Contribution à la modélisation d'un système interactif d'aide à la conduite d'un procédé industriel

Dragos Dobre

Nancy-Université
FACULTE DES SCIENCES ET TECHNOLOGIES

U.F.R. Sciences et Technologies
École Doctorale IAEM Lorraine
Département de Formation Doctorale Automatique

Thèse

Présentée pour l'obtention du titre de

Docteur de l'Université Henri Poincaré, Nancy 1

(Spécialité Automatique, Traitement du Signal et Génie Informatique)

par

Dragoş DOBRE

CONTRIBUTION A LA MODELISATION D'UN SYSTEME INTERACTIF D'AIDE A LA CONDUITE D'UN PROCEDE INDUSTRIEL

Soutenue publiquement le 15 Novembre 2010 devant la commission d'examen :

Président et rapporteur	Bernard THIRION	Professeur, Université de Haute Alsace
Rapporteur	Pascal BERRUET	Professeur, Université de Bretagne-Sud
Examinateurs	Catherine DEVIC	Responsable de groupe EDF R&D/STEP
	Gérard MOREL	Professeur, UHP Nancy I (Directeur de thèse)
	Eddy BAJIC	Professeur, UHP Nancy I (Codirecteur de thèse)
	David GOUYON	Maître de conférences, UHP Nancy I

Centre de Recherche en Automatique de Nancy
Faculté des Sciences et Technologies – 54500 Vandœuvre-lès-Nancy

A Simona, Ioana et Andrei

Remerciements

Je tiens à remercier avant tout, les membres du jury qui me font l'honneur de participer à l'examen de ce travail.

Je souhaite tout d'abord remercier Gérard Morel, Professeur à l'Université Henri Poincaré Nancy 1, de m'avoir encadré, soutenu et fait partager sa passion pour la recherche et sa rigueur scientifique tout au long de cette thèse. Un grand merci pour sa patience et sa disponibilité, ainsi que ses qualités humaines et scientifiques.

Je remercie également Eddy Bajic, Professeur à l'Université Henri Poincaré Nancy 1, d'avoir encadré mon stage de Master 2, de m'avoir encouragé à poursuivre en thèse, et de m'avoir co-encadré. J'ai apprécié ses connaissances technologiques qui m'ont permis d'acquérir de nouvelles compétences que j'ai pu exploiter dans le développement de la thèse.

Je remercie David Gouyon, Maître de Conférences à l'Université Henri Poincaré Nancy 1, pour son encadrement, pour les nombreux échanges que nous avons eus et pour sa disponibilité et son aide dans des moments de doute.

A Pascal Berruet, Professeur à l'Université de Bretagne-Sud, et Bernard Thirion, Professeur à l'Université de Haute Alsace, j'adresse mes sincères remerciements pour l'honneur qu'ils m'ont fait en acceptant d'évaluer ce travail en tant que rapporteurs. J'ai pu apprécier la pertinence des remarques faites sur mon travail.

Je suis reconnaissant envers Catherine DEVIC, Responsable de groupe EDF R&D/STEP, pour l'enthousiasme avec lequel elle a accepté de participer à ce jury de thèse. Ses remarques nombreuses et constructives m'ont permis d'apporter plus de précisions et de pertinence à mon travail de thèse.

Cette thèse ayant été réalisée au Centre de Recherche en Automatique de Nancy, je tiens donc à remercier tout naturellement son directeur, Alain RICHARD ainsi que Thierry DIVOUX responsable du groupe thématique SYMPA pour leur accueil, toute l'attention qu'ils m'ont portée, et pour les moyens mis à ma disposition durant ces années.

J'associe à ces remerciements tous les membres du CRAN, enseignants chercheurs, IATOS et doctorants, avec qui j'ai pu travailler et échanger durant ces années de thèse.

Mon dernier remerciement est pour mon épouse Simona. Je te remercie d'avoir cru en moi durant tous ces années et de m'avoir aidé à me dépasser. Mes travaux n'auraient pu aboutir sans ton soutien indéfectible. Merci d'être à mes cotés à chaque instant.

Table des matières

Abréviations

AFIS	Association Française d'Ingénierie Système
ACT	Diagramme d'activité (Activity Diagram)
BDD	Diagramme de définition de bloc (Block Definition Diagram)
GE	Guide d'Exploitation
eFFBD	Enhanced Functional Flow Block Diagram
IBD	Diagramme de bloc interne (Internal Block Diagram)
IDM	Ingénierie Dirigée par les Modèles
INCOSE	International Council on Systems Engineering
IS	Ingénierie Système
ISBM	Ingénierie Système Basée sur des Modèles
IVV	Intégration/Vérification/Validation
MES	Manufacturing Execution System
MFA	Module Fonctionnel d'Automatisation
MO	Mode Opératoire
MOA	Maître d'ouvrage
MOE	Maître d'œuvre
NIAM	Natural language Information Analysis Method
ORM	Object-Role Modelling
OTB	Objet Technique de Base
OTBi	Objet Technique de Base instrumenté
OTBp	Objet Technique de Base non instrumenté
PAR	Diagramme paramétrique (Parametric Diagram)
PC	Partie Commande
PE	Phase d'Exploitation
PF	Problem Frame
PO	Partie Opérative
PPI	Partie Prenante Intéressée
PPR	Partie Prenante Réalisatrice
REQ	Diagramme d'exigences (Requirement Diagram)
RFID	Radio Frequency Identification
SD	Diagramme de séquence (Sequence Diagram)
SIAC	Système Interactif d'Aide à la Conduite
STM	Diagramme d'état (State Machine Diagram)
SysML	System Modelling Language
UC	Diagramme de cas d'utilisation (Use Case Diagram)

Introduction

Nous défendons dans cette thèse l'intérêt d'améliorer l'interactivité numérique entre un système technique de soutien et des agents d'exploitation (opérateur, rondier, …) qui appliquent des procédures de conduite sur le terrain.

Ces travaux ont été réalisés dans le cadre de trois projets :

- Projet LABIME (LAngage d'expression des Besoins en Informations pour les Métiers d'Exploitation, Galara, et al. (2008)) inscrit dans le Groupement d'Intérêt Scientifique sur la Surveillance, Sûreté et Sécurité des Grands Systèmes (GIS 3SGS) ;
- Projet SafeTech soutenu par le Contrat de Projets Etat-Région (CPER) Modélisation, Information et Systèmes Numériques (MISN) 2007-2013 ;
- Projet Systèmes Interopérants inscrit dans le Groupe Thématique Systèmes de Production Ambiants (SYMPA) au sein du Centre de Recherche en Automatique de Nancy (CRAN).

Dans la pratique, la conduite des grands systèmes industriels à risque s'avère souvent problématique dès lors que des agents d'exploitation ont à faire face à des situations mal connues ou mal perçues. Cela peut être, par exemple, le cas dès lors que l'on n'est plus dans des phases nominales de fonctionnement, par exemple durant les phases d'arrêt et de remise en route consécutives à des opérations de maintenance ou de remise à niveau des installations. C'est plus particulièrement le cas lors des opérations de consignation[1] qui mettent en œuvre un nombre important d'objets techniques (vannes manuelles, vannes régulées, documents papier, …) selon des Modes Opératoires qui ne réfléchissent pas forcément de façon exhaustive le combinatoire des situations d'état, réelles ou inférées, auxquelles ces agents de terrain doivent faire face.

Une cause possible de ces situations récurrentes de dysfonctionnements est une conception insuffisamment intégrée du système de soutien (exploitation, maintenance, …) et du système principal qui a pour effet de ne pas suffisamment rationnaliser la distribution des rôles entre le système humain et le système technique de soutien (Galara D. , 2006). La solution proposée *a posteriori* par Galara, et al. (2008) dans le projet LABIME pour faciliter et rendre plus sûre cette interopération Homme-Système consiste à améliorer la sémantique du langage de conduite du procédé par un ensemble d'expressions logiques qui formalisent le comportement logique attendu du système physique tout en inhibant les actions non-prévues qui peuvent le détériorer, voire engendrer des situations dangereuses.

[1] La consignation renvoie à l'activité de pose et dépose des régimes : « La mise sous régime est un acte d'exploitation destiné à fournir les conditions de sécurité permettant au personnel d'exécuter des interventions sur les ouvrages déterminés, après que le chargé de travaux, d'intervention immédiate ou d'essais ait pris les mesures de sécurité qui lui incombent. » (EDF, 1992)

La mise en conformité des installations, par exemple en réponse à de nouvelles directives relatives aux risques de contamination des opérateurs de conduite en situation critique (Décret 2003-296, 2003) (Figure 1), peut être aussi une opportunité d'un changement de paradigme pour la conception *a priori* de l'exploitation de ces installations.

«requirement» «inducedRequirement» {indReqSource = Décret 2003-296} Valeurs limites de dose applicables aux personnes exposées aux rayonnements
txt Art. R. 231-76. - I. - La somme des doses efficaces reçues par exposition externe et interne ne doit pas dépasser 20 mSv sur douze mois consécutifs.

Figure 1 : Exigences induite par le Décret 2003-296 (2003)

Par exemple, les progrès des technologies « infotroniques » (RFID, Capteurs sans fil, PDA, …) permettent d'envisager la mise en œuvre du paradigme de l'Intelligence Ambiante (Bruns, 2006; Cook, et al., 2009) pour la conception du système d'exploitation des installations industrielles. Ainsi Pirus (2006) explore l'intérêt de l'utilisation locale en exploitation des technologies de l'information, et Dionis et Pirus (2007) proposent des solutions pour faciliter l'interopération numérique Homme – Homme, Homme – Système et Système – Système sur le terrain, par exemple pour surveiller la configuration du système (Figure 2).

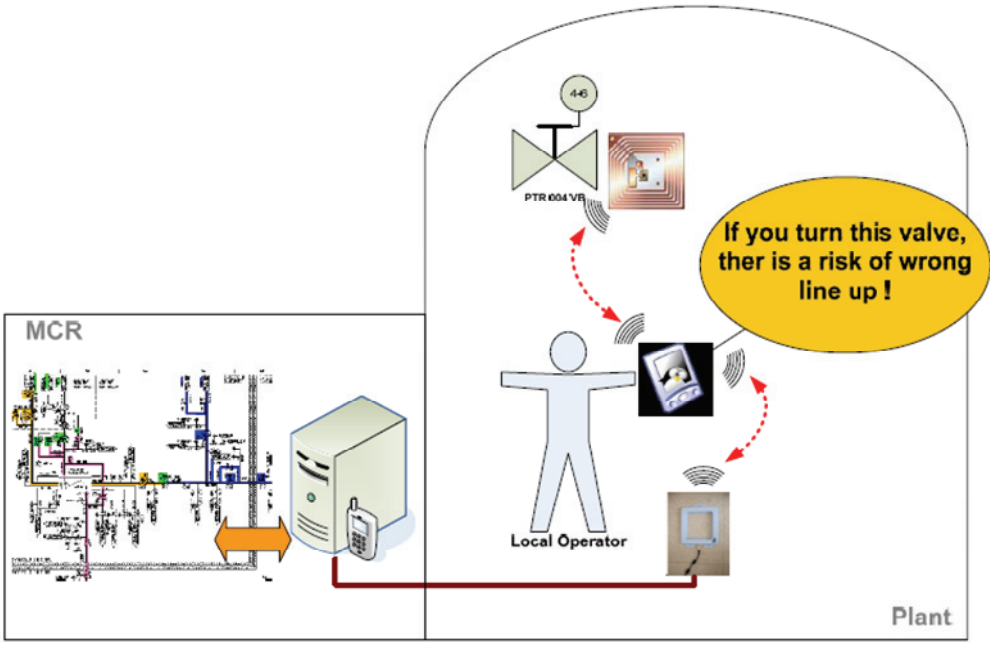


Figure 2 : La surveillance des mauvaises configurations (Dionis, et al., 2007)

Par ailleurs, des travaux antérieurs (Figure 3) ont montré dans d'importants programmes Européens l'intérêt d'architectures techniques interopérables, distribuant au plus près du procédé une forme d' « intelligence technique » afin d'intégrer, en un système de conduite, de maintenance et de gestion technique (CMMS), des systèmes d'actionnement et de mesure intelligents

(IAMS) autour d'un bus de terrain (Iung, 1992; Pétin J.-F. , 1995; Neunreuther, 1998; Leger, 1999; Liu, 2002).

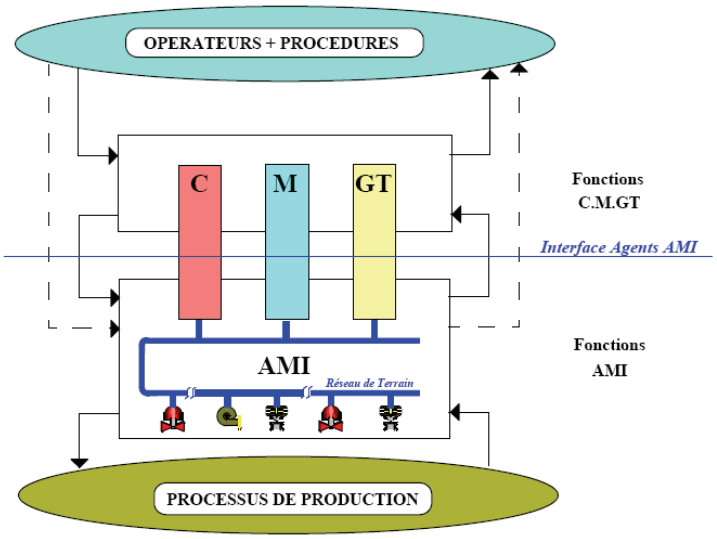


Figure 3 : Système intégré de contrôle, maintenance et gestion technique (CMMS) basé sur des systèmes d'actionnement et de mesure intelligents (IAMS) (Pétin J.-F. , 1995)

Cependant, ces travaux se sont limités à l'interactivité entre les opérateurs et les équipements instrumentés par l'intermédiaire du CMMS sans prendre en compte la boucle de contrôle par l'humain (Figure 3 : flèches en pointillés) qui est une caractéristique d'exploitation de ce type d'installation. Nos travaux combinent cette « intelligence technique » distribuée au plus près du procédé (Figure 4c) avec l'intelligence cognitive de l'opérateur humain (Figure 4b) en proposant

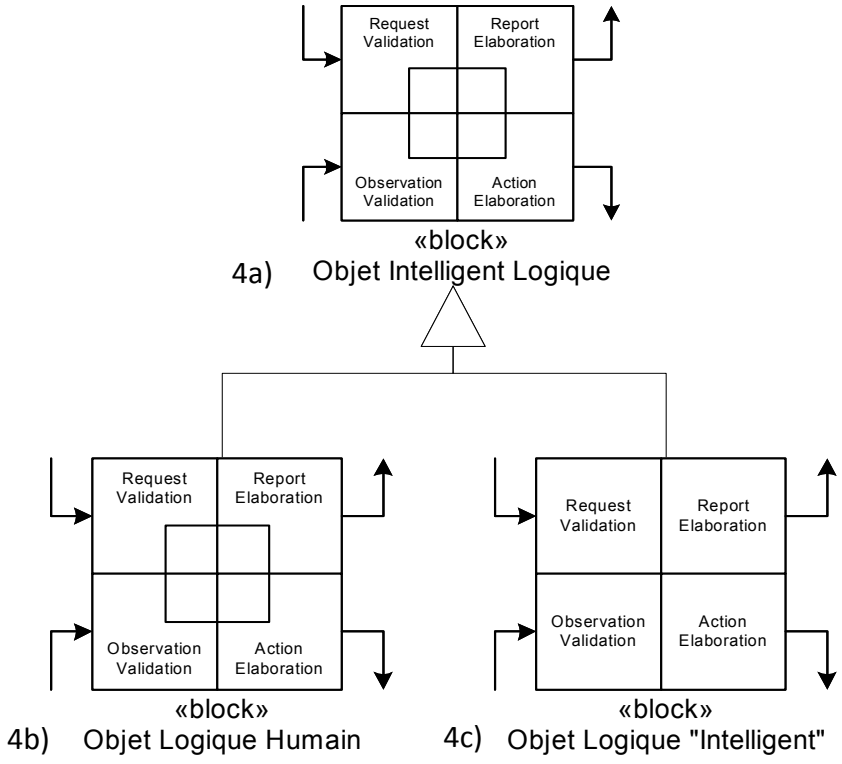


Figure 4 : L'humain composant logique d'un système intelligent d'actionnement et de mesure

un patron architectural (Figure 4a) permettant, en conception, d'allouer un composant technique ou humain pour exécuter une fonction d'exploitation. Ils s'appuient sur ces visions complémentaires de la conduite d'un système industriel par des agents d'exploitation, en se limitant cependant à l'interopération entre le Rondier et le Système Physique à consigner (Figure 3 : flèches en pointillés) afin d'augmenter leur capacités numériques d'interaction.

Notre proposition porte sur un concept d'architecture (Figure 5) de Système Interactif d'Aide à la Conduite (SIAC) qui encapsule le canal physique (de flux matière/énergie) par un canal d'informations articulant trois types d'objets logiques :

«requirement»
«architecturalRequirement»
Système Interactif d'Aide à la Conduite

id#

These 0

txt

Le rondier doit être aidé numériquement pour interopérer avec le système technique de soutien.

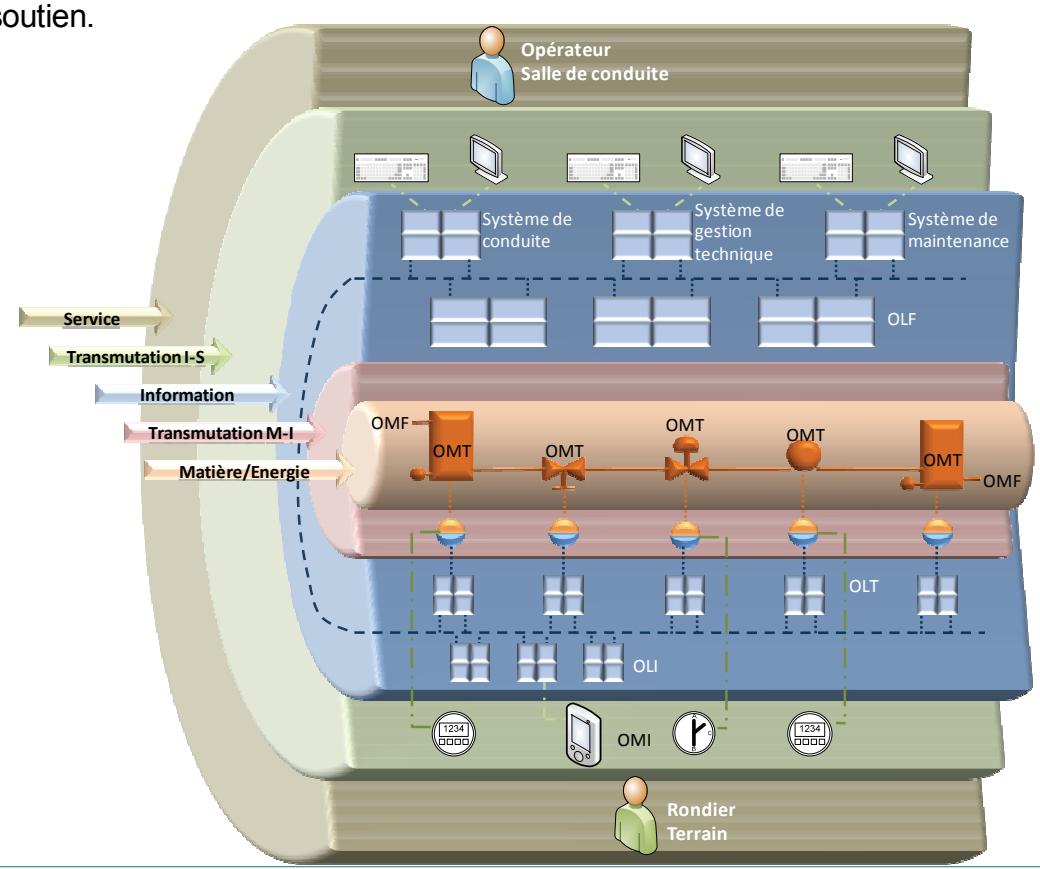


Figure 5 : Exigence initiale (concept) du Système Interactif d'Aide à la Conduite

- techniques et physiques assurant la transmutation[2] entre le canal physique et celui de l'information,

[2] La transmutation assure la transformation de nature

- interactifs, assurant la transmutation entre le canal d'informations et celui des services.

Ce continuum d'informations a pour objectif de permettre au rondier, lui-même composé par un système cognitif (OHC) et un système sensori-moteur (OMT), d'interopérer numériquement en tout lieu, à tout instant et pour toute action de conduite via les objets interactifs composés (Figure 6) :

- d'un Objet Logique Interactif (OLI) qui fournit au rondier des services d'assistance au cours de ces activités,
- d'un Objet Matériel Interactif (OMI) qui fournit l'interface Homme-Système.

Par ailleurs, ces objets logiques interactifs (OLI) interagissent :

- soit avec les Objets Logiques Techniques (OLT), qui reflètent le comportement des Objets Matériels Techniques (OMT) et assurent le filtrage des actions du rondier par rapport à la situation d'état du processus à consigner, en embarquant éventuellement une forme d'intelligence technique,
- soit avec les Objets Logiques de Flux (OLF), qui représentent le comportement des Objets Matériels du Flux physique (OMF) et assurent le filtrage des actions du rondier par rapport à l'état du procédé, mais en se limitant cependant aux modes opératoires.

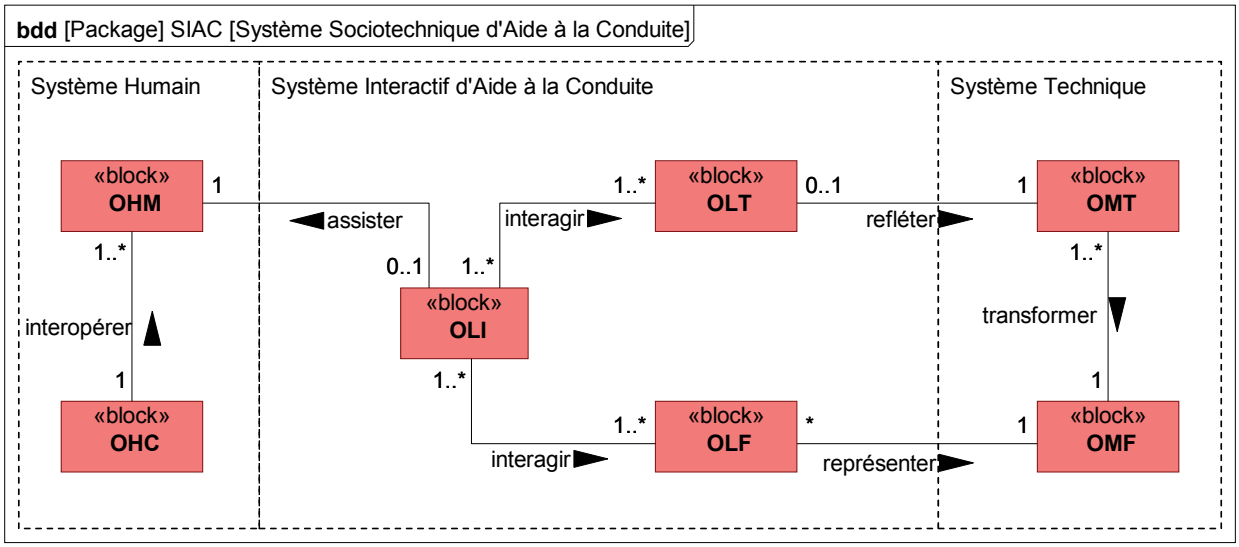


Figure 6 : Objets Logiques du Système Interactif d'Aide à la Conduite

La preuve de ce concept de Système à Faire[3] est présentée dans les chapitres 3 et 4 de la deuxième partie de ce mémoire, d'abord en spécifiant les exigences techniques de ce Système Interactif d'Aide à la Conduite à partir des besoins d'exploitation de ce type d'installation industrielle, puis en développant une solution technique sur la plateforme CISPI (Conduite Interactive et Sûre de Procédés Industriels) du projet SAFETECH du CRAN.

[3] Le « système à faire » est initialement un concept, une abstraction, le résultat concret de sa réalisation est un « produit » (produit-système) ayant les caractéristiques d'un système et répondant à une définition qui en précise toutes les caractéristiques techniques, de production, d'exploitation, de maintenance, … (AFIS, 2009)

De façon plus générale, ce contexte de Recherche & Développement est en phase avec les recommandations du groupe d'experts du CNRS :

> Baptiste, et al. (2007) : « *Un deuxième défi scientifique est de rendre interactif le comportement de l'opérateur humain dans la boucle cybernétique de conduite d'un procédé en examinant les possibilités de le doter de capacités d'interaction numérique avec les objets du système dans leur environnement ambiant de travail. Cette interactivité numérique accrue vise à mieux distribuer la prise de décision entre opérateurs technologiques et humains en tirant parti des compétences de ces derniers pour pallier les difficultés d'observabilité de certains phénomènes, voire de commandabilité de certains équipements, y compris dans des situations critiques. De façon alternative, il s'agit aussi d'étudier les limites techniques et cognitives de ces technologies afin de définir de façon plus pertinente les parties du procédé à automatiser pour éviter que cette numérisation des interactions entre opérateurs d'un système ne finisse par compliquer son pilotage et par pénaliser sa performance, voire la sûreté de son fonctionnement.* »

La nature de ces interactions homme-système pose clairement la nécessité d'un procédé de modélisation systémique qui les prenne en compte de façon concourante comme composante d'un même système. Nous avons interprété les travaux de Hall et Rapanotti (2005) pour formuler le prédicat à satisfaire dans un procédé de spécification d'un système sociotechnique selon :

$$W \wedge S \wedge I \wedge UI \Longrightarrow R$$

dans lequel :

- W décrit le contexte environnant du SIAC,
- R décrit les besoins exprimés du SIAC,
- S prescrit les exigences techniques du SIAC,
- UI prescrit les exigences techniques de l'interface interactive du SIAC,
- I prescrit les exigences du comportement requis pour le rondier afin de conduire le système technique avec le SIAC.

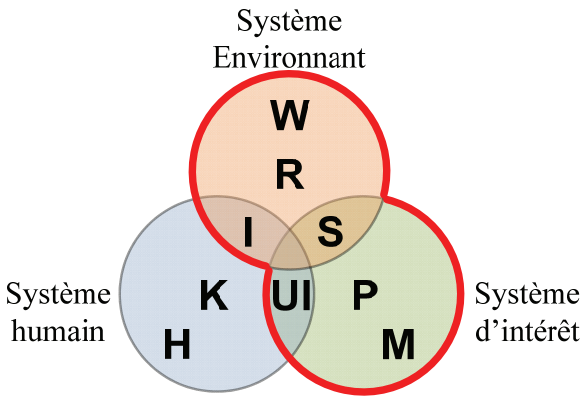


Figure 7 : Modèle de référence pour l'analyse des exigences d'un système sociotechnique (Hall, et al., 2005)

Ces diverses spécifications S, UI et I sont ensuite raffinées de façon itérative et allouées à des constituants système pour concevoir une solution technique selon :

$$M \wedge P \Longrightarrow S \wedge I \wedge UI$$

dans lequel :

- M représente la description organique du SIAC,
- P représente la description comportementale du SIAC.

