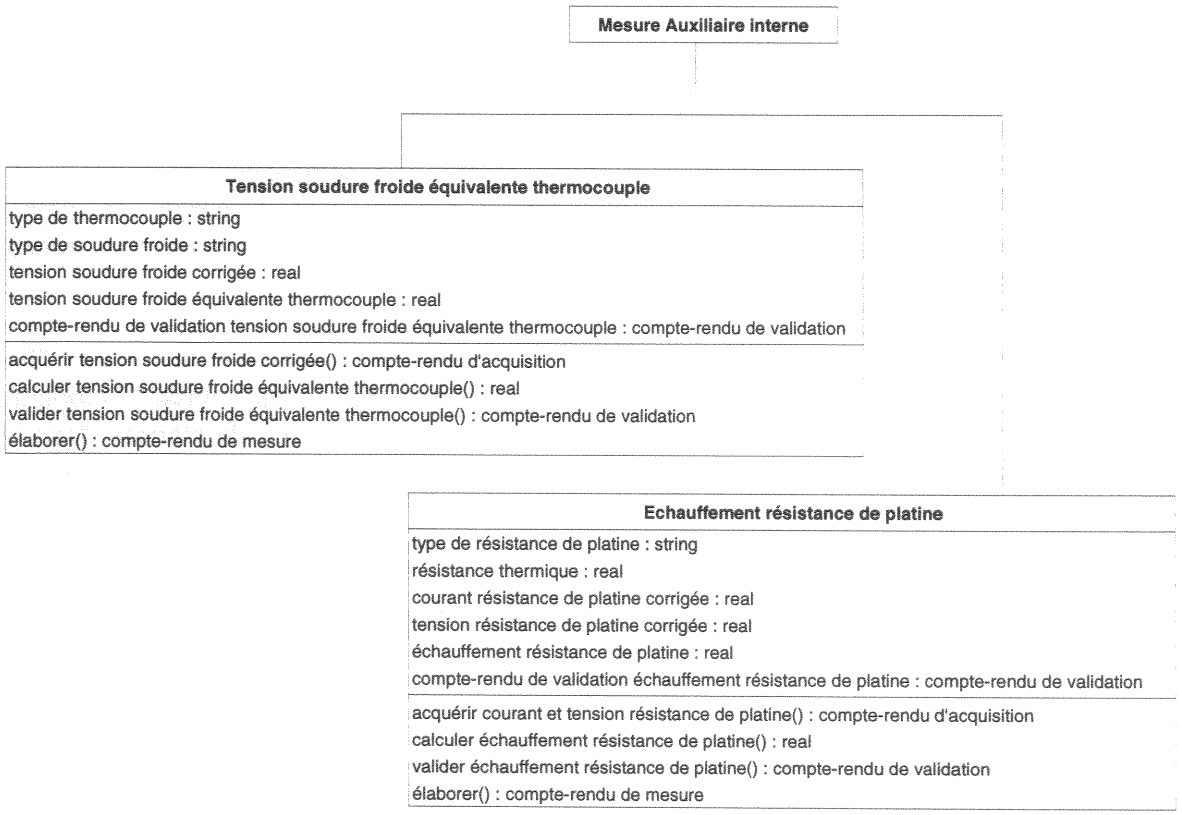


Figure VI.9 : Portion du diagramme d'objet de la hiérarchie de mesure montrant la hiérarchie d'héritage pour la classe "Mesure Auxiliaire Interne".

Les opérations réalisant l'acquisition dans les classes "mesure primaire" et "mesure auxiliaire" sont en relation directe avec le "contrôleur d'acquisition".

L'obtention d'une mesure fonctionnelle correspond, dans le cas d'un thermocouple (classe "température compensée thermocouple"), après compensation de la f.é.m du thermocouple par la tension de soudure froide, à la température mesurée par ce transducteur. Dans le cas d'une résistance de platine (classe "température compensée résistance de platine"), la mesure fonctionnelle correspond à la température mesurée par ce transducteur élaborée à partir de l'échauffement et de la température de la résistance de platine traversée par le courant d'excitation. La mesure fonctionnelle est ensuite validée (validation métrologique) par rapport au domaine d'utilisation du transducteur.

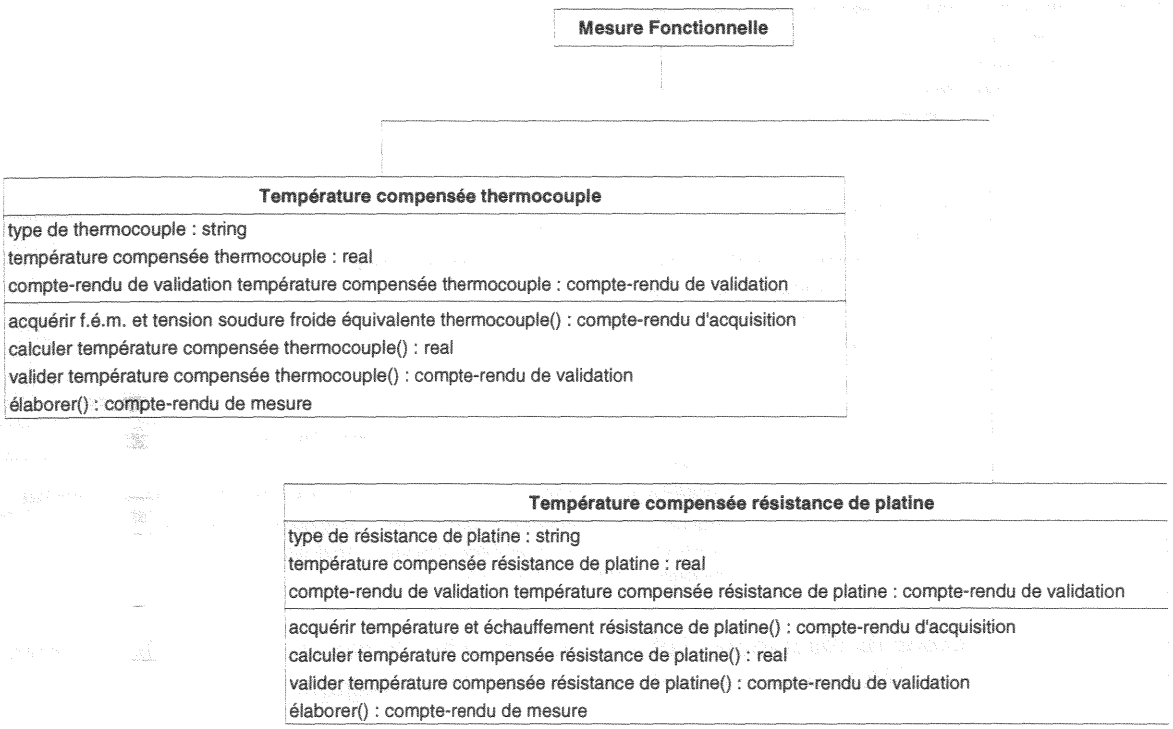


Figure VI.10 : Portion du diagramme d'objet de la hiérarchie de mesure montrant la hiérarchie d'héritage pour la classe "Mesure Fonctionnelle".

Une validation portant sur la mesure d'une tension de référence et le zéro de la chaîne d'acquisition permet de déterminer l'intégrité de cette dernière lors de la phase d'acquisition de la grandeur principale et des grandeurs d'influence. Ces deux grandeurs sont donc considérées ici comme grandeurs d'autocontrôle.

Une autre validation portant sur la température ambiante permettra de déterminer si l'intervalle "température de fonctionnement du capteur intelligent" est bien respecté. La mesure de la température ambiante est réalisée par un composant spécifique qui pourra être, dans le cas où l'on utilise un thermocouple, le composant de soudure froide (la tension de soudure froide sera considérée également comme grandeur d'autocontrôle).

De plus, dans le cas où l'on utilise une résistance de platine comme transducteur principal, une validation portant sur le courant passant dans la résistance de platine permet la détection et la correction des dérives éventuelles du générateur de courant. Le courant passant dans la résistance de platine sera considéré ici comme grandeur d'autocontrôle.

Toutes ces mesures technologiques permettent l'adjonction à la mesure fonctionnelle d'une information supplémentaire caractérisant l'état de fonctionnement de certains constituants matériels du capteur à l'instant où s'effectue la mesure formant ainsi une mesure validée (classe "mesure validée").

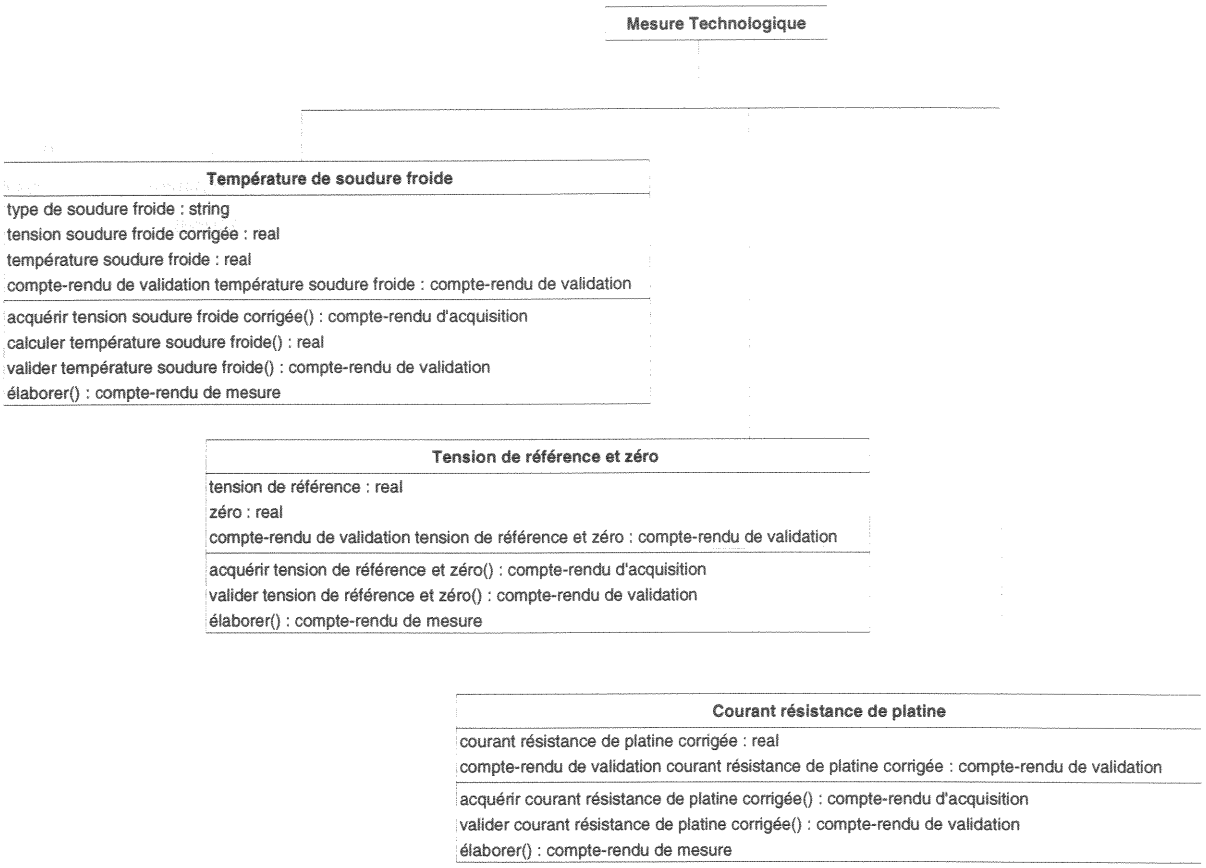


Figure VI.11 : Portion du diagramme d'objet de la hiérarchie de mesure montrant la hiérarchie d'héritage pour la classe "Mesure Technologique".

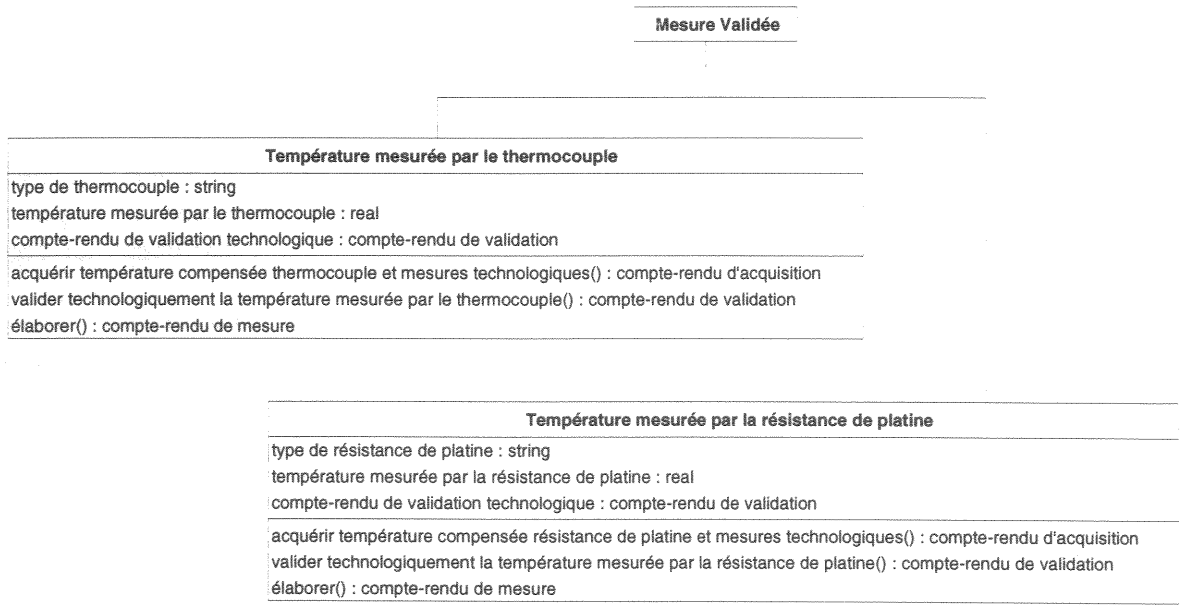


Figure VI.12 : Portion du diagramme d'objet de la hiérarchie de mesure montrant la hiérarchie d'héritage pour la classe "Mesure Validée".

La crédibilité des mesures est accrue grâce notamment à cet ensemble de validations. Ces corrections sont essentiellement celles des défauts dus à l'électronique (dérive d'alimentation, ...), liés aux conditions d'utilisation (dépassement de l'étendue de mesure, ...).

Toutes ces classes définies précédemment sont associées au gestionnaire du temps interne défini dans le chapitre IV permettant la datation des différentes mesures élaborées, ainsi qu'à un historique permettant de stocker les informations les plus pertinentes comme les informations de validation élaborées tout au long de l'élaboration de la mesure opérationnelle susceptibles d'aider l'utilisateur à planifier la maintenance du capteur.

VI.5 Les modèles

Comme il a été mentionné au chapitre I, les capteur intelligents intègrent évidemment des composants numériques, mais conservent souvent des modules analogiques, nécessaires à l'extraction du signal primaire. Ainsi, le principal service attendu d'un capteur intelligent est la fourniture d'une information la plus crédible et la plus exacte possible pour une plage d'utilisation la plus grande possible. Cette dualité, caractérisée par la rangeabilité, implique le plus souvent des traitements numériques qui permettent de compenser les non-linéarités intrinsèques soit des lois physiques, soit de certains composants analogiques.

Par exemple, la compensation de la jonction de référence pour un thermocouple peut être de deux sortes :

- la compensation par matériel, où l'on insère une tension de compensation en série avec l'un des conducteurs de manière que la f.é.m. mesurée corresponde à la température de la jonction chaude. De nombreux circuits sont réalisables tels qu'un pont de Wheatstone, ou des circuits intégrés assurant à la fois la compensation de la jonction froide et l'amplification;
- la compensation par logiciel, où le calculateur est informé de la température de jonction froide, lui permettant ainsi de calculer la température de la jonction chaude. Le temps supplémentaire pour effectuer ces calculs retarde légèrement la saisie des températures, par rapport à la compensation par matériel.

Pour pouvoir élaborer la température mesurée par un thermocouple ou une résistance de platine, le calculateur doit contenir les tables correspondant aux thermocouples et aux résistances de platine susceptibles d'être utilisés. En pratique, cela encombrerait la mémoire de stockage d'une manière prohibitive. Cet inconvénient peut être évité en calculant les valeurs approchées des tables à l'aide d'un polynôme de degré relativement élevé (ordre : 9) ou de plusieurs segments dans lesquels un polynôme de degré moins élevé (ordre : 3) donne une bonne précision. Cette méthode est plus précise et plus rapide que celle qui fait appel à un unique polynôme.

Tous les modèles élaborés sont ici très simples car il s'agit de modèles mathématiques n'intégrant pas, les dérives des caractéristiques des composants électroniques dues notamment à leurs vieillissement. Pour remédier à ce problème, on pourra toutefois recalculer certains de ces modèles lors des phases d'étalonnage, afin de disposer toujours des modèles correspondant au plus près aux composants qu'ils modélisent. On peut citer par exemple les modèles des amplificateurs à gain programmables, ainsi que le modèle du composant de soudure froide.

Les deux figures ci-dessous montrent quels sont les modèles mis en jeux lors de l'élaboration de la mesure de température par une résistance de platine et un thermocouple.

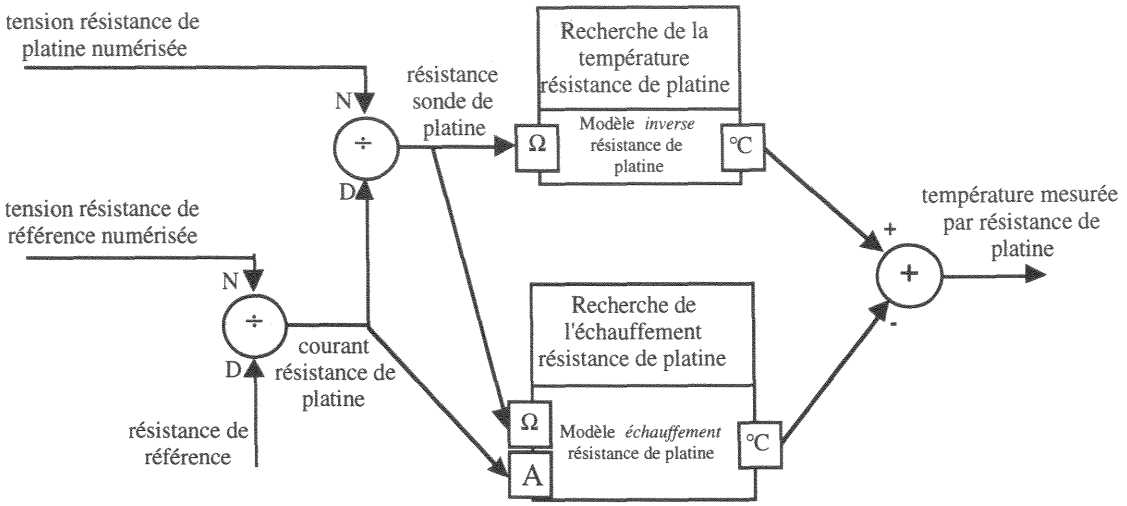


Figure VI.13 : Modèles et grandeurs utilisés pour le calcul d'une température mesurée par une résistance de platine.

Dans le cas d'un thermocouple, deux principes de mesure de jonction froide sont proposés :

- interne où la mesure de la jonction froide est réalisée par le composant de soudure froide,
- et externe où la mesure de la jonction froide est réalisée par une résistance de platine additionnelle.

La méthode de recherche de la température mesurée par le thermocouple (jonction chaude) assure une indépendance maximale du programme vis à vis du matériel. En effet, le changement du principe matériel de mesure de la jonction froide n'implique alors que la modification du modèle inverse de la jonction froide (tension en température).

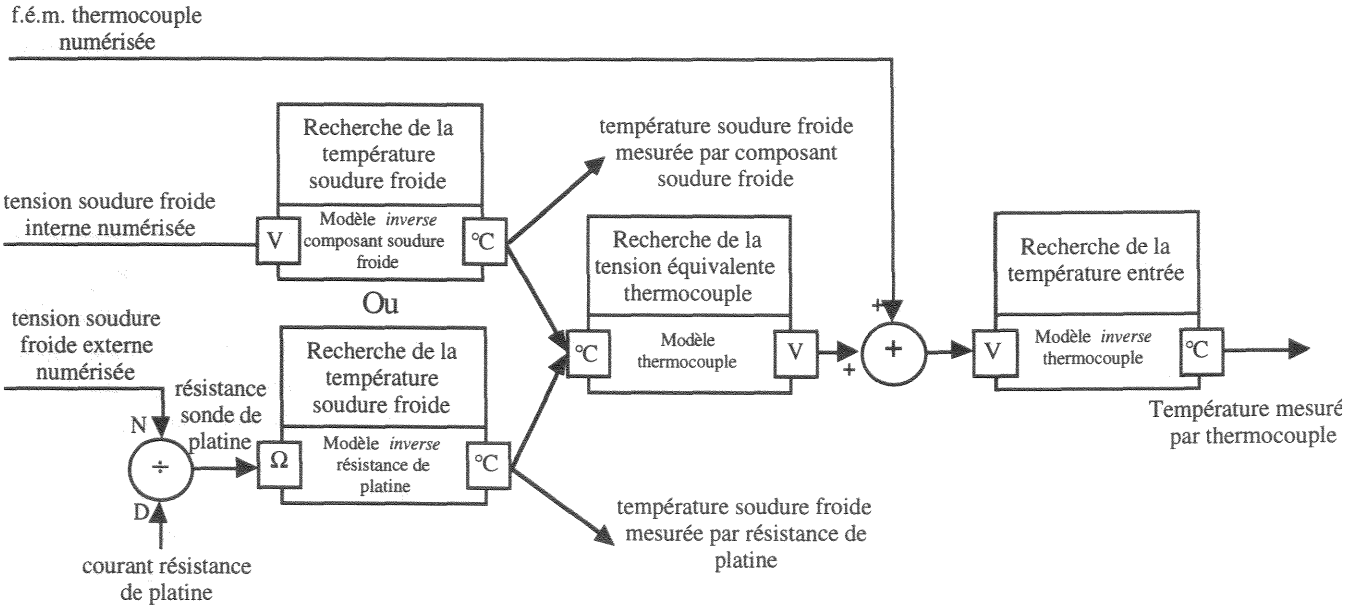


Figure VI.14 : Modèles et grandeurs utilisés pour le calcul d'une température mesurée par un thermocouple.

Les solutions classiques : linéarisation par morceaux, régression polynomiale par morceaux, reposent sur des techniques d'identification issues des moindres carrés; cependant les réseaux de

neurones étant destinés par essence à la modélisation des lois et systèmes non linéaires, nous avons testé et utilisé cette approche et comparer ses performances vis-à-vis des solutions classiques antérieurement implantées.

VI.5.1 Les modèles neuronaux

Réseaux de neurones

Rappelons que l'utilisation des réseaux de neurones comporte deux étapes :

- une phase dite d'apprentissage, au cours de laquelle le réseau est alimenté par des couples de valeurs (entrée, sortie). A chaque entrée, le réseau fait correspondre une sortie estimée qui, comparée à la sortie désirée, fournit une erreur d'estimation. La rétro propagation de cette erreur dans le réseau va alors faire évoluer les poids associés aux connexions inter-neurones jusqu'à ce que la sortie simulée par le réseau tende vers la sortie mesurée.
- une phase d'utilisation, au cours de laquelle la valeur en entrée est simplement propagée au sein du réseau, afin de fournir une valeur de sortie.

Architecture d'un réseau de neurones

Dans notre application, les réseaux de neurones utilisés sont des réseaux comportant une seule couche cachée.

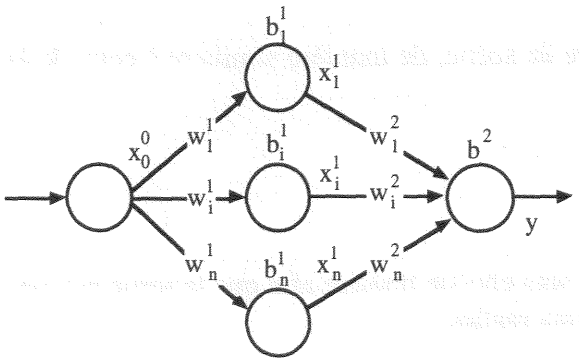


Figure VI.15 : Réseau de neurones à 3 couches.

Le réseau représenté figure VI.15 est composé de neurones interconnectés en trois couches successives.

La première couche, ou couche d'entrée, est composée d'un neurone "transparent" n'effectuant aucun calcul mais distribuant uniquement l'entrée à l'ensemble des neurones de la couche suivante appelée couche cachée. Cybenko [CYBE-89] ou Funahashi [FUNA-89] ont établi que les réseaux de neurones avec une seule couche cachée sont suffisants pour l'identification de n'importe quelle fonction continue. La dernière couche, appelée couche de sortie, est ici constituée d'un seul neurone fournissant la sortie estimée de la fonction à identifier.

Les neurones de la couche cachée sont représentés à la figure VI.16.

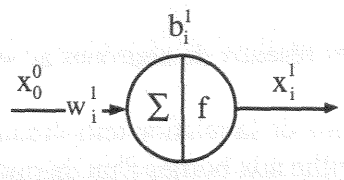


Figure VI.16 : Neurone i de la couche cachée.

Le neurone i de la couche cachée reçoit l'entrée $\left\{x_0^0\right\}$ de la couche d'entrée qui est associée à un poids $\left\{w_i^1\right\}$

Ces neurones calculent tout d'abord :

$$z_i^1 = w_i^1 x_0^0 + b_i^1 \qquad \text{(E.4)}$$

Où $\left\{b_i^1\right\}$ est un facteur de biais. La sortie du neurone est le résultat de (E.4) à travers une fonction d'activation :

$$x_i^1 = f\left(z_i^1\right) \qquad \text{(E.5)}$$

Cette fonction d'activation a été choisie classiquement parmi les fonctions sigmoïdes ; elle est de la forme :

$$f(x) = \frac{2}{\left(1+e^{-2x}\right)} - 1 \qquad \text{(E.6)}$$

Comme cette fonction s'active de -1 à +1, il est nécessaire au préalable pour des problèmes d'initialisation du réseau en phase d'apprentissage, de normaliser l'entrée dans cet intervalle.

Le neurone de la couche de sortie, de manière similaire à ceux de la couche cachée, calculera la somme pondérée suivante:

$$y = \sum_{i=1}^{n} w_i^2 x_i^1 + b^2 \qquad \text{(E.7)}$$

Sa fonction d'activation sera choisie linéaire afin que la sortie estimée du réseau puisse prendre en compte l'ensemble des valeurs réelles.