Ce modèle de référence de Hall et Rapanotti (2005) est une extension au contexte sociotechnique des travaux menés antérieurement par Jackson (Gunter, et al., 2000) pour la spécification des systèmes informatiques. Notons cependant que les capacités K requises par l'humain H pour l'exécution interactive des procédures de conduite ne font pas partie de notre étude mais sont abordées dans le cadre de la physiologie intégrative dans les travaux de Fass et Lieber (2009).

Cette formulation illustre la nature collaborative du procédé de modélisation système puisque divers domaines métiers sont impliqués pour chacune des Parties du Système à Faire comme un Tout. Dans notre cas, les domaines métiers auxquels nous nous sommes intéressés sont :

- $M_a \wedge P_a \Longrightarrow S_a$, en ingénierie d'automatisation,
- $M_i \wedge P_i \Longrightarrow UI_i$, en ingénierie d'informatique,
- $M_c \wedge P_c \Longrightarrow I_c$, en ingénierie de conduite.

Une interprétation complémentaire que nous faisons des travaux de Jackson (Jackson, 1997) est de considérer que la partie optative d'une exigence R qui décrit le besoin attendu d'un domaine de problème doit être complétée par une partie indicative R' qui décrit les connaissances apportées par un domaine de solution pour spécifier une exigence système.

Cette approche des « Problem Frames » nous a permis de faire une analogie avec certains artéfacts essentiels du procédé d'Ingénierie Système (IS) dans lequel nous positionnons nos travaux de façon plus générale. Notamment, nous proposons de faire du changement d'état de l'exigence le « moteur » du raisonnement du procédé itératif et collaboratif en IS.

Le langage de modélisation système SysML, spécifique au domaine de l'IS, ne s'est pas avéré suffisamment mature et complet dans un premier temps pour supporter formellement ce procédé de spécification de notre application (Figure 6). Il nous a fallu le spécialiser comme langage de spécification au niveau :

- Syntaxique et sémantique, par l'ajout de stéréotypes à son méta-modèle, comme par exemple « userNeed », et de patrons de conception, comme par exemple le patron de séquence (« preparation », « realization », « closing ») et le filtre de comportement ;

- Procédé, par une démarche itérative de spécification qui passe par des activités de définition des besoins, de prescription des exigences, de conception de l'architecture fonctionnelle et de conception de l'architecture organique ;
- Preuve, par la vérification et la validation de propriétés pertinentes à l'aide de la simulation ou du model checking.

L'ensemble de ces résultats relatifs au Système pour Faire[4] est présentée dans les chapitres 1 et 2 de la première partie de ce mémoire.

La construction du mémoire suit ces préceptes d'ingénierie (Figure 8).

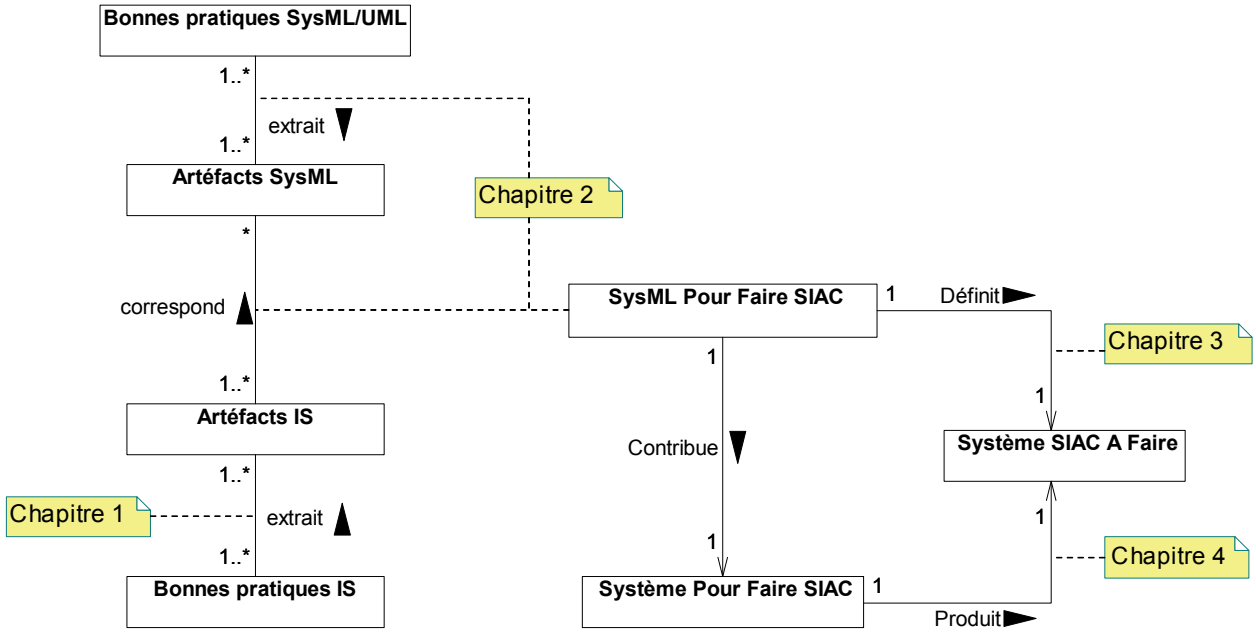


Figure 8 : Architecture du mémoire

En conclusion, nous montrons le caractère préliminaire de nos travaux sur la base des résultats acquis et ouvrons sur diverses perspectives tant pour le Système à Faire, notamment le passage à l'échelle du SIAC proposé, que pour le Système pour Faire, notamment le passage à l'échelle d'un procédé de spécification Système basé sur le changement d'état de l'exigence.

Des annexes complètent ce mémoire pour décrire les exigences initiales de notre démonstrateur CISPI à partir desquelles nous avons défini, développé et déployé notre solution ainsi que pour décrire les modes opératoires de son exploitation.

[4] Le « système pour faire » met en place les ressources humaines et techniques pour organiser, exécuter et coordonner toutes les activités qui conduisent de l'énoncé de la finalité à la réalisation et à la mise à disposition du système à faire (ou système étudié). Ce système est organisé sous forme d'un ou plusieurs projets. (AFIS, 2009)

PARTIE I :
Système pour Faire

Chapitre 1
Artéfacts de modélisation système

Ce chapitre a pour but de construire l'armature conceptuelle des artéfacts[5] - objets et « rationales[6] » - essentiels de modélisation de notre Système SIAC à Faire sur la base des bonnes pratiques de l'Ingénierie Système (IS) ainsi que sur d'autres approches que nous avons jugées nécessaires pour compléter ce corpus.

1.1 Artéfacts de l'Ingénierie Système

Cette synthèse s'appuie dans un premier temps sur un **Référentiel** de documents sur lequel la communauté IS s'accorde en réponse à la nécessité de proposer un cadre normatif de modélisation des systèmes de grandes dimensions. Nous illustrons d'abord la difficulté de s'accorder formellement sur nombre d'artéfacts, comme par exemple celui-clé de **Système**. Cela infère des perceptions multiples de l'**Ingénierie d'un Système** justifiant que des travaux de standardisation unifient les artéfacts IS mais aussi que des travaux scientifiques formalisent petit à petit les fondements de cette multidiscipline IS qui n'a pas la maturité de disciplines conventionnelles. C'est la raison pour laquelle nous ne reprenons pas systématiquement les définitions par trop descriptives de ce référentiel pour chacun des artéfacts. Nous raffinons cette synthèse dans un deuxième temps sur la base de travaux récents qui nous fournissent plus formellement des **méta-modèles d'artéfacts IS** à analyser sans nous attacher à ceux portant sur l'unification des échanges de données entre outils-méthodes du procédé d'IS (par exemple AP233[7] (ISO 10303-233)).

En effet, l'objectif n'est pas de décrire ces artéfacts IS mais plutôt de dégager certains **artéfacts-clé** pour comprendre leurs relations afin de définir le procédé de modélisation que nous proposons.

---

[5] The sense of artifacts as tangible byproducts is similar to the use of the term artifact in science to refer to something that arises from the process in hand rather than the issue itself, i.e., a result of interest that stems from the means rather than the end. (Wikipedia)

[6] A design rationale is the explicit listing of decisions made during a design process, and the reasons why those decisions were made. Its primary goal is to support designers by providing a means to record and communicate the argumentation and reasoning behind the design process. (Wikipedia)

[7] Application Protocol 233: www.ap233.org

1.1.1 A propos de l'Ingénierie de Système

Référentiel d'Ingénierie Système

Le Put et Meinadier (2009) présentent, dans leur tutoriel de la 5ème conférence annuelle d'Ingénierie Système, les raisons et les choix d'une base conceptuelle de l'IS reconnue par l'AFIS[8] pour travailler aussi bien en approfondissement que pour servir d'aide à l'organisation de la connaissance. Notons, dans le même sens, que le projet BKCASE[9] en cours vise à fonder à l'international le référentiel des artéfacts de l'IS. Nous retenons, en l'état, ces constituants d'un référentiel IS (Définition 1.1).

Définition 1.1 : Référentiel d'Ingénierie Système

Trois ouvrages se concentrant sur différents aspects de la connaissance :

- « NASA[10] SE Handbook » (NASA, 2007), faisant référence à NPR-7123.1A (2009), plus sur « Quoi Faire » et « Comment Faire »,
- « Découvrir et comprendre l'IS » (AFIS, 2009) plus sur « Pourquoi Faire » et « Comment Faire » pour mieux comprendre les concepts et les approches de l'IS,
- « INCOSE[11] SE Handbook » (INCOSE, 2010) plus sur « Quoi Faire » que sur « Comment Faire ».

Trois normes couvrant plus ou moins le cycle de vie d'un système (Figure 1.1) :

- IEEE 1220 (2005) : « *Standard for application and Management of the Systems Engineering Process* »,
- EIA–632 (1999) : « *Processes for Engineering a System* »,
- ISO/IEC 15288 (2008) : « *Systems and software engineering - System life cycle processes* ».

Système

Les définitions d'un **Système** que nous prenons en considération sont les suivantes :

Définition 1.2 : Système (*en*. System, NASA (2007))

A « system » is a construct or collection of different elements that together produce results not obtainable by the elements alone. The elements, or parts, can include people, hardware, software, facilities, policies, and documents; that is, all things required to produce system-level results. The results include system-level qualities, properties, characteristics, functions, behavior, and performance. The value added by the system as a whole, beyond that contributed independently by the parts, is primarily created by the relationship among the parts; that is, how they are interconnected.

[8] Association Française d'Ingénierie Système : www.afis.fr
[9] Body of Knowledge and Curriculum to Advance Systems Engineering : www.bkcase.org
[10] National Aeronautics and Space Administration : www.nasa.gov
[11] International Council on Systems Engineering : www.incose.org

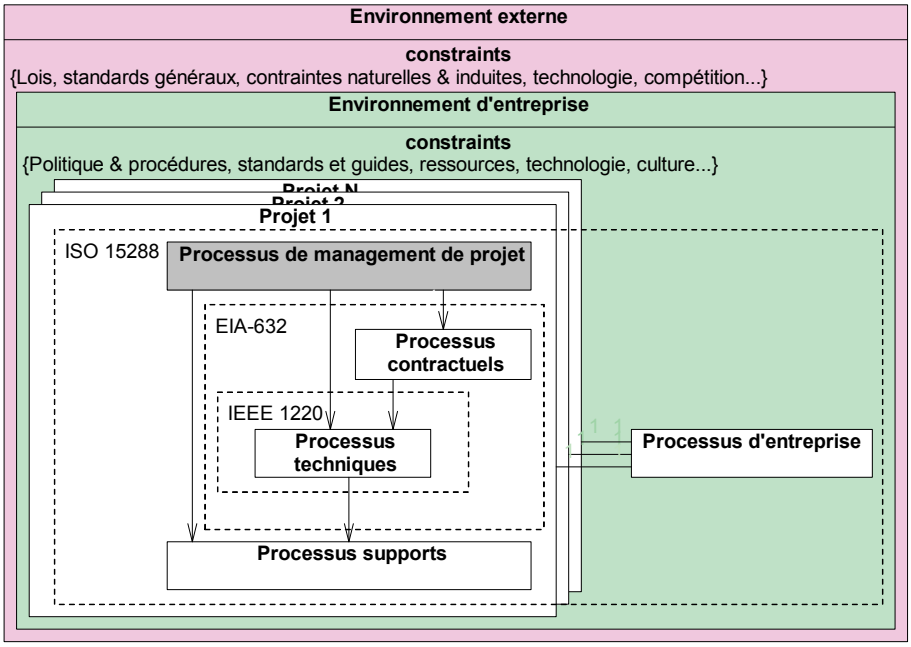


Figure 1.1 : Couverture des types de processus et du cycle de vie d'un système par les normes IS

Définition 1.3 : Système (AFIS, 2009)

Construire ou utiliser un objet technique complexe fait appel à la notion de système. Cette notion, ancienne dans les sciences physiques et humaines, est maintenant courante dans les pratiques industrielles et opérationnelles. Par définition, tout système est constitué d'un ensemble d'éléments dont la synergie est organisée pour répondre à une finalité dans un environnement donné.

Définition 1.4 : Système (*en*. System, INCOSE (2010))

The systems are man-made, created and utilized to provide services in defined environments for the benefit of users and other stakeholders. These systems may be configured with one or more of the following: hardware, software, humans, processes (e.g., review process), procedures (e.g., operator instructions), facilities, and naturally occurring entities (e.g., water, organisms, minerals). In practice, they are thought of as products or services. The perception and definition of a particular system, its architecture and its system elements depend on an observer's interests and responsibilities. One person's system-of-interest can be viewed as a system element in another person's system-of-interest. Conversely, it can be viewed as being part of the environment of operation for another person's system-of-interest.

Ces diverses définitions, extraites de ce référentiel, sur lesquelles chacun peut s'accorder en raison de leur caractère descriptif, voire évasif, montrent, de façon non exhaustive, la difficulté d'unifier ces visions de la définition d'un système complexe. Elles ne sont pas satisfaisantes pour fonder scientifiquement cet **artéfact-clé IS**. Feliot (2007) pose les bases mathématiques d'une sémantique formelle de la vision système en termes de transformateur de prédicat qui aboutit à l'identification de trois formes canoniques de spécification interprétant formellement la grille d'analyse systémique proposé par Penalva (1997). L'objectif à terme est de proposer une articulation de logiques de construction système par niveau d'abstraction. En ce sens, Luzeaux (2009) pose également les bases d'une théorie mathématique des systèmes qui vise à

définir des cadres logiques de modélisation sur des catégories de systèmes en conservant les sources structuralistes de la systémique. L'intérêt de ce cadre unificateur à terme est aussi d'expliquer *a posteriori* l'intérêt de certains artéfacts proposés intuitivement en IS. Il en est de même en systémique (Lavigne, et al., 2003) pour expliquer plus formellement la construction relationnelle d'un système (comme une molécule) à partir d'une relation élémentaire (atomique entre Objet Finalisant - Environnement) proposée par Mayer (1995) (Figure 1.2).

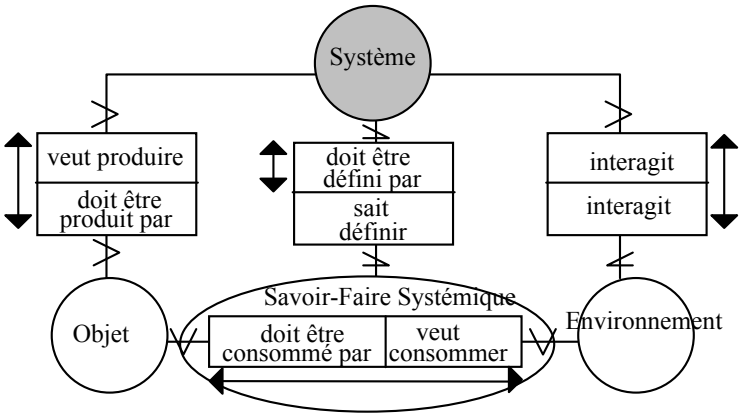


Figure 1.2 : Méta-modèle élémentaire de construction Système par « substantivation » en formalisme NIAM/ORM (Mayer, 1995)

Cette vision relationnelle de la construction d'un système se retrouve aussi dans le cadre de modélisation proposé par Kuras (2006) qui en souligne la dimension multi échelle à partir de toutes les conceptualisations possibles, notamment celle d'Holon (Koestler, 1967) comme archétype TOUT et PARTIE d'un système (Figure 1.3).

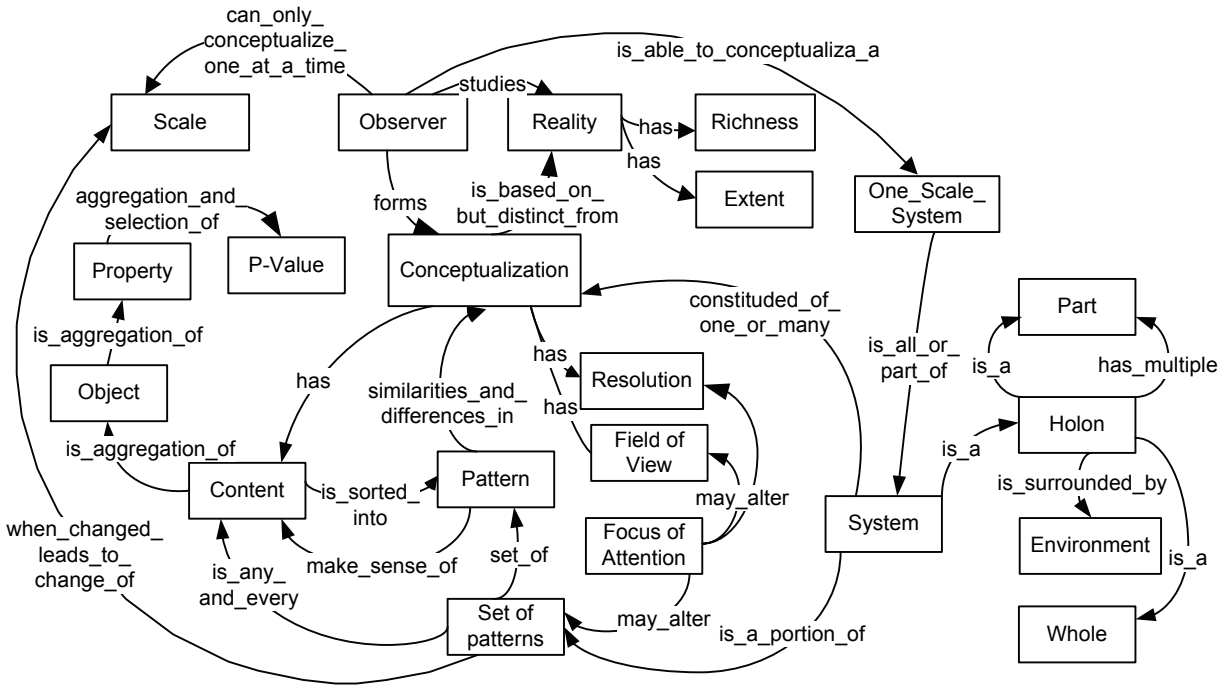


Figure 1.3 : Méta-modèle du cadre de modélisation multi-échelle d'un système (Bjelkemyr, et al., 2007)

Ce cadre permet notamment de comprendre en quoi la construction d'un système à un niveau s'imbrique logiquement avec la construction du même système à un autre niveau d'observation. Il fournit aussi un ensemble d'artéfacts de modélisation qu'Auzelle (2009) applique partiellement en Ingénierie de Systèmes d'Informations techniques de grandes dimensions dans une perspective Système de Systèmes (Luzeaux, et al., 2008).

Nos travaux se placent dans cette vision relationnelle de construction d'un système guidée par certains artéfacts clés d'IS à partir d'une relation initiale élémentaire entre les services attendus par le système en exploitation.

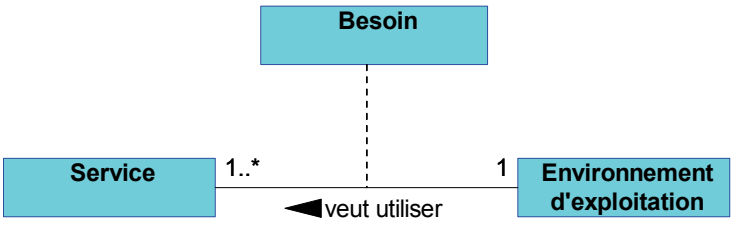


Figure 1.4 : Relation initiale entre les services (fournis par SIAC) et l'environnement d'exploitation

Ingénierie Système

Définition 1.5 : Ingénierie Système (*en*. Systems Engineering, NASA (2007))
Systems engineering is the art and science of developing an operable system capable of meeting requirements within often opposed constraints. Systems engineering is a holistic, integrative discipline, wherein the contributions of structural engineers, electrical engineers, mechanism designers, power engineers, human factors engineers, and many more disciplines are evaluated and balanced, one against another, to produce a coherent whole that is not dominated by the perspective of a single discipline.

Définition 1.6 : Ingénierie Système (AFIS, 2009)
L'Ingénierie Système est une démarche méthodologique coopérative et interdisciplinaire qui englobe l'ensemble des activités adéquates pour concevoir, développer, faire évoluer et vérifier un ensemble de produits, processus et compétences humaines apportant une solution économique et performante aux besoins des parties prenantes et acceptable par tous. Cet ensemble est intégré en un système, dans un contexte de recherche d'équilibre et d'optimisation sur tout son cycle de vie.

Définition 1.7 : Ingénierie Système (*en*. Systems Engineering, INCOSE (2010))
Systems engineering is an interdisciplinary approach and means to enable the realization of successful systems.

Ces diverses définitions, extraites du référentiel, ne font que traduire la vision multi échelle d'un système de Kuras (2006) en mettant en évidence le nécessaire interfaçage entre l'ingénierie du Métier d'Architecte Système et les Ingénieries des Métiers de développement des Composants-Système ainsi que des domaines d'Applications (Figure 1.5). C'est cette relation d'interopération notée S (Spécification) que nous mettons en correspondance par rapport aux artéfacts de l'IS que nous étudions.

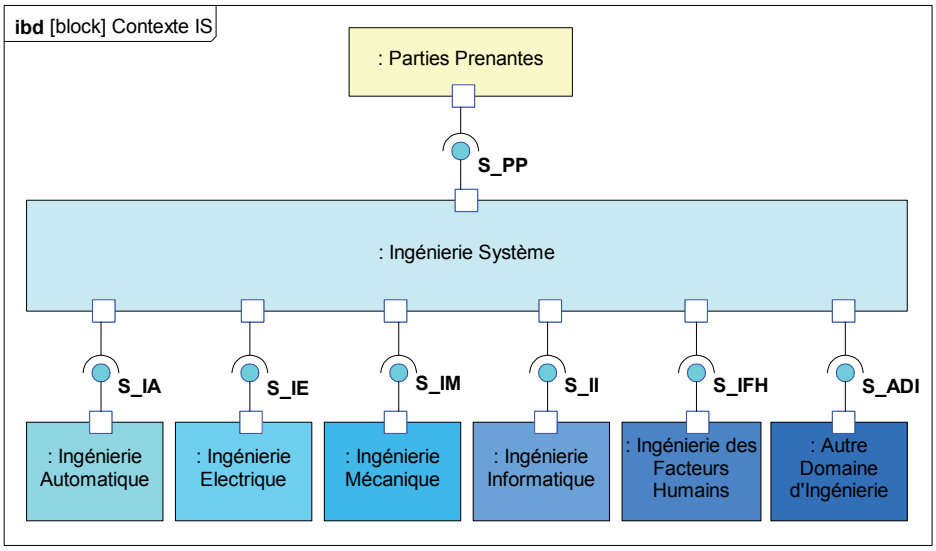


Figure 1.5 : Relations d'interopération S entre Parties Prenantes, Ingénieur Système et Ingénieurs Métiers

Notons, que pour notre étude de cas, nous nous sommes intéressés aux domaines de l'IS, de l'ingénierie d'automatisation, de l'ingénierie informatique et en partie de l'ingénierie du procédé.

Méta-Modèles d'Ingénierie Système

Le choix d'un modèle de processus étant primordial pour en formaliser les artéfacts, nous raffinons cette première synthèse en analysant trois méta-modèles qui se basent sur la même norme ISO/IEC 15288 (2008). Nos travaux se limitent à certains processus opératoires de l'IS (Tableau 1.1 : processus en italique) couvrant en partie les processus techniques de développement d'un système dans cette norme.

Tableau 1.1 : Les processus de la norme ISO/IEC 15288 (2008)

Processus d'entreprise	Processus du projet	Processus techniques
Processus de management de l'environnement de l'entreprise	Processus de planification du projet	*Processus de définition des besoins des parties prenantes*
Processus de management de l'investissement	Processus d'évaluation du projet	*Processus d'analyse des exigences*
Processus de management des processus du cycle de vie système	Processus de pilotage du projet	*Processus de conception de l'architecture*
Processus de management des ressources	Processus de décision	*Processus d'implémentation*
Processus de management de la qualité	Processus de management des risques	*Processus d'intégration*
Processus contractuels	Processus de gestion de configuration	*Processus de vérification*
Processus d'acquisition	Processus de management de l'information	Processus de transition
Processus de fourniture		*Processus de validation*
		Processus d'exploitation
		Processus de maintenance
		Processus de retrait de service

Les travaux de l'AFIS ont exigé un important consensus sémantique de définition des artéfacts IS entre différents groupes de travail, qui ne représentaient qu'une fraction des multiples rela-

tions clients/fournisseurs impliquées dans un processus industriel d'IS. Un effort important de méta-modélisation (Caron, 2005) a aussi été accompli pour ne pas opposer le paradigme systémique (fonctionnel) couramment adopté en maitrises d'ouvrage et d'œuvre IS (Figure 1.6) au paradigme objet de plus en plus adopté en maitrise d'œuvre et en réalisation informatique, dans le sens du consensus naissant d'un langage unifié de modélisation système (SysML).