Algorithmes d'apprentissage

La phase d'apprentissage repose sur l'utilisation d'un algorithme de rétro-propagation du gradient [RUME-86] basé sur l'évolution d'un critère quadratique, pour lequel les paramètres, valeurs des poids et des biais, évoluent dans la direction opposée à celle du gradient lié au critère.

La technique d'optimisation utilise une approximation de la méthode de Newton connue sous l'appellation approximation de Levenberg-Marquardt [LEVE-44] [MARQ-63].

Diverses boîtes à outils informatiques sont disponibles sur le marché. Les algorithmes ont été implémentés par l'intermédiaire de la boite à outils Neural Network Matlab (the MathWorks, Inc.).

Résultats, performances

Nous avons illustré l'intégration des réseaux de neurones au sein des capteurs intelligents à travers une application :

- représentation mathématique de la relation température/f.é.m., permettant de convertir une force électromotrice recueillie aux bornes d'un thermocouple en une température.

Exemple : cas du thermocouple

Afin de respecter, en terme d'exactitude, le cahier des charges initial, la fonction Température = f(f.é.m.) a été caractérisée sur 7 segments à l'aide de polynômes de degré 4. Cette solution a été implantée au sein d'un transmetteur intelligent de température et nous servira de référence par la suite.

Le modèle neuronal utilisé pour modéliser cette fonction sur cette même plage de température utilise quant à lui une couche cachée comportant 10 neurones.

Dans les deux cas, les valeurs utilisées en apprentissage constituent un ensemble de 670 couples (température, f.é.m.).

	Régression Polynomiale par Segments	Réseau de Neurones
Erreur maximale en °C	0.266	0.023
Somme des carrés des erreurs en $°C^2$	0.273	0.011
Nombre de coefficients	45	35
Rapidité d'obtention des coefficients	quelques mn	plusieurs dizaines de mn

Résultats de l'apprentissage pour un thermocouple de type T.

Le tableau ci-dessus synthétise les performances de chacune des 2 techniques.

L'examen de ce tableau amène les remarques suivantes :

- le modèle neuronal est meilleur que le modèle obtenu par régression polynomiale, en particulier, la valeur absolue de l'erreur maximale est réduite dans un rapport supérieur à 10,
- le nombre de coefficients permettant d'exprimer la relation Température = f(f.é.m.) est diminué de plus de 20%. Cette remarque est importante si l'on considère le fait que plus d'une dizaine de thermocouples différents sont susceptibles d'être raccordés au capteur intelligent; ce qui implique une réduction de la taille de la zone mémoire (non volatile) nécessaire au stockage des coefficients,
- la durée de la phase d'établissement des modèles est beaucoup plus importante dans le cas Réseau de Neurones, cependant ce calcul n'est à faire qu'une seule et unique fois.
- le temps de calcul en phase d'utilisation est sensiblement le même pour les deux méthodes de modélisation.

Les réseaux de neurones constituent donc un bon compromis entre la précision des modèles et l'occupation mémoire, mais il y a un compromis à trouver entre la durée d'apprentissage, l'exactitude souhaitée pour les modèles, et l'occupation mémoire.

Cette technique nous semble la plus satisfaisante, nous l'avons donc utilisée pour l'élaboration de tous les modèles mathématiques utilisées pour l'élaboration d'une mesure opérationnelle (modélisation des thermocouples, des résistances de platine, et des caractéristiques des amplificateur à gain programmables qui présentent des non-linéarités).

VI.6 Validation des modèles comportementaux des objets

La validation des modèles comportementaux est une phase nécessaire dans l'établissement d'un modèle de référence de capteurs intelligents.

VI.6.1 Contrôle par tâche concurrentes

Un objet peut être implémenté comme une tâche d'un système d'exploitation ou d'un langage de programmation. C'est l'approche la plus générale, car elle préserve la concurrence inhérente des objets réels. Les événements sont implémentés comme des appels inter-tâches en employant les services du langage ou du système d'exploitation. Certains langages, comme Concurrent C++, prennent en charge la concurrence, mais l'utilisation de ces langages, en production de logiciel, est encore limitée. Ada prend également en charge la concurrence, pourvu qu'un objet soit lui-même une tâche Ada, mais le coût en support système est élevé. Les langages à objet les plus diffusés ne prennent pas encore en compte la concurrence.

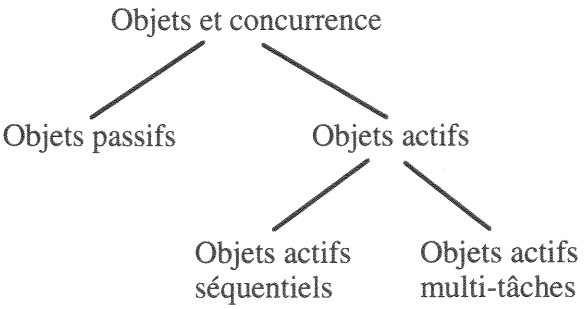


Figure VI.17 : Les différentes catégories d'objet.

On peut distinguer deux classes d'objet :
- les objets passifs
- les objets actifs (objets qui ont un modèle dynamique)

Les objets actifs sont des objets qui peuvent être assimilés à des tâches concurrentes, c'est à dire fonctionnant en même temps. Chaque objet actif peut être représenté par un modèle dynamique (diagramme d'états) qui montre l'évolution de son comportement pour des sollicitations extérieures. La communication entre les objets actifs s'effectue uniquement au moyen de messages (l'échange de messages étant asynchrone). Par contre, les objets passifs sont des objets dit statiques puisqu'ils n'ont pas de modèle dynamique. L'accès à ces objets se fait uniquement par l'appel de leurs méthodes.

Un objet est parfois obligé d'exécuter deux ou plusieurs activités concurremment. Les étapes internes de ces activités ne sont pas synchronisées, mais les deux activités doivent être achevées avant que l'objet puisse progresser vers l'état suivant.

Il existe quatre types d'interaction entre les différents objets, qu'ils soient actifs ou passifs :
- interaction objet passif / objet passif : la communication se fait par l'appel de procédures,
- interaction objet actif / objet actif : la communication se fait par l'envoi de messages,
- interaction objet actif / objet passif : la communication se fait par l'appel de procédures,
- interaction objet passif / objet actif : la communication se fait par l'envoi de messages.

La communication par message est basée sur des opérations explicites d'envoi à destination d'une boîte à lettres et de réception à partir d'une boîte à lettres. Les boîtes à lettres sont des instances de la classe "boite à lettres" et sont associées à chaque objet actif.

Dans la pratique on utilise seulement les trois premiers types d'interaction, à savoir l'interaction entre objets actifs, l'interaction entre objets passifs, et l'interaction d'un objet actif vers un objet passif.

La communication entre les objets actifs doit pouvoir être synchrone ou asynchrone. Comme la méthode OMT définit uniquement la communication asynchrone entre les objets, la communication synchrone se fera alors par l'envoi de deux messages asynchrones. Le contrôle de la communication synchrone doit être intégré dans les machines à états finis de chaque objet actif participant à la communication. L'objet actif émetteur restant bloqué jusqu'à ce qu'une réponse lui soit donnée (un intervalle de temps maximum d'attente durant laquelle la réponse doit être reçue peut être spécifié). Pendant cet intervalle, l'objet ne peut répondre à aucun autres messages provenant d'autres objets.

VI.6.2 Translation de LOV/OMT vers GEODE

L'outil LOV/OMT (cf. chapitre IV pour la description de l'outil) ne permettant pas de simuler l'ensemble des différents diagrammes d'états définis pour les classes actives (modèle dynamique), il est nécessaire si l'on veut valider la vue comportementale, d'utiliser un outil adéquat. Pour ce faire, nous utiliserons l'outil GEODE commercialisé par la société VERILOG, qui permet de simuler les interactions entre les différents objets actifs du modèle OMT. Ainsi, après une vérification de la cohérence de l'ensemble des diagrammes d'états nous pourrons alors passer à la phase d'implémentation.

Le passage de LOV/OMT vers GEODE ne se fait pas sans mal. En effet, les notations, pour représenter les modèles dynamiques, de ces deux outils sont différentes et nécessitent une transformation. Le passage de l'un vers l'autre est présenté ci-dessous.

Il aurait été préférable d'avoir un outil intégré (solution tout objet) qui permette à la fois l'édition, la simulation et l'implémentation des modèles (tout en utilisant les mêmes notations, OMT entre autres). Celui-ci n'existant pas encore sur le marché, nous avons du nous contenter des outils disponibles, à savoir LOV/OMT pour l'édition du modèle de référence du capteur intelligent, GEODE pour la simulation.

La schéma ci-dessous présente le principe de transformation utilisé pour le modèle, afin de pouvoir le simuler.

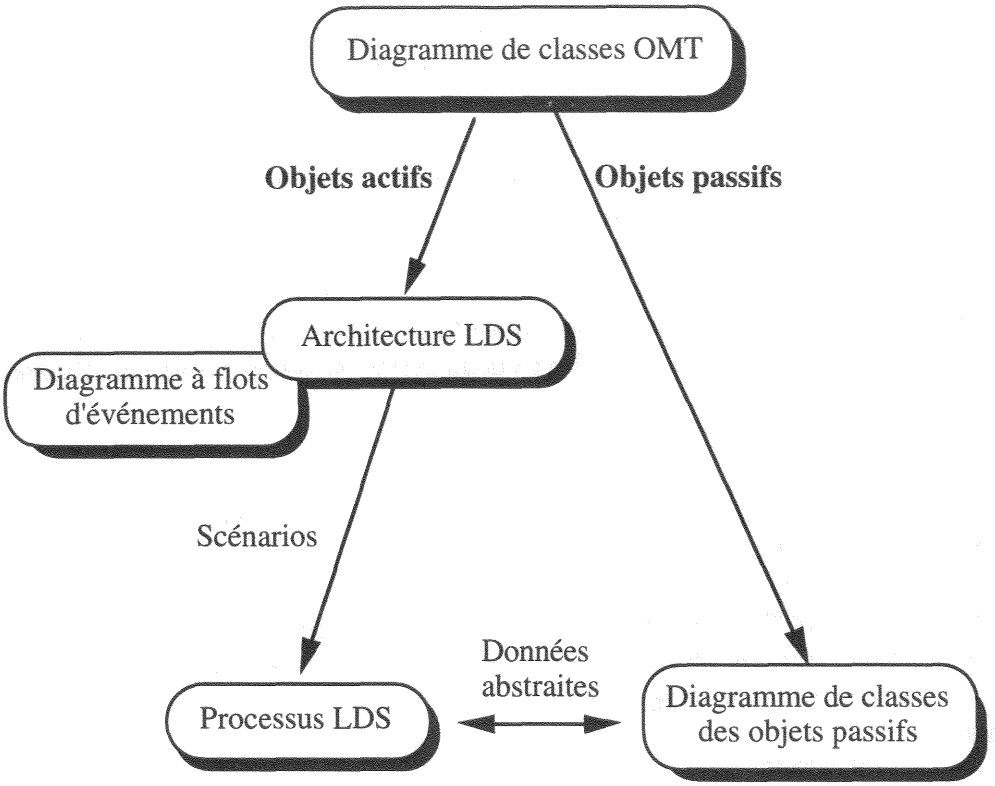


Figure VI.18 : Translation de LOV/OMT vers GEODE.

Nous avons décrit précédemment les deux différents types d'objets (objets actifs et passifs) pouvant intervenir dans un modèle objet, ainsi que les interactions permettant la communication entre ces deux types d'objet.

En utilisant le générateur "LOV/OMT vers LDS (Langage de Déscription et de Spécification)" fourni avec les deux outils, la description OMT du système est directement transformée en une architecture LDS, suivant les principes ci-dessous :

Pour les objets actifs (figureVI.19) :

- Les classes représentant les objets actifs sont transformées en "bloc" ou en processus LDS représentant l'architecture du système (un "bloc" correspond à la classe origine ou à une classe intermédiaire dans une relation d'agrégation; un processus correspond à une classe de bas niveau dans les relations d'agrégation);
- Les relations entre les objets sont transformées en canaux LDS entre les "blocs" ou les processus;
- Les attributs des classes sont transformés en variables pour les processus;
- Les opérations peuvent être transformées en activités.

Pour les objets passifs (figure VI.20):

- Les classes sont transformées en type abstrait (ADT : Abstract Data Types);
- Les attributs et opérations peuvent également êtres transformés en types abstrait.

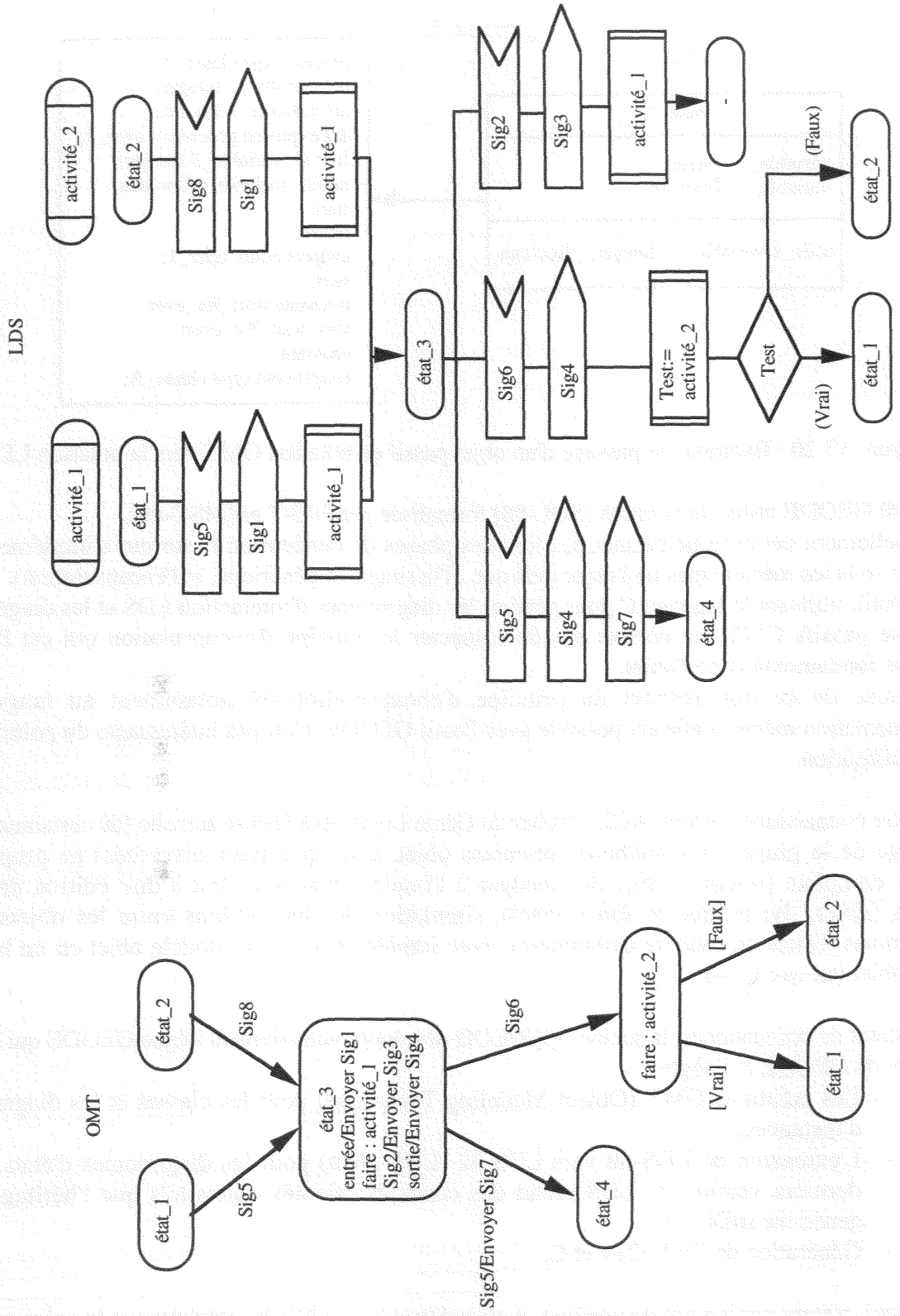


Figure VI.19 : Exemple de passage d'un diagramme dynamique d'un objet actif en notation OMT vers un diagramme dynamique en notation LDS.

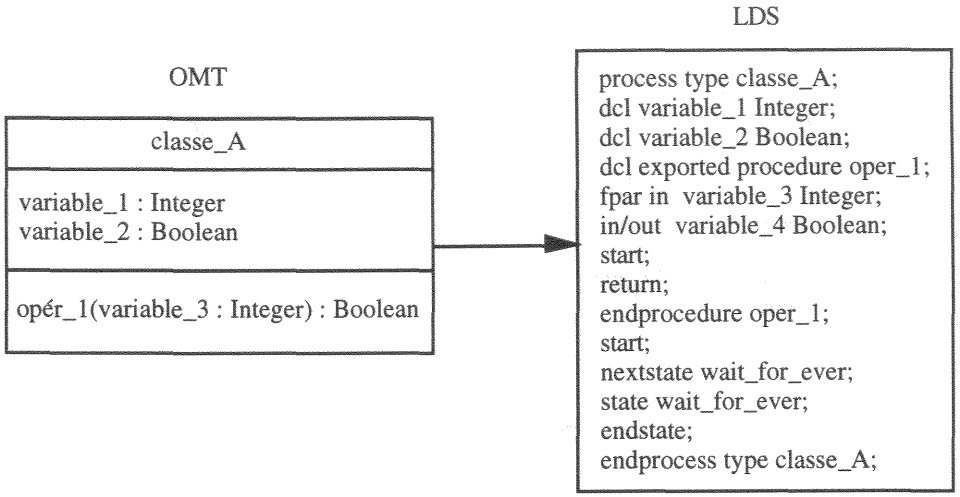


Figure VI.20 : Exemple de passage d'un objet passif en notation OMT vers la notation LDS.

L'outil GEODE utilise la notation [SDL-88] formalisée par ITU-T et [MSC-88].

Actuellement cet outil ne permet pas lors des phases de conception et surtout d'implémentation d'utiliser tous les mécanismes de l'objet (tels que : l'héritage, la généricité, et l'encapsulation).

Cet outil, utilisant le langage C pour générer les diagrammes d'interaction LDS et les diagrammes de classe passifs OMT, ne permet pas de respecter le principe d'encapsulation qui est l'un des principes fondamentaux de l'objet.

A cause de ce non respect du principe d'encapsulation dû notamment au langage C, l'implémentation même si elle est possible avec l'outil GEODE, n'est pas intéressante du point de vue de la réutilisation.

A notre connaissance, aucun AGL (Atelier de Génie Logiciel) à l'heure actuelle (dû certainement au jeune âge de la plupart des méthodes orientées objet, ainsi qu'à leurs diversités) ne propose de solution complète (orientée objet de l'analyse à l'implémentation), c'est à dire édition des trois modèles (objet, dynamique et fonctionnel), simulation des interactions entre les objets actifs (diagrammes d'états du modèle dynamique), avec implémentation du modèle objet en un langage orienté objet (tel que C++).

Conscient de ces manques, la société VERILOG développe actuellement ObjectGEODE qui est une extension de GEODE et intègre :

- Les notations OMT (Object Modeling Technique) pour les classes et les diagrammes d'instances;
- L'extension de LDS-88 vers LDS-92 (LDS objet) pour les diagrammes d'états. Cette dernière version de LDS inclut des concepts orientés objets tels que l'héritage et la généricité [SDL-93];
- Génération de Code C++ et C.

Cet outil, n'étant par à notre disposition, il est préférable une fois la simulation et la validation des modèles comportementaux (modèle dynamique) effectués par GEODE, d'implémenter les modèles OMT suivant une solution orientée objet. C'est à dire, une implémentation des modèles OMT en un langage orienté objet (ce que ne fait pas GEODE) sur un système d'exploitation si possible orienté objet et temps réel.

Nous avons donc implémenté dans un souci de respect du mécanisme d'encapsulation (qui est une des conditions nécessaires pour la réutilisation, voir chapitre V) dans un langage orientée objet (C++) à partir de l'outil LOV/OMT la partie du modèle objet qui permet l'élaboration d'une mesure opérationnelle, ainsi que la gestion des modes d'utilisation sur une station UNIX (Sun IPC avec SunOS4.1.1 comme système d'exploitation et Openwindows 3 comme interface graphique).

Nous avons lors de l'implémentation défini quelques objets de présentation, correspondant à une vision métier d'un ou plusieurs objets du domaine. Leur intérêt essentiel est d'isoler la représentation qui est faite des objets du domaine, des objets eux-mêmes (figure VI.21). Par exemple, la représentation qui est faite de la mesure opérationnelle pourra être différente en fonction du métier des utilisateurs; un opérateur de maintenance s'attachera plus aux informations de validation associées à la mesure qu'à la mesure elle-même. Ces informations seront alors mises au premier plan. Les moyens techniques sont les solutions en terme de techniques de dialogue à la représentation des objets de l'application. Les objets d'application que nous avons utilisés pour faire l'interface entre les utilisateurs et le capteur intelligent se trouvent en annexe (ce sont essentiellement des fenêtres interactives qui permettent entre autres la configuration du capteur intelligent).

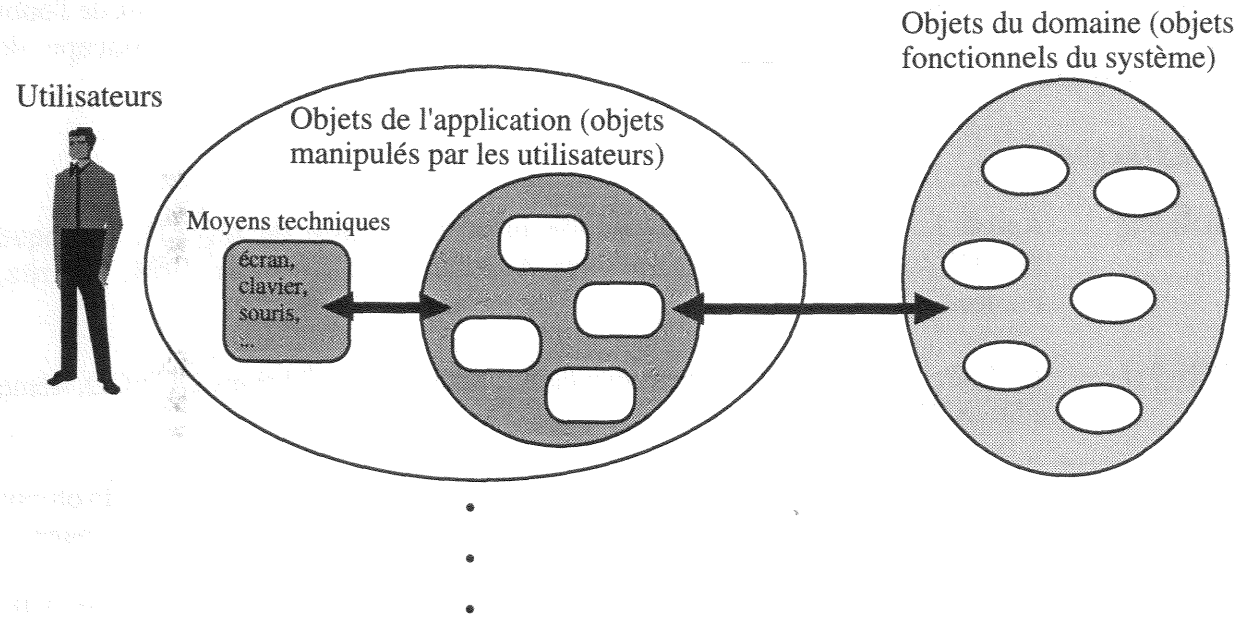


Figure VI.21 : Objets de l'application et objets du domaine [ANDR-94].

VI.7 Conclusion

La simulation dans l'environnement GEODE en vue de la validation des modèles comportementaux des objets, est une étape nécessaire dans l'établissement d'un modèle de référence de capteur intelligent. Cette étape permet la validation des diagrammes d'états de chaque classe, ainsi que la vérification de la cohérence de l'ensemble des classes du modèle objet. La simulation a permis de vérifier les descriptions génériques, puisque le modèle objet n'a pas été spécialisé à un cas particulier de capteur intelligent pour la simulation. La description comportementale des objets spécifie ce que font les objet en réponse aux événements. Les opérations attachées aux états ou aux événements sont exécutées en réponse aux états ou aux événements correspondants. Les opérations utilisées pour la simulation sont abstraites, leur forme est définie, mais pas leur implémentation. L'aspect (contrôle) du système qui décrit des séquences d'opérations activée par des stimuli externes, ne tient pas compte de l'activité de ces opérations ou de leur mode d'implémentation.

On peut citer les quelques difficultés rencontrées lors de la conversion des machines à états finis en LDS. Dans les machines à états finis, un état peut contrôler une activité continue qui persiste jusqu'à ce qu'un événement y mette fin en déclenchant une transition de l'état. Par contre, en LDS (sous GEODE) les activités sont exécutées dans les transitions d'états, ces derniers ne pouvant contrôler les activités. La gestion des files d'attente pour les événements en LDS est également différente du mécanisme de boîte aux lettres utilisée par les objets. Une mémorisation des événements est nécessaire afin de pouvoir gérer les files d'attentes LDS comme des boîtes aux lettres. Ceci implique de légères modifications des diagrammes d'états convertis en LDS.

L'instanciation a un cas particulier de capteur intelligent (dans notre cas : un capteur intelligent de température) permet de valider les techniques de réutilisation mentionnée dans le chapitre IV (mécanisme d'héritage), et de valider l'implémentation des opérations qui a été faite, dans les classes des hiérarchies de mesure pour les thermocouples et les résistances de platine.

Deux solutions distinctes doivent être examinées :

Puisque GEODE tel que nous l'avons ne permet pas une implémentation suivant le concept objet (pas d'encapsulation et d'héritage) l'implémentation des modèles OMT doit se faire à partir de l'outil LOV/OMT ou à partir d'un autre outil permettant une implémentation suivant les concepts de l'orientation objet.

VI.8 Références

[ANDR-94] André J., Brisson G., "modélisation objet d'un poste de travail", Colloque ERGO'IA, Ergonomie&Informatique Avancée, 26-28 octobre 1994, Biarritz, France.

[ASCH-91] Asch G. "Les capteurs en instrumentation industrielle", Éditions DUNOD, 4ème édition, 1991.

[BELI-91] Belina F., Hogrefe D., Samara A., "SDL with application from Protocol specification", Édition Redwood Books, Trowbridge, Wiltshire, 1991, 275 pages.

[BRIG-94] Brignell J., Taner A., "Aspects of smart sensor reconfiguration", Eurosensors VIII conference, Toulouse, France, septembre 25-28, 1994.

[CYBE-89] Cybenko G., "Approximation by superpositions of a sigmoidal function", Math. Control Signals Systems, N°2, 1989, pages 303-314.

[DELA-93] Delannoy C., "Programmer en langage C++", Édition Eyrolles, 1993, 438 pages.

[FUNA-89] Funahashi K., "On the approximate realization of continuous mappings by neural networks", Neural Networks, Vol.2, 1989, pages 183-192.

[GEHI-94] Géhin A.L., "Analyse fonctionnelle et modèle générique des capteurs intelligents: Application à la surveillance de l'anesthésie", Thèse de doctorat de l'université des sciences et technologies de Lille, 26 Janvier 1994.

[GEOF-94] Geoffroy P., "Réalisation d'un transmetteur intelligent de température", Mémoire CNAM, Nancy, Mars 1994.

[KOUR-95] Khoury A., "Transmetteur intelligent de temperature", Mémoire CNAM, Nancy, 26 janvier 1995.

[LEVE-44] Levenberg K., "A method for the solution of certain nonlinear problems in least squares", Quart. Appl. Math., Vol 2, pages 164-168, 1944.

[MAGI-93] Magison E., "Selecting the right temperature sensor", INTECH, February 1993, pages 29-32.

[MARQ-63] Marquardt, "An algorithm for least squares estimation of nonlinear parameters", J. SIAM, Vol.11, pages 431-441, 1963.

[MILL-84] Millet F., "La mesure des temperatures: de l'ambiante à 2500K", Edition pyc, 1984, 171 pages, ISBN 2-85330-070-6.

[MSC-88] Recommendation Z.120, "Message Sequence Chart, MSC", ITU, Switzerland, 1988.

[NOIZE-91] Noizette J.L., "Méthodologie de conception des capteurs intelligents : application a un capteur granulometrique", Thèse de doctorat de l'université de Nancy 1, 29 novembre 1991.

[NOIZE-95] Noizette J.L., Khoury A., Thiriet J.M., Robert M., "Un transmetteur intelligent de température au service de l'environement", SEE, 3ème colloque Européen, Grenoble, 30-31 mars 1995, pages 35-40.

[PEYR-93] Peyrucat J.F., "Mesures de précision: Les mesures de tension", Mesures, N°657, Septembre 1993, pages 105-109.

[RIVI-93] Rivière, J.M., M. Robert, F. Hermann and C. Aubrun, "Modelling of smart sensors : application to a smart temperature transmittert", Imeko TC4 Symposium Intelligent instrumentation for remote and on site measurement, May 12-13, Brussels, Belgium, pages 173-180, 1993.

[RIVI-95] Riviere J.M, "Contribution à la définition d'un modèle de référence de capteur intelligent : Application à un capteur intelligent de température", Thèse de doctorat de l'université Henri Poincaré Nancy 1, 4 janvier 1995.

[RUMB-93] Rumbaugh J., "Controlling code : How to implement dynamic models", Journal of Object-Oriented Programming, Vol.6, N°2, May 1993.

[RUME-86] Rumelhart D.E., Hinton G.E., Williams R.J., "Learning internal representations by error propagation", Parallel Data Processing, Vol.1, Chapter 8, the M.I.T. Press, Cambridge, MA 1986, pages 318-362.

[SDL-88] Recommendation Z.100, "Specification and Description Language, SDL", Blue Book, Volume X.1, and appendices A, B, C1, C2, D, E, F1, F2, F3, ITU, Switzerland, 1988.

[SDL-93] Recommendation Z.100, "Specification and Description Language, SDL", Blue Book, Volume X.1, and appendices A, B, C,D, F1, F2, F3, ITU, Switzerland, 1993.

[STRO-95] Stroustrup B., "Le langage C++", 2ème édition, Addison-Wesley Publishing Company, inc , 1995.

[YICK-94] Yick P., White N.M., J.E. Brignell, "A reconfigurable ASIC front end, Application specific integrated circuits for measurement systems", colloquium organised by professional group E1 (measurements and instruments), savoy place, 24 february 1994.

Conclusions et perspectives

Les capteurs intelligents constituent sans nulle doute une étape importante dans l'évolution de l'instrumentation industrielle. Leur utilisation dans le milieu industriel est réelle, bien que limitée. Un frein à leur développement est lié à l'inexistence d'un langage commun à tous les constructeurs, autrement dit, bien que des travaux soient en cours, il n'existe aucune norme internationale reconnue pour la conception des capteurs intelligents.

Une normalisation du concept de capteur intelligent peut se résumer à la définition des besoins des différents utilisateurs impliqués dans le cycle de vie du capteur, et en partie rationaliser par le savoir faire concepteur. Cette définition des besoins utilisateurs doit être exprimée de manière formelle, à travers un modèle de référence, servant alors de guide de conception des capteurs intelligents.

Cette étude constitue une brique à la construction de ce modèle de référence de capteur intelligent. A travers une application à un capteur intelligent de température, nous avons tenté d'ébaucher une méthodologie pour la formalisation de ce modèle de référence, basé sur une approche orientée objet. La première étape est l'utilisation des cas d'utilisation pour la formalisation des spécifications où les fonctionnalités du capteur intelligent sont définies du point de vue de l'utilisateur. La méthode OMT est employée pour la représentation des différentes vues : la vue informationnelle qui se concentre sur la description des objets définis par les cas d'utilisation, la vue architecturale ainsi que la vue comportementale. Cette dernière est examinée au travers de l'outil GEODE permettant une vérification de la complétude des échanges entre les objets. L'outil GEODE présente quelques inconvénients quant à la description des ressources matérielles du capteur.

Une des perpectives de l'étude entreprise est l'implémentation du modèle objet en un langage orienté objet sur un système d'exploitation orienté objet temps réel susceptible d'être intégré dans un capteur intelligent. Ce système d'exploitation doit permettre le contrôle par tâches concurrentes ainsi que la communication par messages.

A notre connaissance seul l'outil SoftWorks propose une librairie de classes temps réel pouvant être utilisées lors des phases d'analyse et de conception, permettant ainsi de faire une implémentation des modèles sur un système temps réel orientée objet (SoftKernel) pouvant être implanté dans un capteur intelligent.

SoftWorks est un atelier de développement logiciel sous UNIX, composé principalement d'un compilateur de langage C++, d'un cross-compilateur de langage C optimisé pour des microprocesseurs cibles de type 68000, d'un débogueur symbolique multitâche, d'une interface intégré de programmation sous X11 et de librairies Temps Réel.

SoftKernel s'articule autour des notions de Micro-Noyaux et d'objets accessibles en C++. La structure de SoftKernel est orientée objets. Les différents micro-noyaux sont constitués d'objets : l'objet tâche, l'objet événement, l'objet sémaphore...

L'innovation la plus significative de SoftKernel est que le principe des micro-noyaux est étendu aux programmes applicatifs : chaque objet développé par l'utilisateur est encapsulé sous forme d'un micro-noyau par le compilateur SoftWorks rendant ainsi l'application beaucoup plus souple, modulaire et sécurisée. L'application accède aux objets des micro-noyaux depuis le langage C++ de l'atelier SoftWorks par l'intermédiaire de librairies d'objets en tant qu'instance de classes.

Le développement de SoftWorks et SoftKernel, entièrement réalisé en C++, a permis de sélectionner les classes standards indispensables à tout concepteur de projets Temps Réel et multitâches avec les techniques objets.

Parmi les méthodes d'analyse et de conception objet, OMT de J. Rumbaugh s'avère parfaitement adaptée aux langages tels que C++. Aussi, tout en commercialisant l'AGL SELECT/OMT (éditeur),

Microprocess complète son offre en proposant avec l'atelier, une librairie de classes dédiées au temps réel.

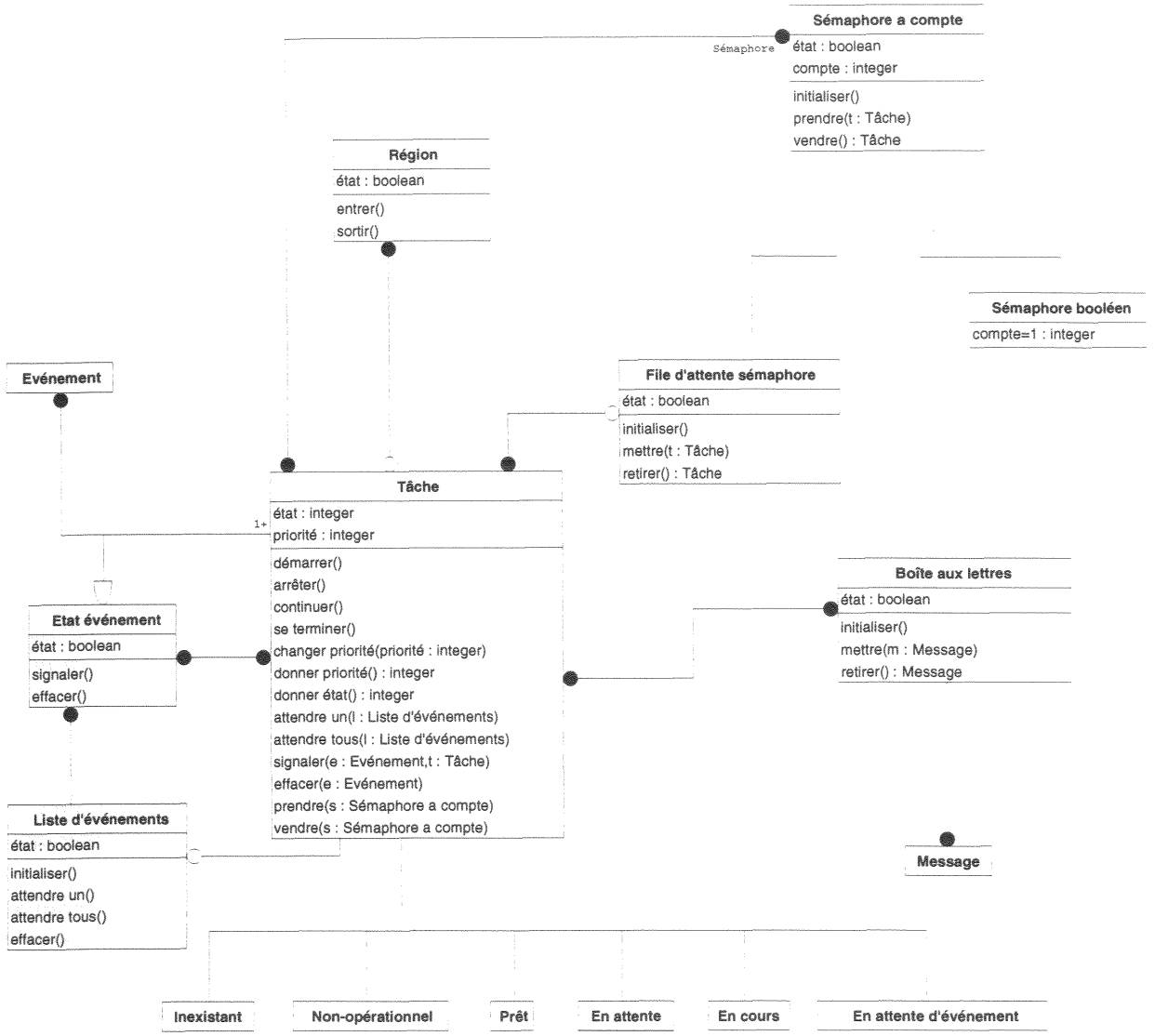


Figure C.1 : Diagramme d'objets montrant les classes dédiées au temps réel [DORS-91] [PERE-90].

Une autre perspective de recherche liée à cette étude consiste en la possibilité d'utiliser un outil capable de réaliser, en plus de la description logicielle, une description des ressources matérielles du capteur intelligent. Au delà de la formalisation du modèle de référence de capteur intelligent, cet outil permet d'envisager la constitution d'un environnement pour la spécification et la conception de capteurs intelligents.

L'outil en cours de réalisation résultant du projet INSYDE (une approche formelle pour la conception des systèmes hybrides) nous semble tout indiqué pour la constitution d'un tel environnement.

L'objectif principal de ce projet INSYDE (INtegrated methods for evolving SYstem DEsign) (ESPRIT III P8641)(durée du projet : deux ans, 1994-1995) est de définir une méthodologie de conception pour les systèmes hybrides, c'est à dire, système intégrant à la fois des composants logiciels et matériels. Cette méthodologie basée sur l'orientation objet, intègre une partie de la méthode OMT (modèle objet pour l'analyse) et combine SDL et VHDL (Very high speed Hardware Description Language) lors de la conception.

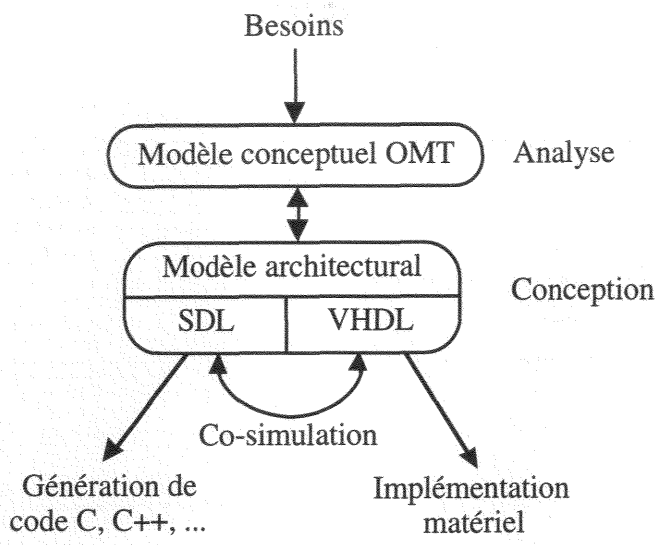


Figure C.2 : La méthodologie INSYDE.

Il m'a été offert de participer à un groupe d'évaluation technologique. Les membres de ce groupe sont tous extérieurs au projet, et évaluerons le premier prototype de l'AGL (Atelier de Génie Logiciel) intégrant la méthodologie INSYDE.

Les différents partenaires sont :

- Verilog SA,
- Alcatel Bell Telephone,
- Dublin City University,
- Humbolt University Berlin,
- Intracom SA,
- Vrije University Brussels.

Références

[DELP-94] Delpiroux J.P. et al., "INSYDE, Toolset specification", 15 septembre 1994.

[DORS-91] Dorseuil Alain, Pillot Pascal, "Le temps réel en milieu industriel", Edition Dunod, 1991, 300 pages.

[PEET-95] Peeters J. et al., "INSYDE, Final demonstration of the telecom application", 15 decembre 1995.

[PERE-90] Jean-Paul Perez, "Systèmes temps réel", Edition Dunod, 1990, 244 pages.

[SINC-96] Sinclair D. et al., "INSYDE, Integrated methods for Evolving System design", Esprit Ref: p8641, User Manuel, 21 february 1996.

Références URL (Uniform Resource Locator) sur le projet INSYDE :

{URL-INS} "http://www.compapp.dcu.ie/~gclynch/insyde.html", The INSYDE ESPRIT III Project.

Glossaire des principaux termes employés en approche orientée objet

Les définitions suivantes sont tirées de [GILL-94], [RUMB-91], [RUMB-93].
Entre parenthèses se trouve la traduction en anglais du vocabulaire qui est utilisé dans la littérature.

[GILL-94] Gillian C., "An approach for using OMT in the development of large systems", Journal of object-oriented programming, vol. 6, N°9, Feb. 94.

[RUMB-91] Rumbaugh J., M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, "Object-Oriented Modeling and Design", Prentice-Hall International Editions, 1991.

[RUMB-93] Rumbaugh J., "Object Modeling Technique", Séminaire technique intensif, 21 Juin 1993 Versailles.

Action (Action) :
Concept utilisé au niveau de l'analyse.
Une opération instantanée. Les actions sont, en général, associées aux événements mais nous pouvons les trouver associées aux états.

Activité (Activity) :
Concept utilisé au niveau de l'analyse.
Une opération consommant du temps. Les activités sont toujours associées aux états.

Agrégation (Aggregation) :
Concept utilisé au niveau de l'analyse.
L'agrégation est une relation "composé-composant" ou "partie de" dans laquelle les objets représentant les composants d'une chose sont associés à un objet représentant l'assemblage entier.

Association (Association) :
Concept utilisé au niveau de l'analyse.
Réalisation entre instances de deux ou plusieurs classes décrivant un groupe de liens ayant une sémantique et une structure communes.

Attribut (Attribute) :
Concept utilisé au niveau de l'analyse.
Une propriété décrivant les objets d'une classe. Un nom d'attribut est unique dans une classe. Un attribut a une valeur pour un objet.

Attribut d'événement (Event attribute) :
Concept utilisé au niveau de l'analyse.
Données portées par un événement d'un objet à un autre.

Attribut de lien (Link attribute) :
Concept utilisé au niveau de l'analyse.
Données portées par un lien dans une association.

Classe (Object class) :

Concept utilisé au niveau de l'analyse.
Description d'un groupe d'objets ayant des propriétés (attributs) identiques, un comportement commun (des opérations), des relations communes (associations) avec d'autres objets et des sémantiques communes. Chaque classe définit un ensemble infini d'objets individuels.

Classification (Classification) :

Concept utilisé au niveau de l'analyse.
Un groupe d'objets ayant même structure statique et même comportement.

Encapsulation (Encapsulation) :

Concept utilisé au niveau de l'analyse, de l'implémentation.
Technique de modélisation et d'implémentation permettant de séparer les aspects externes d'un objet des détails internes de l'objet (détails d'implémentation).

Evénement (Event) :

Concept utilisé au niveau de l'analyse, de la conception (système, objet).
Quelque chose apparaissant instantanément à un point donné dans le temps. Il n'a aucune durée et porte l'information d'un objet à un autre (attribut d'événement).

Etat (State) :

Concept utilisé au niveau de l'analyse.
Les valeurs des attributs et des liens (association) d'un objet à un instant particulier.

Héritage (Inheritance) :

Concept utilisé au niveau de l'analyse, de l'implémentation.
Mécanisme permettant à des classes de partager des attributs et des opérations. Ce mécanisme est basé sur la relation de généralisation.

Héritage multiple (Multiple inheritance) :

Concept utilisé au niveau de l'analyse, de l'implémentation.
Type d'héritage permettant à une classe d'avoir plus d'une super-classe et d'hériter de leurs caractéristiques (statiques, dynamiques).

Héritage simple (Single inheritance) :

Concept utilisé au niveau de l'analyse, de l'implémentation.
Type d'héritage autorisant une classe à n'avoir qu'une super-classe.

Identité (Identity) :

Concept utilisé au niveau de l'analyse.
Les objets sont distinguables par leur existence même et non par leurs propriétés (attributs) descriptives qu'ils possèdent. Deux objets sont distincts même si toutes leurs valeurs de propriétés sont identiques.

Instance (Instance) :

Concept utilisé au niveau de l'analyse.
Un élément (objet) d'une classe. Chaque instance a ses propres valeurs pour chaque attribut mais partage les noms des attributs et des opérations avec d'autres instances

de la classe. Un objet contient une référence implicite à sa propre classe d'appartenance.

Instanciation (Instantiation) :

Concept utilisé au niveau de l'analyse, de l'implémentation.
Processus de création d'instances de classes.

Lien (Link) :

Concept utilisé au niveau de l'analyse.
Instance d'une association. Une connexion conceptuelle ou physique entre objets.

Méthode (Method) :

Concept utilisé au niveau de la conception objet.
Implémentation d'une opération pour une classe.

Module (Module) :

Concept utilisé au niveau de l'analyse.
Un sous-ensemble cohérent d'un système contenant un groupe de classes et leurs relations. Un module, identifié au cours de l'analyse, se transforme en sous-systèmes durant la conception du système.

Multiplicité (Multiplicity) :

Concept utilisé au niveau de l'analyse.
Nombre d'instances d'une classe pouvant être reliées à une instance d'une autre classe. (synonyme : cardinalité).

Objet (Object) :

Concept utilisé au niveau de l'analyse, de la conception (système, objet).
Un concept, une abstraction, une chose ayant une signification pour le problème étudié. Tous les objets ont une identité et sont distinguables.

Opération (Operation) :

Concept utilisé au niveau de l'analyse.
Une fonction ou une transformation pouvant être appliquée à ou par des objets d'une classe. Chaque opération a un objet cible comme argument implicite. Le comportement de l'opération est fonction de la classe cible.

Orienté objet (Object-oriented) :

Orienté-objet = identité+classification+polymorphisme+héritage

Polymorphisme (Polymorphism) :

Concept utilisé au niveau de l'analyse.
Une même opération peut avoir un comportement différent pour des classes différentes.

Processus (Process) :

Concept utilisé au niveau de l'analyse.
Quelque chose qui transforme des données.

Propagation (Propagation) :

Concept utilisé au niveau de l'analyse.
Application automatique d'une opération à des objets sélectionnés dans un réseau d'objets.

Scénario (Scenario) :

Concept utilisé au niveau de l'analyse.
Une séquence d'événements apparaissant durant une exécution particulière d'un système.

Surcharge (Override) :

Concept utilisé au niveau de l'implémentation.
Redéfinition de méthode pour une opération héritée.

Suivi d'événement (Event trace) :

Concept utilisé au niveau de l'analyse.
Diagramme montrant l'émetteur et le récepteur d'événements ainsi que la séquence d'événement.

Transition (Transition) :

Concept utilisé au niveau de l'analyse.
Un changement d'état causé par un événement.

Glossaire des principaux termes employés en instrumentation

INF = Glossaire associé au projet INF
NF = Norme AFNOR
CIAME = définition ou précision apporté par le groupe de travail "fonctionnalités capteurs intelligents" du CIAME
CEI = Norme CEI

Auto-surveillance [INF] :
Mécanisme matériel et/ou logiciel surveillant en permanence la mission d'un dispositif

Auto-test (Self-test) [INF] :
C'est la réalisation de la vérification du bon fonctionnement de différents composants matériel et/ou logiciel, par analyse de leur réponse à des stimuli internes. Ces tests peuvent être automatiques ou à la demande d'un utilisateur.

Calibrage d'un appareil de mesures (Gauging of measuring instrument) [NF X 07-001] :
Positionnement matériel des repères (éventuellement de certains repères principaux seulement) d'un appareil de mesure en fonction des valeurs correspondantes de la grandeur mesurée.

Capteur (Sensor) [NF X 07-001] :
Element d'un appareil de mesure ou d'une chaîne de mesure auquel est directement appliquée une grandeur à mesurer.

Datation [INF] :
Association à une information des attributs de type année, mois, jour, minute, seconde ...

Conduite [CEI comité 65] :
Ensemble des opérations de commande et de contrôle d'un système, éventuellement de celles qui assurent la fiabilité et la protection.

Conduite de processus [CEI comité 65] :
Régulation ou manipulation d'un certain nombre de grandeur influençant le fonctionnement d'un processus en vue d'atteindre des objectifs spécifiés.

Configuration [CEI comité 65] :
Choix des modules matériels ou logiciels, établissement de leurs emplacements, définition de leurs interconnexions et affectation des constantes.

Configuration fonctionnelle [CIAME] :
Ensemble d'actions et leur validation visant à rendre le capteur mesurant et communiquant.

Configuration opérationnelle [CIAME] :
Ensemble d'actions et leur validation visant à dédier le capteur à l'application.

Configuration technologique [CIAME] :
Ensemble d'actions et leurs validations visant à intégrer le capteur dans son environnement physique.

Diagnostic (Diagnosis) [NF X 60-011] :
Identification de la cause probable de la (ou des) défaillances à l'aide d'un raisonnement logique fondé sur l'ensemble d'informations provenant d'une inspection d'un contrôle ou d'un test.

Disponibilité (Disponibility) [NF X 60-500] :
Aptitude d'une entité à être en état d'accomplir une fonction requise dans des conditions données, à un instant donné ou pendant un intervalle de temps donné, en supposant que la fourniture des moyens extérieurs nécessaires soit assurée.

Fiabilité (Reliability) [NF X 60-500] :
Aptitude d'une entité à accomplir une fonction requise, dans des conditions données, pendant un intervalle de temps donné.

Grandeur d'influence (Influence quantity) [NF X 07-001] :
Grandeur qui ne fait pas l'objet du mesurage mais qui influe sur la valeur du mesurande ou sur les indications de l'instrument de mesure.

Interchangeabilité (Interchangeability) [club FIP] :
L'interchangeabilité permet de remplacer un composant de marque X par un composant de marque Y in situ tout en respectant le cahier des charges initial.

Interopérabilité (Interoperability) [club FIP] :
L'interopérabilité est réalisée entre les différents abonnés, si les informations transmises sont interprétables par le destinataire sans obliger l'utilisateur à écrire des programmes de traduction particuliers.

Maintenabilité (Maintenability)[NF X 60-500] :
Aptitude d'une entité à être maintenue ou rétablie, sur un intervalle de temps donné, dans un état dans lequel elle peut accomplir la fonction requise, lorsque la maintenance est accomplie dans des conditions données, avec des moyens prescrits.

Maintenance (Maintenance)[NF X 60-010, NF X 60-500] :
Ensemble des actions destinées à maintenir ou rétablir une entité dans un état dans lequel elle peut accomplir une fonction requise.

Mesurande (Mesurand) [NF X 07-001] :
Grandeur soumise à mesurage. On emploie également le terme grandeur principale pour les capteurs intelligents.

Mesurage (Measurement) [NF X 07-001] :
Ensemble des opérations ayant pour but de déterminer la valeur d'une grandeur.

Mesure auxiliaire :

La mesure auxiliaire correspond à l'image d'un grandeur d'influence et sert au calcul de la mesure fonctionnelle (chapitre I).

Mesure fonctionnelle :

Mesure élaborée à partir de la mesure primaire et des mesures auxiliaires à l'aide d'algorithmes permettant des corrections, des compensations, des linéarisations, ... (chapitre I)

mesure opérationnelle :

Mesure délivrée par le capteur intelligent, directement exploitable (chapitre I)

Mesure primaire :

La mesure primaire correspond à l'image du mesurande (chapitre I)

Mesure qualifiée :

Mesure obtenue à partir de la mesure fonctionnelle et des mesures technologiques (chapitre I)

Mesure technologique :

La mesure technologique correspond à l'image d'une variable ou d'une grandeur d'auto-contrôle. Ces mesures caractérisent l'état de fonctionnement de certains constituants matériels du capteur à l'instant ou s'effectue la mesure (chapitre I).

Protocole (Protocol) [INF] :

Ensemble de conventions permettant de faire communiquer des équipements. Il comporte la spécification précise des unités de données échangées et leur enchaînement.

Rangeabilité (Rangeability) [CEI comité 65] :

Rapport de l'intervalle maximal à l'intervalle minimal pour lequel un instrument ou un appareil peut être étalonné avec la précision assignée.

Réseau local [J.O. 87-0507] :

Ensemble connexe à caractéristique privatif de moyen de communication établie, pourvu de règles de gestion du trafic et permettant des échanges internes d'informations de toute nature.

Sureté de fonctionnement (Dependability) [CEI 1069-5] :

Mesure dans laquelle on peut se fier au système pour qu'il exécute exclusivement et correctement la (les) tache(s) qui lui est (sont) attribuée(s) dans les conditions de fonctionnement et d'environnement définies, à un instant donné ou sur une durée définie.