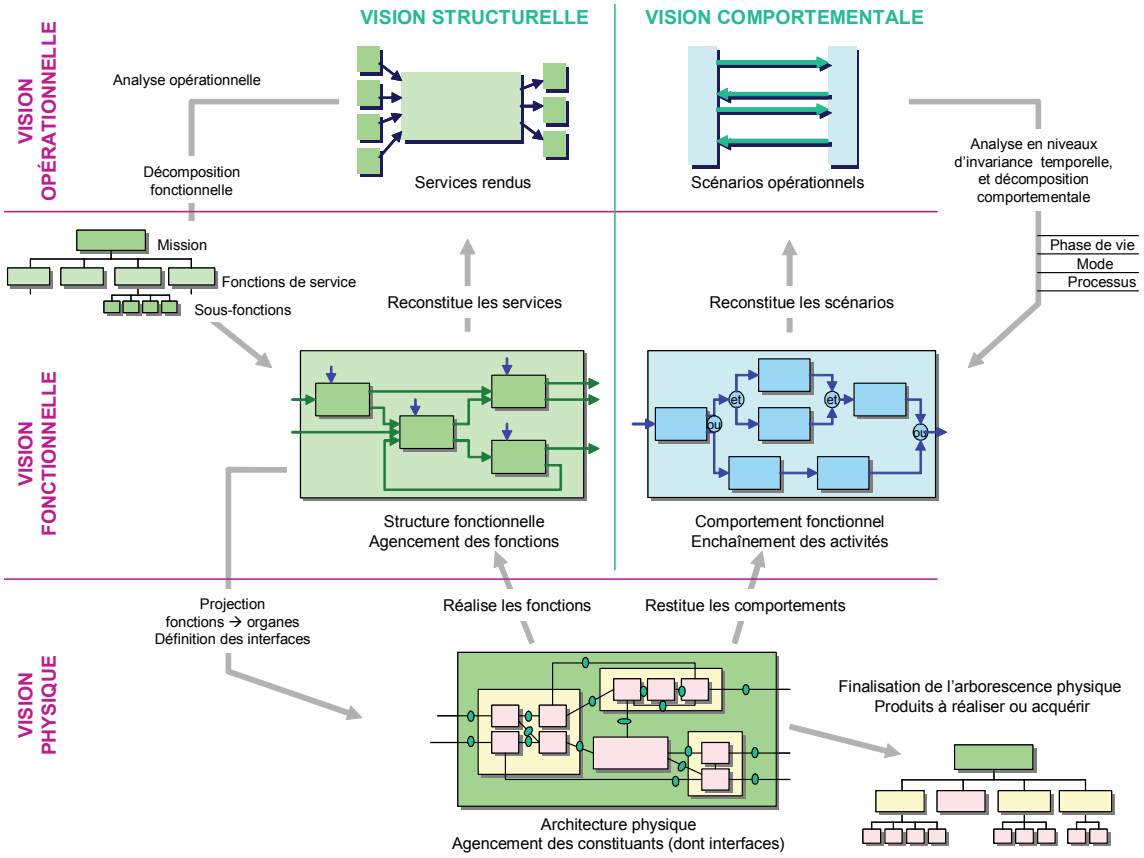


Figure 1.6 : Modélisation systémique traditionnelle (AFIS, 2009)

Le méta-modèle des artéfacts d'IS de l'AFIS (Figure 1.7) interprète principalement les travaux de groupes de travail « Méthodes et Outils » et « Ingénierie Système » qui ont conduit respectivement à la proposition d'un Modèle de Données d'IS (AFIS, 2005) et à la rédaction de diverses ébauches de ce qui deviendra « Découvrir et Comprendre l'IS » (AFIS, 2009) sur la base de travaux antérieurs de Jean-Pierre Meinadier (Meinadier, 1998; Meinadier, 2002).

La structuration de ces artéfacts en cinq vues place celle de l'Ingénierie des Exigences au cœur du procédé de modélisation pour relier la vue des affaires avec les vues de l'ingénierie et de l'intégration système conduisant à une solution architecturale validée tout en assurant la gestion de la traçabilité des données sur le cycle de vie du système.

Les travaux de MAP Système (Faisandier A. , 2010) ont donné naissance à un deuxième méta-modèle générique d'IS (Figure 1.8), structuré autour trois vues : besoins/exigences, architecture fonctionnelle et scénarios, et architecture organique. Chaque vue comporte des entités manipulées dans un processus d'ingénierie ; les entités sont liées entre elles par des relations afin d'établir la cohérence et la traçabilité de bout en bout du besoin à la solution et inversement.

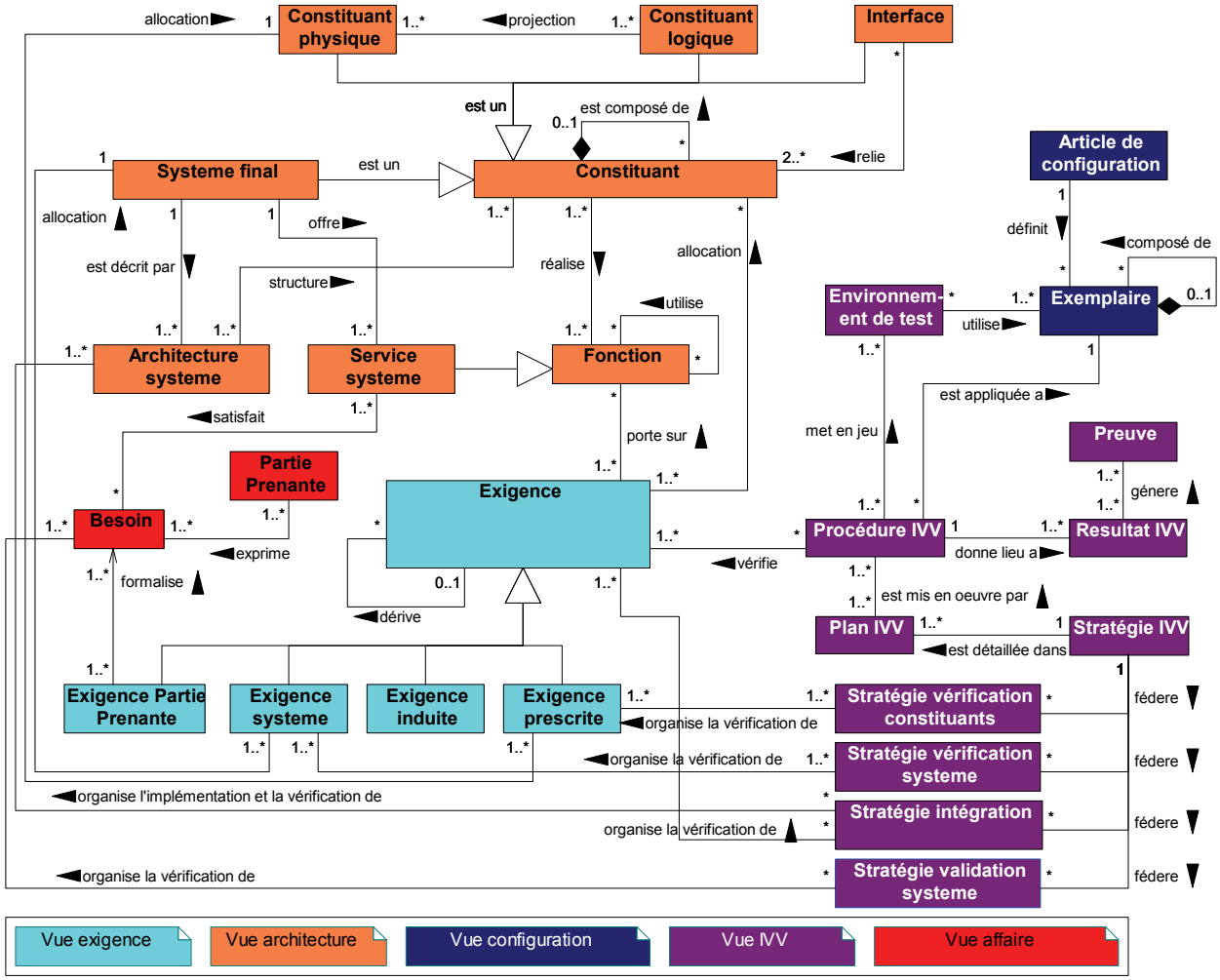


Figure 1.7 : Vue partielle du modèle de données de l'Ingénierie Système proposé par AFIS (2005)

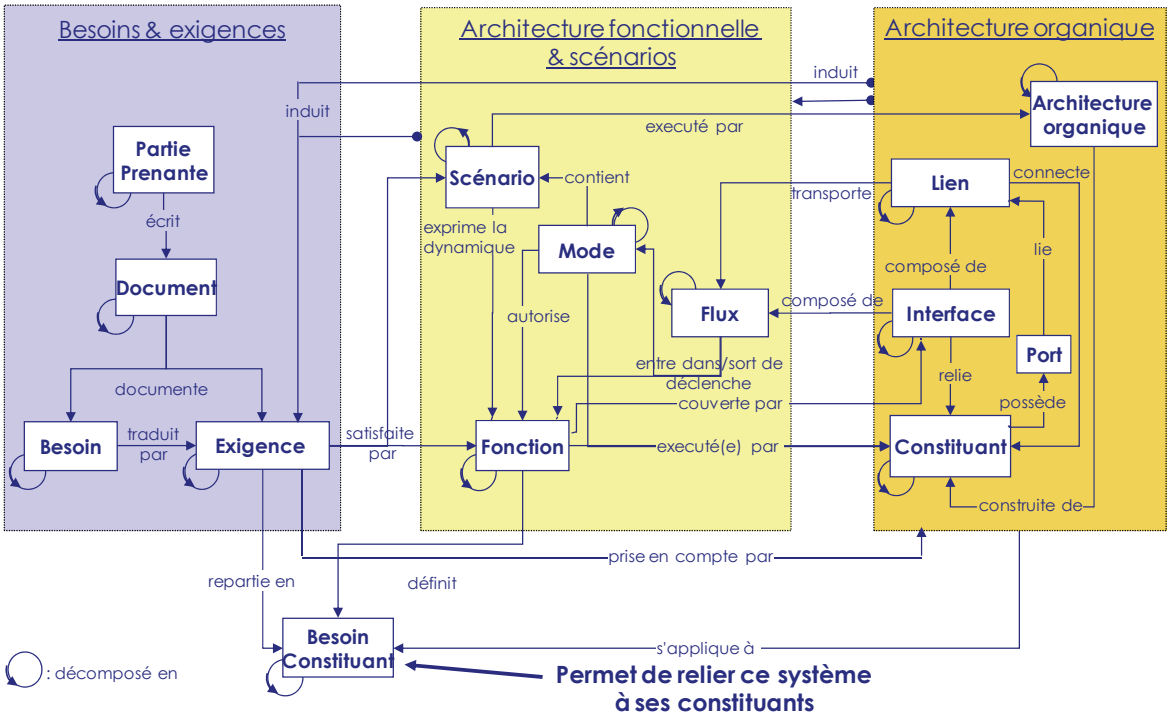


Figure 1.8 : Méta-modèle générique de l'Ingénierie Système d'après Faisandier (2008)

Ces travaux ont permis le développement par SODIUS[12] de l'outil de modélisation MDWorkbench[13]. MDWorkbench est un environnement de développement intégré basé sur Eclipse qui permet la conception et la manipulation de modèles, leur transformation ainsi que la génération de code et de documentation. Il existe en différentes « versions », adaptées à des domaines spécifiques. Cette adaptation est réalisée par la spécialisation du méta-modèle au domaine spécifique. Par exemple, dans le cas de MDWorkbench for Defense, le méta-modèle est adapté à la modélisation de systèmes de défense (il supporte notamment la génération de modèles DoDAF).

Les travaux de Holt et Perry (2008) nous fournissent un troisième méta-modèle d'artéfacts d'IS qui combine « INCOSE SE Handbook » (INCOSE, 2010)[14] et la norme ISO/IEC 15288 (2008) en structurant les objets IS en quatre groupes (Figure 1.9) : processus, cycles de vies, projets et systèmes. Contrairement aux autres méta-modèles qui mettent plus l'accent sur le Système, ils mettent au préalable l'accent sur les besoins indiscutables en Ingénierie d'un Système vu comme un TOUT en regard des ingénieries n'en considérant qu'une PARTIE. Mais ils soulignent aussi le manque de fondements de la multidiscipline d'IS auxquels ils remédient en positionnant le langage de modélisation système SysML (et UML) en regard des artéfacts de ce cadre normatif d'IS.

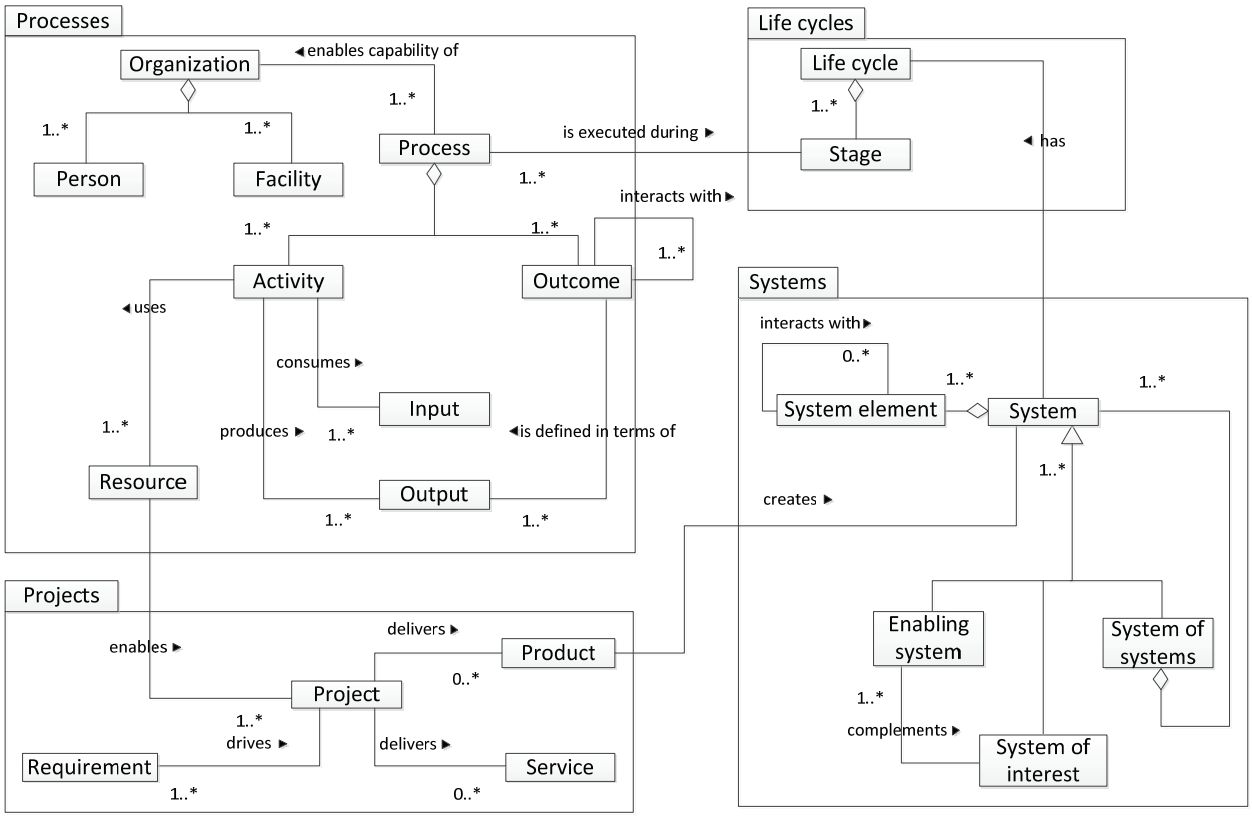


Figure 1.9 : Méta-modèle générique de l'Ingénierie Système par Holt et Perry (2008)

[12] SODIUS : www.sodius.com
[13] MDWorkbench : www.mdworkbench.com
[14] Nous citons ici la dernière version du « INCOSE SE Handbook »

En ce sens, la démarche suivie dans les deux premiers chapitres de ce mémoire est analogue puisqu'elle vise à méta-modéliser les artéfacts essentiels d'un procédé d'IS avant de les mettre en correspondance avec ceux qu'offre le langage SysML pour en faire ensuite une spécialisation à notre propre procédé de modélisation.

1.1.2 Artéfacts-clés de la relation d'interopération *S*

Nous nous appuyons sur ce référentiel et ces méta-modèles pour définir plus précisément ce que nous considérons comme artéfacts-clés d'IS pour supporter la relation itérative S entre les différents métiers d'ingénierie dans notre cas d'étude.

Processus

L'objectif est de guider la construction collaborative et itérative d'un système par une organisation systématique des « Savoir-Faire » multiples comme s'y attache l'IS en structurant et standardisant les « bonnes pratiques » industrielles.

Définition 1.8 : Processus (*en*. Process, INCOSE (2010))
Set of interrelated or interacting activities which transform inputs into outputs.

La difficulté est de délimiter la part d'expertise qui relève du seul métier d'architecte de celles qui relèvent des métiers d'application dans le cycle de développement d'un même système (Figure 1.10).

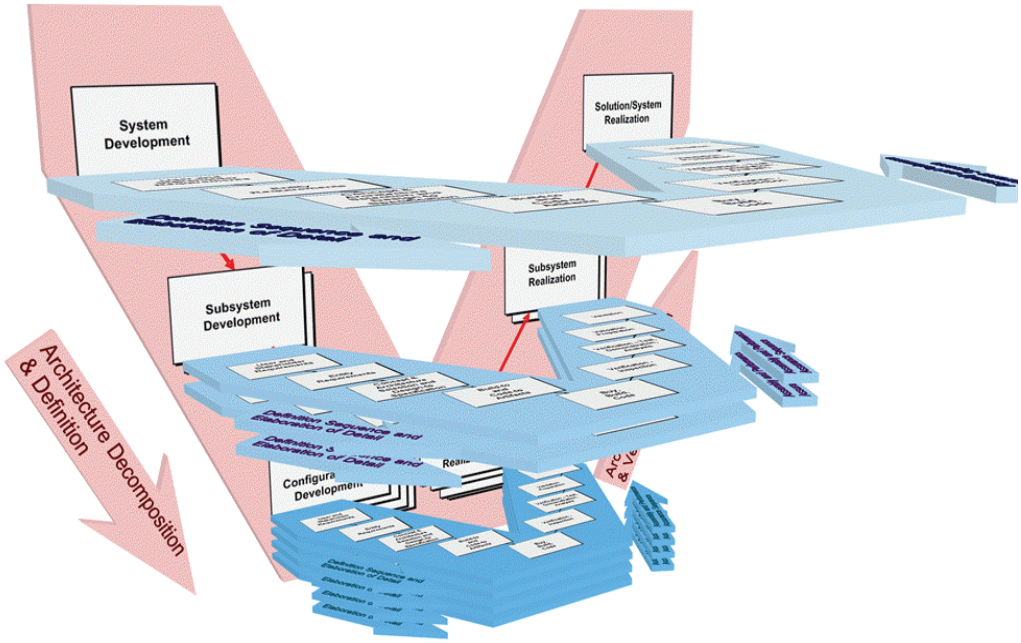


Figure 1.10 : Articulation des cycles en V d'un système et de ses sous-systèmes (Forsberg, et al., 2005) : le V vertical concerne le cycle de développement du système alors que les V horisontaux concernent les métiers associées aux differents niveaux de décomposition du système

La limite est que la seule définition de processus-clé d'IS et des artéfacts associés n'est pas suffisante si les règles de construction d'un système pour enchainer ces processus ne sont pas elles-mêmes méta-modélisées (Figure 1.11).

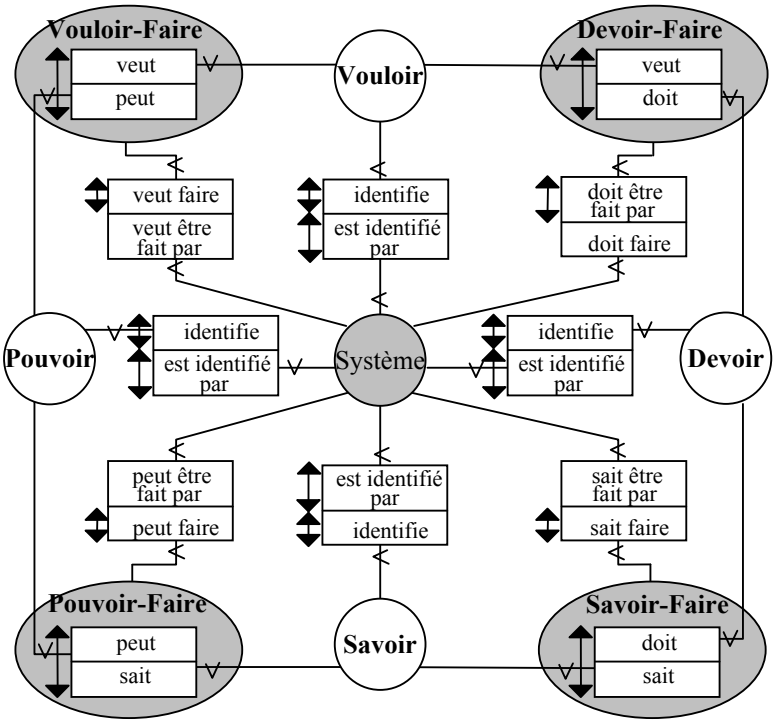


Figure 1.11 : Modalités DF / SF / PF / VF de Construction Système (formalisme NIAM/ORM par Mayer (1995))

La solution proposée par les normes est de formaliser sous forme de processus, notamment techniques (Figure 1.12), des invariants d'IS permettant de paralléliser des activités opératoires

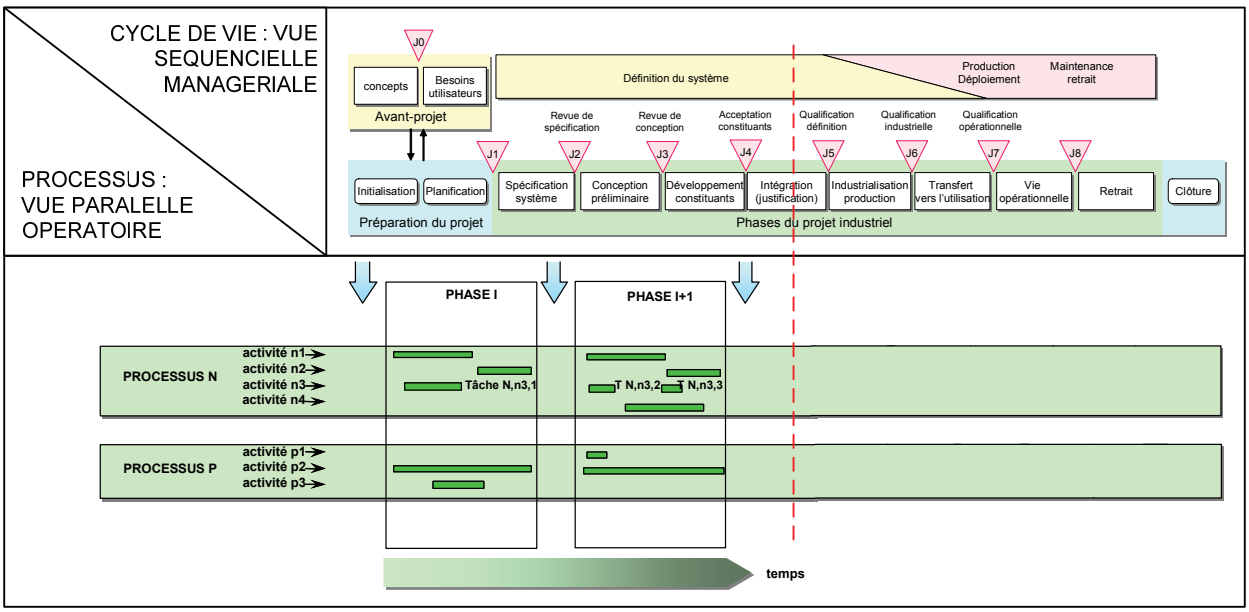


Figure 1.12 : Complémentarité entre cycle de vie du projet et processus opératoire IS (limité dans le cas du développement d'un système unique)

d'ingénierie répétitives et génériques dans le cadre plus séquentiel des activités de management du projet quelque soit le cycle de vie du système.

L'intérêt est que ces processus s'enchainent logiquement en articulant une démarche descendante avec une démarche ascendante qui assure de ne prescrire que ce qui peut être construit. En d'autres termes, elle permet de séparer la description « boite noire » du problème à résoudre (ce qui est attendu de la solution pour répondre au besoin, le « Pourquoi ? ») de la prescription « boite blanche » de la solution (ce qu'est la solution, le « Quoi ? ») en réponse.

Ceci permet de dépasser la vision statique des artéfacts d'IS par une vision selon une abstraction en boucle temporelle (Figure 1.13). Cette « boucle d'ingénierie » met en évidence, d'une part, les fortes interactions entre les processus de la boucle et, d'autre part, les itérations de l'ensemble de la boucle sur le système, ses sous-systèmes, les constituants, …

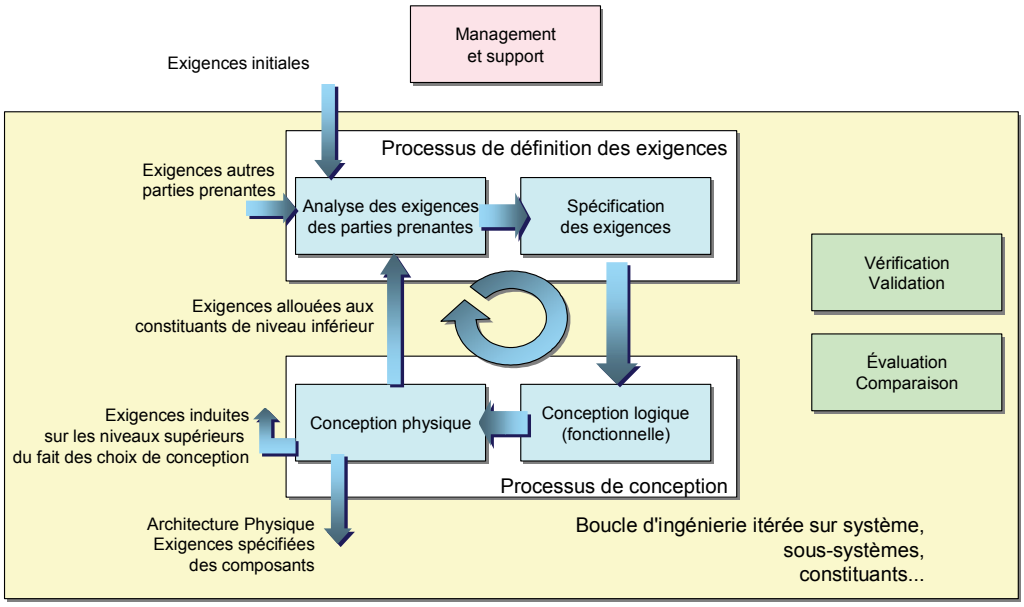


Figure 1.13 : Vision synthétique de la boucle d'ingénierie selon les 3 normes IS (AFIS, 2009)

Les travaux de Holt et Perry (2008) montrent aussi qu'il faut compléter le méta-modèle structurel des artéfacts d'IS par un méta-modèle du processus de modélisation pour assurer la cohérence entre les artéfacts mis en œuvre selon différentes vues. Par exemple, les exigences peuvent être « revisitées » de façon itérative à n'importe quelle étape du projet selon une *vue conceptuelle* (Figure 1.14) ou une *vue réalisationnelle*.