Système [CEI comité 65] :

Ensemble d'éléments interdépendants constitués en vue d'atteindre un but déterminé au moyen d'un fonctionnement spécifié.

Annexe I

Ateliers de Génie Logiciel supportant la méthode OMT

Tous les AGL présentés supportent la méthode OMT dans sa globalité (c'est à dire intégrent des éditeurs graphiques pour le modèle objet, le modèle dynamique et le modèle fonctionnel). De plus ils permettent tous la génération de code C++.

Les AGL GraphTalk/OMT, ObjectMaker, ObjectTeam/Rumbaugh, Stp/OMT et LOV/OMT ont été évalués pendant un mois afin de déterminer lequel de ces outils correspondait le mieux à nos besoins.

Platformes supportées par la majorité des environnements :

- Station UNIX (X11) : RS6000/AIX 3.2.x, SparcStation/SunOS 4.1.x & Solaris 2.x, DecAlpha/OSF1, HP/HPUX 9.x, SiliconGraphics/IRIX 5.x
- PC Windows et OS/2 : MS-Windows NT & 95 & 3.1x
- Macintosh : MacOS 7.x

LOV/OMT et ObjectGEODE	
Description brève	Outils sélectionnés par les différentes équipes du CRAN. LOV/OMT supporte les trois modèles de la méthode OMT avec contrôle de cohérence entre eux. Il intègre un générateur de code C++ et permet de faire de la rétro-ingénierie à partir d'un code écrit en C++. La société VERILOG organise en outre des sessions de formation sur la méthode (prix moyen d'une journée de formation : 5 kF).
Méthodologie(s)	OMT et LDS (pour ObjectGEODE)
Distributeur	VERILOG 52, Avenue Aristide Briand 92220 BAGNEUX Tél. : 01 46 65 70 70 Fax : 01 46 65 77 38

ISF/AD OMT	
Description brève	(Intelligent Software Factory, Analysis&Design) Il supporte les trois modèles de la méthode OMT avec des extensions issues de ASM et RD afin d'afiner les constructions durant la phase de conception. Il intègre un générateur de code SQL, C++ et Smalltalk.
Méthodologie(s)	OMT, ASM (Analysis Scripting Method), RD (Responsability driven Design)
Distributeur	REICH TECHNOLOGIES 16 bis, rue d'Odessa - 75014 PARIS Tél. : 01 43 20 19 46 Fax : 01 43 20 20 31

Westmount I-CASE OMT	
Description brève	
Méthodologie(s)	OMT
Distributeur	Westmount USA Inc. 1555 Wilson Blvd. Suite 3000 Arlington, VA 22209 U.S.A. Tél : (1)703 875 8799 Fax : (1)703 527 5709

OpenTool/OMT	
Description brève	Méta-atelier développé sous Smalltalk supportant la méthode OMT avec certaines extensions. Ils permet également d'utiliser d'autres méthodes comme Shlaer/Mellor, SYS_P_O
Méthodologie(s)	OMT
Distributeur	TNI (Techniques Nouvelles d'Informatique) Technopôle Brest-Iroise Case Postale 1 29608 BREST CEDEX Tél. : 02 98 05 27 44 Fax : 02 98 49 45 33

DESIGNER/OMT	
Description brève	AGL disponible uniquement sous PC Windows.
Méthodologie(s)	OMT
Distributeur	CERUS 48 Bd Flandrin 75116 PARIS Tél. : 01 45 04 30 67

Stp/OMT et OMTool	
Description brève	Ces AGL ont été développés respectivement par GE (General Electric) et ACC (Advanced Concepts Center de Martin Marietta), et sont distribués par la société IDE. Ils supportent complètement la méthode OMT et intègrent un générateur de documentation, un générateur C++ et/ou ADA (à partir du modèle objet) et un générateur de schéma relationnel (le méta-modèle des outils est celui décrit dans le livre de Rumbaugh)
Méthodologie(s)	OMT
Distributeur	IDE Europe SARL (Interactive Development Environments) 7ter, rue de la porte de Buc 78000 Versailles Tél. : 01 39 02 26 02 Fax : 01 39 02 05 37

ObjectTeam/Rumbaugh	
Description brève	AGL développé à partir du méta-outil Paradigm Plus/Cadre Edition
Méthodologie(s)	OMT
Distributeur	CADRE Technologies 19, Avenue de Norvège 91953 LES ULIS CEDEX Tél. : 01 69 28 12 13 Fax : 01 69 28 76 36

GraphTalk/OMT	
Description brève	AGL offrant la possibilité de générer des instructions DDL (Data Definition Language)/SQL en vue d'intégrer les données persistantes de l'application dans de multiples bases de données (Oracle, Sysbase par exemple). Il permet également la génération de code C++ correspondant aux structures statiques du modèle objet.
Méthodologie	OMT
Distributeur	Parallax Software Technologies 5, rue Bellini 92806 PUTEAUX CEDEX Tél. : 01 46 96 07 07 Fax : 01 46 96 07 10

ObjectMaker	
Description brève	AGL développé par MARK V Systems, Ltd Cet AGL est multi-méthodes, il dispose d'une trentaine de méthodes de base. Il permet trés rapidement et trés simplement de faire des mises à jour régulières.
Méthodologie	OMT, BOOCH, Coad-Yourdon ..;
Distributeur	PROCESS (Electronique Informatique Industrielles) 285, Rue Camille Krantz 88000 DINOZE Tél. : 03 29 33 01 69 Fax : 03 29 35 37 52

Paradigm Plus	
Description brève	AGL développé par ProtoSoft et commercialisé par Scientific COMPUTERS.
Méthodologie(s)	OMT
Distributeur	Scientific COMPUTERS 11 C, Quai Conti 78840 LOUVECIENNES Tél. : 01 30 82 77 07 Fax : 01 30 82 72 78 E-mail : info@scientific.fr

Select/OMT	
Description brève	AGL developpé par Select Software Tools, Ltd, 1526 Brookhollow Dr., Santa Ana, CA 92705, Tél. : (714)957-6633, Fax : (714)957-6219 Il intègre une bibliothèque de classes temps réel développées par Microprocess pouvant être utilisées lors de l'implémentation des modèles.
Méthodologie(s)	OMT
Distributeur	Microprocess 97 bis rue de Colombes BP 87 92405 COURBEVOIE CEDEX Tél. : 01 47 68 80 80 Fax : 01 47 88 97 85

Rational Rose	
Description brève	AGL intégrant les nouveaux concepts de la méthode unifiée formalisé par Rumbaugh, Booch et jacobson.
Méthodologie(s)	OMT et BOOCH
Distributeur	RATIONAL Software Corporation Immeuble de la Gare 1, Place Charles de Gaulle F-78180 Montigny le Bretonneux France Tél. : 01 30 12 09 50 Fax : 01 30 12 09 66

Annexe II

Notation de la méthode OMT

La méthode OMT (Object Modeling Technique) [RUMB-91a, 91b, 93a] [LARO-95] fournit un formalisme de description graphique du modèle objet assez complet, un modèle d'abstraction de l'aspect dynamique du système et une perspective d'intégration du modèle fonctionnel. Le formalisme est unique pour les phases d'analyse et de conception. Les modèles sont utilisables durant toutes les phases du développement et s'enrichissent au fur et à mesure de détails au cours de la progression dans le cycle de développement.

Le modèle objet [RUMB-93e, 93f, 94d, 94e, 94f, 95a, 95d]

Le modèle objet décrit la structure des objets du système :

- l'identité,
- les relations entre les objets,
- les attributs,
- les opérations.

Le modèle objet sert de squelette pour les modèles dynamique et fonctionnel. C'est le plus important des trois modèles. Dans ce modèle, l'objet est l'unité qui sert à diviser le monde. Le but du modèle objet est d'extraire les concepts du monde réel qui sont importants pour l'application. Ce modèle est représenté graphiquement par des diagrammes objet contenant des classes d'objets. Ces classes sont arrangées hiérarchiquement et sont associées entre elles. Les classes définissent les valeurs d'attributs pour chaque instance d'objet et les opérations que chaque objet utilise.

Un diagramme objet dispose d'une notation graphique formel pour modéliser les objets, les classes et les relations entre objet. Ces diagrammes servent à l'abstraction des données.

Le diagramme d'objets

Il existe deux types de diagrammes d'objets :

- les diagrammes de classes,
- et les diagrammes d'instances.

Un diagramme de classes est un schéma, une structure permettant de décrire un grand nombre d'instances possibles de données. Il décrit les classes d'objets.

Un diagramme d'instances décrit comment un ensemble particulier d'objets est lié à d'autres. Ces diagrammes sont souvent utiles pour expliquer des cas de tests et pour présenter des exemples. A un diagramme de classes donné correspond un ensemble infini de diagrammes d'instances.

La figure AII.1 montre la notation de modélisation des objets. La classe "Capteur" possède les attributs "Modèle", "Unité de mesure" ...

Un objet de la classe "Capteur" possède les valeurs "Rosemount 3051CG4A22A1AJ2" pour nom de modèle et "Pascal" pour unité de mesure ...

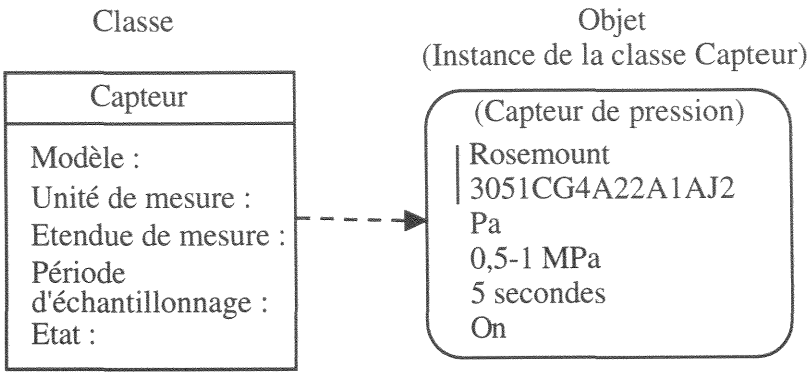


Figure AII.1 : Notions de Classe et d'Objet.

La notation de modélisation par objets pour les classes est une boîte qui possède jusqu'à trois zones. Ces zones contiennent, de haut en bas : le nom de la classe, la liste des attributs et la liste des opérations. Chaque nom d'attribut peut être suivi de détails facultatifs tels que le type et la valeur par défaut. Chaque nom d'opération peut être suivi de détails facultatifs tels que la liste des arguments et le type du résultat. les attributs et les opérations peuvent ou non être affichés, selon le niveau de détail souhaité.

Relations entre les classes

Des liens et des associations sont utilisés pour établir des relations entre les objets et les classes. Ces liens correspondent à une connexion physique ou conceptuel entre les instances d'objet (un lien est une instance d'une association). Une association décrit un groupe de liens qui ont une structure commune (tous les liens d'une même association connectent tous les objets d'une même classe). La multiplicité spécifie combien d'instances d'une classe sont en relation avec d'autres instances d'une autre classe. La multiplicité restreint le nombre d'objets reliés ensemble. Elle est souvent décrite comme étant "un" ou "plusieurs" mais, plus généralement, c'est un sous-ensemble d'entiers positifs.

Pour modéliser les associations, on trouve des constructions additionnelles comme l'agrégation, la qualification, des attributs de liens et des "rôles". L'agrégation correspond à une relation d'appartenance où des objets représentent les composants et sont associés à un objet représentant le composé [BLAH-93]. Un rôle est une extrémité de l'association. Le nom de rôle est un nom qui identifie de façon unique une extrémité de l'association. Une association qualifiée met en relation deux classes d'objets et un qualificatif. Le qualificatif est un attribut spécial qui réduit la multiplicité effective d'un association.

Toutes ces notations sont illustrées à la fin de cet annexe.

Généralisation et Héritage [RUMB-93c]

La généralisation permet d'organiser les classes sous forme hiérarchique basée sur leurs similarités et leurs différences. Le terme généralisation se réfère à la relation entre classes. Le terme héritage se réfère au mécanisme d'attributs et d'opérations utilisant la structure générale. L'héritage peut être simple ou multiple. L'héritage multiple permet à une classe de dépendre de plus d'une superclasse et d'hériter de ses parents. Ceci permet de mixer les informations provenant de deux ou plusieurs sources.

La figure AII.2 montre une généralisation de capteur. Chaque généralisation doit couvrir une propriété unique. Si une classe peut être affinée sur plusieurs dimensions distinctes et indépendantes, alors il faut utiliser des généralisations multiples (classe "Capteur de niveau différentiel").

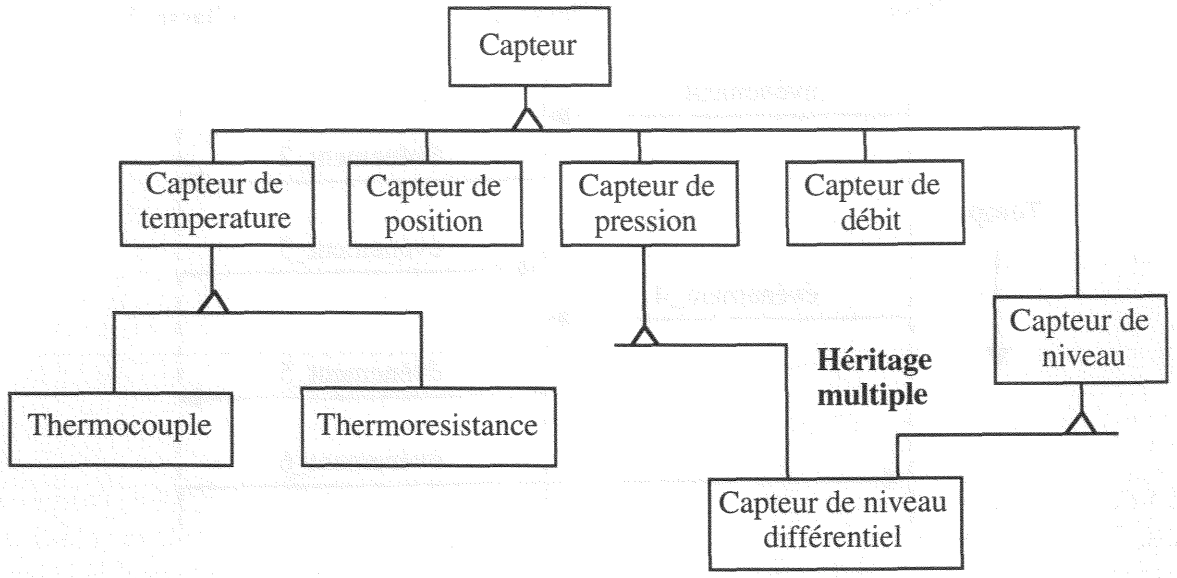


Figure AII.2 : Exemple d'héritage pour les capteurs.

Le modèle dynamique [RUMB-88, 93b, 95b]

Le modèle dynamique décrit les aspects du système du point de vue temporel, des séquences d'opérations, des événements qui opèrent des changements dans le système et des états qui sont définis par le contexte des événements. Le modèle dynamique est représenté graphiquement par des diagrammes d'état utilisés pour spécifier et implémenter le contrôle du système. Chaque diagramme d'état montre l'état et les séquences d'événements permis dans le système pour une classe d'objet. Ces diagrammes se réfèrent aux autres modèles. En effet, les actions dans les diagrammes d'état correspondent aux fonctions du modèle fonctionnel et les événements deviennent les opérations sur les objets dans le modèle objet.

Les valeurs d'attributs et les liens d'un objet sont appelés son état.

Dans le temps, les objets se stimulent entre eux, il en résulte une série de changements de leurs états. Un stimulus d'un objet à un autre est appelé un événement. Un scénario est une séquence d'événements qui apparaissent pendant une exécution partielle du système. Ces séquences d'événements sont représentées par des diagrammes de flots d'événements (figure AII.3). Une condition est une fonction booléenne qui contrôle une transition et est valide sur un intervalle de temps donné.

Les états et les événements sont organisés hiérarchiquement et utilisent le mécanisme d'héritage. En effet une sous classe hérite du diagramme d'état de son ancêtre.

Le diagramme de flots d'événements

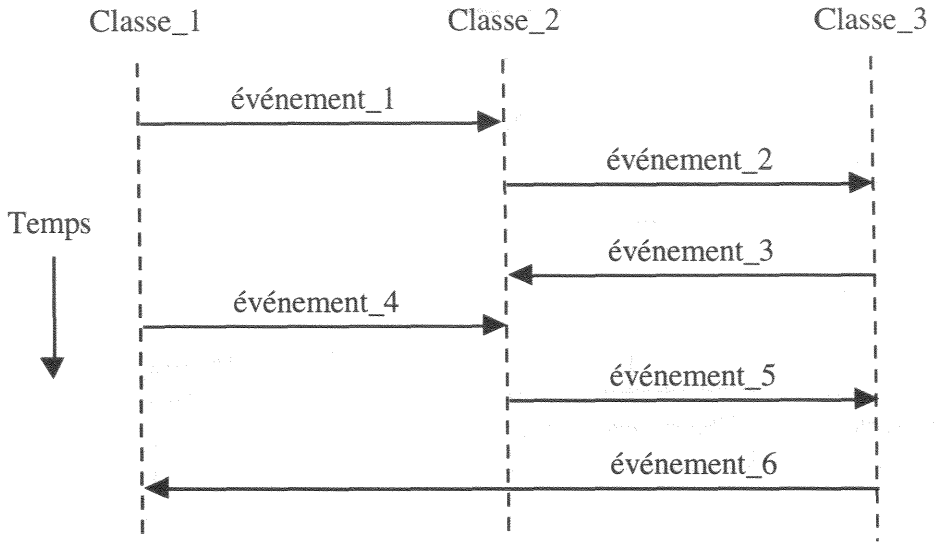


Figure AII.3 : Exemple de diagramme de flots d'événements.

Le diagramme d'états

Les objets répondent aux événements en exécutant des opérations. Une action est une opération instantanée associée à un événement. Une activité est une opération qui prend du temps pour s'exécuter et est associée à un état.

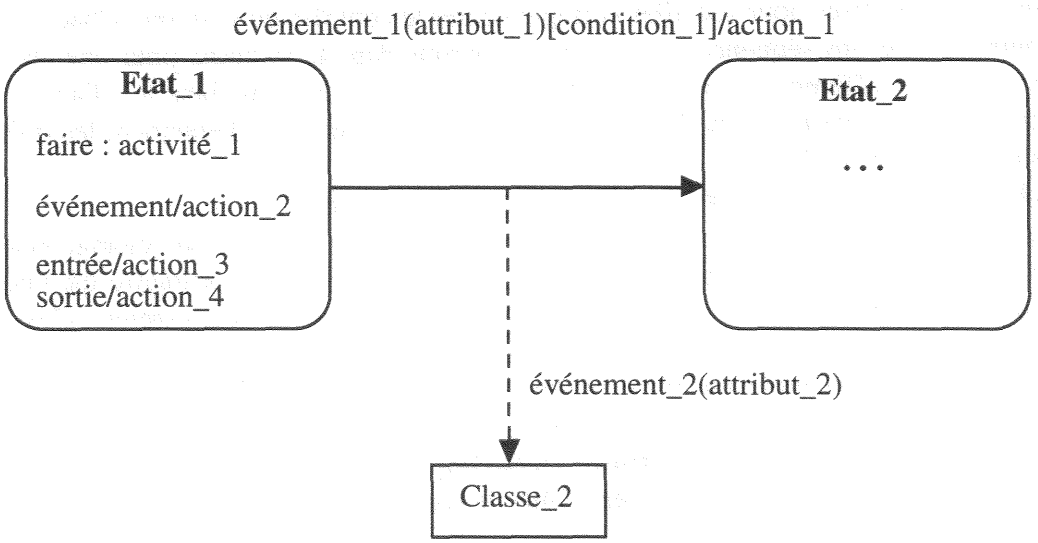


Figure AII.4 : Notation du diagramme d'états.

La méthode permet également de synchroniser les activités, comme illustré ci dessous.

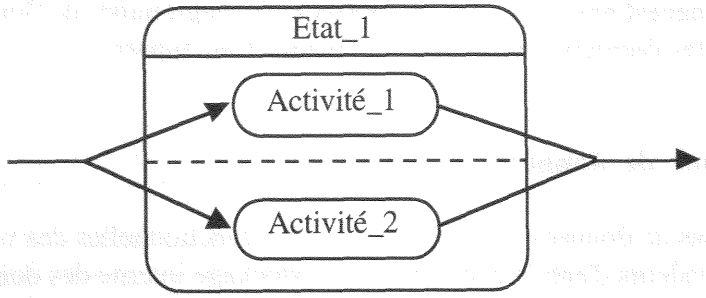


Figure AII.5 : Synchronisation d'activités.

Le modèle fonctionnel [RUMB-94b, 95c]

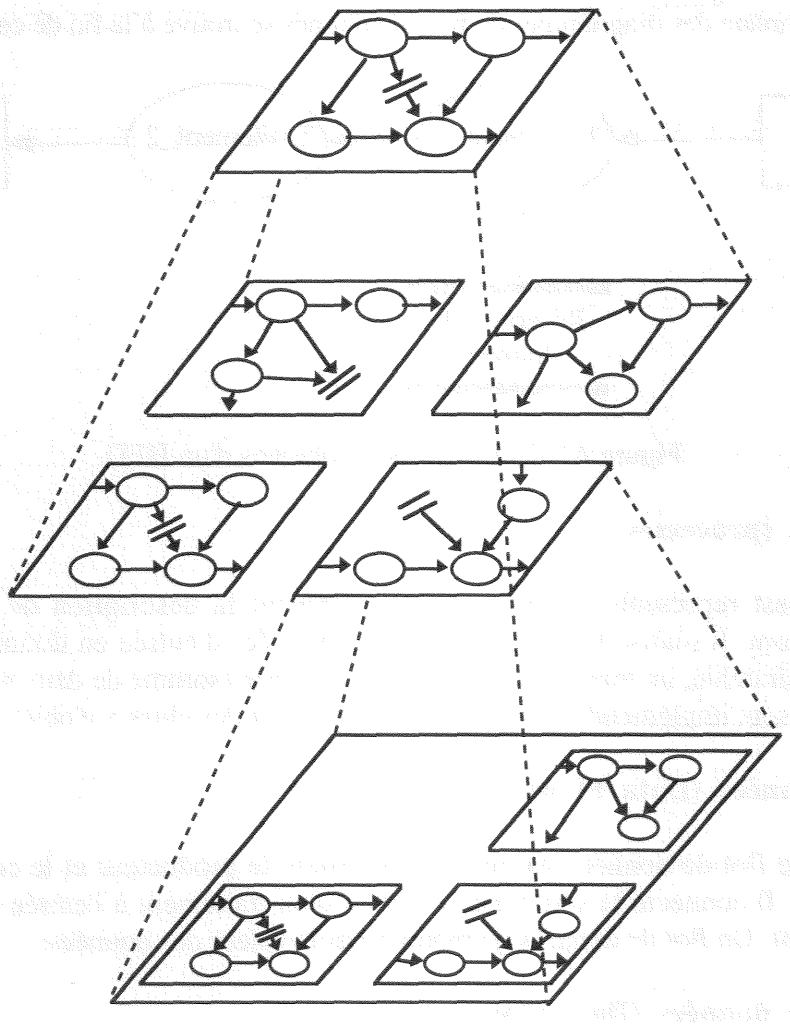


Figure AII.6 : Hiérarchie du modèle fonctionnel.

La dimension fonctionnelle du système est représentée par le modèle fonctionnel qui offre une vision globale des fonctions du système (les transformations). Le modèle fonctionnel montre comment les données de sortie dans un processus sont dérivées des données d'entrée sans s'intéresser à l'ordre dans lequel les fonctions seront exécutées. Le résultat de l'exécution d'un processus dépend du comportement du système spécifié par le modèle dynamique.

Le modèle fonctionnel est représenté par un ensemble de diagrammes de flots de données ordonnés hiérarchiquement, où les derniers niveaux de "raffinage" font apparaître des fonctions élémentaires (figure AII.6).

Le diagramme à flots de données

Un diagramme à flots de données explicite les relations fonctionnelles des valeurs calculées par un système, ainsi que les valeurs d'entrée et de sortie et le stockage interne des données.

Un diagramme à flots de données est un graphe qui contient des traitements qui transforment les données, des flots de données qui transportent les données, des objets acteurs qui produisent et consomment des données et des objets réservoirs de données qui stockent passivement les données.

Le diagramme à flots de données (DFD : Data Flow Diagram) se construit à partir de quatre éléments graphiques (figure AII.7).

La notation complète des diagrammes de flots de donnés se trouve à la fin de cet annexe.

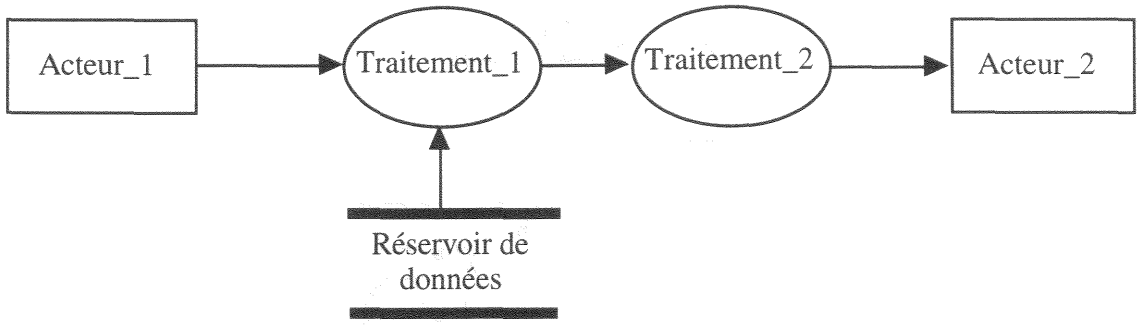


Figure AII.7 : Eléments graphiques d'un DFD.

Les traitements (processes)

Un traitement est représenté par une ellipse contenant la description de la transformation, usuellement son nom. Il réalise la transformation de données d'entrée en données de sortie. A un niveau bas de la hiérarchie, un traitement est une pure fonction (somme de deux nombres, ...).

Les traitements sont implémentés comme des méthodes sur des classes d'objets.

Les flots de données (Data Flows)

On représente le flot de données par une flèche reliant le producteur et le consommateur de la valeur de données. Il connecte la sortie d'un objet ou d'un traitement à l'entrée d'un autre objet ou d'un autre traitement. Un flot de données ne modifie pas la valeur des données.

Les réservoir de données (Data Store)

Un réservoir de données (exemples : base de données, fichiers, ...) est symbolisé par deux traits parallèles contenant le nom du réservoir. Cet élément à été introduit pour immobiliser sous certaines conditions l'information qui circule, afin de pouvoir y accéder plusieurs fois, et ceci, dans un ordre quelconque.

Les acteurs (Actor)

L'acteur est symbolisé par un rectangle pour montrer qu'il s'agit d'un objet. Il indique la provenance ou la destination des données traitées par le système.

Les flots de contrôle (Control Flows)

Un diagramme de flots de données montre toutes les transformations possibles pour les données mais ne décrit pas l'ordre d'exécution. La décision et le séquencement sont issus du contrôle qui fait parti du modèle dynamique. Il est souvent utile d'inclure des flots de contrôle dans le modèle fonctionnel pour montrer la dépendance de certaines données. Un flot de contrôle est symbolisé par une flèche en pointillé reliant le traitement qui produit une valeur de contrôle au traitement qui doit être contrôlé.

Les activités de la démarche

La méthode OMT peut être mise sous forme d'étapes. Ceci implique que l'ordre est important.

Les développeurs expérimentés pourront combiner plusieurs étapes ou mener plusieurs étapes en parallèle sur des portions de projet.

L'itération des étapes est nécessaire à des niveaux de plus en plus bas d'abstraction, pour ajouter des détails au modèle.

Une fois l'analyse d'ensemble terminée au plus haut niveau d'abstraction, les sous-systèmes au sein d'un projet plus vaste peuvent être conçus indépendamment et parallèlement dans les niveaux inférieurs d'abstraction.

Le modèle d'analyse doit inclure les informations significatives dans la perspective du monde réel et présenter la vue externe du système. Il doit être compréhensible par le client du système et fournir une base utile pour définir et choisir les vraies spécifications du système. Les vaies spécifications sont celles qui sont vraiment indispensables, cohérentes entre elles et dont la réalisation est possible.

En revanche, le modèle de conception est lié par la dépendance vis-à-vis d'une implémentation informatique. Le modèle de conception doit s'occuper de détails de plus bas niveau que ceux traités dans le modèle d'analyse. Les deux modèles se combinent pour fournir une documentation valide du système depuis deux points de vue différents mais complémentaires.

1. L'Analyse [RUMB-92a, 92b, 93d] [BLAH-94]

Le but de l'analyse est de développer un modèle de ce que le système doit faire.

Ecrire ou obtenir une description initiale du problème.

Construire un modèle objet.

Identifier les classes d'objets, leurs attributs et leurs associations.
Construire un dictionnaire de données (descriptions des classes, attributs et associations).
Organiser et simplifier les classes en utilisant le mécanisme d'héritage.
Tester le modèle en utilisant des scénarios.
Itérer les étapes précédentes.
Grouper les classes en module (sous-ensembles cohérents).

Modèle Objet = diagramme du modèle objet + dictionnaire des données

Développer un modèle dynamique.

Préparer des scénarios de dialogue entre un utilisateur et le système (séquences d'événements entre des acteurs et des objets du système).
Identifier les événements entre objets et préparer un diagramme de suivi d'événements pour chaque scénario.

Préparer un diagramme des flux d'événements pour le système.
Développer un diagramme d'états pour chaque classe ayant un comportement dynamique important.
Vérifier la cohérence et la complétude des événements des différents diagrammes d'états au niveau système.

Modèle dynamique = diagramme d'états + diagramme global des flots d'événements

Construire un modèle fonctionnel.

Construire un arbre fonctionnel du système.
Construire un "Data Flow Diagram" du système.
Identifier les entrées-sorties du système.
Décrire ce que réalise chaque fonction.
Vérifier la couverture de l'arbre fonctionnel.
Identifier les contraintes.
Spécifier les critères d'optimisation (minimiser le nombre de messages, ...).

Modèle fonctionnel = diagramme à flots de données + contraintes

Vérifier, itérer et affiner les trois modèles.

Ajouter les opérations principales (issues du modèle fonctionnel) dans le modèle objet (encapsulation).

document d'analyse = description du problème + modèle objet + modèle dynamique + modèle fonctionnel

2. Conception système [RUMB-94a]

Organiser le système en sous-systèmes.
Identifier les problèmes de concurrence.
Allouer les sous-systèmes à des processus et des tâches.
Choisir les stratégies pour implémenter les zones de stockage en terme de structure de données, de fichier, de base de données.
Identifier les ressources globales et déterminer les mécanismes de contrôler des accès.
Choisir une approche d'implémentation pour le contrôle du logiciel.
Établir les compromis de priorités.

Document conception système = structure de l'architecture + choix stratégiques

3. Conception objet

Obtenir des opérations pour le modèle objet à partir du modèle fonctionnel (pour chaque traitement, trouver une opération) et du modèle dynamique (pour chaque événement, définir une opération) fonction des choix d'implémentation.
Concevoir des algorithmes pour implémenter les opérations.
Optimiser les chemins d'accès aux données.
Organiser les classes pour accroître l'héritage.
Concevoir l'implémentation des associations.
Déterminer la représentation des attributs.
Mettre classes et associations dans des modules physiques.

Document conception Objet = modèle objet détaillé + modèle dynamique détaillé + modèle fonctionnel détaillé

4. Implémentation [RUMB-93b, 94c] [ACKR-95] [COOK-94]

Génération dans les langages et système de base de données choisis.

Remarque :

Les règles de passage d'une conception objet à une implémentation sont plus des règles de principes que des règles formelles.

Évolution de la méthode OMT : vers une méthode Orientée Objet unifiée

La méthode OMT n'a cessé d'évoluer depuis la parution initiale de l'ouvrage de référence en 1991. Ces évolutions ont diverses sources, certaines sont dues à des utilisateurs, d'autres ont pour origine des concepts développés dans d'autres méthodes et qui se sont avérés utiles, d'autres enfin proviennent des auteurs eux-mêmes qui ont adopté de nouveaux points de vue. Il n'est pas possible de présenter tous les concepts de la méthode, aussi avons-nous dû omettre de nombreux détails et explications. Ce résumé permettra aux utilisateurs avisés d'OMT de commencer à utiliser tout de suite la méthode OMT enrichie de ces nouvelles extensions.

Notation du modèle objet

La notation et les concepts utilisés pour le modèle objet d'OMT ont recueilli un grand succès. D'autres méthodes, telles que Booch et FUSION, ont adopté la plupart des concepts et une grande partie de la notation. Les changements apportés au modèle objet sont assez mineurs ; il s'agit soit de rectifications pour clarifier le modèle ou le rendre plus cohérent, soit d'ajouts de cas particulier nécessaires dans certains contextes.

Un nombre peut être associés au symbole rectangulaire d'une classe dans le coin supérieur droit pour indiquer sa multiplicité, c'est-à-dire le nombres d'objets de la classe qui peuvent coexister. Ce nombre peut être un entier ou un intervalle ; une valeur illimitée pour la limite supérieure est indiquée par une astérisque (*). La valeur par défaut est illimitée tout comme auparavant. Le cas particulier le plus important est le singleton, c'est-à-dire la classe qui ne s'instancie qu'en un seul objet.

Les noms d'associations peuvent contenir des flèches afin d'indiquer le sens dans lequel le nom doit être lu (généralement les noms se lisent mieux dans un sens, très peu étant symétriques).

Notation du modèle dynamique

La méthode originale ne possédait pas de notation particulière pour dénoter un message de création d'un objet. La même notation représentant l'envoi d'un message peut être utilisée pour indiquer la création d'un objet par un autre objet. Le créateur envoie un message de création à la classe cible (en général l'événement est "créer"). L'événement de création doit apparaître sur le diagramme d'états de la classe cible en tant qu'événement initial. De plus, il est possible pour une classe cible de définir plusieurs événements de création ; l'état initial du diagramme d'états aura alors plusieurs transitions (lorsque plusieurs libellés d'événements initiaux sont les mêmes, le choix de la transition se fait par exemple en fonction du nombre ou du type d'arguments transmis avec l'événement de création).

Le modèle dynamique peut être libellé pour indiquer les informations temporelles d'analyse temps-réel. Les contraintes temporelles peuvent être indiquées entre des parenthèses. Ce type d'information est d'ordinaire le plus utile sur un diagramme de suivi d'événements pour un scénario, bien qu'il puisse aussi être montré sur un diagramme d'états.

Notation du modèle fonctionnel

Aujourd'hui, l'accent est mis sur les description textuelles des opérations. Comme c'est souvent le cas ; il n'est pas nécessaire de dessiner un diagramme lorsque du texte ou de simples équations suffisent. Il est préconisé de ne pas faire un diagramme à flots de données de l'ensemble du système avec toutes les données d'entrée et de sortie comme cela se fait dans la méthode Structured Analysis. Un diagramme à flots de données dans l'approche objet doit décrire les effets d'une seule opération en termes de valeurs d'entrée et de sortie pouvant êtres liées à des objets en tant qu'attributs.

Diagramme d'interactions d'objet (DIO)

Un diagramme d'interactions d'objets est un diagramme qui représente le flot des messages échangés entre les objets. Ce type de diagramme n'est guère utile durant la phase d'analyse, mais le devient particulièrement lors de la phase de conception, pour montrer le flot de contrôle et de données entre les objets et les méthodes. Plusieurs auteurs utilisent différentes variations de ces diagrammes d'interactions d'objet ; la notation est ici issue de Booch et FUSION [BOOC-93] [COLE-94] et indirectement des diagrammes de suivi d'événement d'OMT. Les diagrammes d'interaction entre objets sont particulièrement utiles pour montrer le flot de contrôle lors de la conception, mais peuvent aussi être utilisés pour montrer les flots de contrôle du monde réel ainsi que les flots d'information dans un modèle d'entreprise.

La méthode unifiée est une méthode de troisième génération d'analyse et de conception orientée objet développée par Gady Booch, Jim Rumbaugh et I. Jacobson à partir de l'unification des méthodes Booch et OMT.

Les métamodèles décrivent les constituants d'un modèle ainsi que leurs relations. Un métamodèle est lui même un modèle qui peut être décrit en utilisant ces propres notations. Les métamodèles décrivent la sémantique des concepts de la méthode unifiée et pose ainsi une approche standard pour l'analyse et la conception orientée objet.

Références

[ACKR-95] Ackroyd M., "Object-Oriented design of a finite state machine", Journal of Object-Oriented Programming, Vol.8, N°3, June 1995.

[BLAH-93] Blaha M., "Aggregation of parts of parts of parts", Journal of Object-Oriented Programming, Vol.6, N°5, September 1993.

[BLAH-94] Blaha M., "Finding object in object diagrams", Journal of Object-Oriented Programming, Vol.6, N°9, February 1994.

[BOOC-93] Booch G., "Object-Oriented Analysis and Design with Applications", (Second Edition), Benjamin and Cummings, Redwood City, CA, 1993.

[COLE-94] Coleman D. et al., "Object oriented development : the fusion method", Prentice-hall international editions, 1994, 314 pages.

[COOK-94] Cook S., Daniels J., "Object communication", Journal of Object-Oriented Programming, Vol.7, N°5, September 1994.

[LARO-95] Laroque P., Perrin P., "Présentation de la métode Object Modeling Technique (OMT)", AFCET, Groupe de travail COOSI, 1995.

[RUMB-88] Rumbaugh J., "State trees as structured finite state machines for user interface", ACM SIGGRAPH Symposium on User Interface Software, Banff, Alberta, October 17-19, 1988.

[RUMB-91a] Rumbaugh J. et al., "Object-Oriented modeling and design", Prenice-Hall International Editions, 1991, pages 500, ISBN 0-13-630054-5.

[RUMB-91b] Rumbaugh J. et al., "Solutions Manuel: Object oriented modeling and design", Prentice hall, 1991, ISBN 0-13-629858-3, pages 265.

[RUMB-92a] Rumbaugh J., "An object or not an object ?", Journal of Object-Oriented Programming, Vol.5, N°3, June 1992.

[RUMB-92b] Rumbaugh J., "Let there be objects: a short guide to reification", Journal of Object-Oriented Programming, Vol.5, N°7, November-December 1992.

[RUMB-93a] Rumbaugh J., "Object Modeling Technique", Séminaire technique intensif, 21 Juin 1993 Versailles.

[RUMB-93b] Rumbaugh J., "Controlling code : How to implement dynamic models", Journal of Object-Oriented Programming, Vol.6, N°2, May 1993.

[RUMB-93c] Rumbaugh J., "Disinherited: Examples of misuse of inheritance", Journal of Object-Oriented Programming, Vol.5, N°9, February 1993.

[RUMB-93d] Rumbaugh J., "Object in the twilight zone", Journal of Object-Oriented Programming, Vol.6, N°3, June 1993.

[RUMB-93e] Rumbaugh J., "On the horns of the modeling dilemma : choosing among alternate modeling constructs", Journal of Object-Oriented Programming, Vol.6, N°7, November-December 1993.

[RUMB-93f] Rumbaugh J., "What's in a name ? A qualified answer", Journal of Object-Oriented Programming, Vol.6, N°2, May 1993.

[RUMB-94a] Rumbaugh J., "Building boxes : Subsystems", Journal of Object-Oriented Programming, Vol.7, N°6, October 1994.

[RUMB-94b] Rumbaugh J., "Going with the flow: Flow graphs in their various manifestations", Journal of Object-Oriented Programming, Vol.7, N°3, June 1994.

[RUMB-94c] Rumbaugh J., "Modeling models and viewing views : A look at the model-view-controller framework", Journal of Object-Oriented Programming, Vol.7, N°2, May 1994.

[RUMB-94d] Rumbaugh J., "The life of an object model : How the object model changes during development", Journal of Object-Oriented Programming, Vol.6, N°10, March-April 1994.

[RUMB-94e] Rumbaugh J., "Trouble with twins : Warning signs of mixed-up classes", Journal of Object-Oriented Programming, Vol.7, N°4, July-August 1994.

[RUMB-94f] Rumbaugh J., "Virtual worlds : Modeling at different levels of abstraction", Journal of Object-Oriented Programming, Vol.6, N°8, January 1994.

[RUMB-95a] Rumbaugh J., "Driving to a solution : Reification and the art of system design", Journal of Object-Oriented Programming, Vol.8, N°4, July-August 1995.

[RUMB-95b] Rumbaugh J., "OMT : The dynamic model", Journal of Object-Oriented Programming, Vol.7, N°9, February 1995.

[RUMB-95c] Rumbaugh J., "OMT : The fonctionnal model", Journal of Object-Oriented Programming, Vol.8, N°1, March-April 1995.

[RUMB-95d] Rumbaugh J., "OMT : The object model", Journal of Object-Oriented Programming, Vol.7, N°8, January 1995.

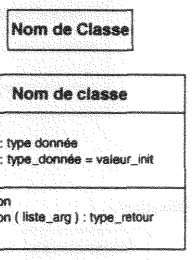
Notation du modèle objet : concepts de base

MEMENTO OMT

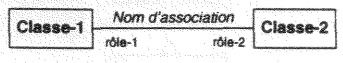
Notation de la méthode OMT décrite dans
OMT - Modélisation et conception orientées objet (Masson, 1994).
Ce mémento tient compte des dernières évolutions de la notation.

Notation du modèle objet : concepts de base

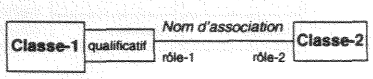
Classe :



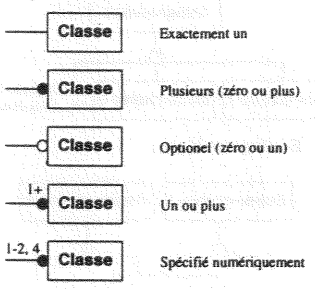
Association :



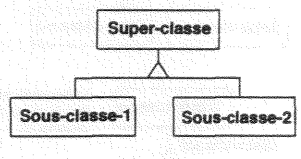
Association qualifiée :



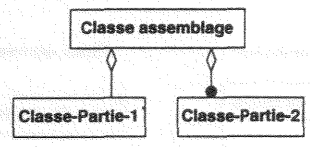
Multiplicité des associations :



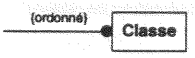
Généralisation (Héritage) :



Agrégation :

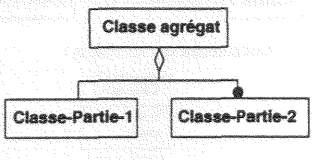


Ordre :

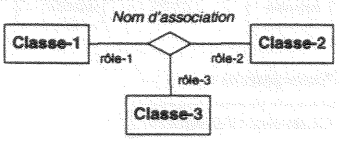


Attribut de lien :

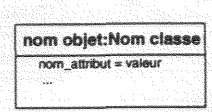
Agrégation (forme alternative) :



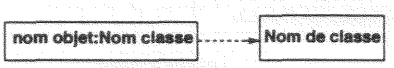
Association ternaire :



Instances de classe :

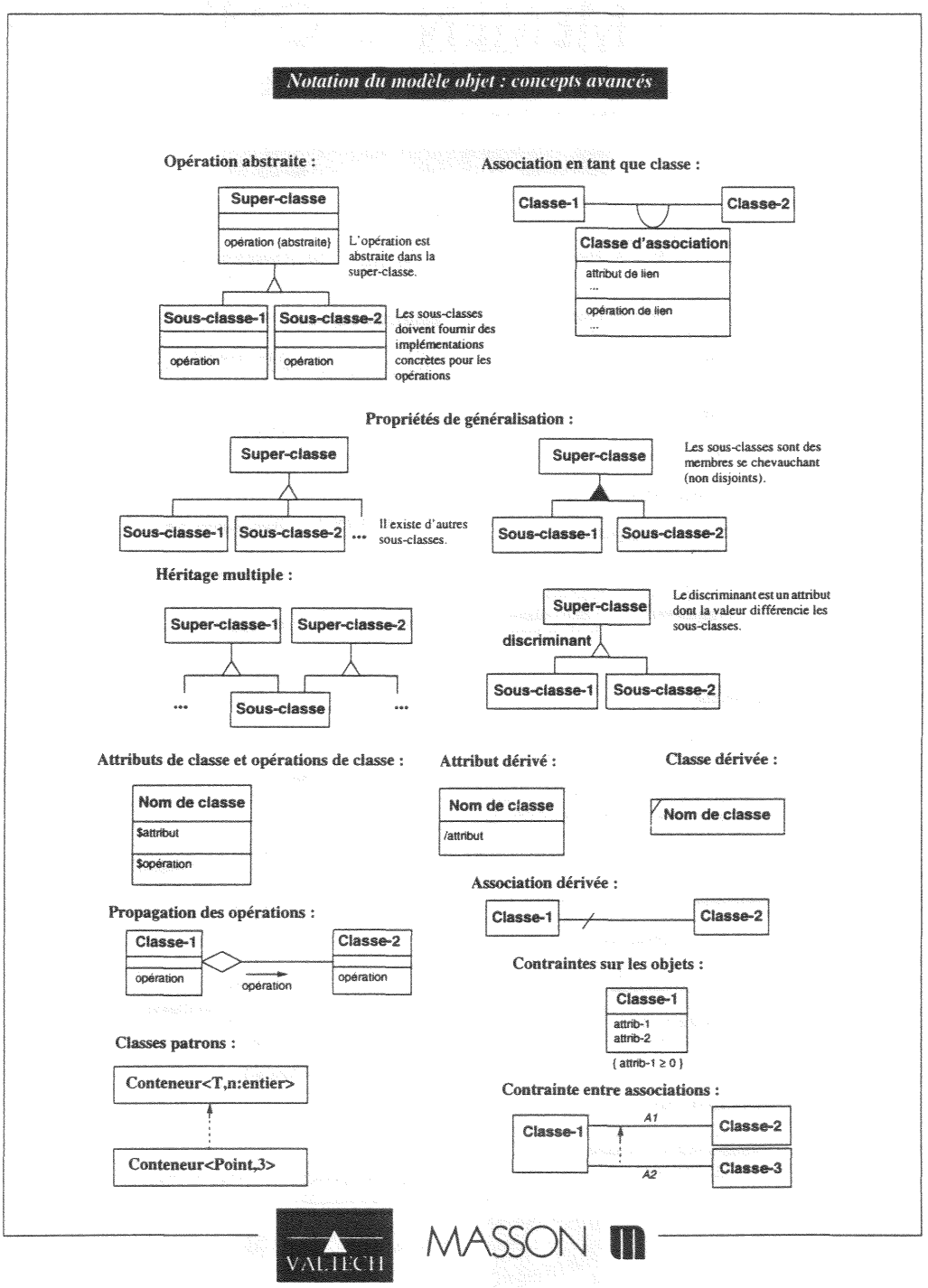


Relation d'instantiation:

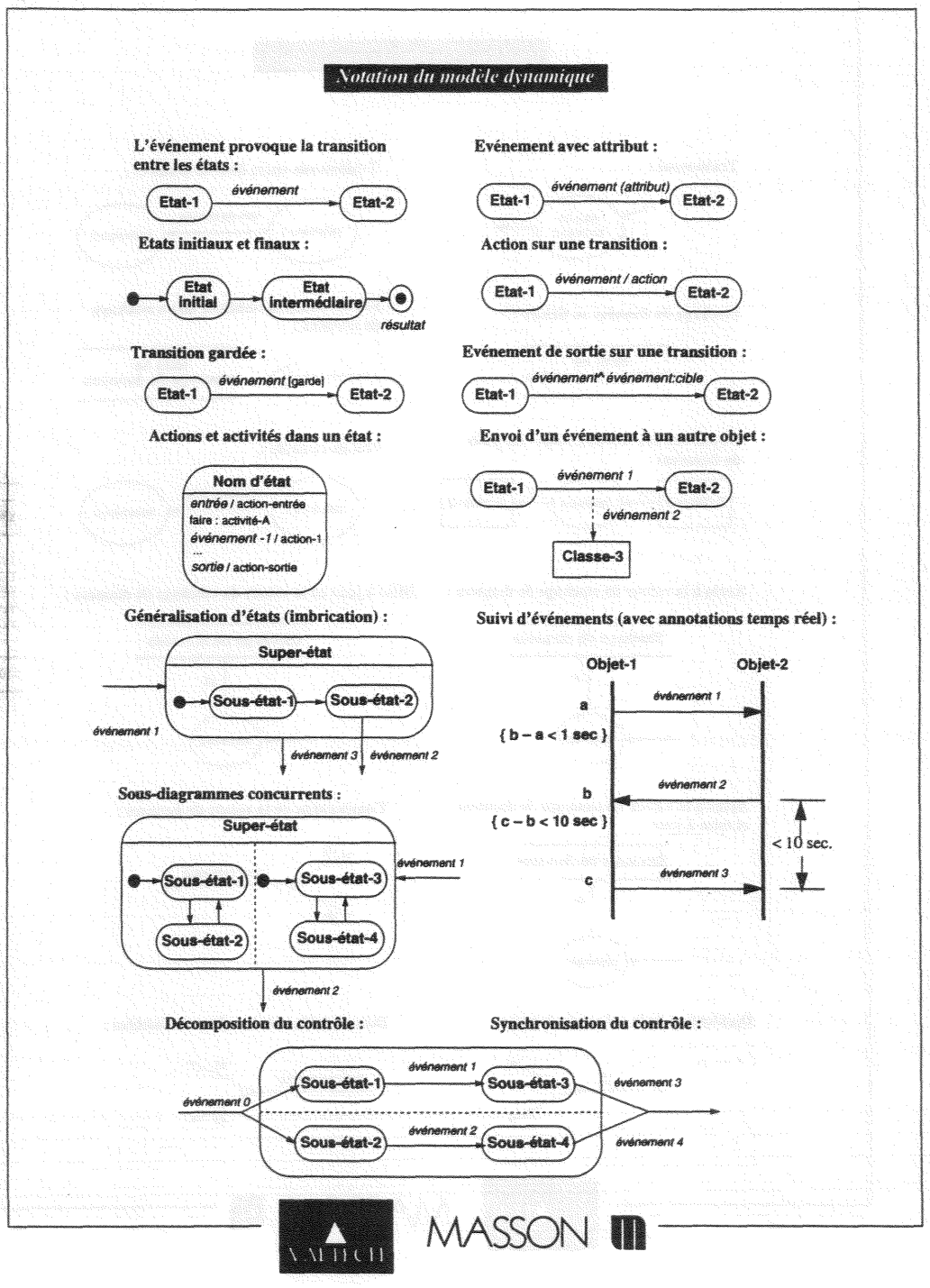


VALTECH MASSON

Notation du modèle objet : concepts avancés



Notation du modèle dynamique

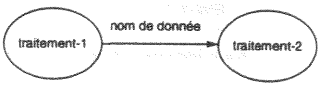


Notation du modèle fonctionnel

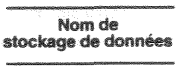
Notation du modèle fonctionnel

Traitement :

Traitements entre flots de données :



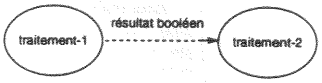
Stockage de données ou fichier :



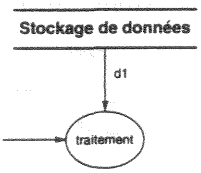
Flot de données résultant en un stockage de données :

Objets acteurs (comme source ou puits de données) :

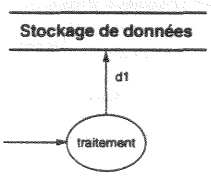
Flot de contrôle :



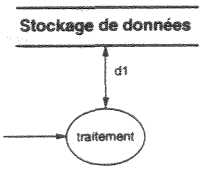
Accès à la valeur du stockage de données :



Mise à jour de la valeur du stockage de données :

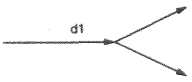


Accès à la valeur du stockage de données et mise à jour :

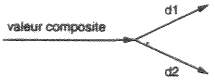


Composition de la valeur de données :

Duplication de la valeur de données :



Décomposition de la valeur de données :



VALTECH MASSON

Annexe III

Les modes d'utilisation et les services

Le tableau AIII.1 montre les droits d'accès des utilisateurs aux modes d'utilisation. On remarque que certains modes d'utilisation sont accessibles par plusieurs utilisateurs. Chacun de ces utilisateurs a accès pour le même mode d'utilisation à un ensemble de services qui peut être différent (voir tableau AIII.2).

Mode d'utilisation / Utilisateurs	Configuration			Opérationnel	Maintenance
	Configuration constructeur	Étalonnage	Configuration responsable		
Constructeur (sur le site d'exploitation)	X				
Responsable sur le site d'exploitation		X	X	X	
Opérateur de conduite				X	
Opérateur de maintenance				X	X
Opérateur de gestion technique				X	
Équipement du SA				X	

X : le mode d'utilisation est accessible par l'utilisateur

Tableau AIII.1 : Droit d'accès des utilisateurs aux modes d'utilisation lorsque le capteur est sur le site d'exploitation.

Le constructeur est un utilisateur particulier puisqu'il accède à tous les modes d'utilisation ainsi qu'à tous les services utilisateurs lorsque le capteur intelligent est hors du site d'exploitation, afin de pouvoir vérifier, entre autres, le bon fonctionnement de celui-ci (voir chapitre II). En revanche, lorsque le capteur intelligent est en exploitation sur le site, le constructeur n'a qu'un accès très limité (seulement le mode d'utilisation "configuration constructeur").