De même, la logique du procédé de modélisation dans l'atelier MDWorkbench est exprimée à la fois par les relations entre entités de chaque vue du méta-modèle, mais aussi par les relations entre ces trois vues : « *Une Partie Prenante écrit un Document qui documente un Besoin traduit par des Exigences. Une Exigence est satisfaite par des Fonctions et / ou prise en compte par des Constituants, des Liens, des Architectures organiques, …* » (Faisandier A. , 2010)

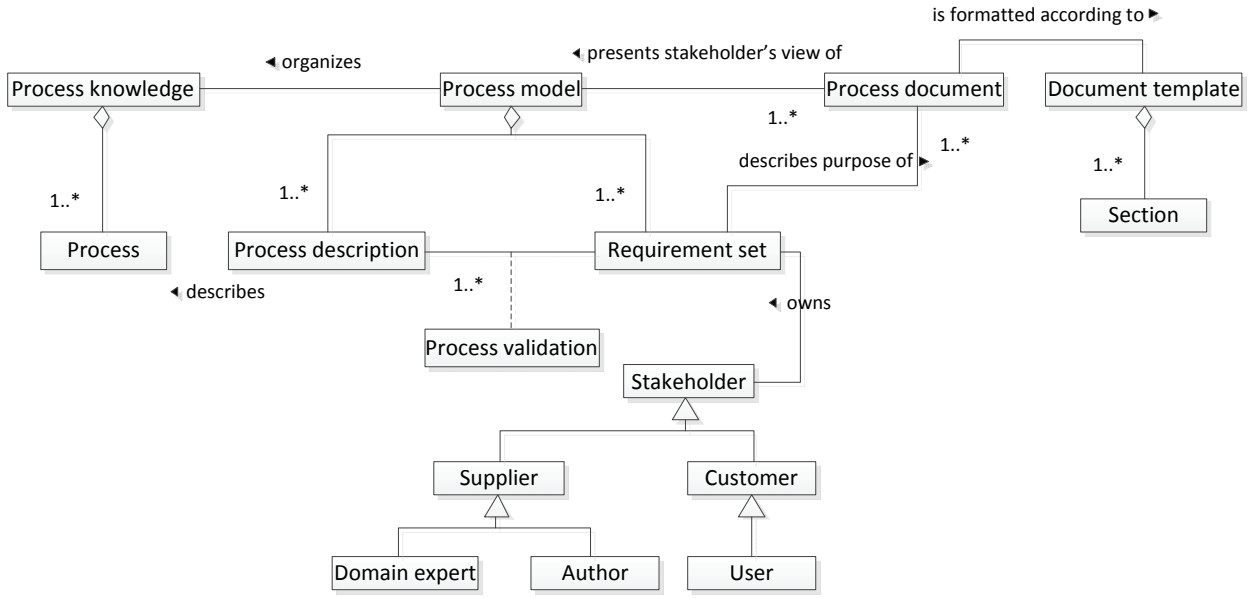


Figure 1.14 : Méta-modèle du « Process model » (Holt, et al., 2008)

L'intérêt de formaliser un processus d'IS est de guider l'architecte système et les métiers associées à la conception, et ceci de façon statique pour la construction du système lui-même et de façon dynamique pour l'enchainement des différentes activités d'IS.

Bloc système

Cette logique consiste à appliquer systématiquement de façon itérative la boucle d'ingénierie par « **Bloc Système** » (Faisandier A. , 2010) comme artéfact-clé de construction architecturale spécifiant un élément de solution à une phase-clé de développement d'un système (Figure 1.15). Nous détaillons plus précisément cette structuration par blocs systèmes qui est à la base de l'approche processus de l'IS comprenant les tâches d'ingénierie sur la branche descendante du V, et en miroir les tâches d'intégration sur la branche montante. Remarquons aussi que la réalisation proprement dite, telle que présentée dans le Chapitre 4, ne fait pas partie des activités de l'IS.

Chacun de ces blocs est constitué pour chaque activité d'ingénierie de :

- Processus de définition du besoin des parties prenantes qui produit un document (Cahier des Charges) qui contient des expressions de besoins, des attentes des parties prenantes, une vision orientée demandeur, l'ensemble exhaustif des besoins dans toutes les phases de vie ;
- Processus de définition des exigences techniques produit un document (Spécification Technique) qui contient des exigences techniques cohérentes, formalisées, vérifiables (des caractéristiques techniques, voire physiques) utiles pour le concepteur (quelque soit le niveau d'abstraction) afin qu'il puisse faire le lien avec des réalités, des lois naturelles, des expressions mathématiques, … ;

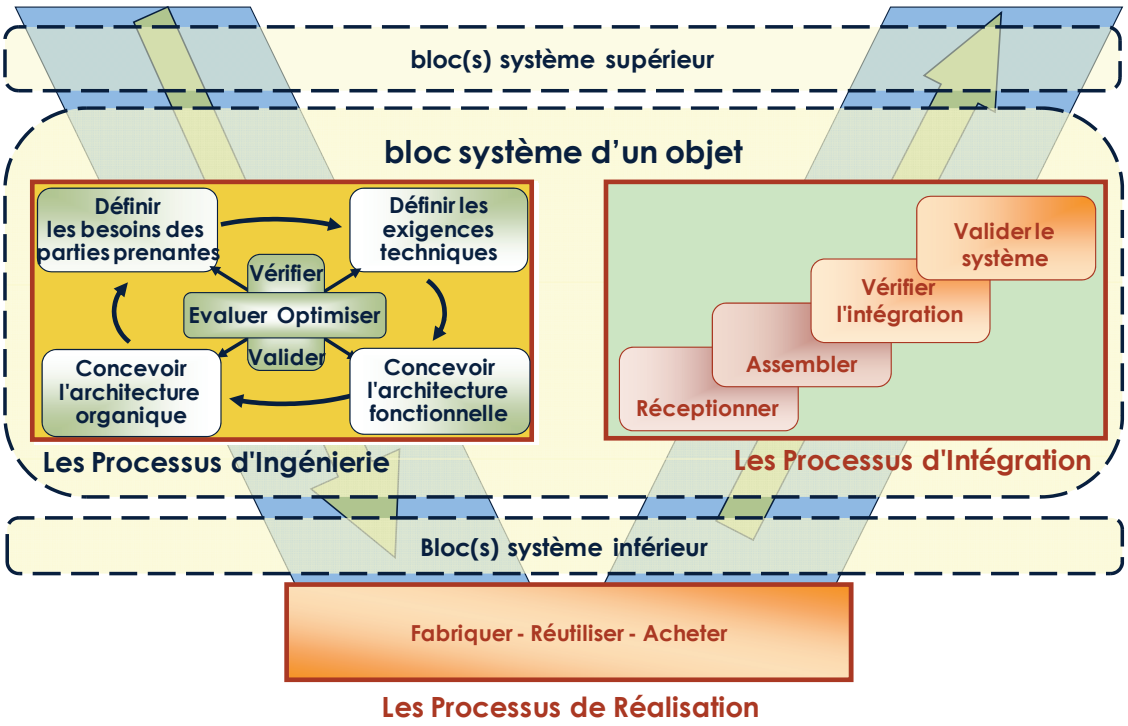


Figure 1.15 : Le développement par niveaux avec des processus récursifs (Faisandier A. , 2010)

- Processus de conception fonctionnelle et organique produisent un document (Dossier de Conception) qui contient la description de la solution retenue en termes d'architectures fonctionnelle et organique, d'interfaces ;
- Processus d'évaluation – optimisation, vérification – validation produisent un document (Dossier Justificatif) qui contient des éléments justifiant les choix de besoins, d'exigences techniques, de conception, la trace des exigences.

Chacun de ces blocs est constitué pour chaque activité d'intégration de :

- Processus d'intégration qui inclut la réception et l'assemblage des constituants conforme au Dossier de Conception ;
- Processus de vérification qui a pour but de démontrer qu'un objet réalisé est conforme à ses caractéristiques de conception ;
- Processus de validation qui consiste à s'assurer que l'objet satisfait ses exigences techniques, qu'il remplit ses fonctions.

Les activités de Vérification et Validation permettent de relier par niveau de bloc système les activités d'ingénierie avec celles d'intégration pour apporter des preuves de conformité.

Ainsi, à chaque constituant de l'architecture organique est associé un bloc-système (Figure 1.16) pour lequel il faut satisfaire des relations d'interopération avec des blocs amont décrivant le problème à résoudre et des blocs aval prescrivant une solution.

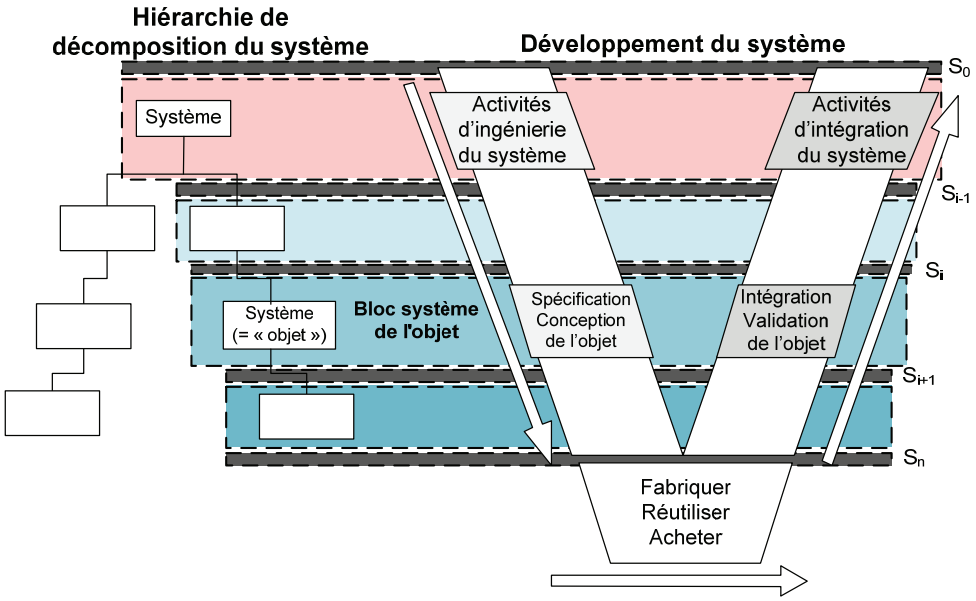


Figure 1.16 : Notion de bloc système avec des interfaces (d'après Faisandier (2010))

Chaque bloc système N (Figure 1.17) récupère les besoins des parties prenantes d'un niveau amont N-1 et fournit au bloc aval N+1 (qui peut être la réalisation ou un autre bloc de spécification de constituants) des besoins liés aux constituants du système (intégrant les exigences et les fonctions associées à ces constituants).

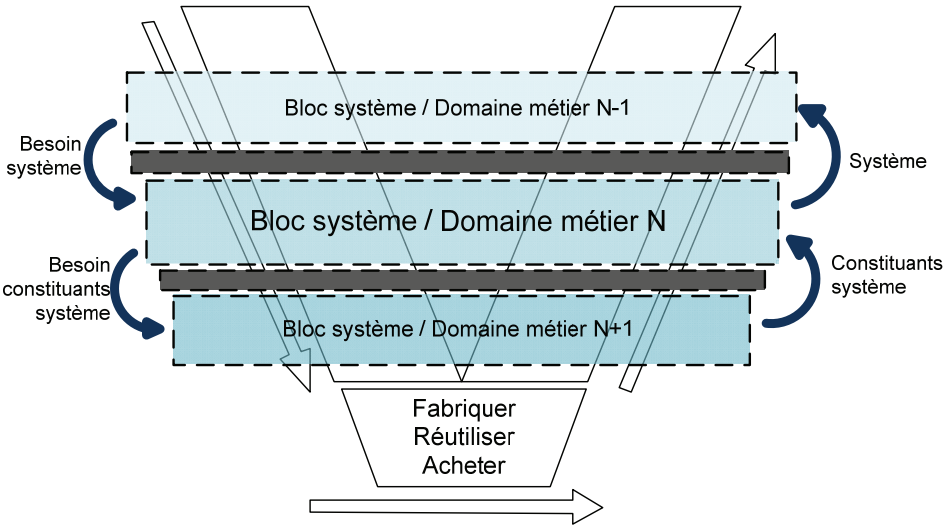


Figure 1.17 : Tout bloc système a deux voisins

Notons aussi que ces blocs systèmes peuvent être associés en pratique à des métiers différents, ce qui nécessite de caractériser la nature de ces multiples relations d'interopérations S dans un processus de modélisation IS. C'est la raison pour laquelle nos travaux se sont principalement attachés à formaliser cette relation S de description/prescription de type client/fournisseur, « boite noire »/« boite blanche », problème/solution dans le procédé itératif et collaboratif d'IS.

Nous retenons ce principe comme base du processus d'ingénierie de notre application. Dans la suite nous allons présenter les principaux artéfacts que nous considérons importants pour concevoir notre système SIAC.

Besoins

Intégrés dans la vue affaire du méta-modèle de l'AFIS, il s'agit de l'interface S d'entrée de tout bloc système.

Nous retrouvons les besoins sous la forme des documents d'expression des besoins dans la relation S (Figure 1.18). Ainsi, du point de vue de Faisandier A. (2010), la logique d'enchainement des processus d'ingénierie (côté descendant du bloc système) se fait par ces documents qui décrivent ce qui est demandé par le niveau du bloc système en aval. De plus, le processus de définition des besoins des parties prenantes assure la structuration et la formalisation des besoins pour valider avec le client, dans une première itération, l'ensemble des besoins consolidés.

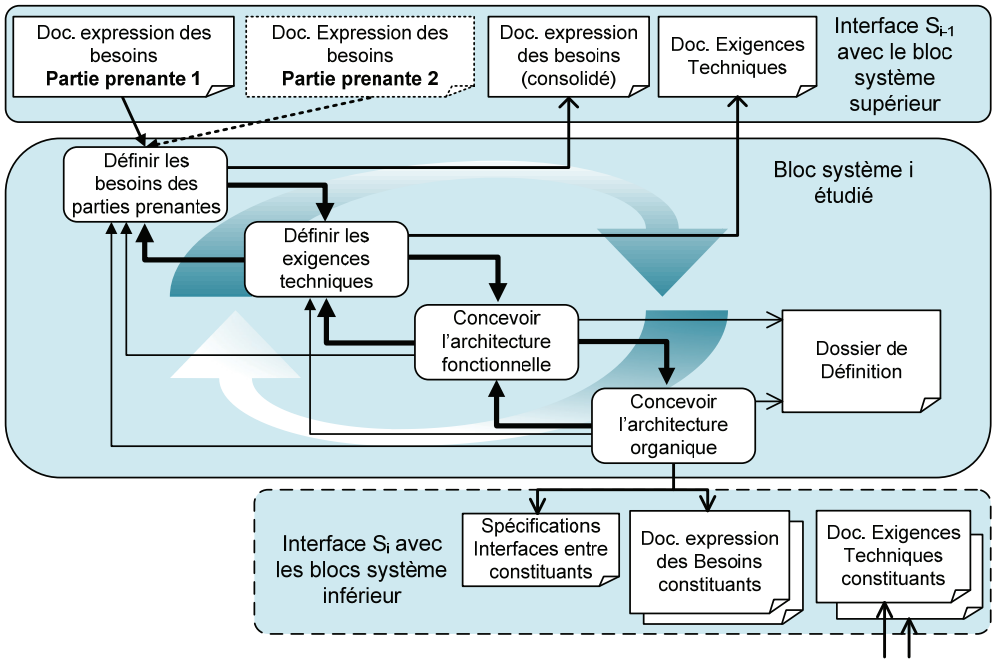


Figure 1.18 : Logique d'enchaînement des processus d'ingénierie (Faisandier A. , 2010)

Une solution pour lever les ambigüités du langage naturel pour éliciter les besoins est de prescrire en Langage Naturel Binaire leur compréhension en utilisant la technique ORM/NIAM[15] (Natural language Information Analysis Method, Halpin (1998), Blaise (2000)).

Notons l'importance après cette phase préliminaire d'élicitation des besoins de formaliser la finalité, la mission et l'objectif du système.

Définition 1.9 : Finalité (après Faisandier A. (2010))
La finalité décrit pourquoi le système doit exister, sa pertinence dans le contexte.

[15] Nous retrouvons aussi sous le nom de « Nijssen Information Analysis Method »

Elle décrit le « POURQUOI ».

La finalité correspond par rapport à la Figure 1.2 à l'objet finalisant qui est produit par le système et qui est consommé par l'environnement. Nous considérons que la finalité doit être exprimée sous la forme d'un objet et non pas en utilisant un verbe comme Faisandier le propose.

Définition 1.10 : Mission (après Faisandier A. (2010))
La mission décrit ce que le système doit faire, ce qu'il transforme, le service qu'il rend. Elle décrit le « QUOI ».

La mission décrit l'action principale exécutée par le système. Ainsi, nous allons exprimer la mission avec un verbe décrivant cette action.

Définition 1.11 : Objectif (après Faisandier A. (2010))
L'objectif décrit le nombre d'éléments traités par le système, son efficacité, … Il décrit le « COMBIEN ».

Les objectifs quantifient le système par rapport à ce qu'il doit faire.

La finalité, la mission et l'objectif du système peuvent être combinés dans une phrase sous forme de « Sujet (finalité) + Verbe (Mission) + Complément d'objet (Objectif) ». Cette première formulation doit ensuite être raffinée pour qualifier ces éléments : où, quand, comment, combien, …

Exemple 1.1
La **finalité** du SIAC correspond aux services d'aide à la conduite.
La **mission** du SIAC est de fournir les services d'aide à la conduite.
L'**objectif** du SIAC est de faciliter l'interaction numérique entre le rondier et le procédé en tout lieu et à tout instant.

Nous pouvons formuler l'ensemble finalité/mission/objectif du SIAC dans la phrase :
« *Des **services d'aide à la conduite** doivent **être fournis** pour **faciliter l'interaction numérique entre le rondier et le procédé en tout lieu et à tout instant***. »

Le document d'expression des besoins est habituellement appelé « Cahier des Charges » (ou « Expression du besoin utilisateur »), document par lequel le demandeur exprime son besoin (ou celui qu'il est chargé de traduire) en terme de services et de contraintes (interfaces avec l'environnement et contraintes sur la solution). Son établissement implique que des études aient permis de cerner avec précision les besoins des utilisateurs.

Exigences

Cet artéfact-clé d'IS est le pivot de tout projet quelle que soit la phase considérée (définition, spécification, validation, test, …) puisqu'il représente une première réponse (solution) à l'ensemble de besoins (Figure 1.19). C'est également sur ces exigences que se base la conception du système final. Les exigences définissent ce que le système DOIT FAIRE et DOIT ÊTRE.

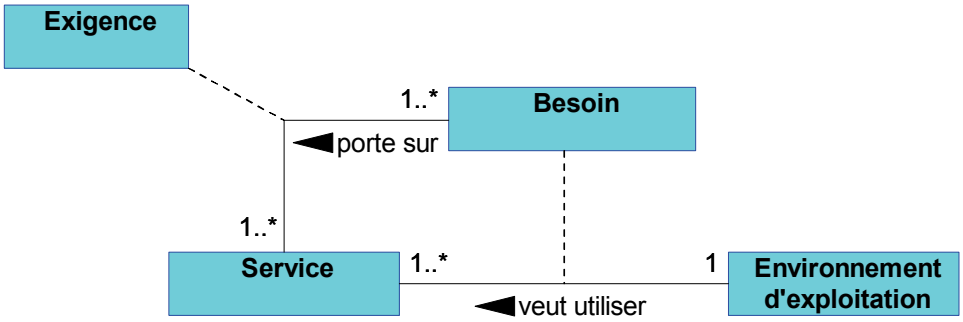


Figure 1.19 : Relation entre besoin, services et exigence

Définition 1.12 : Exigence (AFIS, 2009)

Une exigence prescrit une propriété dont l'obtention est jugée nécessaire. Son énoncé peut être une fonction, une aptitude, une caractéristique ou une limitation à laquelle doit satisfaire un système, un produit ou un processus.

Un type d'exigence caractérise dans les méta-modèles des relations de dérivation/allocation entre des artéfacts sources et puits de vues différentes. Nous nous sommes particulièrement intéressés à la relation de description/prescription d'une exigence qui traduit in-fine la spécification d'une solution système en réponse à un besoin système qui doit nécessairement faire l'objet d'un processus de vérification et/ou de validation. De même, les transformations multiples (décomposition, dérivation, raffinement, …) de cette exigence à différents niveaux d'échelle (d'observation) représentent d'autres sources de relations à satisfaire par allocation de constituants-système.

L'état d'exigence traduit le fait qu'une exigence a un cycle de vie de par son allocation au cours du cycle de vie d'ingénierie, y compris en relation avec le cycle de vie du système.

Un attribut d'exigence ajoute des propriétés intrinsèques telles que la source de l'exigence, la méthode de vérification, …

Dans l'approche par bloc système de Faisandier (2010), nous retrouvons les exigences techniques qui sont définies par le processus de même nom. Le résultat de ce processus est un document à caractère contractuel entre demandeur et fournisseur, appelé aussi « Spécification technique », établi par le fournisseur d'un produit à l'intention du concepteur et par lequel il exprime les exigences techniques applicables. Envoyé au niveau du bloc système amont par la relation S (Figure 1.18), ce document doit être validé par le client. Il doit exprimer ce que l'on attend du produit en terme de fonctions, de performances, d'interfaces, les contraintes d'utilisation, d'environnement et de soutien (logistique), les contraintes pour la conception, la production et la validation du produit (les conditions de vérification du respect des exigences), …

Les travaux de Holt et Perry (2008) considèrent cet artéfact comme un élément structurel d'un processus d'ingénierie des exigences en le catégorisant en trois types selon qu'il adresse des problèmes relatifs aux aspects « affaires » des processus d'entreprise ou aux aspects techniques des processus opératoires d'IS en terme de fonctionnalités désirées du système et de contraintes qui restreignent ces fonctionnalités (Figure 1.20).

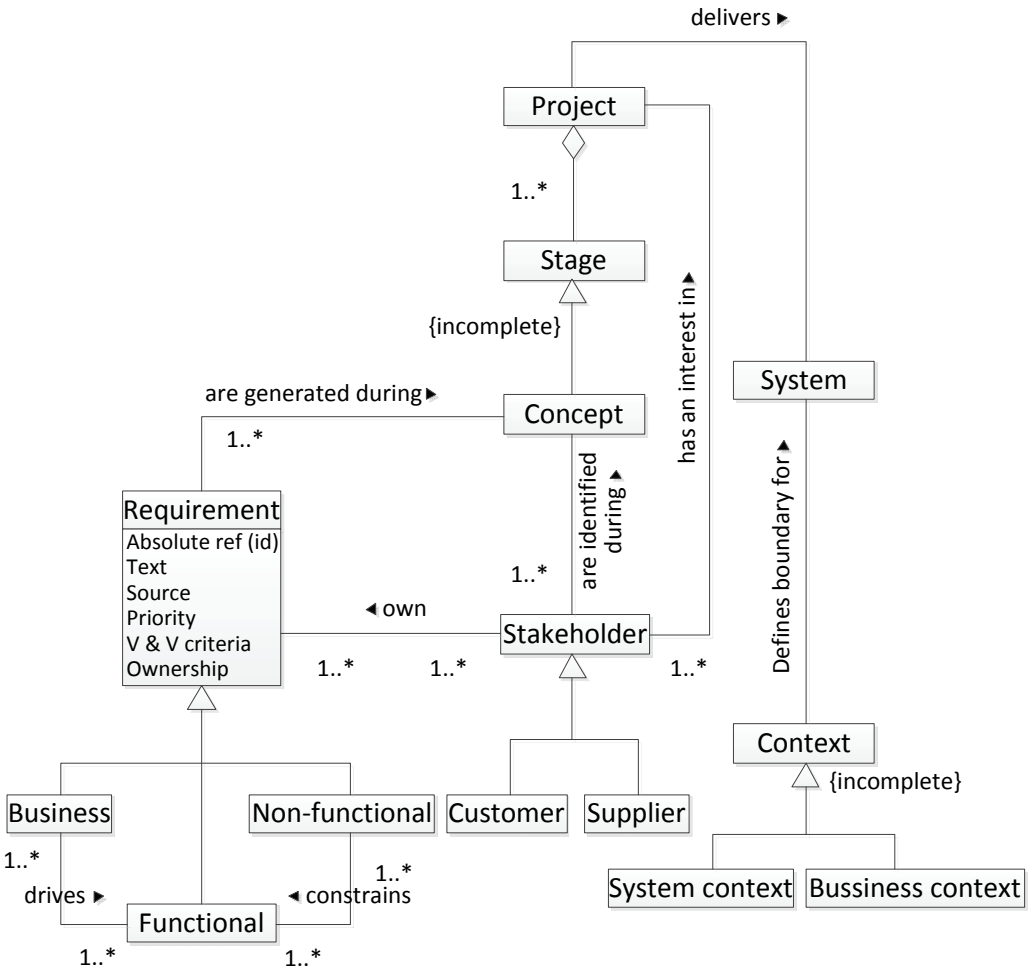


Figure 1.20 : Méta-modèle des exigences d'un système (Holt, et al., 2008)

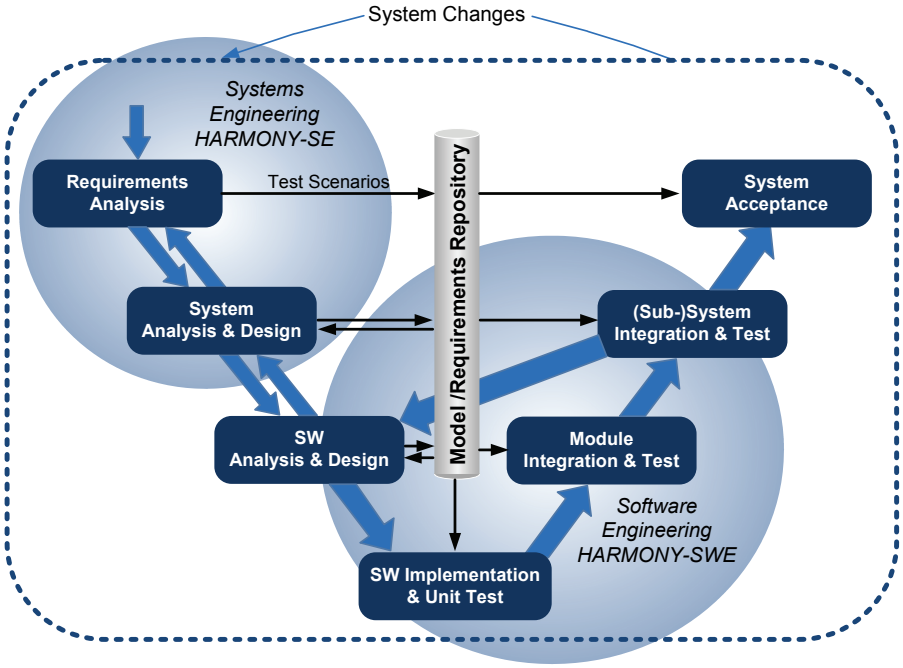


Figure 1.21 : Le processus d'ingénierie des systèmes intégrés et des applications informatiques (HARMONY) proposé par I-Logix (Hoffmann, 2004)

Nous dépasserons en Chapitre 2 cette vision structurelle statique qui conduit à nombre de guides (REGAL (Dick, et al., 2006), Volere (Robertson, et al., 2006)) et d'outils (IBM Rational DOORS[16], Reqtify[17], …) de définition, de gestion et de traçabilité des exigences sans pallier la difficulté de les structurer *a posteriori* dans un référentiel des exigences (Figure 1.21).