Le tableau AIII.2 définit quels sont les services accessibles aux utilisateurs en fonction du mode d'utilisation.

Les utilisateurs lorsque le capteur est sur le site d'exploitation sont :

U1 : Constructeur sur site(service après vente sur site)
U2 : Responsable du capteur sur le site d'exploitation (installateur, configurateur, étalonneur)
U3 : Opérateur de conduite
U4 : Opérateur de maintenance
U5 : Opérateur de gestion technique
U6 : Équipement du Système d'Automatisation (Intelligent ou pas)

Modes d'utilisation / Services	configuration constructeur	étalonnage	configuration responsable	opérationnel		maintenance
				configuration mesure opérationnelle	mesure	
configurer la mesure opérationnelle pour l'équipement consommateur				U2 U3 U4 U6		
lire la configuration de la mesure opérationnelle relative à un équipement consommateur				U2 U3 U4 U6	U2 U3 U4 U6	
demande de passage en mode mesure				U2 U3 U4 U6		
demande de passage en mode configuration mesure opérationnelle					U2 U3 U4 U6	
demande de suspension de la mesure opérationnelle relative à un équipement					U2 U3 U4 U6	
demande de reprise de la mesure opérationnelle relative à un équipement					U2 U3 U4 U6	
demande de passage en mode configuration constructeur			U1			
demande de passage en mode étalonnage			U2	U2		U2
demande de passage en mode configuration responsable	U2	U2		U2		U2
demande de passage en mode opérationnel		U2	U2			U4
demande de passage en mode maintenance		U4	U4	U4		
changer un mot de passe	U1		U2	U2 U3 U4 U5		U4
modifier les paramètres constructeur	U1					
lire la configuration constructeur	U1					
demande de test						U4
demande d'arrêt du capteur			U2	U5		U4

modifier les paramètres de mesure			U2		
lire la configuration de mesure		U2	U2	U2	
étalonnage de la chaîne de mesure		U2			
demande d'ajout d'un utilisateur			U2		
demande de suppression d'un utilisateur			U2		
demande de consultation de l'historique		U2	U2	U2 U3 U4 U5	U4
demande de génération d'un rapport		U2	U2	U2 U3 U4 U5	U4
demande de consultation de la liste des utilisateurs			U2	U2 U3 U4 U5	U4
modifier les paramètres d'un utilisateur			U2		
configurer la chaîne de mesure			U2		
lire les paramètres de la chaîne de mesure		U2	U2	U2	
demande d'initialisation du capteur			U2		U4

Tableau AIII.2 : Droits d'accès aux services.

Annexe IV

Outil de simulation d'un capteur intelligent de température

Nous présentons dans cet annexe l'outil de simulation, réalisé à partir des modèles objet décrits précédemment.

Nous avons lors de l'implémentation défini quelques objets de présentation, correspondant à une vision métier d'un ou plusieurs objets du domaine. Leur intérêt essentiel est d'isoler la représentation qui est faite des objets du domaine, des objets eux-mêmes.

La fenêtre interactive (figure AIV.1) permet la configuration du capteur intelligent : d'une part, au niveau du constructeur où celui-ci choisi les transducteurs en fonction des conditions d'exploitation du capteur intelligent (domaine nominal d'emploi); et d'autre part, au niveau du responsable du capteur intelligent sur le site d'exploitation où celui-ci choisi les paramètres relatifs à la mise en service (libellé, adresse physique ...) ainsi que ceux relatifs à la mesure (unité de mesure, valeur de repli ...) et déclare les équipements du système automatisé de production qui pourront établir une communication bidirectionnelle avec le capteur intelligent (postes de conduite, actionneurs intelligents ...).

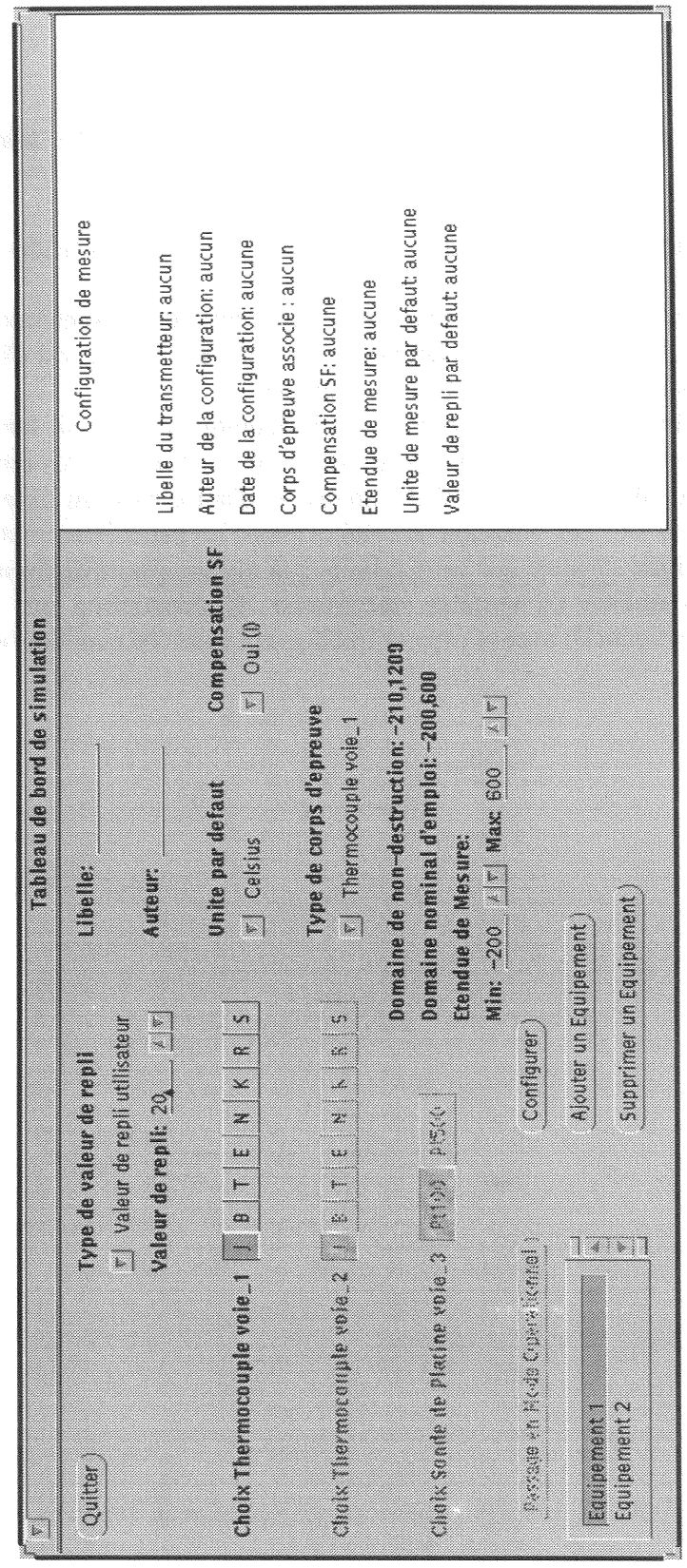


Figure AIV.1 : Fenêtre interactive permettant la configuration du capteur intelligent.

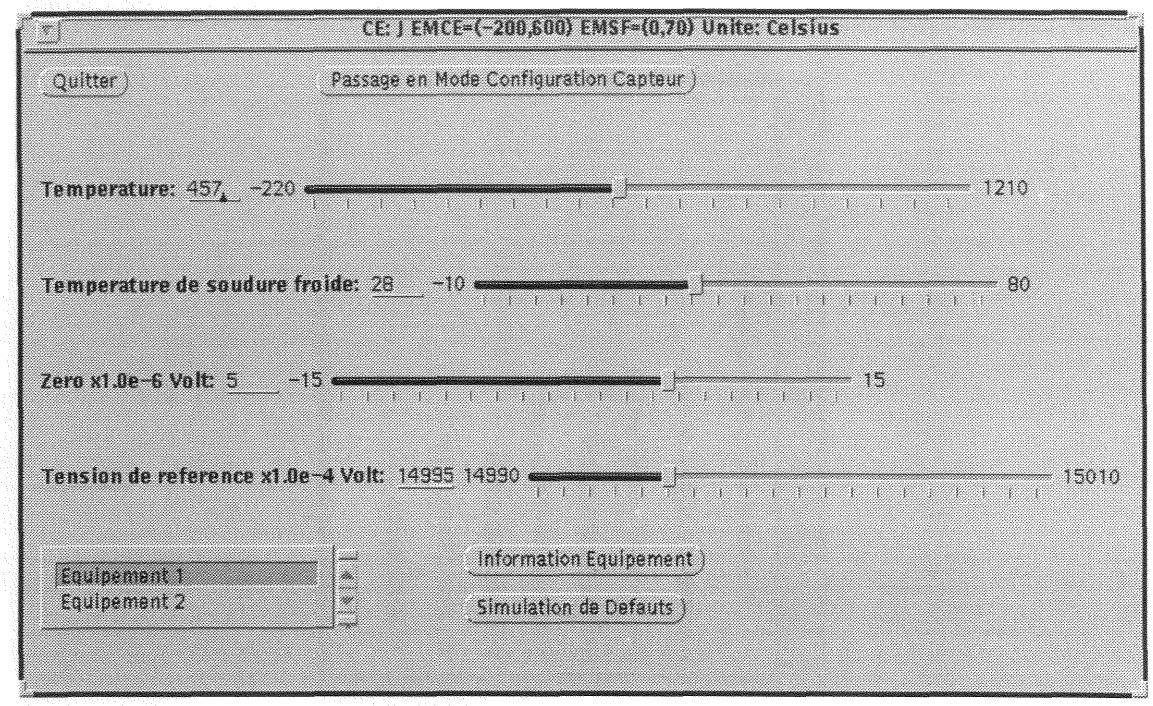


Figure AIV.2 : Tableau de bord de simulation pour les grandeurs physiques.

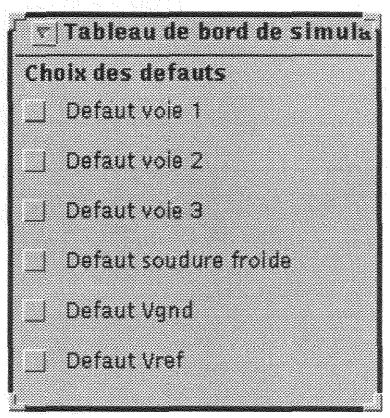


Figure AIV.3 : Tableau de bord de simulation pour les défauts.

Le tableau de bord ci-dessus (figure AIV.2) permet lorsque le capteur intelligent se trouve en phase d'exploitation (mode d'utilisation "opérationnel"), après avoir été configuré et mis en service sur le site d'exploitation, de simuler les grandeurs physiques, à savoir la température à mesurer correspondant au mesurande, la température de soudure froide (ambiante) correspondant à une grandeur d'influence, le zéro et la tension de référence correspondant à deux grandeurs d'autocontrôle.

Le tableau de bord (figure AIV.3) permet lors du mesurage de simuler la non présence d'une information servant à l'élaboration de la mesure opérationnelle (par exemple : la non présence de l'information image de la température de soudure froide lorsqu'un thermocouple est utilisé comme corps d'épreuve principal). Ceci permet de montrer le comportement des hiérarchies de mesure participant à l'élaboration de la mesure opérationnelle ainsi que les informations de validation élaborées au cours du mesurage. L'indisponibilité d'une information se fait au niveau de l'acquisition ; les défauts menant à cette indisponibilité ne sont pas détaillés, mais ils peuvent être par exemple pour l'information image de la température de soudure froide : une rupture du corps

d'épreuve, une défaillance d'un des composants de la chaîne de mesure (conditionneur, multiplexeur analogique, convertisseur analogique/numérique) participant à l'élaboration de cette information.

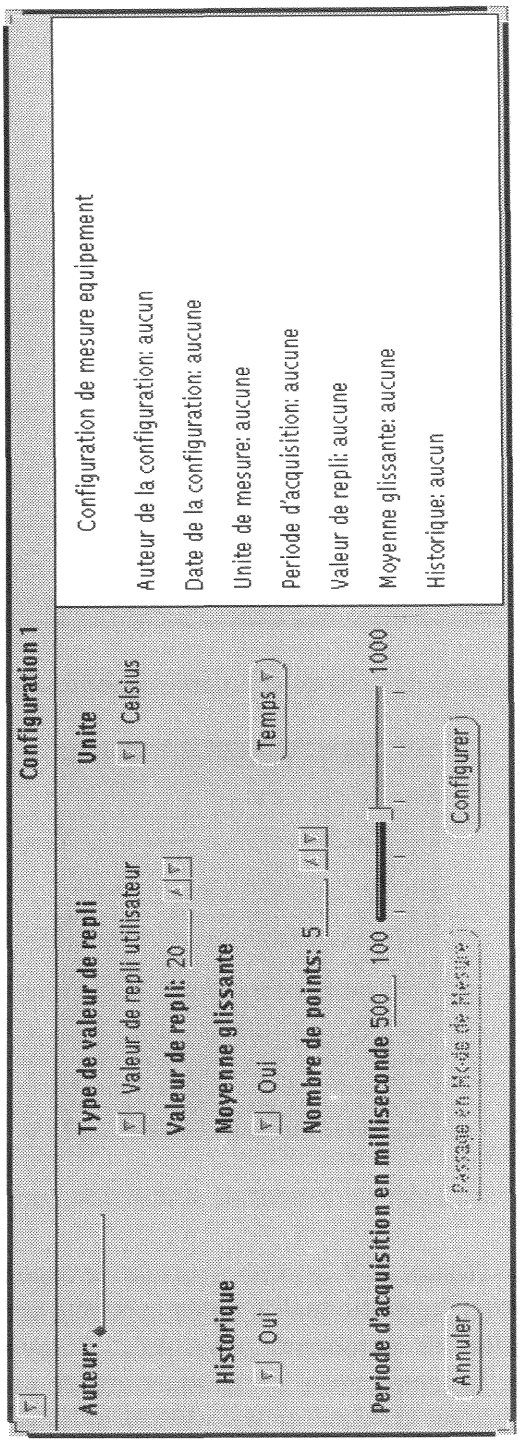


Figure AIV.4 : Fenêtre interactive permettant le paramètrage de la mesure opérationnelle relative à un équipement.

La fenêtre interactive ci-dessus (figure AIV.4) permet, lorsque le capteur intelligent est dans le mode d'utilisation "opérationnel", de paramètrer la mesure opérationnelle afin d'avoir une mesure dans le référentiel de l'utilisateur (unité, période d'acquisition, ...).

Après chaque mesurage, la mesure opérationnelle est affichée avec les informations de validation qui ont été élaborées tout au long du mesurage. La moyenne glissante est également affichée avec sa

validation ; cette dernière indique le nombre de valeurs non aberrantes utilisées lors de l'élaboration de la moyenne glissante. Un historique est également disponible ; il est composé : des mesures opérationnelles et leur validation ainsi que des moyennes glissantes et leur validation, stockées dans l'ordre chronologique dans le format défini par les paramètres de mesure de l'équipement (unité ...).

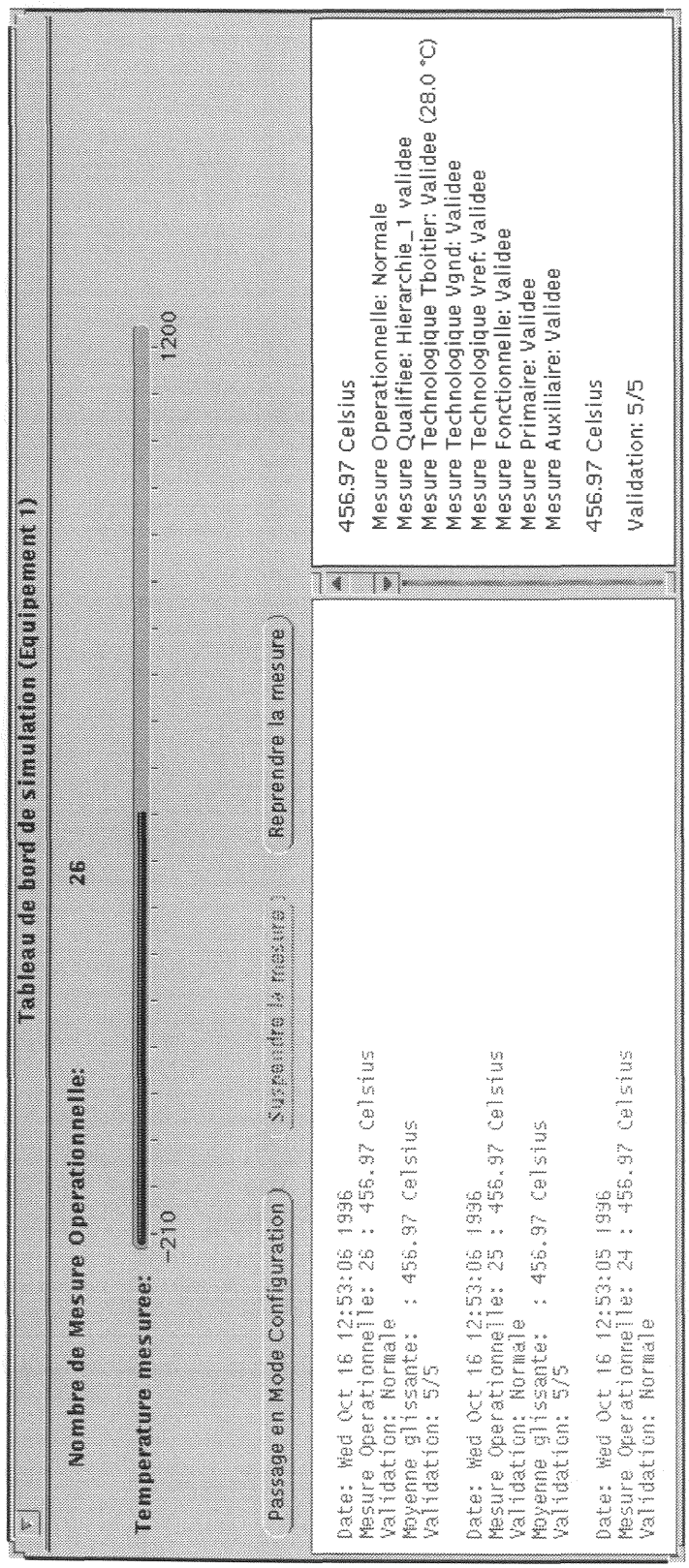


Figure AIV.5 : Tableau de bord de simulation relatif à un équipement lors du mesurage.

Annexe V

Complément des modèles OMT et GEODE

Description OMT du capteur intelligent concernant le mesurage (par ordre d'apparition) :

- Diagramme de classes montrant la bibliothèque des modèles utilisés pour l'élaboration de la mesure opérationnelle.

- Diagramme de classes montrant l'historique où sont archivés les différentes mesures ainsi que les différents comptes-rendus d'erreurs survenus lors du mesurage.

- Diagramme de classes montrant le sous-système d'acquisition où s'opère la conversion des grandeurs physiques en grandeurs numériques.

- Diagramme de classes montrant le sous-système de communication permettant entre autres de gérer les messages provenant des opérateurs ou des automatismes du systèmes automatisé de production.

- Diagramme de classes montrant le superviseur qui élabore la liste des états des services et des ressources en indiquant la disponibilité de ceux-ci.

- Diagramme d'états montrant le contrôle des requêtes provenant d'un équipement dit "interfacé".

- Diagramme d'états de la classe "mesure opérationnelle".

- Diagramme d'états de la classe "mesure validée".

- Diagramme à flots de données pour l'élaboration d'une mesure opérationnelle.

Transformation de la description OMT du capteur intelligent en une architecture LDS :

- Diagramme d'interactions représentant le flot des messages échangés entre les différents sous-systèmes.

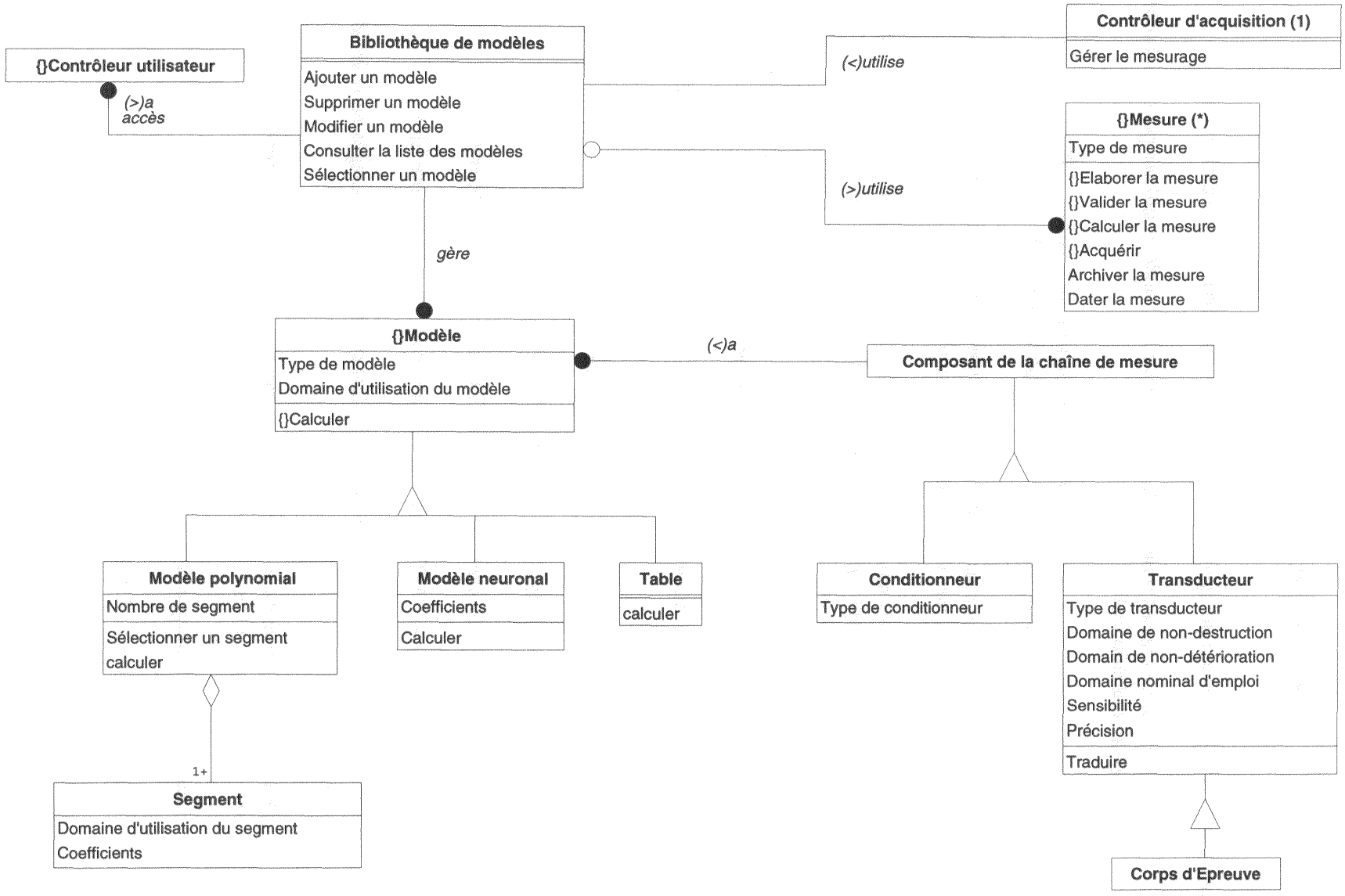
- Diagramme d'interactions représentant le flot des messages échangés entre les objets du sous-système de mesure.

- Diagramme dynamique de la mesure opérationnelle en notation LDS.

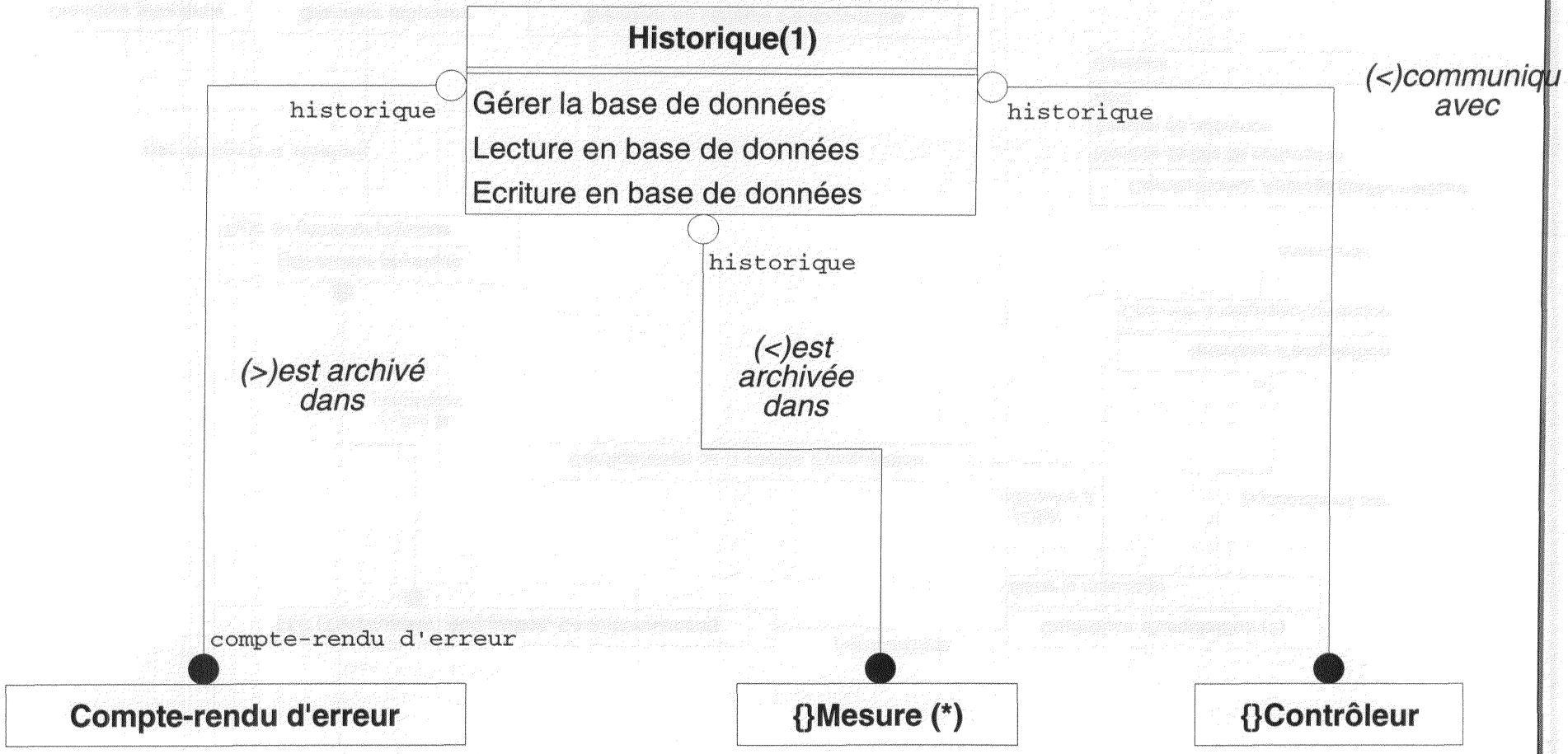
- Diagramme de suivi d'événements pour la mesure opérationnelle.