Nous étendrons la formalisation de l'exigence comme artéfact de modélisation transverse S en distinguant la partie décrite par un espace de problème de la partie ajoutée par un espace de solution pour prescrire un élément de solution. Cette composition qui interprète la proposition de Jackson (§1.2.1.1) aide, à notre sens, à structurer le référentiel des exigences systèmes en n'y conservant que le résultat prescrit et pas les connaissances y contribuant qui demeurent dans les référentiels métiers.

Architecture système

Cet artéfact-clé d'IS concrétise la solution système et ses variantes construites à partir des exigences système prescrites. La démarche est la décomposition du problème à résoudre pour satisfaire aux exigences en leur allouant des éléments de solution puis leur recomposition pour agencer et intégrer ces éléments en un système final. En effet, nous nous sommes limités à cette relation de prescription/construction d'une solution.

Cependant, Caron (2005) précise que selon les méthodes en vigueur dans les entreprises et l'expérience professionnelle des architectes, tous n'ont pas la même vision de ce qu'est une architecture système. En essayant de fédérer les différentes visions, Caron se base sur le principe qu'un système donné n'a pas plusieurs architectures mais bien une seule, que l'on peut représenter selon différents points de vue qui ne sont pas nécessairement antagonistes.

Définition 1.13 : Architecture (AFIS, 2009)
L'architecture est un modèle constructif du système définissant son organisation (ses éléments et leurs interfaces) ainsi que les principes régissant sa conception et son évolution. L'architecture peut être représentée selon différents points de vue. Elle peut être logique (fonctionnelle) ou physique. Elle peut se représenter à différents niveaux d'abstraction ou de granularité.

Ainsi, deux types d'architectures peuvent être définis : d'une part l'architecture fonctionnelle qui rassemble les fonctions-système et qui définit le comportement du système, et d'autre part l'architecture organique qui structure les constituants-système et qui définit la structure du système.

Fonctions et architectures fonctionnelles

Ces artéfacts sont directement liés au comportement d'un système et définissent ce que le système PEUT FAIRE.

[16] IBM Rational DOORS : www-01.ibm.com/software/awdtools/doors/
[17] Reqtify : www.geensoft.com/en/article/reqtify

Définition 1.14 : Fonction
Comme un processus, une fonction est une action qui traite des entrées et qui génère des sorties. Cependant, nous préférons utiliser le terme de « fonction » pour parler de comportement d'un système et le terme de « processus » pour se référer à une activité du cycle de vie d'un système.

Dans les architectures fonctionnelles[18], les fonctions sont enchaînées en utilisant des « schémas de conception » (design patterns connus). Pour Faisandier (2010), cet enchaînement des fonctions prend la forme des scénarios opérationnels, qui expriment leur dynamique par l'échange des flux. Faisandier ajoute aussi les modes opérationnels pour exprimer les états d'un système pendant son utilisation.

AFIS fait la distinction entre les services systèmes, fonctions de plus haut niveau fournies à la limite de l'environnement du système, et les fonctions internes au système (utilisées par ces services systèmes).

La représentation de l'architecture fonctionnelle peut être faite de trois manières (« vision fonctionnelle » dans Figure 1.6) :

- Décomposition fonctionnelle ;
- Représentation structurelle sous forme de diagramme de flux, sans flux de contrôle ;
- Représentation comportementale sous forme de l'enchaînement des fonctions.

Etant assez génériques, les architectures fonctionnelles permettent de proposer plusieurs solutions organiques qui peuvent être évaluées afin de choisir la meilleure. C'est pour cela que nous considérons l'approche systémique mise en œuvre par l'IS, qui conçoit les fonctions avant de les rassembler dans des constituants, une démarche complémentaire à l'approche orienté objet proposée par le génie logiciel, pour la spécification des systèmes.

Constituants et architectures organiques

Ces artéfacts sont directement liés à la structure d'un système et définissent ce que le système PEUT ÊTRE.

Définition 1.15 : Constituant
Un constituant d'un système est un élément qui rassemble et réalise une à plusieurs fonctions du système.

D'après le méta-modèle de l'AFIS, on peut remarquer deux types de constituants : logiques et physiques. Un constituant logique est la vue des fonctions, des caractéristiques et des propriétés d'un constituant, décrite indépendamment des solutions physiques et technologiques choisies pour la réalisation. Le constituant physique est le constituant concret réalisable, offrant des fonctions et possédant des caractéristiques et propriétés.

[18] AFIS utilise le terme d'architecture logique, mais nous préférons de l'appeler architecture fonctionnelle

En complément de l'architecture fonctionnelle, l'architecture organique propose un arrangement de constituants interfacés entre eux pour définir une solution conçue (« vision physique » dans Figure 1.6).

La décomposition du système à réaliser par un bloc système en constituants permet de définir de nouveaux blocs systèmes en les attachant à ces constituants. C'est pour cela que l'ensemble des exigences, fonctions, ainsi que le constituant assimilé vont faire partie du document d'expression des besoins pour le bloc système du constituant en aval.

Patrons de conception

Les patrons de conception (« Design Patterns ») font l'objet d'un grand intérêt en IS pour formaliser des bonnes pratiques de construction système, comme par exemple la transformation en eFFBD (enhanced Function Flow Block Diagram) bien formés à des fins de vérification d'une spécification décrite en FFBD (Function Flow Block Diagram) (Seidner, 2009).

Définition 1.16 : Patron de conception (*en*. Design Patterns, Alexander, et al. (1977))
Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.

D'une manière plus générale, les patrons de conception fournissent un mécanisme pour qu'un spécialiste puisse transmettre les connaissances suffisantes pour faciliter la résolution du problème par un non spécialiste (architecte) dans son processus de modélisation de la solution. Nous y attacherons une attention particulière parce que cette manière de capitaliser des savoir-faire devient importante pour une Ingénierie Système Basée sur des Modèles (Cloutier, 2006; Estefan, 2008) sans qu'il soit évident que la construction du « Design Pattern » lui-même soit robuste.

L'utilisation du « patron composite » (Gamma, et al., 1995) dans (Holt, et al., 2008) pour la construction d'un système (Figure 1.22) et la définition des éléments de son contexte (« Boundary ») confirme, comme le font aussi Baïna (2006) et Auzelle (2009), l'intérêt de s'appuyer sur des « Design Patterns » établis, comme dans d'autres domaines (Perronne, et al., 2006), pour méta-modéliser les artéfacts IS.

Par exemple, ces travaux distinguent trois principaux types de systèmes que sont le système d'intérêt objet de la mission principale et ses systèmes contributeurs (conception, maintenance, …) supports à cette mission principale ainsi que la composition d'un Système de Systèmes à différents niveaux d'échelle (d'abstraction, d'intégration) sans cependant de critères de catégorisation comme proposé par Maier (1998).

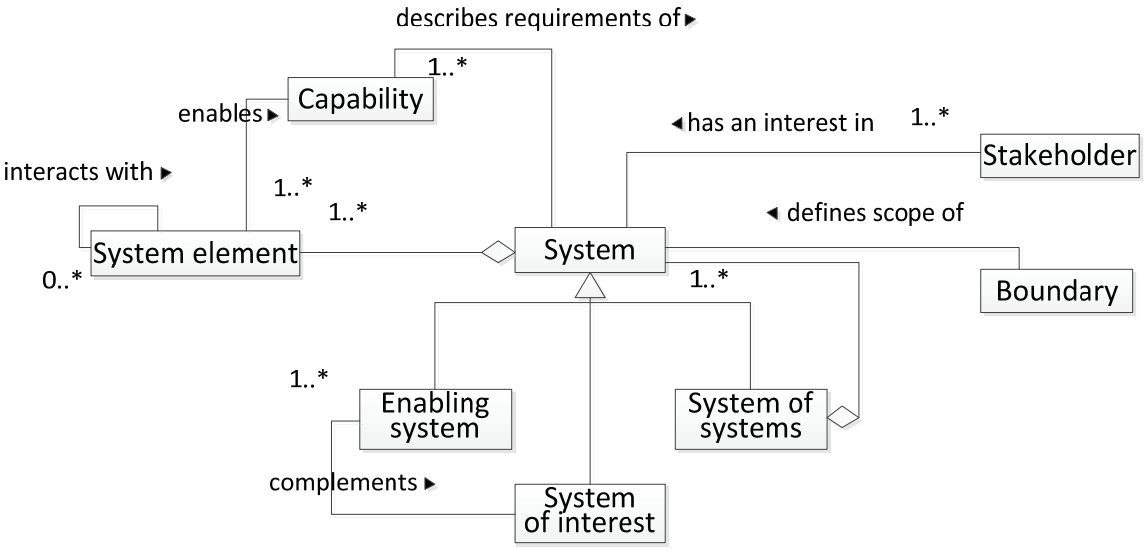


Figure 1.22 : Meta-modèle d'un système (Holt, et al., 2008)

Interfaces

Comme nous avons pu le voir, nous pouvons définir deux types d'interfaces.

Définition 1.17 : Interface (AFIS, 2009)
Lors de toute décomposition, fonctionnelle ou organique, les interactions entre éléments résultant de la décomposition sont identifiées, ce qui génèrera des interfaces dans les architectures.

D'une part, les interfaces physiques apparaissent lorsque des éléments fonctionnels en interaction sont alloués à des constituants physiques différents. Elles peuvent nécessiter des fonctions d'interface (fonctions d'adaptation entre les constituants telles que lubrification en interface mécanique, adaptation sémantique et syntaxique en interface informationnelle).

D'autre part, le passage entre les blocs systèmes nécessite la définition des interfaces (relations) pour échanger les différents documents générés par chaque bloc système. Ce sont les relations S qui nous intéressent.

Vérification et Validation

Une des caractéristiques de qualité des exigences introduite par AFIS (2009) est la vérifiabilité : à toute exigence est associée au moins une méthode permettant de vérifier son obtention (il serait inutile d'écrire une exigence dont on ne saurait montrer que le système, une fois réalisé, y satisfait). Aussi, toute exigence système doit avoir une justification. C'est pour cela qu'AFIS introduit des stratégies de vérification et validation des exigences sur des exemplaires du système final, intégrés dans différents environnements de test, afin de s'assurer que celui-ci satisfait bien ces exigences.

Définition 1.18 : **Vérification**

La *vérification* est la confirmation, par examen et collecte d'évidences objectives, que les exigences à partir desquelles un système est construit, codé, ou assemblé, sont satisfaites.

Elle répond à la question : « A-t-on construit *UN BON* système ? »

La mission principale de la vérification est de déterminer que la solution du système, qu'elle soit logique ou physique, est conforme aux exigences.

Définition 1.19 : **Validation**

La *validation* est la confirmation par examen et collecte d'évidences objectives qu'un système, ou l'agrégation de systèmes, fonctionne comme attendu par le client dans son environnement opérationnel.

Elle répond à la question : « A-t-on construit *LE BON* système ? »

La validation est l'action de déterminer que le système fait tout ce qu'il est censé faire (il répond aux besoins). La validation est souvent réalisée par une tierce personne autre que le développeur et l'utilisateur.

Par conséquent, il est important de mettre en œuvre des processus de Vérification/Validation pour chacun des niveaux ainsi qu'entre niveaux :

- Pour la vérification sur un même niveau par l'exécution des modèles de spécification ;
- Pour la validation de la solution réelle dans la phase d'IVV ;
- Pour la validation de la spécification qui vient du niveau inférieur.

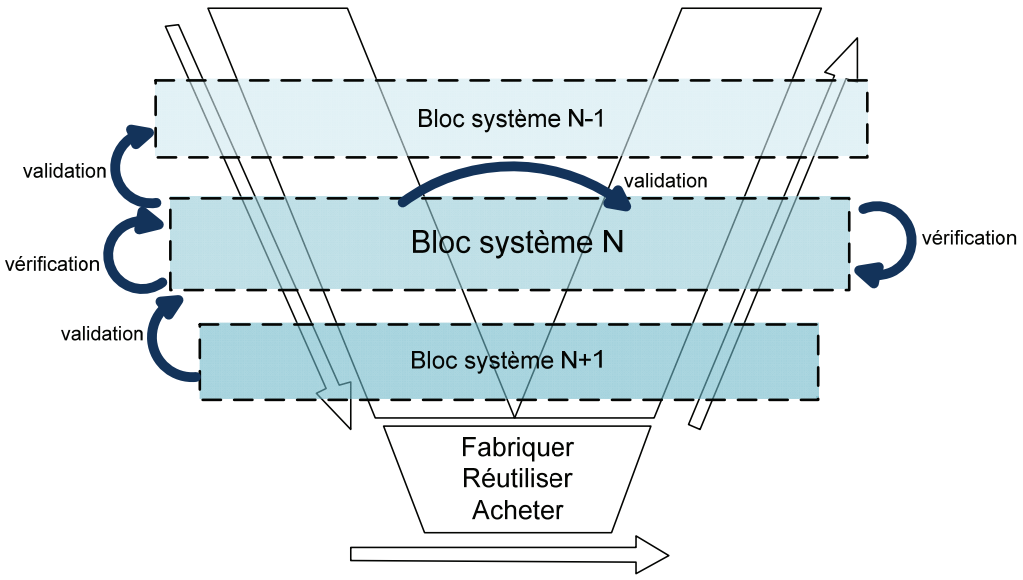


Figure 1.23 : Vérification et validation par rapport à un bloc système

1.2 Artéfacts connexes

Afin de formaliser la relation S de description/prescription, nous nous sommes intéressés à l'approche des « Problem Frames » en informatique. Cette étude nous a permis ainsi de faire une analogie avec des travaux antérieurs du laboratoire relatifs au paradigme d'automatisation, et avec d'autres approches.

1.2.1 Approche des « Problem Frames »

Le Problem Frame (PF) (Jackson, 2000) est une approche orientée sur le problème (Figure 1.24) qui offre un cadre pour décrire les interactions entre les logiciels et les autres composants d'un système. Il aide le développeur à comprendre le contexte dans lequel réside le problème et quels sont les aspects pertinents pour la conception d'une solution. Dans cette approche, un besoin[19] est une contrainte de bout en bout sur les phénomènes du domaine du problème, qui ne sont pas nécessairement contrôlés ou observés par la machine. Au cours du développement ultérieur (Seater, 2009), le besoin est généralement pris en compte par une spécification[20] (d'une machine à mettre en œuvre) et un ensemble d'hypothèses sur le domaine (sur le comportement des périphériques physiques et les opérateurs qui interagissent directement ou indirectement avec la machine).

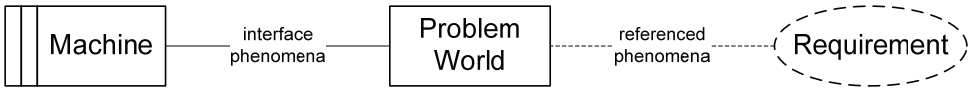


Figure 1.24 : Une description dans le Problem Frame (Jackson, 2000)

Une description dans le PF met en évidence la séparation entre les phénomènes qui sont contrôlés par la machine (« interface phenomena ») et ceux contraints par le besoin (« referenced phenomena »). Cette description est appelé aussi un diagramme de problème (*en*. problem diagram).

Définition 1.20 : Problème (Wikipedia)
Un problème dans son acception la plus courante, est une situation dans laquelle un obstacle empêche de progresser, d'avancer ou de réaliser ce que l'on voulait faire.

Les concepts qui sont à la base des PF fournissent un cadre de spécification des systèmes informatiques.

1.2.1.1 Spécification des systèmes informatiques

On retrouve des travaux de recherche sur la notion de la spécification principalement dans le domaine de l'informatique. Nous nous sommes attachés principalement aux travaux de Mi-

[19] Jackson utilise le terme « Requirement » pour définir des exigences liées au domaine du problème. Nous considérons que la meilleure correspondance dans l'IS est la notion de besoin décrivant le problème à résoudre.
[20] La notion de spécification est utilisée pour représenter les exigences du système à réaliser.

chael Jackson (Jackson, et al., 1995; Jackson, 1997) qui étudie les exigences et les spécifications d'un système en séparant l'environnement du système réalisé.

L'objectif de Jackson est de réaliser un programme informatique P implémentable sur une machine M qui est intégré dans un environnement W. Afin de faciliter la compréhension des définitions que Jackson a données en termes de besoin et de spécifications, nous allons introduire d'abord le modèle de référence (Figure 1.25) proposé par Gunter, et al. (2000).

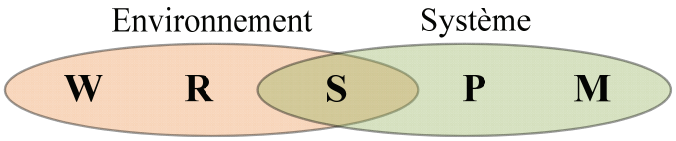


Figure 1.25 : Modèle de référence pour les exigences et les spécifications (Gunter, et al., 2000)

Ce modèle de référence présente cinq artéfacts répartis entre l'environnement et système :

- la connaissance du domaine (World) fournit des connaissances sur l'environnement ;
- les besoins (Requirements) des utilisateurs du système, décrits sous la forme d'effets sur l'environnement ;
- les spécifications (Specifications) fournissent des informations suffisantes pour qu'un programmeur puisse construire le système qui satisfait les exigences ;
- le programme (Program) implémente les spécifications en utilisant la plateforme programmable ;
- la plateforme programmable (Machine) fournit la base pour programmer un système qui satisfait les exigences et les spécifications.

Il est à noter dans Jackson que le terme « requirement » est utilisé dans ce modèle pour exprimer des besoins ou « customer needs ». Pour Jackson et Zave (1995), W et R représentent la description complète des besoins qui consiste en :

- La description des propriétés de l'environnement qui vont permettre à la machine de satisfaire les exigences : c'est la *partie indicative*. Une description indicative décrit comment sont les choses et représente les connaissances sur l'environnement (W dans le modèle de référence).
- La description des conditions demandées sur les phénomènes[21] de l'environnement : c'est la *partie optative*[22]. Une description optative décrit comment doivent être les choses (R dans le modèle de référence).

[21] Phenomena (Bjørner D. , 2009) : By a phenomenon we understand an observable fact, that is, a temporal or spatio/temporal individual (particular, "thing") of sensory experience as distinguished from a noumenon[21], that is a fact of scientific interest susceptible to scientific description and explanation.

[22] L'optatif (du latin « optare » = souhaiter) est un mode de la conjugaison grecque, ainsi appelé parce que très souvent il exprime le souhait, le désir, le vœu.

Définition 1.21 : Besoin (d'après Jackson et Zave (1995) et Jackson (1997))
Un besoin spécifie les relations souhaitées dans l'environnement qui seront induites ou maintenues par la machine. Le besoin est entièrement consacré à l'environnement où les effets et les avantages de la machine se feront sentir et évalués : la machine est uniquement le moyen pour réaliser l'effet requis dans l'environnement.
Un besoin est une propriété optative, destinée à exprimer les désirs des clients concernant le projet de développement.

Gunter, et al. (2000) montrent aussi qu'à la limite de l'environnement et du système, il existe des phénomènes qui sont vus par les deux ensembles. C'est cette partie qui est concernée par la spécification S du système.

Cela revient à définir (Figure 1.26):

- Les termes qui indiquent les phénomènes e_h, e_v et s_v, visibles par l'environnement et utilisés dans W et R.
- Les termes qui indiquent les phénomènes s_h, s_v et e_v, visibles par le système et utilisés dans P et M.

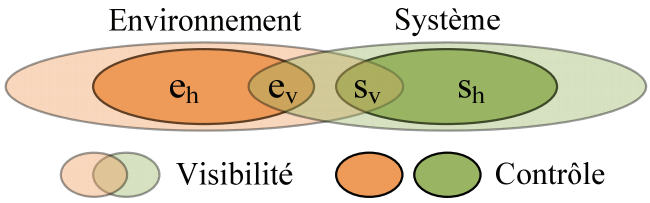


Figure 1.26 : Visibilité et contrôle des phénomènes (Gunter, et al., 2000)

Exemple 1.2
Nous allons illustrer la Figure 1.25 et la Figure 1.26 avec des éléments de notre application. On suppose que l'exigence R est de construire un système de vérification et de validation des opérations réalisées par le rondier sur les vannes manuelles. Pour réaliser cela, nous disposons d'une machine (dispositif) M, attaché à la vanne manuelle, avec des capteurs, pour récupérer son état, et une interface de communication, pour recevoir l'action à exécuter sur la vanne. Un programme P s'exécute sur cette machine et vérifie l'action à réaliser par rapport à l'état de la vanne et la valide. La connaissance sur l'environnement W dit que si le système reçoit une action pour la valider, alors il y a toujours un rondier pour la réaliser. Un scénario est : (1) le rondier reçoit sur son PDA l'action à exécuter ; (2) le PDA transmet au système un message avec l'action à réaliser ; (3) le système vérifie l'état de la vanne par rapport à l'action à réaliser ; (4) dès que l'action est réalisée, le système envoie un message de validation sur le PDA du rondier. Étant donné ce système, les phénomènes sont :

- e_h : le rondier, le PDA du rondier, la vanne manuelle ;
- e_v : le message avec l'action à exécuter, l'état des capteurs de position ;
- s_v : le message de validation ;

- s_h : représentation interne de l'action à réaliser et de l'état de la vanne.

Ainsi, S s'exprime par rapport aux phénomènes communs aux deux espaces : s_v et e_v, ce qui est conforme à notre proposition de structuration du référentiel des exigences.

Définition 1.22 : Spécification (d'après Jackson et Zave (1995) et Jackson (1997))
Une spécification décrit le comportement de la machine à son interface avec l'environnement. Comme un besoin, elle est exprimée exclusivement en termes de phénomènes de l'environnement. Vue de la machine, une spécification est un point de départ pour l'implémentation (la programmation), vu de l'environnement, c'est une sorte restreinte de besoin.
Une spécification est dérivée d'un besoin. Compte tenu d'un besoin, nous nous acheminons vers une spécification en dégageant le besoin de toutes les fonctionnalités, telles que les références aux phénomènes environnementaux qui ne sont pas accessibles à la machine et qui empêcheraient la mise en œuvre. La dérivation est rendue possible par les propriétés de l'environnement qui peuvent être invoquées indépendamment du comportement de la machine. Ces propriétés doivent, bien entendu, être explicitement décrites si elles doivent être exploitées. Cette dérivation des spécifications à partir des besoins est vaguement analogue au raffinement d'un programme.
Une spécification est une propriété optative, destinée à être implémentée directement et à supporter la satisfaction des besoins.

De façon plus formelle, la spécification S selon :

$$\forall e, s. W \wedge S \Rightarrow R \qquad 1\text{-}1$$

permet de prescrire une solution selon :

$$\forall e. (\exists s. S) \Rightarrow (\exists s. M \wedge P) \wedge (\forall s. (M \wedge P) \Rightarrow S) \qquad 1\text{-}2$$

en réponse au besoin selon :

$$\forall e, s. W \wedge M \wedge P \Rightarrow R \qquad 1\text{-}3$$

Problème/Solution

Nous avons interprété le modèle de Gunter, et al. (2000) comme une partition entre domaines, où l'environnement représente le domaine d'exploitation visé et où le système représente le domaine d'IS dans un premier temps et, dans un deuxième temps, comme une partition entre le domaine de l'IS et les domaines métiers des composants du système (Figure 1.27). Ceci de façon récursive jusqu'à la spécification des constituants du système.

Appliquée au procédé d'IS (Figure 1.16), notre interprétation considère ainsi que chaque domaine ou bloc système comporte deux rôles, en tant qu'espace de solution et espace de problème (Figure 1.28), cela étant récursif jusqu'à la réalisation qui est considérée comme espace de solution.

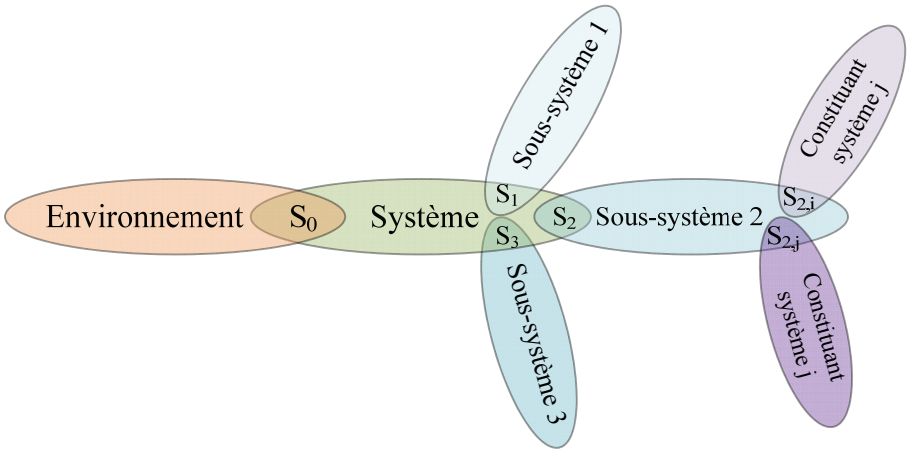


Figure 1.27 : Partition des domaines autour du domaine du Système à Faire

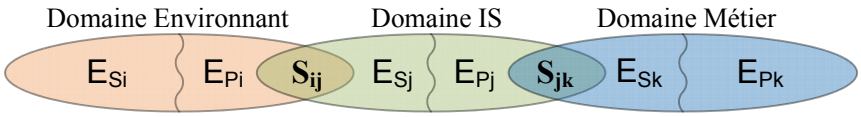


Figure 1.28 : Séparation entre espaces de problème et de solution

Les domaines métiers commencent avec le domaine de l'environnement d'exploitation, qui est un espace de problème, et vont jusqu'aux domaines techniques qui assurent la réalisation de la solution, et qui sont des espaces de solution. C'est pour cela que nous considérons tout domaine intermédiaire (ou bloc système), que ce soit la maîtrise d'ouvrage ou la maîtrise d'œuvre, d'une part comme un espace de solution pour le domaine métier amont et d'autre part comme un espace de problème pour le domaine métier aval. En effet, si un domaine métier propose des solutions aux problèmes d'un autre domaine métier, alors ce premier domaine va assurer les processus de définition des besoins, prescription des exigences, conception de l'architecture fonctionnelle et de l'architecture organique pour arriver à fournir ces solutions. Si ce domaine n'est pas capable de répondre à certaines attentes du domaine amont, alors il devient à son tour un espace de problème, et va chercher un autre domaine métier qui pourrait lui fournir des solutions.

Vision compositionnelle de l'exigence

Les définitions du besoin et de la spécification de Jackson ne couvre pas exactement la signification de l'artéfact clé d'exigence défini précédemment, notamment dans le modèle de l'AFIS qui met en évidence la notion d'état de l'exigence R. Nous avons finalement interprété ces travaux en considérant que toute exigence induite prescrite par un espace de solution est le résultat d'un processus de transformation associant une partie optative décrite par un espace de problème et une partie indicative représentant les connaissances nécessaires dans l'espace de solution (Figure 1.29). Nous faisons de cette transformation besoin/connaissance/exigence le pivot de notre procédé itératif de modélisation système.

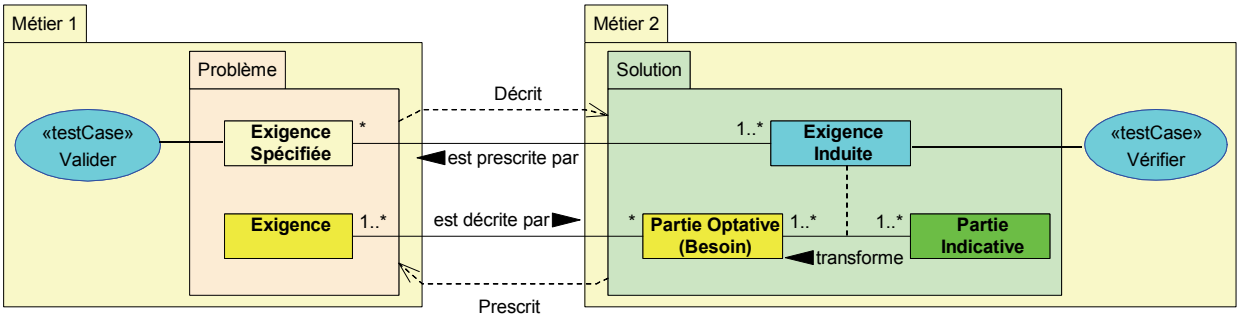


Figure 1.29 : Exigence transformée dans le procédé d'IS

Relation de spécification *S*

Nous considérons la spécification comme l'ensemble d'exigences induites vérifiées et prescrites par l'espace de solution, et validées par l'espace de problème à tous les niveaux du procédé itératif d'IS. Ceci traduit l'interface *S* comme une relation contractuelle entre les deux domaines problème/solution (Figure 1.30).

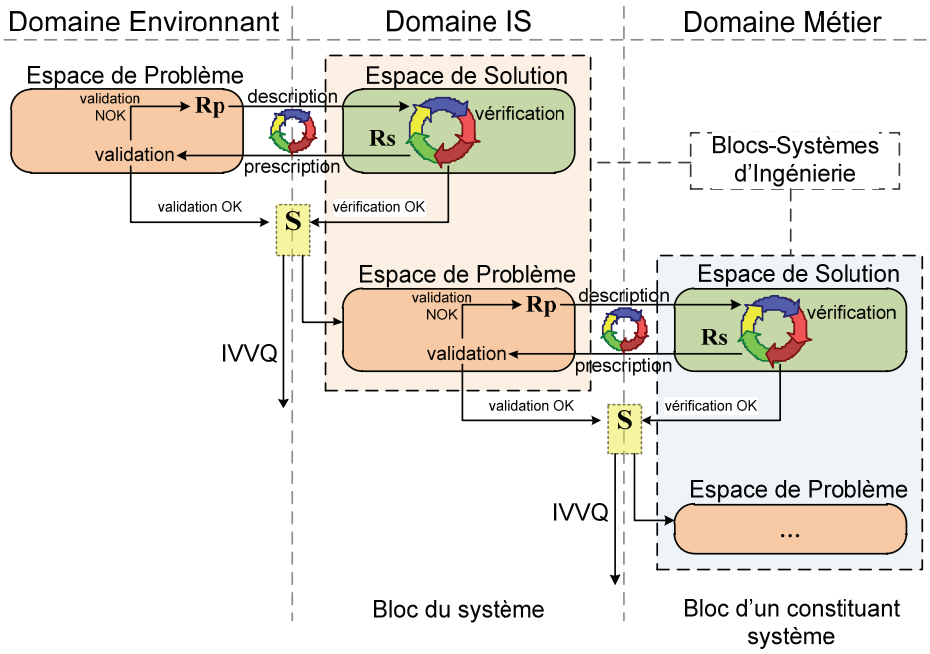


Figure 1.30 : Procédé itératif d'IS par blocs-système

1.2.1.2 Spécification des systèmes sociotechniques

L'extension du cadre des PF aux systèmes sociotechniques par Hall et Rapanotti (2005) considère que le comportement des utilisateurs du système doit être étudié dès l'analyse des exigences. C'est-à-dire que l'humain, comme la machine, doit faire l'objet de descriptions optatives.

Par comparaison avec le domaine de la machine, qui est un domaine causal (prédictif), Hall et Rapanotti qualifient le domaine de l'humain de domaine docile (*en*. biddable) pour exprimer le

manque de prédiction sur le comportement de l'humain. Ainsi, les exigences imposées à l'humain se limitent à des contraintes sur les actions permises entre l'utilisateur et la machine.

Comme résultat, ils proposent de faire la réification du modèle de Gunter, et al. (2000) (Figure 1.25) en ajoutant un troisième domaine : l'humain H avec sa connaissance K (Figure 7).

Avec l'introduction de l'humain H, deux nouvelles spécifications apparaissent :

- Spécifications UI qui déterminent l'interface Homme-Machine ;
- Spécifications I qui déterminent les connaissances et le comportement qui est attendu de l'humain comme un composant du système sociotechnique.

Ainsi, le prédicat de satisfaction des besoins 1-1 se réécrit sous la forme :

$$W \wedge S \wedge I \wedge UI \Rightarrow R \qquad 1\text{-}4$$

qui pourrait servir à définir, dans notre cas, les connaissances K que le rondier doit acquérir pour utiliser notre SIAC.

1.2.2 Autres approches

Nous avons considéré que les travaux de Bjørner (2009) en « Domain Engineering », de van Lamsweerde (1998b) en « Requirement Engineering », et de Fusaoka et al. (1983) et de Pétin, et al. (2006) en « Automation Engineering » apportent des compléments aux artéfacts clés de modélisation système.

1.2.2.1 Domain Engineering

Comme nous l'avons vu dans les sections précédentes, il est important de comprendre le domaine ou l'environnement qui pose les besoins système. C'est aussi l'avis de Bjørner (2009) qui, à travers l'ingénierie des domaines, propose une autre formalisation de la spécification. Ainsi, ses travaux en informatique sont guidés par le prédicat 1-5 (⊨ est l'opérateur de modèle qui signifie « S est un modèle de R »).

$$D, S \vDash R \qquad 1\text{-}5$$

Ce prédicat exprime le dogme : « *Before Software can be designed we must understand the Requirements, and before Requirements can be expressed we must understand the Domain.* »

Définition 1.23 : Domaine (*en*. Domain, Bjørner (2009))

By a domain, or, more precisely an application domain, we shall understand (i) a suitably delineated area of human activity, that is, (ii) a universe of discourse, something for which we have what we will call a domain-specific terminology, (iii) such that this domain has reasonably clear interfaces to other such domains.

Définition 1.24 : Description d'un domaine (*en*. Domain Description, Bjørner (2009))
By a domain description we shall understand (i) a set of pairs of informal, for par exemple, English language, and formal, say mathematical, texts, (ii) which are commensurate, that is, the English text "reads" the formulas, and (iii) which describe the simple entities, operations, events and behaviours of a domain in a reasonably comprehensive manner.

Par « software design » nous comprenons la spécification du programme, et le domaine représente l'environnement dans lequel le programme va être utilisé.

Nous retenons de ces travaux la nécessité de décrire le domaine d'application que nous assimilons au contexte du système à faire. Ainsi, une bonne compréhension du domaine et une bonne description permettent dès les phases initiales d'un projet de diminuer le risque d'interprétations erronées et de proposer une solution adaptée au domaine.

1.2.2.2 Requirement Engineering

Afin d'enlever l'ambiguïté introduite par Jackson entre exigence et besoin, van Lamsweerde (2001) utilise le concept d'objectif pour exprimer ce qui est demandé (partie optative), et réalise une évolution vers exigences, représentant la spécification du système à réaliser.

Définition 1.25 : Objectif (*en*. Goal, van Lamsweerde (2001))
A goal is an objective the system under consideration should achieve. Goal formulations thus refer to intended properties to be ensured; they are optative statements as opposed to indicative ones, and bounded by the subject matter.

Pour décrire ce qui est connu de l'environnement, van Lamsweerde utilise le terme hypothèse.

Définition 1.26 : Hypothèse (*en*. Assumption, van Lamsweerde, et al. (1998a))
An assumption is a fact taken for granted about agents in the environment. Unlike goals, assumptions need not be refined nor enforced. They often appear as auxiliary assertions needed to prove the correctness of refinements or operationalizations. Assumptions are tentatively true and are likely to change.

Contrairement aux exigences, les hypothèses ne peuvent pas être satisfaites par le système à faire ; elles peuvent être satisfaites grâce à des normes et des règlements, des lois physiques, … Les hypothèses correspondent par rapport au modèle de référence à la partie indicative W des exigences.

Van Lamsweerde (2001) propose une autre formalisation de la spécification selon le prédicat de satisfaction des objectifs :

$$R, As, D \vDash g \qquad \text{1-6}$$

où R représente l'ensemble des exigences, As représente l'ensemble des hypothèses de l'environnement, D est un ensemble de propriétés du domaine et g un objectif dans l'ensemble G d'objectifs. Ce prédicat doit être valable pour tous les objectifs $g \in G$, avec $R, As, D \nvDash false$.

Les travaux de van Lamsweerde ont donnée naissance à la méthodologie KAOS[23] (van Lamsweerde, et al., 1998b). Cette méthode d'ingénierie des exigences guidée par les objectifs est supportée par l'outil Objectiver[24].

1.2.2.3 Automation Engineering

En 1983, Fusaoka, et al. (1983) postulent que la conception de systèmes d'automatisation consiste à définir les règles de contrôle (inconnues) de la dynamique (connue) d'un système physique, à partir des objectifs comportementaux (connus) qui doivent être atteints :

$$Control\ Rules \wedge Dynamics \supset Goal \qquad 1\text{-}7$$

Lamboley (2001) revisite la formulation du paradigme d'automatisation proposé par Fusaoka et l'interprète de manière plus générale selon le prédicat 1-8. A noter que l'opérateur ⇒ (équivalent à ⊃) concrétise mieux la relation de spécification.

$$\underbrace{Processus\ de\ commande \wedge Processus\ opérant}_{\substack{Système\ concret \\ (lois\ physiques, comportements)}} \Rightarrow \underbrace{"Système"}_{\substack{Système\ abstrait \\ (missions, objectifs, propriétés)}} \qquad 1\text{-}8$$

A cause de la difficulté d'assurer à la fois la correction intrinsèque de chacun des modèles vis à vis de leurs règles de construction et d'application ainsi que leur cohérence globale (consistency) vis à vis du comportement attendu du système, Lamboley propose de différer l'utilisation de représentations orientées métiers dans la phase de spécification par une représentation commune entre tous les acteurs selon le prédicat :

$$Spécification\ de\ la\ commande \wedge Spécification\ du\ processus \Rightarrow Spécification\ "système" \qquad 1\text{-}9$$

Ce résultat est exploité aussi par Pétin, et al. (2006) qui formalise l'ingénierie des systèmes automatisés en utilisant le langage B. Ainsi, une notation simplifiée du prédicat 1-9 est donnée :

$$P_c \wedge P_o \Rightarrow S \qquad 1\text{-}10$$

Pour Pétin, la partie opérative P_o ainsi que les spécifications S sont connues, ce qui exige plusieurs boucles d'ingénierie afin de réaliser cette P_o et spécifier le S. Ce qu'il cherche à réaliser est la partie commande P_c d'un système automatisé.

Par analogie avec ce qui se fait dans l'informatique (Gunter, et al., 2000), si nous ajoutons l'environnement W, dans lequel sera intégré le système (un système ne peut pas exister sans l'environnement avec lequel il interagit), le prédicat 1-10 devient :

$$W \wedge P_c \wedge P_o \Rightarrow W \wedge S \qquad 1\text{-}11$$

[23] KAOS : Knowledge Acquisition in autOmated Specification
[24] Objectiver : www.objectiver.com

Nous pouvons observer que le terme à droite du prédicat 1-11 représente le terme à gauche du prédicat 1-1. Etant donnée la propriété de transitivité de l'opérateur ⇒ (implication) nous pouvons en déduire le prédicat :

$$W \wedge P_c \wedge P_o \Rightarrow R \qquad 1\text{-}12$$

Ce qui nous amène à reconnaitre le prédicat de satisfaction des besoins 1-3 définit par Gunter, et al. (2000).

1.3 Conclusions

Dans ce chapitre nous avons vu différents artéfacts de modélisation système proposés dans différents domaines d'ingénierie (système, informatique, …).

Comme décrit dans la première partie du chapitre, l'Ingénierie Système (IS) définit deux processus principaux : l'ingénierie du système et l'intégration, la vérification et la validation (IVV) du système. La notion de spécification est principalement utilisée dans le processus d'ingénierie du système et permet de fournir à la réalisation des spécifications bien formalisées et structurées de constituants systèmes.

La spécification, en ingénierie des exigences, est un point crucial de tout cycle de vie d'un système à ingénieriser. Même si la spécification n'est pas un processus proprement dit, la définition du système dans l'IS a pour rôle de spécifier ce que la solution doit être et doit faire. La spécification est le résultat d'une contractualisation entre un espace du problème, qui fournit des besoins, et un espace de la solution, qui fournit une solution, quel que soit le niveau dans le cycle de vie du système.

La spécification permet de concevoir et de réaliser un système composé par une partie structurelle (la machine) et une partie comportementale (le programme exécuté par la machine). Dans l'IS, la spécification est réalisée à partir des besoins et exigences des parties prenantes. Le comportement est donné par des fonctions système qui sont réalisées par une structure définie par une architecture de constituants systèmes.

Des travaux plus formels ont été présentés dans ce chapitre (Tableau 1.2) avec des liens entre objectifs, exigences et spécifications.

Tableau 1.2 : Correspondance entre prédicats 1-1, 1-5 et 1-6

Prédicat 1-1 (Gunter, et al., 2000)		Prédicat 1-5 (Bjørner D. , 2009)		Prédicat 1-6 (van Lamsweerde, 2001)	
W	World	D	Domain	As D	Environnement Domain
S	Specifications	S	Software design	R	Requirements
R	Requirements	R	Requirements	g	Goal

En comparant les trois prédicats de satisfactions des besoins/objectives de Gunter, et al. (2000), Bjørner (2009) et van Lamsweerde (2001) (Tableau 1.2) nous pouvons observer des correspondances entre les trois approches. Ceci-dit, afin de spécifier une solution à un ensemble de besoins/objectifs, il faut dans un premier temps chercher à bien connaître son environnement, voire le domaine auquel il appartient, c'est-à-dire le vrai problème à résoudre, dans lequel le système à faire doit être intégré afin d'apporter les phénomènes, dont la solution, satisfaisant les besoins décrits. Ainsi, avant de proposer une solution il faut déjà bien comprendre et décrire le problème à résoudre.

Ces travaux formels nous ont donné aussi les bases de notre raisonnement autour du processus de spécification système, ce qui nous a permis de séparer l'espace du domaine qui décrit le problème, dont l'environnement, de l'espace du domaine qui prescrit la solution, dont le système, ces deux domaines étant reliés par une relation contractuelle de spécification.

Tous ces travaux sont orientés autour la notion de spécification et fournissent des artéfacts de modélisation que nous pouvons utiliser pour spécifier notre SIAC. Parmi tous les artéfacts d'IS qui sont présentés dans ce chapitre nous retenons principalement ceux qui nous permettent de spécifier un procédé de modélisation du SIAC et donc ceux qui s'articulent autour du concept clé de bloc système : besoins, exigences, fonctions, constituants, architecture, …

De manière à synthétiser les différents artéfacts vus dans ce chapitre, nous proposons un procédé de modélisation basé sur le concept de bloc système (Figure 1.31) que nous pourrons appliquer pour spécifier le SIAC.

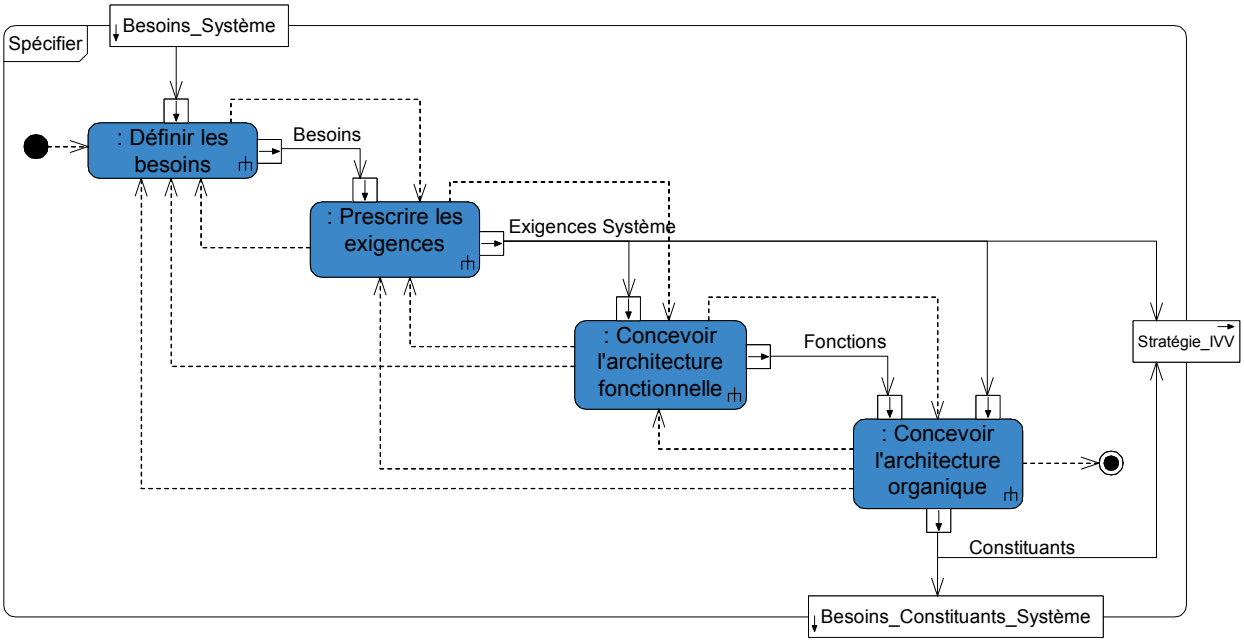


Figure 1.31 : Les phases de la spécification pour un bloc système donné

La mise en application de ce processus nécessite un langage d'ingénierie, comme par exemple SysML qui est un langage standardisé qui rentre dans la catégorie des langages semi-formels, et dont la syntaxe concrète est orientée métier de l'Ingénierie Système. Nous allons pour cela,

dans le chapitre suivant, confronter les artefacts de modélisation SysML aux artéfacts recensés dans ce chapitre.

Chapitre 2
Artéfacts de modélisation système en langage SysML

Ce chapitre a pour but de mettre en correspondance les artéfacts de modélisation système définis précédemment avec ceux proposés par le langage SysML, afin de le spécialiser par le développement d'artéfacts spécifiques à notre application, et de proposer un processus de modélisation système basé sur ce langage.

2.1 Artéfacts du langage SysML

SysML (2008) a pour objectif de supporter l'Ingénierie Système Basée sur des Modèles (ISBM). Il est issu de l'intérêt des ingénieurs système pour la conception orientée objet et le langage UML, ce dernier étant orienté vers l'ingénierie informatique, et fut initié par une collaboration entre l'INCOSE et l'OMG en 2001. Ainsi, SysML a été conçu comme une extension d'UML2 utilisant des stéréotypes, des méta-propriétés (*en*. tagged values) et des contraintes pour minimiser les difficultés de son implémentation dans les outils UML existants. Actuellement, la version officielle de SysML est 1.2, lancée en juin 2010.

SysML a été conçu dans l'idée de supporter l'analyse, la spécification, la modélisation et la vérification des systèmes complexes.

> Friedenthal et Burkhart *(2003)* : « *A system modelling language is intended to support analysis, specification, design, and verification of complex systems by :*
>
> - *Capturing the systems information in an efficient and precise manner that enables it to be integrated and reused in a broader context ;*
> - *Analyzing and evaluating the system being specified, to identify and resolve system requirements and design issues, and to support trade-offs ;*
> - *Communicating systems information correctly and consistently among various stakeholders and participants.* »

Quand il a été proposé, Friedenthal et Burkhart (2003) ont proposé 9 critères que ce langage devrait satisfaire :

- Facilité d'utilisation : le langage doit pouvoir être utilisé et compris (bien interprété) par un large public.

- Non-ambiguïté : le langage doit être basé sur des sémantiques bien définies avec une notation non-ambiguë, donnant lieu à un ensemble consistant de vues de modèles qui adhèrent à des règles de bonnes constructions (théorie des langages).
- Précis : le langage doit spécifier la sémantique qui peut être transformée en une représentation formelle basée sur les mathématiques (voir méthodes Z, B, OCL, …). Ce critère doit faciliter l'exécution de la spécification et des modèles à tout niveau hiérarchique afin de valider les besoins et vérifier que le modèle satisfait bien à ces besoins.
- Complet : le langage doit supporter l'expression de tous les détails émanant de la modélisation de systèmes, de l'analyse à la spécification, la conception et la vérification.
- Adaptatif/taille du système : le langage doit supporter l'abstraction, l'élaboration et le raffinement (généralisation/spécialisation, décomposition, collection, vues multiples, …) pour fournir des solutions adaptées à la modélisation de systèmes complexes.
- Adaptatif/domaine : le langage doit fournir les moyens d'extension vers des domaines spécifiques (aérospatial, télécommunications, automobile, …)
- Evolutif : le langage doit permettre le changement, l'évolution et supporter la compatibilité avec les versions antérieures.
- Echange de modèles et de diagrammes : le langage doit supporter le mapping vers l'AP233[25] (ISO 10303-233) pour l'échange d'informations entre outils. L'AP233 et XMI vont fournir des mécanismes d'échange de données, qui pourront éventuellement être représentées dans d'autres langages de modélisation (diagrammes comportementaux, IDEF0, langages formels, …) aussi bien que des langages/outils de gestion de besoins et d'autres outils/modèles d'analyse et de conception. En plus de l'échange sémantique, le langage doit supporter un échange de diagrammes pour faciliter l'échange des modèles entre outils.
- Indépendant du processus et de la méthode : le langage doit pouvoir supporter des processus d'ingénierie système tels que les standards EIA 632 (1999), ISO/IEC 15288 (2008) et non contraindre un choix de processus ou méthode.

Malgré ces prescriptions, SysML, du fait qu'il s'agit d'un langage de modélisation sans méthode de modélisation associée, reste assez difficile à mettre en pratique, en particulier par ceux qui ne connaissent pas UML.

> Estefan *(2008)* : « *It is important to note that the UML and SysML are not software or systems methodologies but rather visual modelling languages that are agnostic to any one specific methodology.* »

D'autres langages de modélisation existent, comme par exemple MARTE (Modeling and Analysis of Real-Time and Embedded Systems) ou AADL (Architecture Analysis and Description Language). MARTE (2009), comme SysML, est un profil UML2 pour l'analyse et la modélisation de systèmes temps-réel et embarqués. AADL (2009) est un langage de conception d'architecture standardisé par SAE (Society of Automative Engineers) destiné, comme MARTE, à la conception

[25] Application Protocol 233

et à l'analyse de systèmes embarqués complexes et temps-réel. La différence entre ces langages de modélisation et SysML est le niveau d'abstraction des modèles réalisés. De ce point de vue nous considérons SysML comme un langage plus générique qui peut être utilisé pour la modélisation d'un système avec un niveau d'abstraction plus élevé que MARTE et AADL qui sont orientés métiers et spécifiques au domaine d'application temps-réel et embarqué. Il est à noter que ces langages peuvent coexister dans le même projet (*ex*. SysML et MARTE) pour spécifier un même système à des niveaux de détails différents. La généricité de SysML ainsi que notre expérience sur UML nous a conduits à le choisir comme langage de spécification pour notre application. L'outil qui nous a permis de modéliser notre application est Artisan Studio Uno[26].

En 2010, une journée thématique organisée par AFIS autour du langage SysML a mis en discussion l'universalité de ce langage pour la modélisation système. Cette journée a rassemblé des industriels et des scientifiques du monde académique pour confronter leurs pratiques de SysML. Notre intervention (Panetto, et al., 2010) a posé des éléments sur l'utilisation de SysML comme langage de spécification. Il ressort de cette journée que ce langage laisse un grand nombre de questions ouvertes.

Dans SysML nous pouvons retrouver des artéfacts d'IS que nous avons illustrés dans le Chapitre 1 mais aussi des artéfacts connexes qui sont utilisés dans un procédé de modélisation itératif et collaboratif. C'est pour cela que nous allons présenter d'abord les artéfacts SysML rapportés aux artéfacts du chapitre précédent et ensuite les artéfacts SysML connexes, en nous limitant cependant aux objets de modélisation proposés par ce langage.

2.1.1 Artéfacts du langage SysML relatifs aux artéfacts d'IS

Nous allons présenter les artéfacts SysML qui sont en relation avec les artéfacts d'IS retenus. Ces artéfacts sont relatifs aux objets de modélisation utilisés par les processus de modélisation par bloc système : système avec la vision exigences, comportement, structure, ainsi que la vérification et la validation.

Piliers SysML

SysML supporte les activités du cycle de vie d'un système comme la spécification des exigences, la modélisation des différents constituants avec leur structure et leur comportement, l'intégration, ainsi que la spécification des scénarios pour la vérification et la validation. Pour cela, les diagrammes de SysML ont été catégorisés selon quatre piliers : structure, comportement, exigences et paramétriques, avec des relations d'allocation entre eux.

Comme certains auteurs (Holt, et al., 2008), nous limitons cette structuration à trois piliers (Figure 2.1) : structure, comportement et exigences. Ceci est dû à l'utilisation des diagrammes paramétriques dans la représentation de la structure (il s'agit d'une spécialisation du

[26] Artisan Studio Uno : www.atego.com/products/artisan-studio-uno

diagramme de bloc interne) et au fait que les contraintes représentées dans ce type de diagramme ne peuvent être reliées que sur des éléments de type bloc.