- Diagramme de suivi d'événements pour une hiérarchie de mesure.

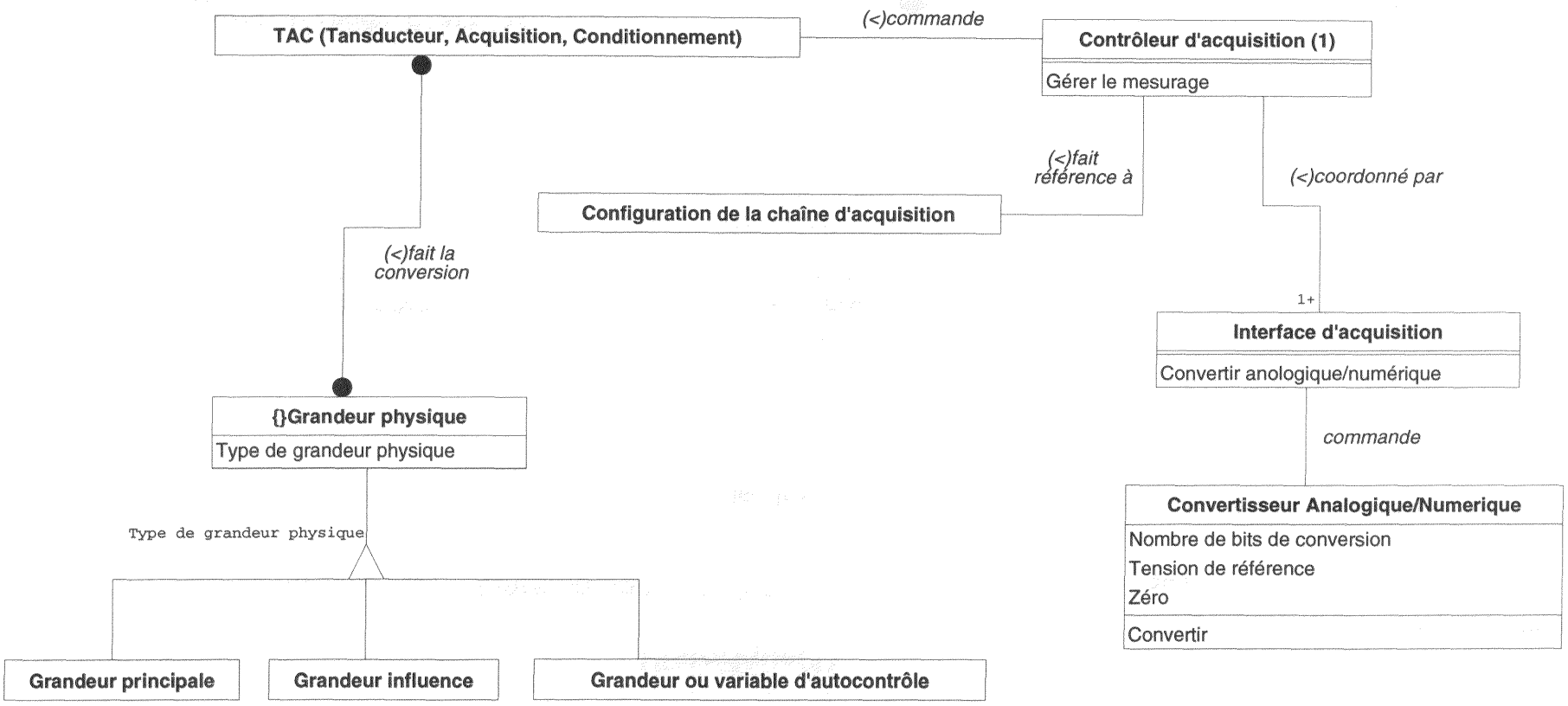
CLASS MODULE Bibliothèque de modèles



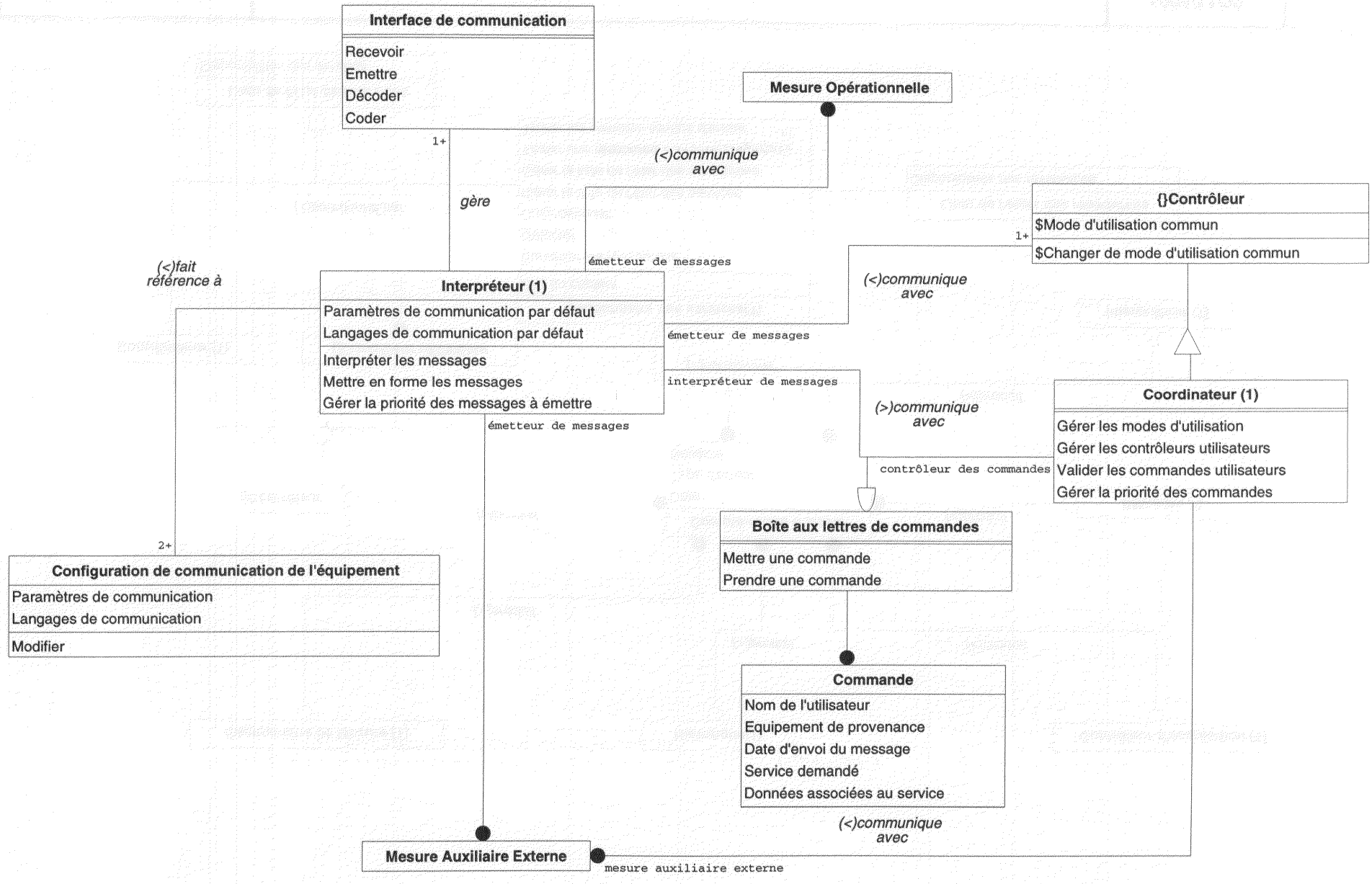
CLASS MODULE Sous-Système d'archivage



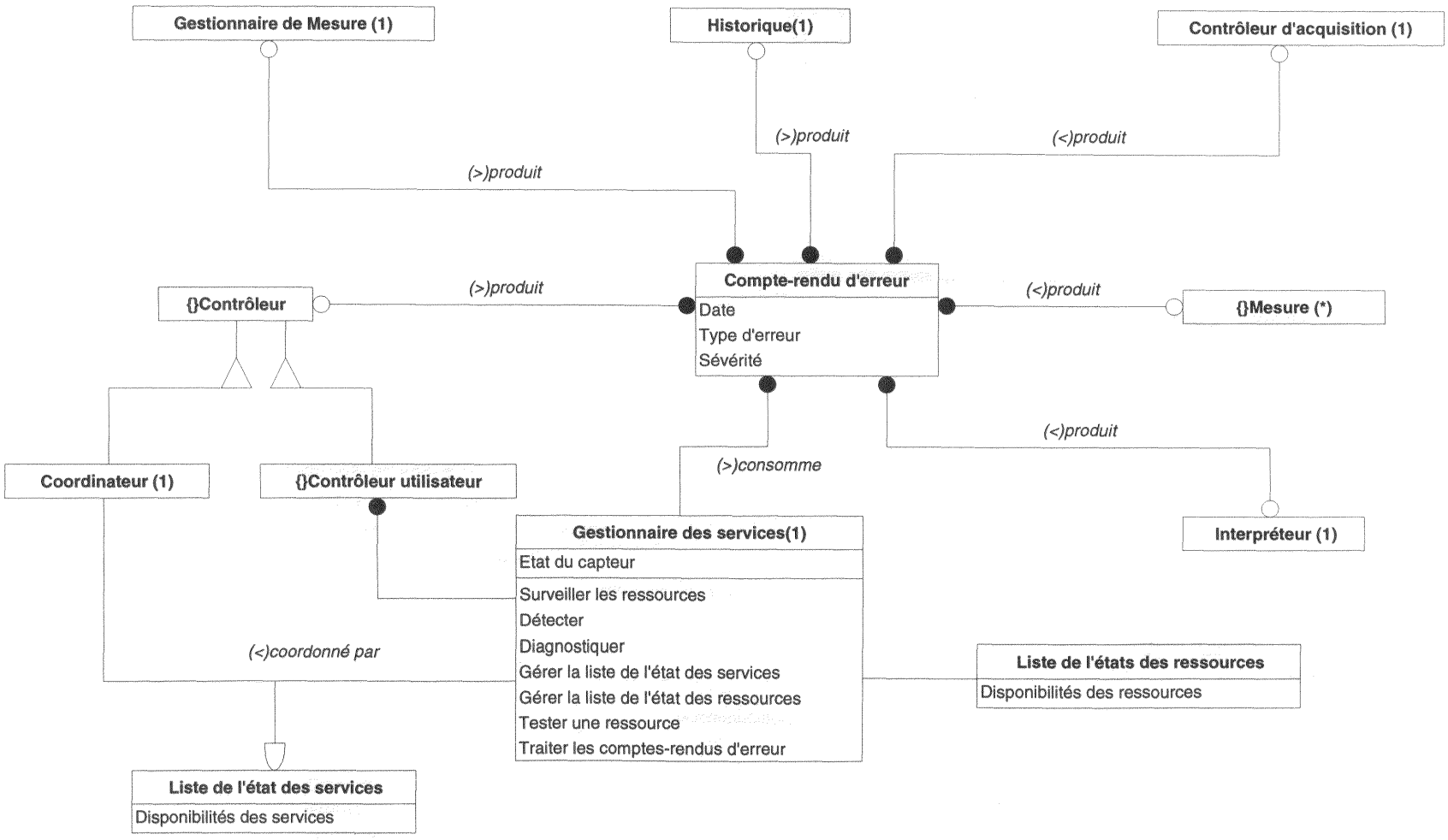
CLASS MODULE Sous-système d'acquisition et Grandeurs physiques



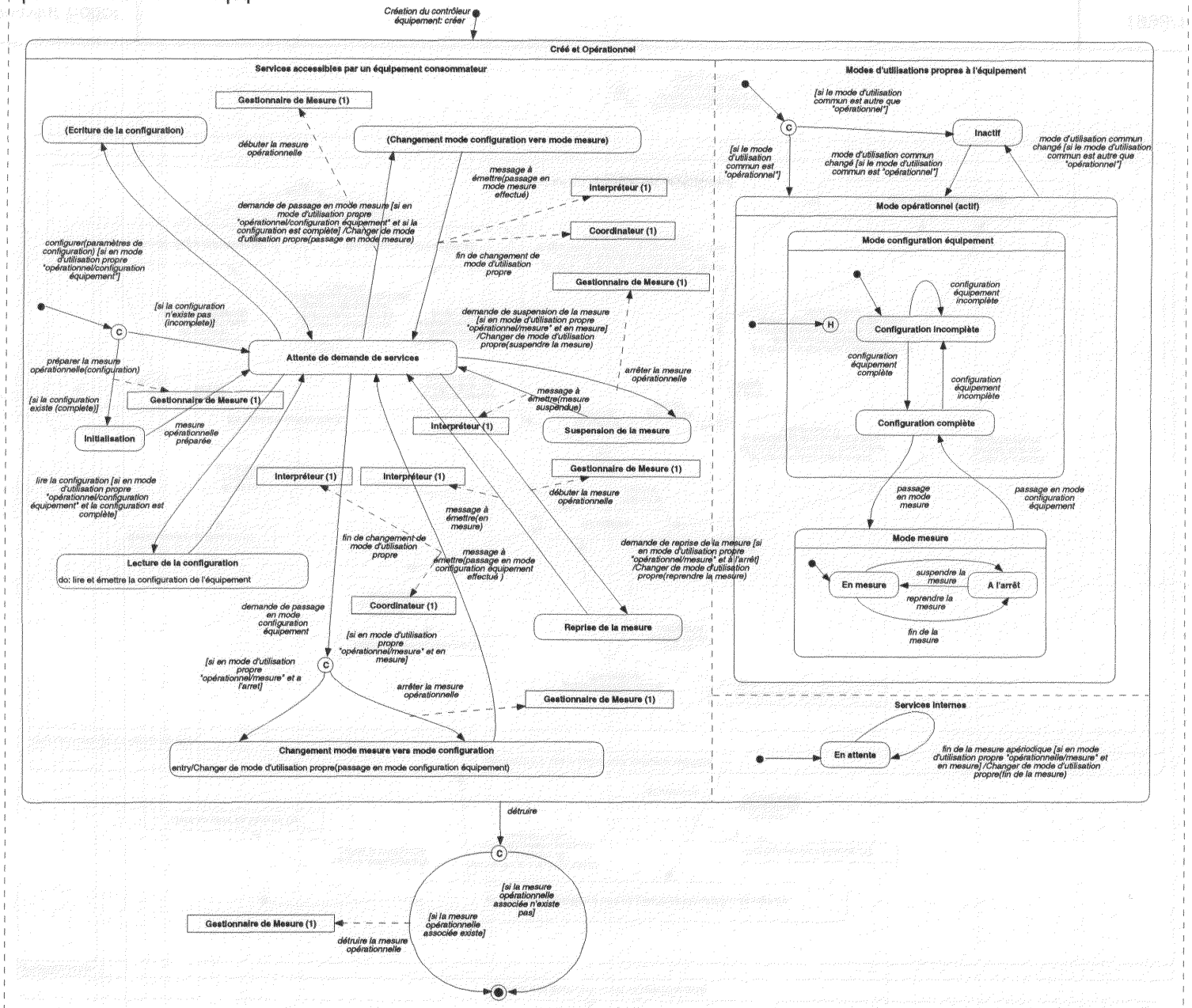
CLASS MODULE Sous-Système de communication



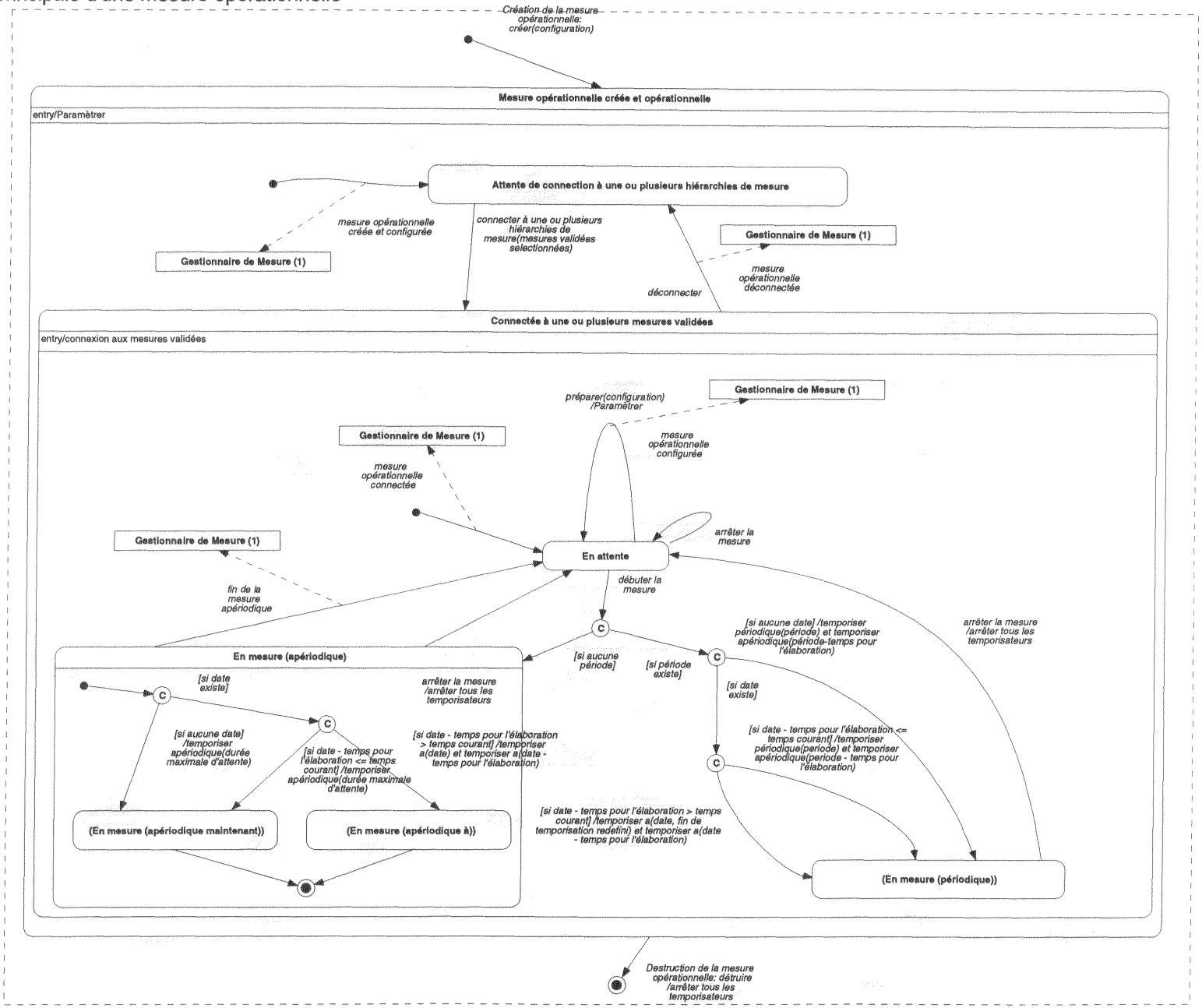
CLASS MODULE Gestion des comptes rendus d'erreur



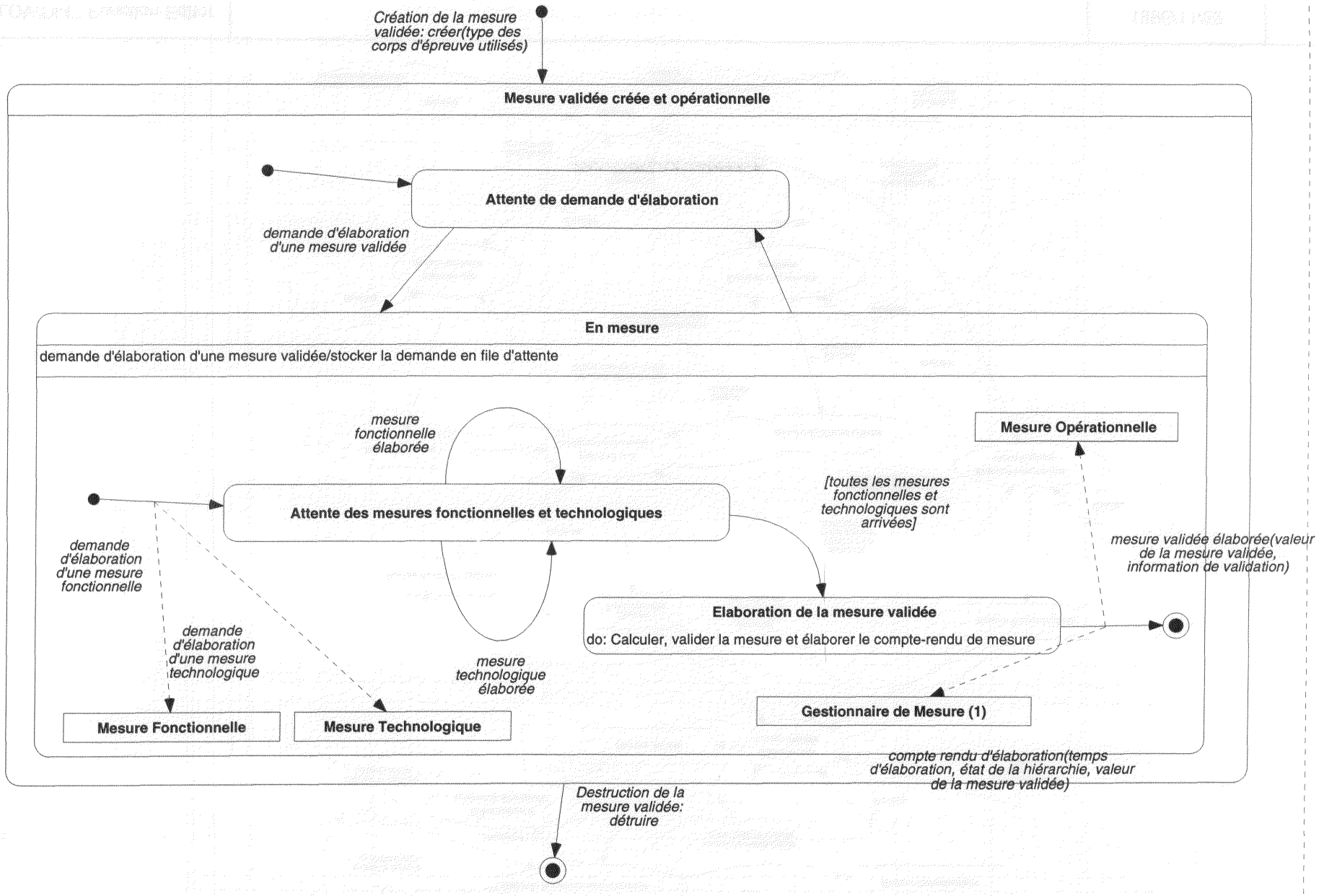
ACTIVITY Activité principale d'un contrôleur équipement



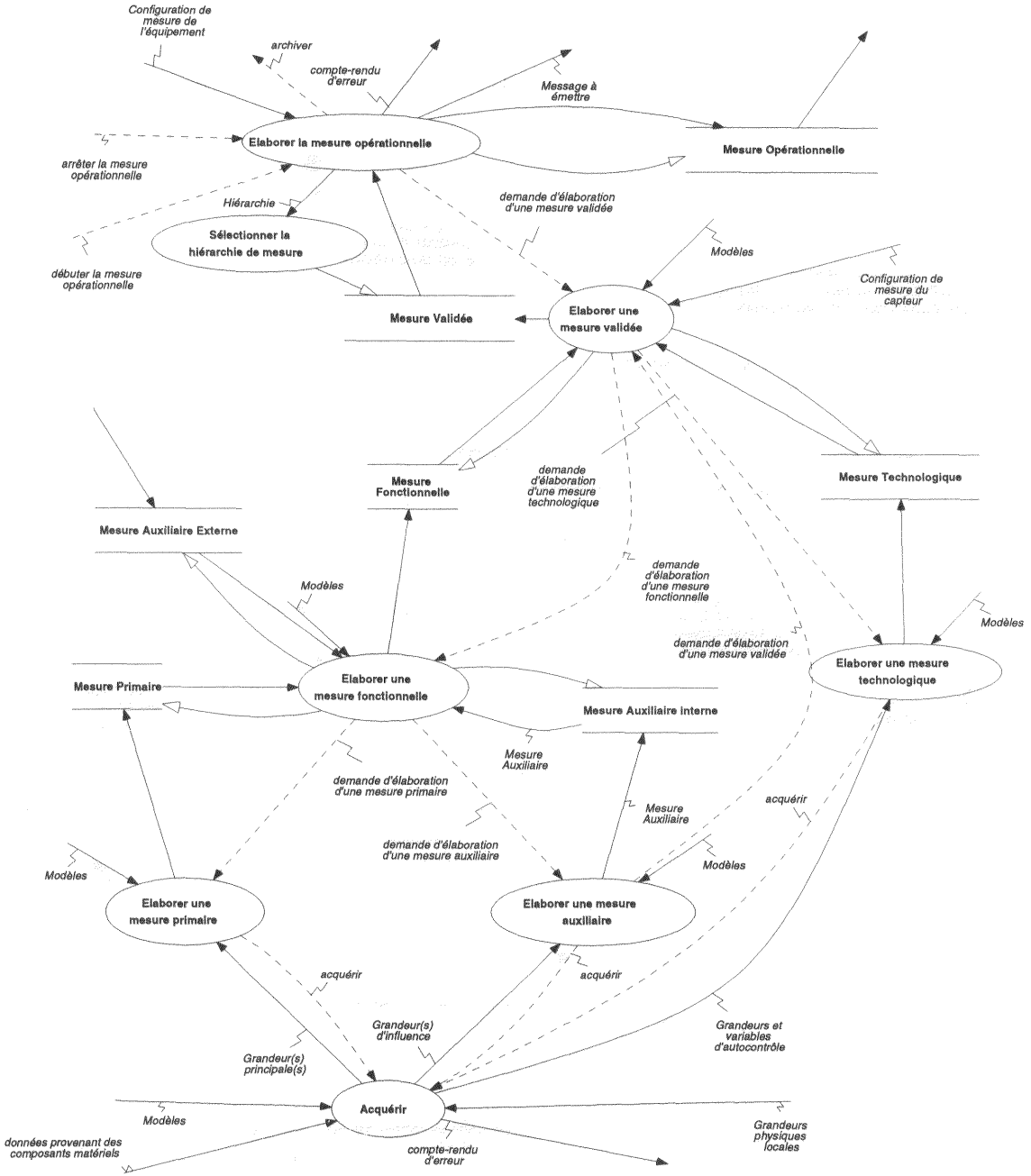
ACTIVITY Activité principale d'une mesure opérationnelle