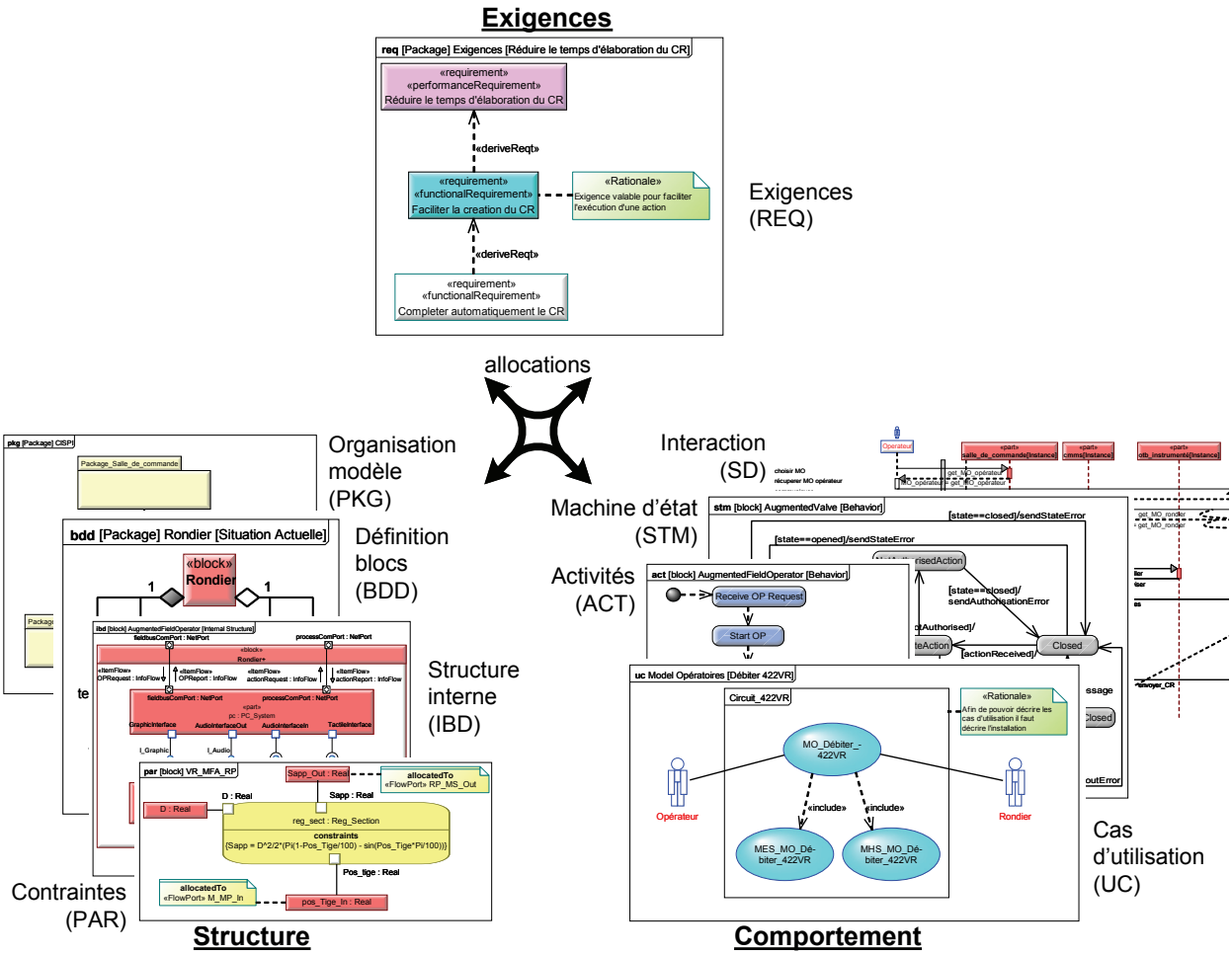


Figure 2.1 : Trois piliers de SysML

Ces piliers sont organisés de telle façon que nous pouvons de prescrire un système selon trois visions complémentaires : exigences, comportement et structure, de l'abstrait vers le concret.

Système

Comme nous avons vu, le système est l'artéfact-clé de l'IS. SysML ne présente pas de notion de système, mais la représentation en SysML d'un système peut cependant être réalisée à l'aide de l'objet de modélisation « block ». Il est à noter que cette notion de bloc dans SysML n'a pas de lien avec la notion de « bloc système » utilisée dans l'IS pour désigner une structuration des processus d'ingénierie (« rationale ») pour un niveau de spécification dans le cycle de vie d'un système. En SysML, le bloc est l'unité de structure de base et peut être utilisé pour représenter des éléments du hardware, du software, du personnel, ou tout autre élément d'un système.

Afin de distinguer le Système à Faire d'autres systèmes (par exemple systèmes externes), Weilkiens (2008) ajoute dans son profile orienté IS, SYSMOD, le stéréotype « system » qui étend l'objet de type « block », ainsi que le stéréotype « subsystem » qui représente une unité

intégrée dans un système plus large. Un autre stéréotype pour la distinction d'un système est donné par Holt et Perry (2008) avec le nom de « System of interest ».

Exigences

SysML, par rapport à d'autres langages de modélisation, ajoute la possibilité de décrire les exigences à l'aide du stéréotype « *requirement* ». Ce stéréotype de classe présente deux propriétés : un identifiant et un texte descriptif de l'exigence. Les diagrammes d'exigences (*Requirement Diagram – REQ*) permettent de construire la hiérarchie des exigences en utilisant des mécanismes tels que la composition et la dérivation, ainsi que des relations avec d'autres éléments de modélisation pour tracer, raffiner, satisfaire ou vérifier ces exigences (Tableau 2.1).

Tableau 2.1 : Liens entre les exigences

Stéréotype	Classe de base	Description
« NestedClassifier »	N/A	Lien de composition des exigences pour leur structuration dans des arborescences
« Trace »	« Dependency »	Lien sans contrainte entre un élément du modèle et une exigence
« DeriveReqt »	« Trace »	Lien pour la dérivation des exigences
« Copy »	« Trace »	Lien pour la création d'une copie d'une exigence avec le même texte mais nom et identifiant différent
« Satisfy »	« Trace »	Lien de satisfaction entre un élément du modèle et une exigence
« Verify »	« Trace »	Lien d'un cas de test à une exigence
« Refine »	« Abstraction »	Lien pour ajouter des détails à une exigence

SysML permet aussi de spécialiser cette notion de « requirement ». Par exemple, les spécifications SysML présentent six spécialisations avec des contraintes supplémentaires (Tableau 2.2). A l'aide de ce mécanisme, il est possible d'ajouter d'autres spécialisations d'exigences.

Même s'il est considéré par certains auteurs (Holt, et al., 2008) comme un élément de structure, le « requirement » est un élément transversal (*en*. crosscutting construct) qui peut être utilisé dans tous les diagrammes de SysML, en particulier pour montrer les relations de satisfaction par les éléments du modèle.

Tableau 2.2 : Stéréotypes d'exigences supplémentaires

Stéréotype	Classe de base	Contraintes	Description
« extended-Requirement »	« requirement »	N/A	Un stéréotype qui ajoute des attributs généralement utiles pour la description des exigences Propriétés : • source : String • risk : RiskKind • verifyMethod : VerifyMethodKind

Stéréotype	Classe de base	Contraintes	Description
« functional-Requirement »	« extended-Requirement »	Satisfaite par une opération ou un comportement	Exigence qui spécifie une opération ou comportement qu'un système ou partie d'un système doit réaliser
« interface-Requirement »	« extended-Requirement »	Satisfaite par un port, connecteur, « item flow » et/ou « constraint property »	Exigence qui spécifie les ports pour connecter des systèmes ou partie d'un système et (optionnel) peut inclure le flux qui passe par le connecteur et/ou les contraintes d'interface
« performance-Requirement »	« extended-Requirement »	Satisfaite par une « value property »	Exigence qui quantifie la mesure dans laquelle un système ou partie d'un système satisfait une capacité requise ou condition
« physical-Requirement »	« extended-Requirement »	Satisfaite par un élément de structure	Exigence qui spécifie les caractéristiques physiques et/ou les contraintes physiques d'un système ou partie d'un système
« design-Constraint »	« extended-Requirement »	Satisfaite par un bloc ou « part »	Exigence qui spécifie une contrainte d'implémentation du système ou partie d'un système comme l'utilisation obligatoire d'un composant sur étagère

Ce que nous pouvons critiquer est la difficulté de choisir entre l'utilisation du lien de composition et celui de dérivation pour la définition des exigences. La composition des exigences, telle que présentée dans les spécifications de SysML, est utilisée pour structurer les exigences, principalement liées entre elles par type (exigences d'interface, physiques, …) ou nature (exigences liées aux mêmes objets). La dérivation quant à elle peut être utilisée pour lier des exigences qui n'ont pas le même type, mais qui sont de même nature.

Vue comportement

En SysML, le comportement d'un système peut être modélisé à l'aide de quatre diagrammes :

- Le diagramme de cas d'utilisation (*Use Case Diagram - UC*) offre une description de fonctionnalités de haut niveau obtenues par l'interaction entre systèmes.
- Le diagramme de séquence (*Sequence Diagram - SD*) permet de représenter les échanges entre différentes parties d'un système, et avec des systèmes ou acteurs externes au système. Il décrit, en particulier, les cas d'utilisation.
- Le diagramme d'activité (*Activity Diagram - ACT*) permet de représenter le flux de données et de contrôle entre différentes actions. En effet, ce diagramme est le plus adapté pour représenter une architecture fonctionnelle du point de vue structurel et comportemental.
- Le diagramme d'états (*State Machine Diagram – STM*) décrit les transitions entre états et les actions qu'un système réalise en réponse à des événements.

Vue structure

Avec l'ajout de la notion de bloc (qui étend la notion de classe d'UML), SysML a complété les possibilités d'UML en termes de modélisation de l'architecture organique d'un système. Cette architecture peut être réalisée avec SysML selon deux points de vue différents :

- Le diagramme de définition de blocs (*Block Definition Diagram* - *BDD*) décrit la composition hiérarchique d'un système (arborescence), ainsi que les associations entre les différents constituants d'un système. Ce diagramme peut également être utilisé pour représenter une décomposition fonctionnelle des activités.
- Le diagramme de bloc interne (*Internal Block Diagram* - *IBD*) décrit la structure interne d'un système en termes de parties (parts), ports, interfaces et connecteurs.

En complément des blocs, les paquetages sont utilisés pour structurer et organiser les modèles. Si le diagramme de paquetage permet de modéliser cette structuration du modèle, les paquetages peuvent aussi être utilisés dans d'autres diagrammes pour grouper un ensemble d'éléments du modèle.

Interfaces

Un élément que nous avons vu dans le méta-modèle de l'AFIS est l'interface. Dans SysML, l'élément qui correspond le mieux à ce concept est le connecteur utilisé dans les IBD, mais il existe également les éléments de type « interface » qui, à travers les ports standards, permettent de spécifier les services fournis ou requis par un bloc. Les objets de type « interface » sont utiles par exemple pour relier deux systèmes de natures différentes.

Vérification et validation

Un élément important pour la vérification et la validation des exigences est la définition des cas de test (« testcase »), qui représentent des scénarios de test des éléments du modèle, afin de voir s'ils répondent bien aux exigences et aux attentes des clients. En temps que stéréotype de comportement (« Behaviour »), un cas de test peut être représenté par n'importe quel diagramme comportemental (d'état, d'activité, de séquence) et permet de vérifier un ensemble d'exigences sur la solution proposée. Nous pensons, comme le montre Balmelli (2007), qu'une représentation pertinente des cas de test est le diagramme d'activité parce qu'il est possible de définir de vrais tests à l'aide d'éléments de type nœuds de décisions.

Ce que nous pouvons reprocher à SysML pour la vérification et la validation des modèles avant la réalisation effective est le manque de sémantique opérationnelle, qui rend difficile leur exécution. Ceci nécessite la définition de transformations vers des langages appropriés qui permettent par exemple la simulation ou le model checking. Il existe cependant des travaux restreints qui définissent des sémantiques opérationnelles, comme pour les diagrammes d'état (Beeck, 2002) ou les diagrammes d'activité (Jarraya, et al., 2009). De plus, l'OMG a récemment mis en place une fondation, Foundational UML (FUML, 2009), pour étudier la sémantique d'UML et

proposer un sous-ensemble exécutable qui peut être utilisé pour définir, dans un style opérationnel, la sémantique structurelle et comportementale des systèmes.

La vérification et la validation des modèles SysML est étudiée plus en détails à la section §2.3.

Processus

SysML a été intégré dans des cadres de modélisation tels que le modèle « 4+1 » (Santos, et al., 2009), Zachman (Morel, et al., 2007) et a été appliqué sur des processus des normes de l'IS tels que IEEE 15288 (Turki, 2008), EIA-632 (Bonhomme, 2008).

Un procédé de modélisation est important pour la mise en pratique d'un langage de spécification. Le manque de maturité du langage SysML fait que beaucoup des procédés proposés (Weilkiens, 2008; Holt, et al., 2008) ne se sont pas encore assimilés dans les pratiques des ingénieurs systèmes.

SysML reste encore un langage controversé pour beaucoup d'ingénieurs systèmes qui se sont retrouvés, comme nous, dans l'« obligation » d'utiliser un langage de modélisation système adopté par l'INCOSE pour faire de l'Ingénierie Système. C'est pour cela que nous nous sommes posés la question :

« Comment faire une spécification avec SysML ? »

2.1.2 Artéfacts connexes du langage SysML

Nous avons recensé d'autres artéfacts du langage SysML qui complètent la liste des artéfacts d'IS. Ces artéfacts connexes portent d'une part sur les bonnes pratiques d'UML et SysML, et d'autre part sur des éléments indispensables dans une ingénierie multidomaine collaborative.

Bonnes pratiques d'UML en SysML

La partie d'UML2 réutilisée par SysML est appelée UML4SysML (Figure 2.2). Des diagrammes d'UML2 ont été étendus pour recouvrir les besoins spécifiques à l'IS. En héritant d'UML, SysML bénéficie de mécanismes d'extension qui lui permettent de s'adapter au domaine et d'être évolutif. En outre, les diagrammes UML permettant la modélisation du comportement d'un système ont été retenus, avec toutefois quelques ajouts dans le cas du diagramme d'activité. D'ailleurs, à partir de la version 1.2 de SysML OMG à ajouté la possibilité d'utiliser le concept d'instance, identique à celui d'UML mais applicable au niveau des blocs.

De par son héritage d'UML, SysML est très teinté logiciel et la juxtaposition de ses vues/diagrammes ne peut, d'après (Wippler, 2010), correspondre véritablement à des modèles orientés IS. En ce sens, la typologie des « modèles » peut ne pas être toujours adaptée au raisonnement système, et le langage manque de certains concepts fondamentaux (par exemple fonction, dans le sens mathématique du terme) : « *Modéliser ce n'est pas dessiner, ou « ingé-*

nieurer » c'est raisonner et non représenter, ou avant de décrire une solution, il faut la trouver, ou … »

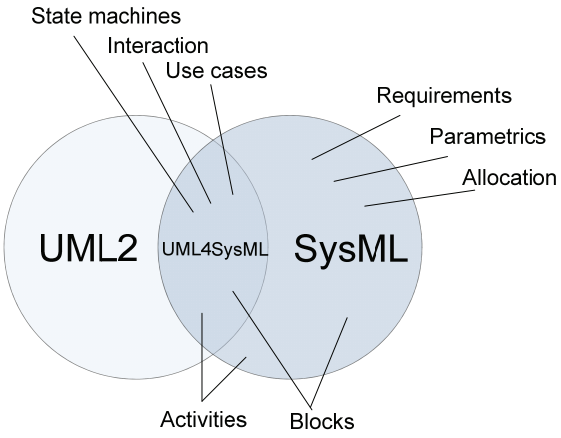


Figure 2.2 : Relation entre UML et SysML

Du fait que certains diagrammes UML et SysML sont similaires (*ex*. diagramme de classe et diagramme de bloc), se pose la question de l'utilisation conjointe de ces langages pour la modélisation d'un même système, comme par exemple dans le cas de son modèle de données (Canals, et al., 2010).

Contexte

Même si SysML ne présente pas d'éléments de modélisation de type « contexte » dans les exemples publiés par sa spécification (SysML, 2008), des modèles de contexte sont présentés sous la forme d'un IBD.

En parallèle, Weilkiens (2008) utilise le BDD pour représenter la composition du contexte et les associations entre le système et les éléments du contexte, avec ou sans flux d'informations. Il utilise également l'IBD qui, en plus de la représentation des flux échangés, permet de modéliser les ports et les interfaces du système avec son environnement.

Une autre façon de représenter le contexte est l'utilisation par Holt et Perry (2008) du diagramme de cas d'utilisation. Ainsi, ils définissent la frontière du système et les utilisateurs ou systèmes qui communiquent avec le système par l'intermédiaire des cas d'utilisations. Deux types de contextes sont présentés :

- le contexte d'affaire (« business context ») qui montre les exigences d'affaire qui relient le système avec les parties prenantes (« stakeholders ») et non pas les utilisateurs directs du système. Un point intéressant est l'utilisation de l'élément de type acteur (« actor ») pour définir les parties prenantes ;
- le contexte du système (« system context ») qui montre les exigences fonctionnelles et non-fonctionnelles reliant le système avec les utilisateurs directs du système.

Holt et Perry (2008) présentent aussi des contextes relatifs au projet et des contextes en fonction des points de vue des différentes parties prenantes.

Scénarios opérationnels

Même si nous l'avons classé dans les éléments permettant de faire la modélisation comportementale, le diagramme de séquence est également approprié pour décrire des scénarios opérationnels, ces derniers représentant les échanges entre le système à faire et les systèmes de l'environnement, ou entre les différents constituants d'un système. Ces scénarios peuvent être un raffinement des exigences fonctionnelles, ajoutant des détails sur le fonctionnement désiré du Système à Faire.

Modèles d'analyse

Le diagramme paramétrique (*Parametric Diagram - PAR*) présente des contraintes sur les propriétés du système (performance, fiabilité, …), généralement représentées sous la forme d'équations mathématiques, et permet d'intégrer les modèles de spécification avec les modèles d'analyse.

Les travaux de Peak, et al. (2007) proposent de transférer des éléments de ces diagrammes vers des outils de simulation et de calcul, tels qu'Excel, MATLAB/Simulink et Mathematica. Cette transformation est dors et déjà implantée dans des outils tels que ParaMagic[27] ou Artisan Studio par exemple, ce qui permet d'interfacer des modèles SysML avec des outils de simulation et de calcul commerciaux.

Types de valeurs, unités et dimensions

Dans une ingénierie multidomaine où l'échange de modèles est essentiel pour la conception et la réalisation du système, l'utilisation des mêmes unités de mesure est primordiale. Pour cela, SysML a introduit l'élément de modélisation « valueType » (type de valeur). Un type de valeur peut inclure une dimension (« dimension ») qui identifie la quantité physique (par exemple la longueur), et une unité (« unit ») qui identifie l'unité de mesure (par exemple le mètre).

« Crosscutting constructs »

De la même manière que les exigences qui peuvent être utilisées dans tous les diagrammes, SysML a introduit le mécanisme d'allocation pour relier des éléments de natures différentes. Nous pouvons notamment citer l'allocation des fonctions (activités représentant le comportement) aux constituants (blocs représentant la structure), la projection des constituants logiques sur les constituants physiques, …

Afin de faciliter la traçabilité des liens qui sont établis par l'allocation ou par les exigences, SysML introduit de plus des vues tabulaires, ce qui facilite la mise en relation des éléments du modèle et donne une vue d'ensemble sur la complétude des relations. Cependant, Caron, et al. (2010) mettent en évidence certaines limitations au niveau de la traçabilité des exigences.

[27] ParaMagic est un plugin pour MagicDraw commercialisé par NoMagic

Points de vue

Du fait que l'IS est transversale à une multitude des domaines métiers, SysML introduit les concepts de « View » et « viewpoint ». Introduite par ISO/IEC 42010 (2007), une vue (« View ») est une description de l'ensemble du système pour un domaine métier spécifique. Un point de vue (« viewpoint ») détermine les ressources et les règles de construction d'une vue.

2.2 Procédé de modélisation système en SysML

Un langage de spécification est un système formel avec une syntaxe, une sémantique et des règles de preuve (Bjørner, et al., 2008). A cela nous ajoutons la nécessité d'avoir un procédé de modélisation, une méthodologie générique pour la spécification, ainsi que des règles de construction telles que les patrons de conception.

Par rapport au langage de spécification, le principal bénéfice de SysML est de fournir une syntaxe et une sémantique cohérentes.

> Willard *(2007)* : « *To provide system engineers with a standard and comprehensive system specification paradigm. This enables consistency in the syntax and underlying semantics of the specification (this includes requirements, diagrams, models, and descriptions). The graphic symbols used in system diagrams have unambiguous meaning, reducing the likelihood of miscommunication.* »

SysML définit une syntaxe abstraite (ou le schéma), définie par son méta-modèle, et une syntaxe concrète (ou la notation) structurée dans les piliers. La sémantique de SysML est définie par son méta-modèle (Figure 2.3) et les règles OCL associées. En effet, la sémantique d'UML est reprise pour partie dans SysML et de nouveaux éléments sont ajoutés, avec des significations particulières (par exemple la notion de « block »). Les règles OCL permettent quant à elles de brider certains degrés de libertés d'UML, en fonction du niveau système visé par SysML.

Ce qui manque à SysML pour être considéré comme un langage de spécification est le système de preuves. Par exemple, même si la notion de raffinement existe en SysML (« Refine[28] »), cette notion n'a pas de signification prouvable, c'est-à-dire qu'on ne peut pas vérifier la correspondance entre l'exigence qui est raffinée et l'élément qui la raffine. Il manque également à SysML une sémantique opérationnelle nécessaire à l'exécution des modèles.

Comme nous l'avons vu dans le premier chapitre, il existe des artéfacts pour les objets de modélisation système et des artéfacts pour les « rationales » précisant des règles de constructions des exigences, fonctions et constituants système. Du fait que SysML définit que les objets de modélisation sans règles de constructions associées, nous avons proposé de bâtir un procédé de modélisation système basé sur les artéfacts d'IS retenus, et plus particulièrement en considérant comme élément pivot l'artéfact de bloc système (Figure 1.15).

[28] « Refine » : lien de raffinement entre une exigence et un élément du modèle, habituellement un cas d'utilisation

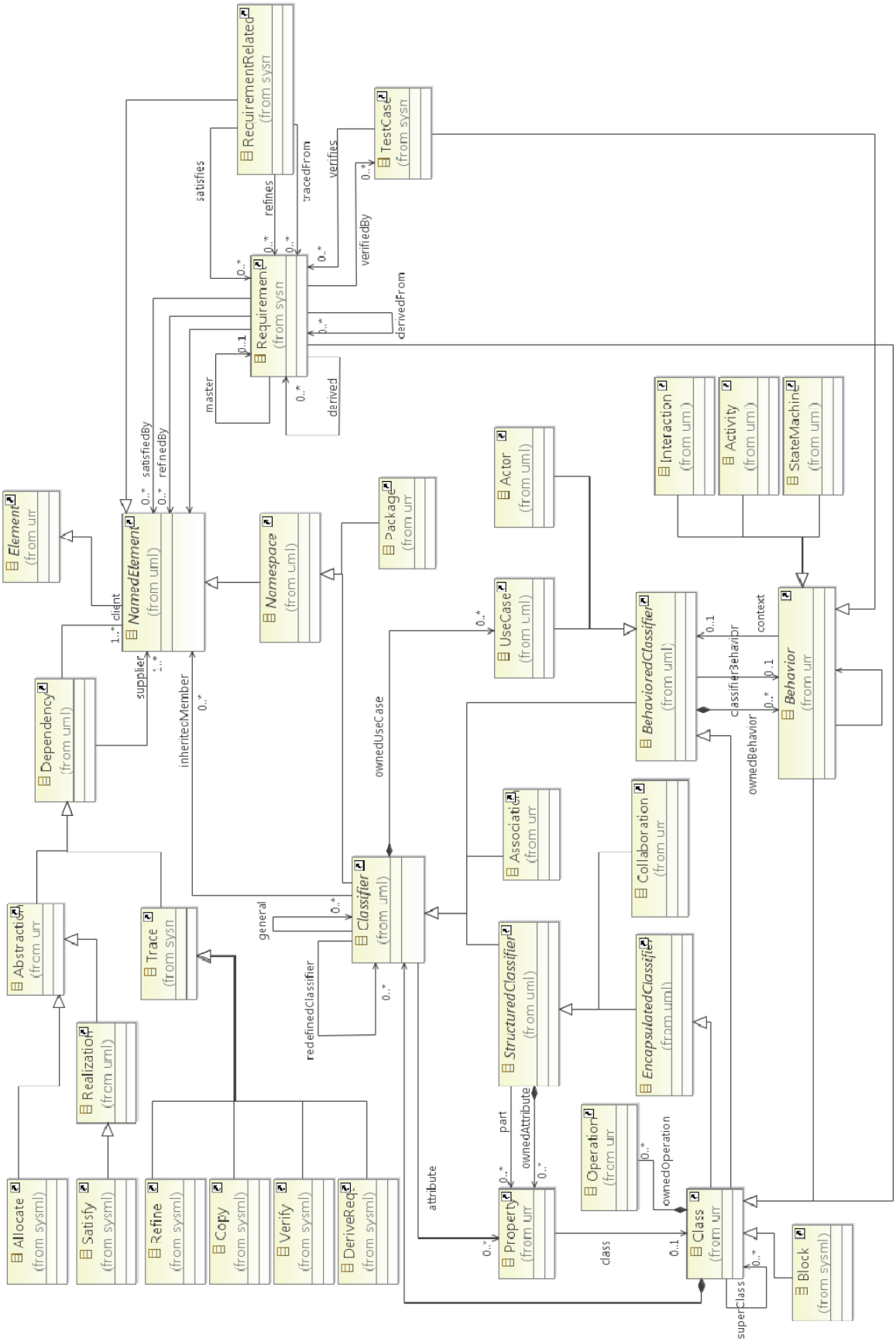


Figure 2.3 : Méta-modèle partiel de SysML utilisé par l'outil Topcased (2010)

Ci-dessous nous allons décrire plus en détail les activités liées à la spécification (Figure 1.31) d'un Système à Faire (SàF), chaque activité étant présentée dans une section séparée : définition des besoins avec SysML, prescription des exigences avec SysML, conception de l'architecture fonctionnelle avec SysML et conception de l'architecture organique avec SysML. Les activités de vérification/validation seront détaillées dans la section §2.3.