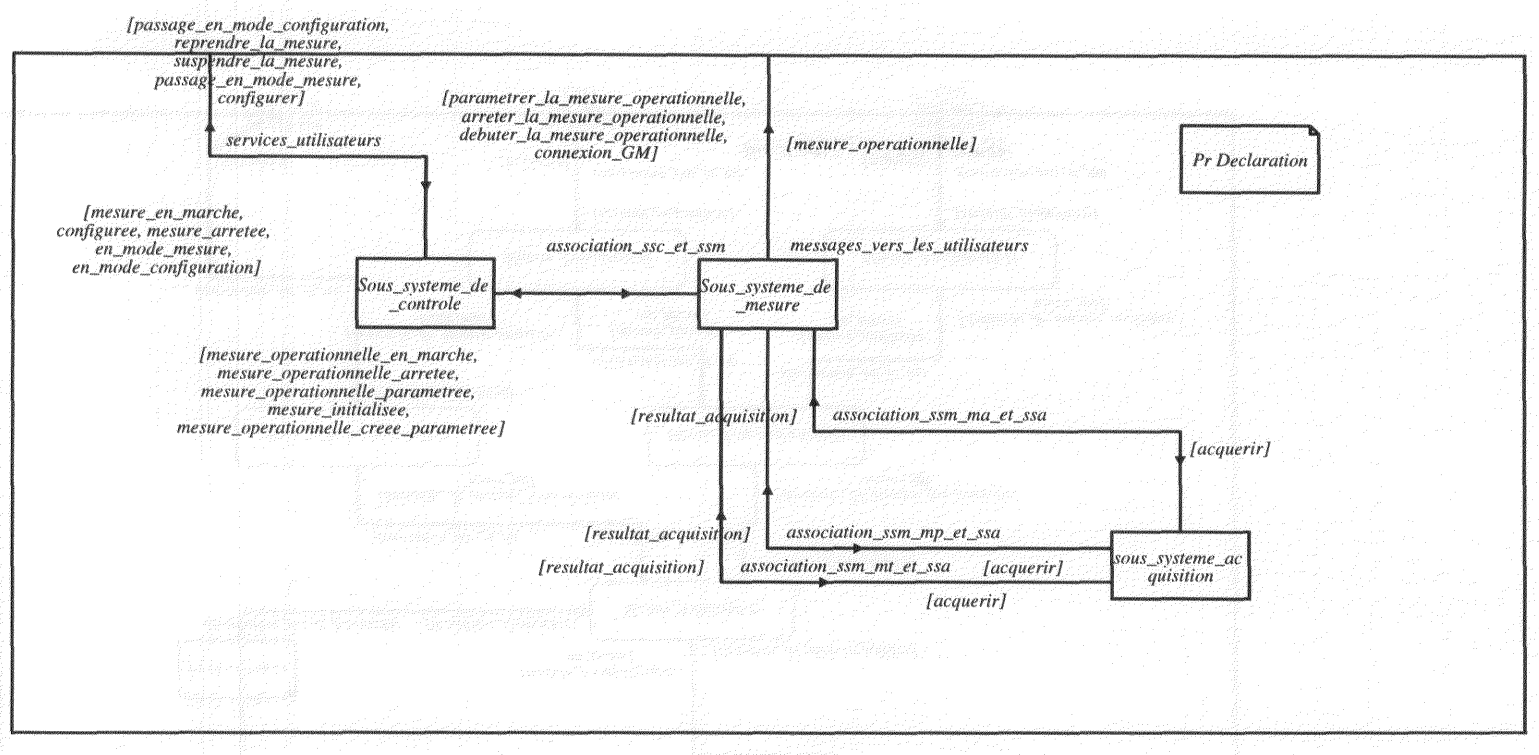
ACTIVITY Activité principale d'une mesure validée



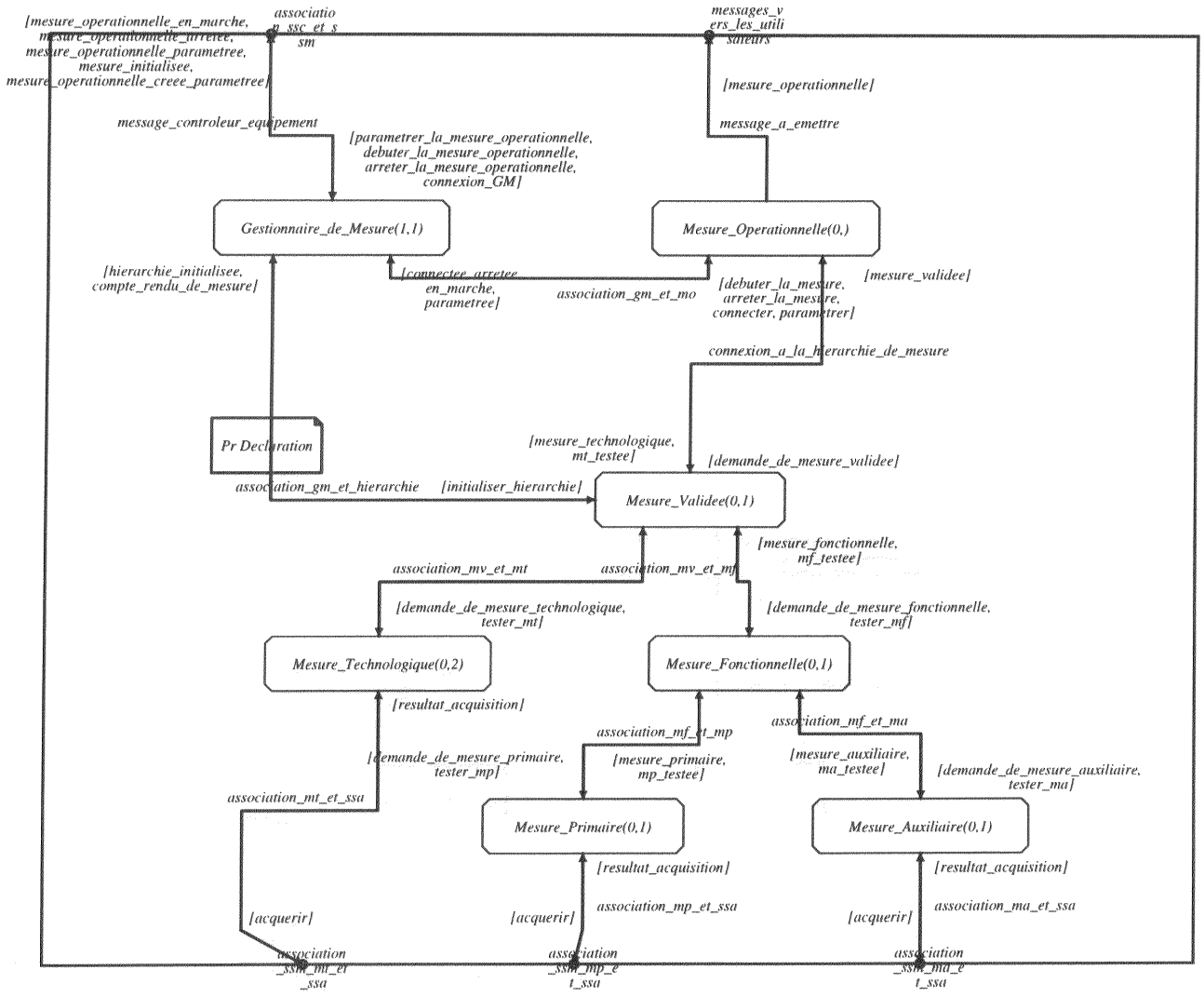
PROCESS Mesurer



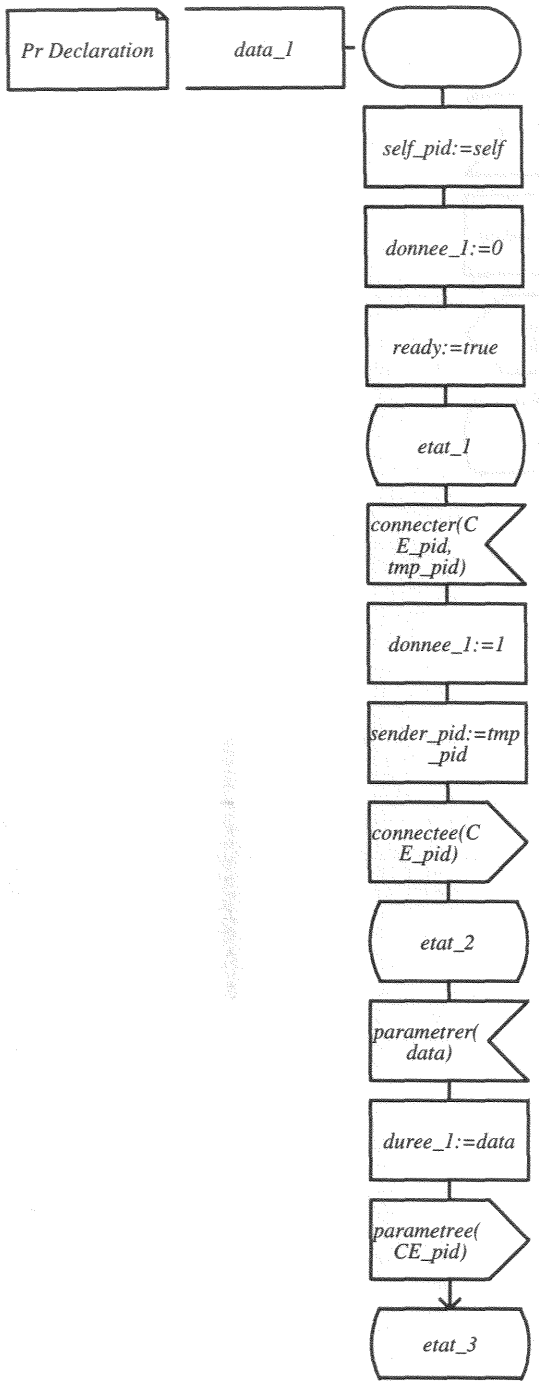
SYSTEM SYS_I


[passage_en_mode_configuration, reprendre_la_mesure, suspendre_la_mesure, passage_en_mode_mesure, configurer]
services_utilisateurs
[parametrer_la_mesure_operationnelle, arreter_la_mesure_operationnelle, debuter_la_mesure_operationnelle, connexion_GM]
[mesure_operationnelle]
Pr Declaration
[mesure_en_marche, configuree, mesure_arretee, en_mode_mesure, en_mode_configuration]
association_ssc_et_ssm
messages_vers_les_utilisateurs
Sous_systeme_de_controle
Sous_systeme_de_mesure
[mesure_operationnelle_en_marche, mesure_operationnelle_arretee, mesure_operationnelle_parametree, mesure_initialisee, mesure_operationnelle_creee_parametree]
[resultat_acquisition]
association_ssm_ma_et_ssa
[acquerir]
[resultat_acquisition]
association_ssm_mp_et_ssa
[resultat_acquisition]
association_ssm_mt_et_ssa
[acquerir]
sous_systeme_acquisition
[acquerir]

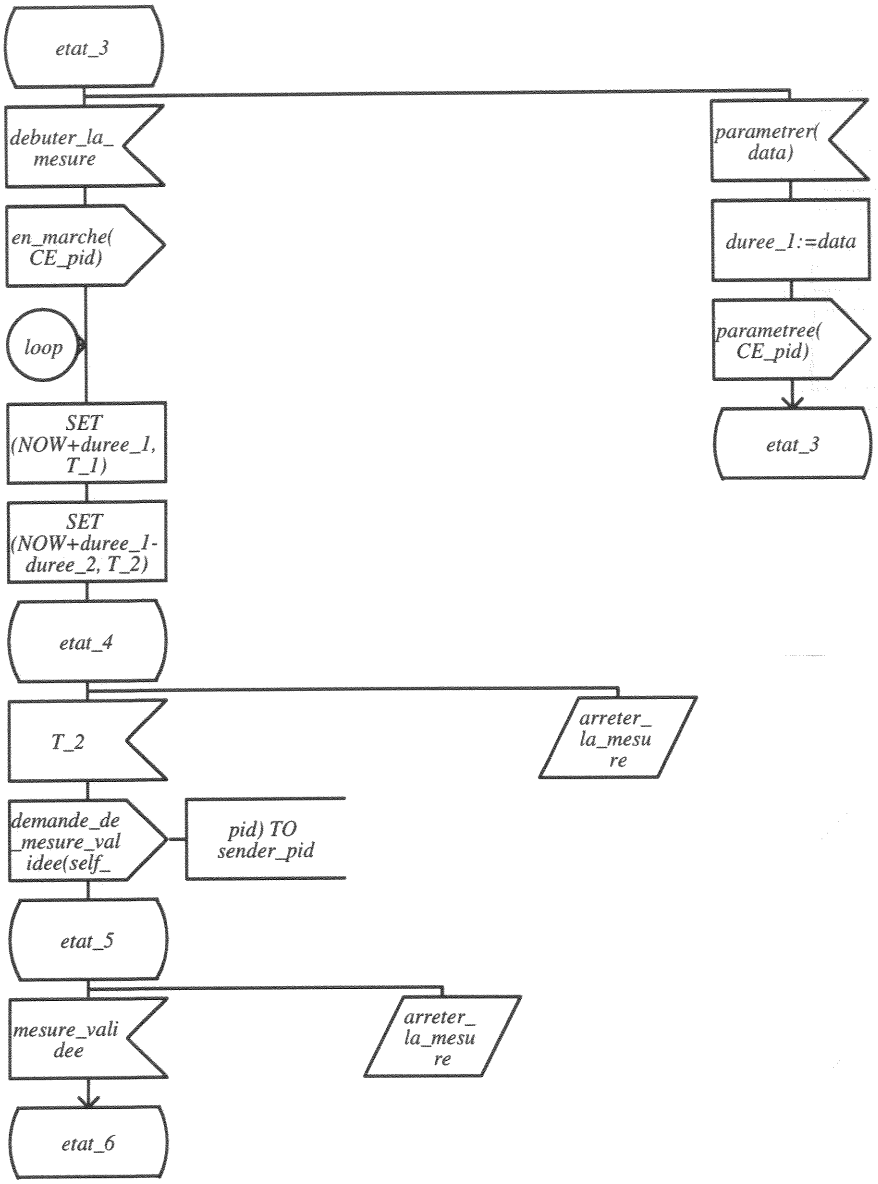
BLOCK SYS_I/Sous_systeme_de_mesure



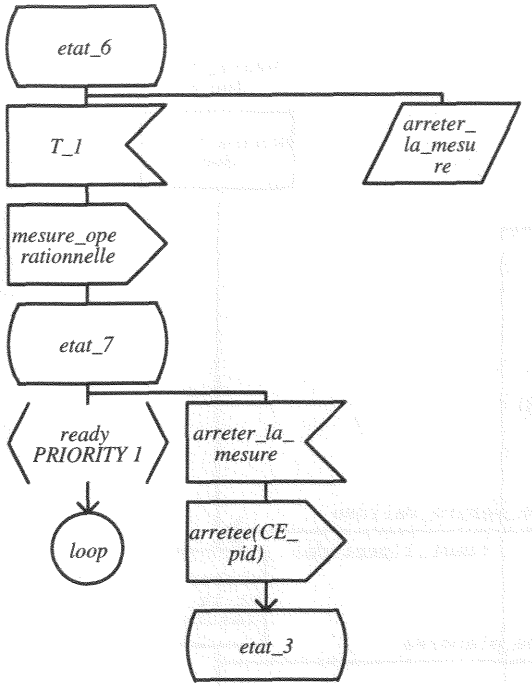
PROCESS SYS_I/Sous_systeme_de_mesure/Mesure_Operationnelle



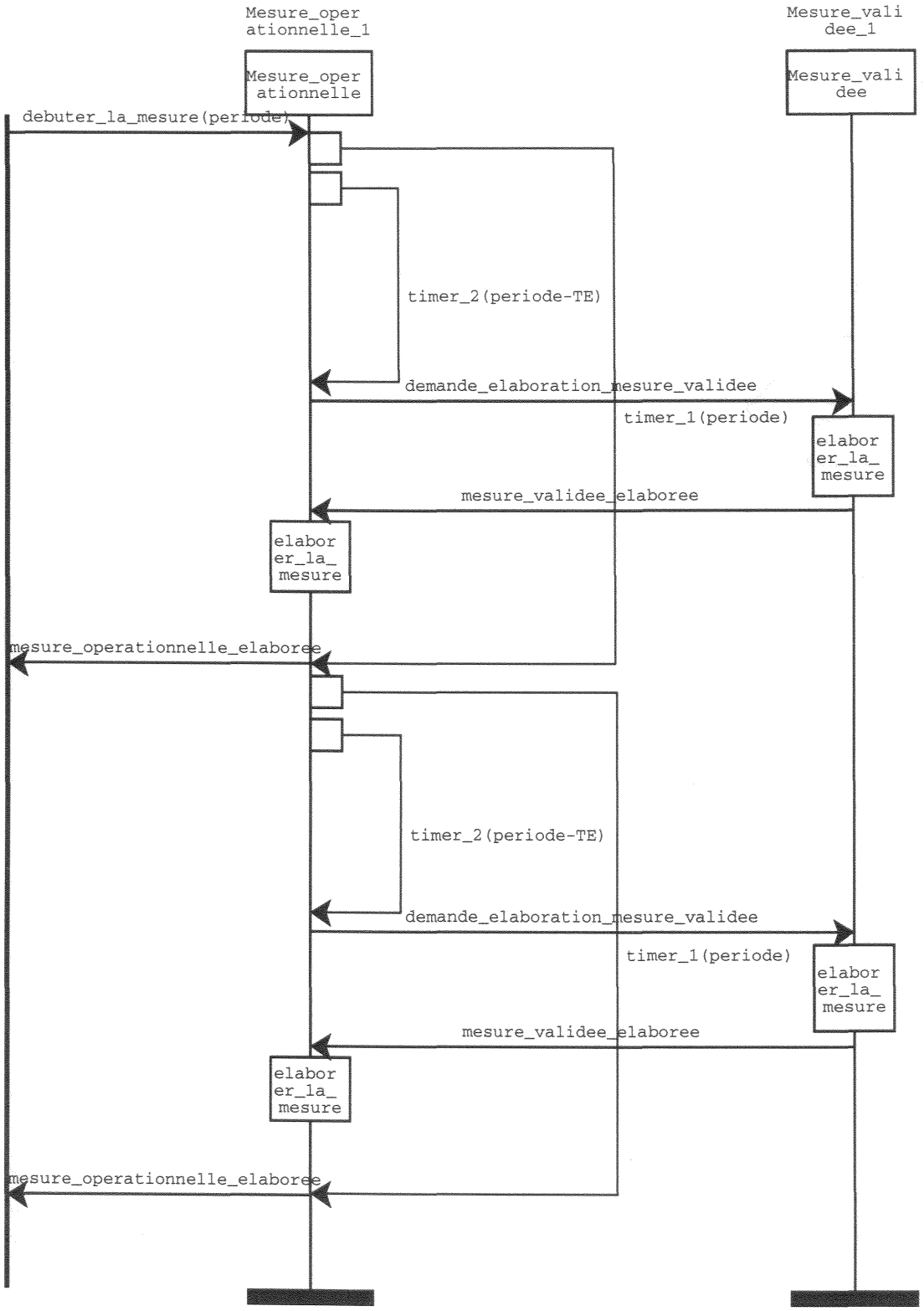
PROCESS SYS_I/Sous_systeme_de_mesure/Mesure_Operationnelle



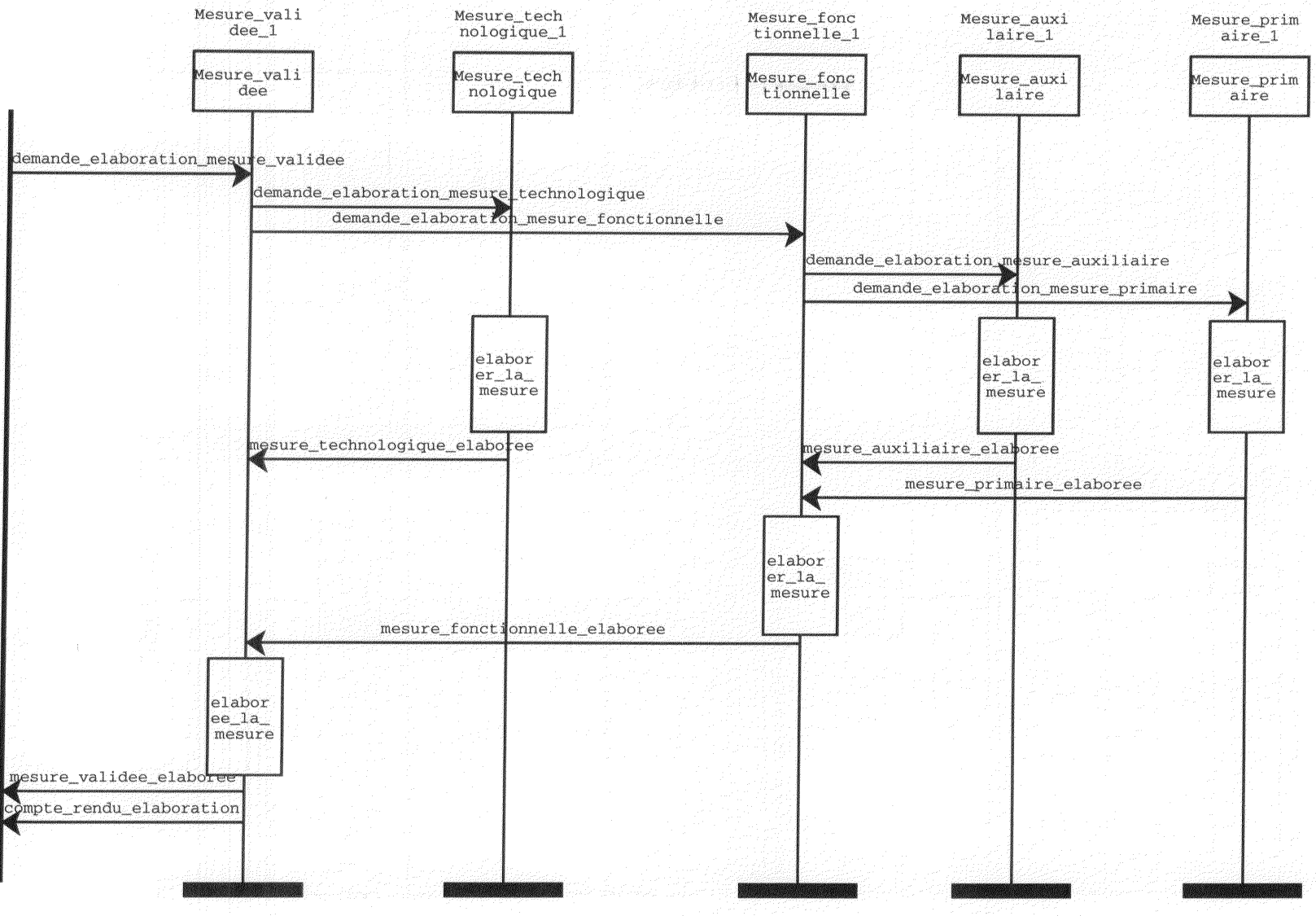
PROCESS SYS_I/Sous_systeme_de_mesure/Mesure_Operationnelle



MSC periodique_sans_date



MSC Scenario(hierarchie_de_mesure)



université HENRI POINCARÉ
NANCY 1

Nom : **LUTTENBACHER**

Prénom : **Damien**

DOCTORAT de l'UNIVERSITE HENRI POINCARE, NANCY-I

en AUTOMATIQUE

VU, APPROUVÉ ET PERMIS D'IMPRIMER

Nancy, le 17 FÉV 1997 UHP 019/97

Le Président de l'Université

UNIVERSITÉ HENRI POINCARÉ, NANCY 1
LE PRÉSIDENT
J.P. FINANCE

3C-F9

UNIVERSITÉ HENRI POINCARÉ - NANCY 1
24-30, rue Lionnois B.P.3069 54013 Nancy cedex
Tél : 83 32 81 81 Fax : 83 32 95 90
Adresse électronique : _____ @presn1.u-nancy.fr

Modélisation du concept capteur intelligent par une approche orientée objet : application à un capteur de intelligent de température

Résumé :

S'inscrivant dans la mouvance des travaux du CIAME (Comité Interprofessionnel pour l'Automatisation et la MEsure) et les réflexions du groupe de travail CRAN-AMI (Actionnements et Mesures Intelligents) relatifs aux capteurs intelligents, cette étude propose un modèle de référence générique du concept capteur intelligent, développé dans un contexte de réutilisabilité. Dans cette optique, la démarche de modélisation adoptée repose sur une approche orientée objet, suivant la technique de modélisation par objets (Object Modeling Technique : OMT) formalisant les aspects structurel, comportemental et fonctionnel du capteur intelligent. La description du capteur intelligent est réalisée à l'aide de l'AGL (Atelier de Génie Logiciel) LOV/OMT (commercialisé par la société VERILOG) support de la méthodologie OMT. La validation dynamique de la description du capteur intelligent est réalisée dans l'environnement GEODE (commercialisé par la société VERILOG).

Les concepts de modélisation et de réutilisation sont appliqués à la modélisation d'un capteur intelligent de température et se concrétisent par le développement d'un outil de simulation, s'appuyant sur les modèles élaborés.

Mots-clés :

Capteur Intelligent - Capteur Intelligent de Température - Modélisation par Objets - Modèle de Référence - OMT.

Modelling of smart sensor concept with an object oriented approach : application to a smart temperature sensor

Abstract :

This work, in line with the works of CIAME (Comité Interprofessionnel pour l'Automatisation et la MEsure) and the reflections of CRAN-AMI (Actionnements et Mesures Intelligents) working group in the field of smart sensors, aim at proposing a generic reference model of a smart sensor established in a reusable context. The modelling approach adopted is based on the object oriented approach, by the way of the object modelling technique (OMT) which emphasises the data, behaviour, and architecture points of view of the smart sensor. The LOV/OMT tool is used to edit the smart sensor description (commercialized by the VERILOG compagny). The dynamic validation is carried out in the GEODE environment.

The modelling and reuse concepts are applied to the modelling of a smart temperature sensor, and are materialised by the code generation of the elaborated models.

Keywords :

Smart sensor, Smart Temperature Sensor, Object Oriented Modelling, Reference Model, OMT.