2.2.1 Définition des besoins avec SysML

Parmi les artéfacts de l'IS, les exigences sont des éléments essentiels dans l'ingénierie d'un système. Ce que SysML apporte de neuf par rapport à d'autres langages de modélisation est la possibilité de décrire textuellement les exigences avec le stéréotype de classe « Requirement ». Avec cet objet de modélisation on peut décrire tout type d'exigences, mais également les besoins, même si ces derniers n'ont pas de correspondance directe dans la spécification SysML

En effet, pour différencier les besoins des exigences, nous proposons un stéréotype spécialisé pour les décrire : « userNeed » (Figure 2.4). L'intérêt de ce stéréotype est la possibilité d'indiquer le lien de traçabilité entre les besoins des parties prenantes (parfois non structurés et ambigus) et les exigences qui sont ensuite prescrites pour répondre à ces besoins. Nous ajoutons comme propriétés de ce stéréotype la source du besoin (« userNeedSource »), qui correspond aux parties prenantes qui l'expriment, et le type de besoin (« userNeedType ») pour différencier par exemple un besoin fonctionnel d'une contrainte.

Comme la description des besoins nécessite la description des parties prenantes, nous proposons de les représenter à l'aide du stéréotype « stakeholder » de type « Actor ». Ceci permet d'assurer la traçabilité entre les besoins et les parties prenantes qui les expriment.

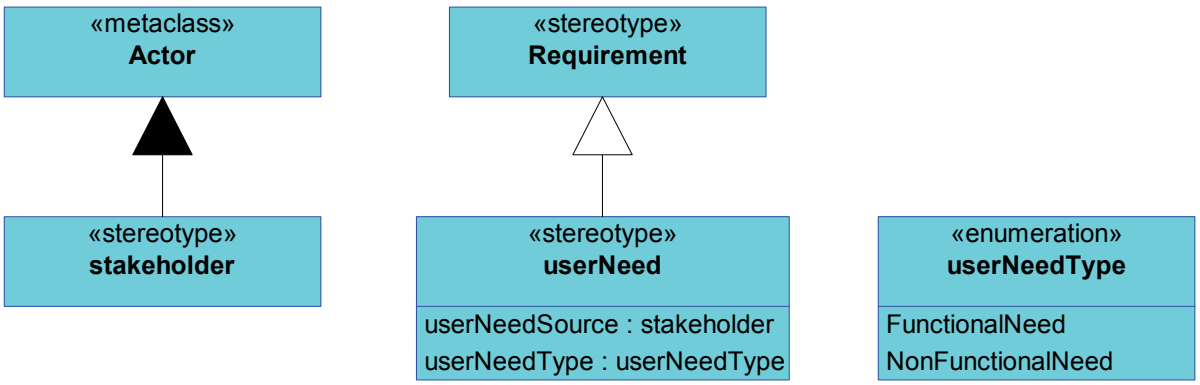


Figure 2.4 : Stéréotypes pour la description des besoins

La structuration des besoins peut être réalisée en utilisant le mécanisme SysML de composition.

Comme nous l'avons vu dans le premier chapitre, les besoins doivent décrire des phénomènes désirés par l'environnement, alors que les exigences doivent prescrire le système. Afin de faire le passage des besoins vers des exigences, nous définissons la règle suivante :

- Si le besoin est de type fonctionnel (fonctions désirées par l'environnement) alors il peut être dérivé en une exigence fonctionnelle qui doit être satisfaite par une fonction du système.
- Si le besoin est une contrainte alors il peut être dérivé en un autre type d'exigence.

2.2.2 Prescription des exigences avec SysML

La compréhension du besoin permet de définir la finalité, la mission et les objectifs du système. Un élément complémentaire qui résulte de cette définition du besoin est le recensement de l'ensemble d'acteurs ou de systèmes qui interagissent avec le SàF et qui définissent le contexte. Ainsi, une étape préalable à la prescription des exigences est la modélisation du contexte. Nous proposons de le représenter d'abord sous la forme d'un diagramme de bloc permettant de montrer la hiérarchie des éléments du contexte (le contexte est représenté comme un bloc qui est composé par les éléments du contexte). Ainsi, ce diagramme permet de définir pour la première fois le SàF (vision « boîte noire ») se qui justifie l'emplacement du contexte dans l'étape de prescription de la solution.

Ce diagramme de bloc est ensuite détaillé sous la forme d'un diagramme de bloc interne (relations entre le SàF et les éléments du contexte).

Au niveau de l'analyse des exigences entre un espace de problème et un espace de solution, nous proposons un procédé itératif de prescription qui illustre la vision compositionnelle de l'exigence. Ce procédé, représenté sous la forme d'un diagramme d'activité à la Figure 2.5, traite un flux d'exigences, qui passe par différentes activités synchronisées par un flux de contrôle, montrant l'ordre d'exécution de ces activités :

1. Dans l'espace de problème, une exigence est décrite à un espace de solution pour qui elle devient une exigence optative.
2. Dans l'espace de solution on combine l'exigence optative avec une exigence indicative représentant des connaissances métiers, pour définir une exigence pour la solution, non vérifiée.
3. Cette exigence est ensuite vérifiée. Si l'exigence n'est pas vérifiée alors on retourne vers la définition des exigences.
4. Si l'exigence est vérifiée, alors elle va être prescrite comme solution à l'espace de problème. Nous avons indiqué cette solution comme des exigences indicatives parce que l'espace de la solution lui amène de nouvelles connaissances.
5. Cette solution est ensuite validée dans l'espace de problème par rapport à l'exigence initiale. Si elle n'est pas validée alors on retourne vers la définition des exigences pour trouver une autre solution. Si la solution est conforme à l'exigence alors, par l'acceptation de la solution, elle devient une spécification ou la relation contractuelle entre le domaine de l'espace de problème et le domaine de l'espace de solution.

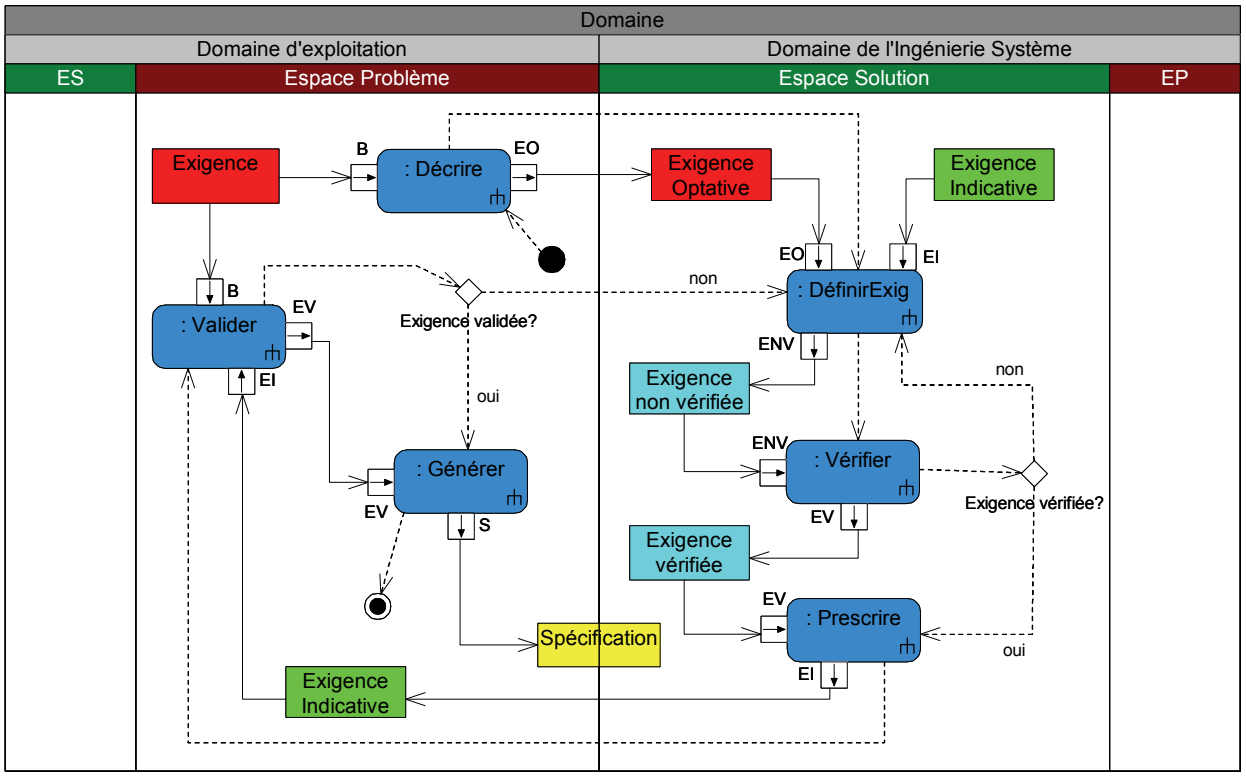


Figure 2.5 : Evolution des exigences entre un espace de problème et un espace de solution

Ce procédé de transformation de l'exigence illustre le changement d'état de l'exigence qui part d'un besoin pour arriver in fine être considéré comme une spécification entre un espace de problème et un espace de solution.

Pour illustrer ce procédé, nous pouvons l'appliquer sur le cas d'un appel d'offre. Dans une première itération de l'algorithme, l'appel d'offre est donné par un espace de problème (domaine du client) et la réponse est proposée par un espace de solution (domaine du fournisseur). On peut observer qu'avant de donner sa réponse, le fournisseur doit vérifier la solution qu'il va proposer et avant d'accepter cette proposition, le client doit valider que cela répond bien à ces besoins. On peut voir aussi qu'en fonction de la complexité du système à réaliser, le fournisseur peut faire appel à d'autres domaines métiers tels que électrique, mécanique, hydraulique, …

En complément de l'algorithme, nous proposons les règles suivantes :

- La description (élicitation) du besoin et la validation de la solution proposée doit toujours être faite dans l'espace de problème,
- La définition des exigences, leur vérification et la prescription ne peuvent être faites que dans un espace de solution.

Les connaissances métiers (normes, règles, …) qui permettent de répondre au besoin de l'espace de problème, étant des connaissances implicites du domaine de la solution, ne sont pas toujours représentées dans un modèle SysML. C'est pour cela que nous pensons qu'il est important de préciser dans les modèles l'utilisation de ces connaissances sous la forme de raisonnements (« Rationale »). En particulier, nous nous posons la question de l'utilisation des

mécanismes de dérivation et de composition entre exigences à partir de ces connaissances métier. En effet, les spécifications SysML proposent des liens de composition pour structurer les exigences, mais sans donner de mécanisme pour expliquer cette décomposition (il est impossible d'ajouter un « rationale »). Par contre, il est possible lors d'une dérivation de spécifier le raisonnement qui a amené à cette dérivation (Figure 2.6).

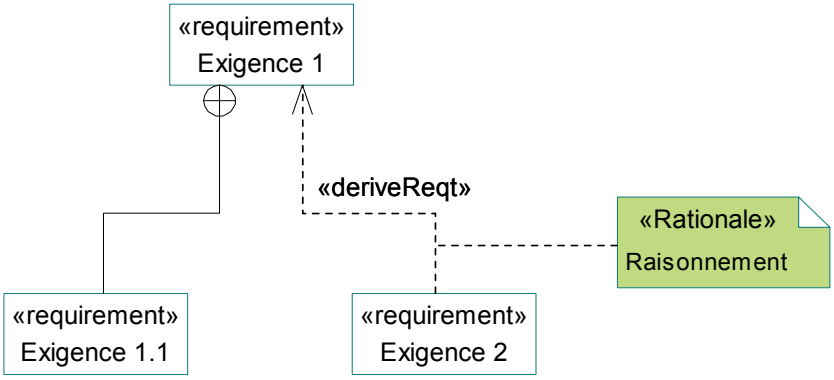


Figure 2.6 : Utilisation des liens de composition et de dérivation des exigences

Cette séparation des exigences en optative et indicative ne nous donne aucune information sur comment raisonner pour exprimer les exigences. Nous considérons qu'il faut raisonner en profondeur et non pas en largeur, c'est-à-dire qu'il faut prendre la finalité du système comme fil conducteur du raisonnement et le suivre pour arriver à exprimer les exigences du système à faire.

Un exemple de modèle qui permet de raisonner sur les exigences d'un système est la boucle cybernétique (Figure 2.7).

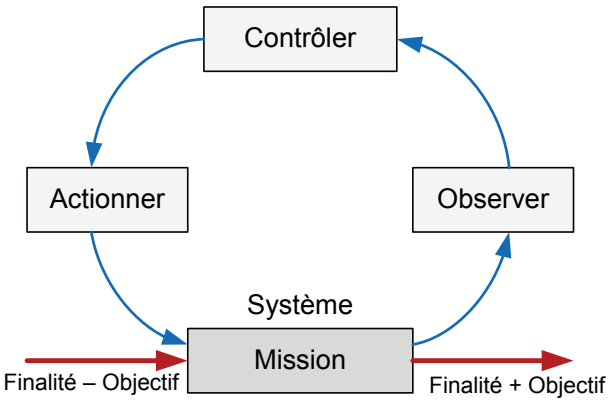


Figure 2.7 : Patron de raisonnement pour l'évolution des exigences : la boucle cybernétique

En effet, comme nous avons vu aussi dans les Problem Frames (Jackson, 2000), les besoins portent sur des phénomènes de l'environnement, ce que le système doit transformer ou apporter à l'environnement (finalité + objectif). Des questions peuvent ainsi se poser sur chaque élément de la boucle :

- Comment peut-on OBSERVER les phénomènes de l'environnement ?
- Comment peut-on ACTIONNER les phénomènes de l'environnement ?

- Comment peut-on CONTROLER l'actionnement à partir des observations ?

Le modèle cybernétique nous permet de définir des exigences sur des éléments du comportement et de la structure du système à réaliser. En effet, le processus d'observation et d'actionnement nous permet de définir des interfaces avec l'environnement du système étudié. Le processus de contrôle permet de définir des fonctions qui permettront de contrôler ces interfaces d'actionnement en fonction des observations réalisées.

Les exigences fonctionnelles peuvent être explicitées par des scénarios opérationnels. En SysML, les scénarios opérationnels peuvent être modélisés avec les diagrammes de séquences qui précisent chacun un cas d'utilisation du système, c'est-à-dire une fonction que le système doit réaliser. Le mécanisme approprié pour faire le lien entre les exigences et les cas d'utilisation est le « refine » qui explicite l'ajout de plus de détails à la description de l'exigence.

Toutes ces exigences, spécifiées en vue de la conception et la réalisation d'un système, doivent être vérifiées et validées sur les éléments du modèle (comportement et structure). Dans SysML, les cas de test (« testcase ») permettent ainsi de déclarer des scenarios pour vérifier et valider des exigences.

Les exigences vérifiés et validés servent comme spécifications pour les phases de conception de l'architecture fonctionnelle et de l'architecture organique.

2.2.3 Conception de l'architecture fonctionnelle avec SysML

En IS, la mission du SàF est de fournir des services à l'environnement dans lequel il va être intégré. Ces services, dans la vision systémique représentent des fonctions que le système réalise.

En effet, les services système sont des fonctions qui ont un résultat à la frontière du système avec l'environnement. Pour réaliser ce résultat, les services système utilisent d'autres fonctions, qui à leur tour utilisent d'autres fonctions. Cette décomposition peut être modélisée avec le patron de conception représenté à la Figure 2.8. D'après Bouhours (2010), ce patron de conception est un patron de conception « abimé », car il n'est pas possible de dire quand on doit s'arrêter.

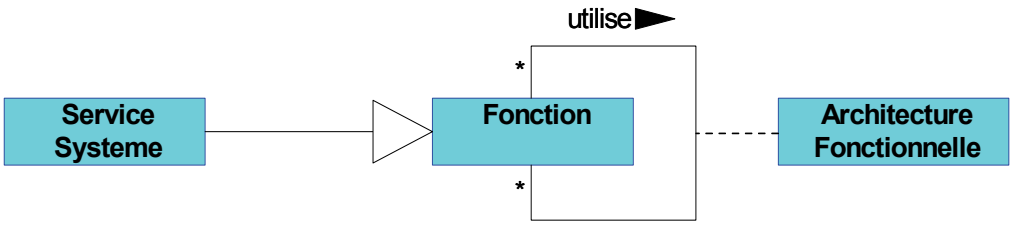


Figure 2.8 : Patron de conception pour décomposition des fonctions

Dans SysML nous pouvons modéliser ces fonctions à l'aide du pilier comportement qui offre trois diagrammes comportementaux : le diagramme d'états, le diagramme de séquence et le diagramme d'activité.

Le diagramme d'états permet de modéliser les états du système (« State ») et les passages d'un état à l'autre (« Transition »). Ces diagrammes sont utilisés plutôt dans la modélisation des systèmes à événements discrets du fait que le passage entre les états est fait par rapport à des événements et des conditions. Dans les systèmes continus, ce diagramme peut être utilisé avec l'échantillonnage du comportement continu, mais cela ne reflète pas le vrai comportement du système. Il peut être utilisé pour modéliser les modes de marche d'un système (Figure 2.9 : nominal, dégradé et en panne) ou pour montrer les différentes phases du cycle de vie d'un système (en conception, en production, en maintenance, …).

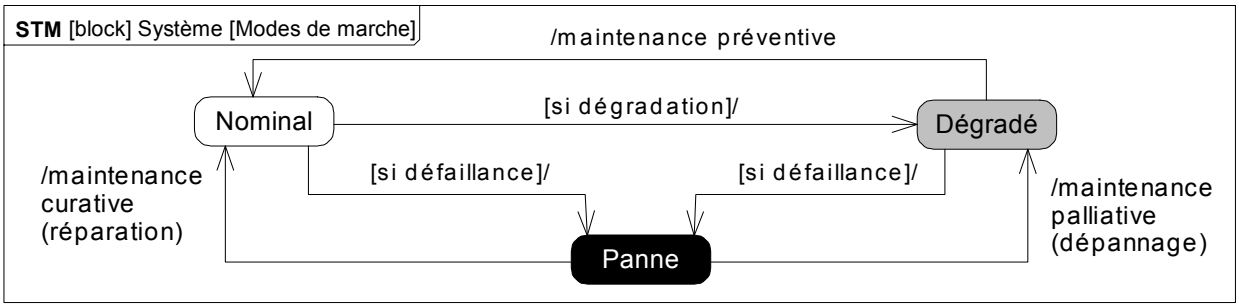


Figure 2.9 : Exemple de diagramme d'état qui modélise les modes de marche d'un système

Le diagramme de séquence permet de montrer l'échange des messages ou d'événements, ou l'appel de méthodes, entre plusieurs éléments du système ou avec des systèmes de l'environnement. Ce diagramme est un bon choix pour décrire les scénarios opérationnels qui raffinent les exigences fonctionnelles et principalement dans une vue boîte noire du système pour définir l'échange entre le système et les éléments de l'environnement.

Le diagramme d'activité est le plus adapté pour la modélisation des architectures fonctionnelles et permet de modéliser le comportement d'un système sous la forme d'un algorithme qui fait appel à des actions. Ces actions sont en effet des objets de type « Activity » que nous proposons d'utiliser pour modéliser les fonctions du système. Les activités permettent de modéliser (Figure 2.10) les trois types d'architectures de l'approche systémique (Figure 1.6) :

- La décomposition fonctionnelle : décomposition des activités à l'aide du lien de composition dans des diagrammes de bloc (BDD),
- L'approche structurelle : diagrammes d'activité (ACT) avec flux d'objets/données entre les activités, sans flux de contrôle,
- L'approche comportementale : diagrammes d'activité (ACT), dans lesquels figure le flux de contrôle pour enchaîner/paralléliser l'exécution des fonctions (on ajoute les aspects temporels).

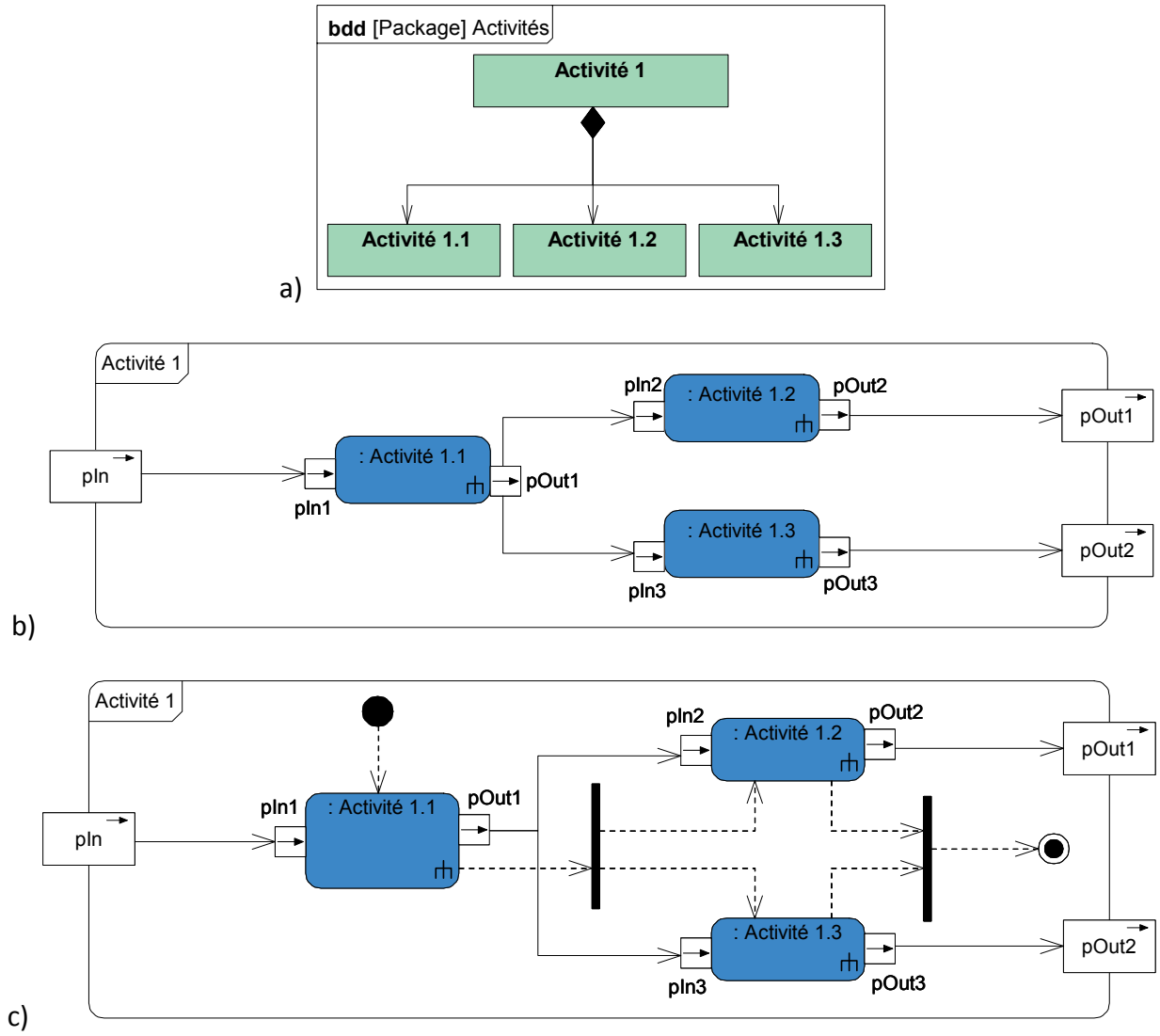


Figure 2.10 : Représentation des architectures fonctionnelles dans SysML : a) décomposition fonctionnelle, b) approche structurelle, c) approche comportementale

Les diagrammes d'activité pourront modéliser un comportement continu dans le cas où nous utilisons l'approche structurelle (Figure 2.10b) pour que chaque activité puisse s'exécuter à tout moment, et où les entrées d'une activité dépendent des sorties d'autres activités (équivalence d'un processus continu où chaque activité représente le comportement d'un élément du processus).

A ces trois diagrammes peut s'ajouter le diagramme paramétrique qui permet de modéliser le comportement continu d'un système à l'aide d'équations mathématiques.

Le Tableau 2.3 montre les deux types de comportements que ces quatre diagrammes permettent de modéliser. A ces égards, on peut conclure que SysML est plus adapté pour modéliser des systèmes ayant un comportement discret.

Tableau 2.3 : Types de comportements décrits par les diagrammes comportementaux de SysML

Type de Diagramme	Comportement discret	Comportement continu
Diagramme de machine d'état	OUI	PARTIEL
Diagramme d'activités	OUI	OUI
Diagramme de séquences	OUI	PARTIEL
Diagramme paramétrique	OUI	OUI

En ce qui concerne les règles de construction des diagrammes comportementaux, principalement dans le cas des diagrammes de séquence et d'activité, nous proposons un patron de conception basé sur l'utilisation des actinomies de Vogel (1988). Cette séquence est composée de trois actèmes (Figure 2.11) :

- ouverture de la séquence : préparation des ressources et de l'objet destinataire du noyau de la séquence ;
- noyau de la séquence : réalisation des événements et des transformations de la séquence ;
- clôture de la séquence : libération des ressources.

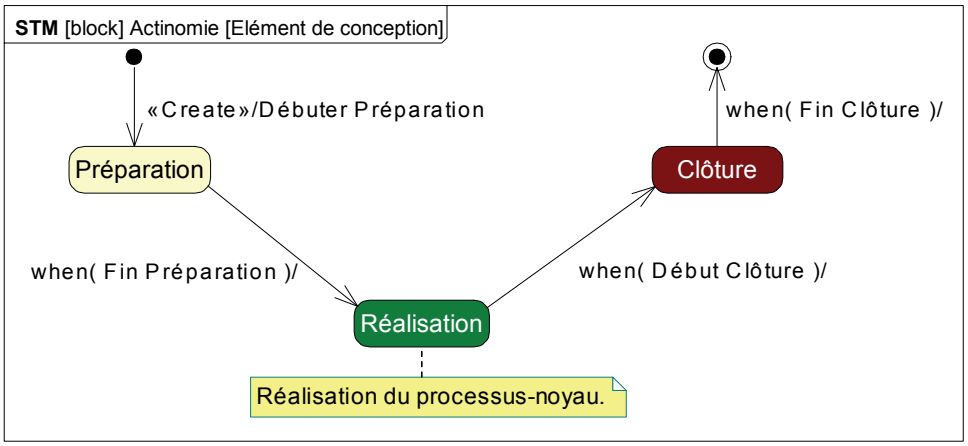


Figure 2.11 : Patron de séquence pour la construction des fonctions (d'après Vogel (1988))

Pour pouvoir modéliser le comportement d'un système en utilisant la logique des actinomies nous proposons les stéréotypes décrits à la Figure 2.12. Ces trois stéréotypes (« Preparation », « Realization », « Closing ») permettent de modéliser le comportement à l'aide d'états, dans des diagrammes d'états, ou à l'aide d'activités, dans des diagrammes d'activité. En effet, c'est dans l'utilisation d'une activité qu'on pourrait dire si elle est en préparation, en réalisation ou en clôture. De cette façon, une activité peut être utilisée dans différentes étapes de préparation, réalisation ou clôture, et cette utilisation est définie par l'objet de modélisation « Action ».

Par rapport au modèle de Gunter, et al., (2000), les fonctions représentent le programme P qui doit être réalisé. Comme le programme est exécuté sur une machine M, les fonctions doivent être regroupées dans des constituants systèmes.

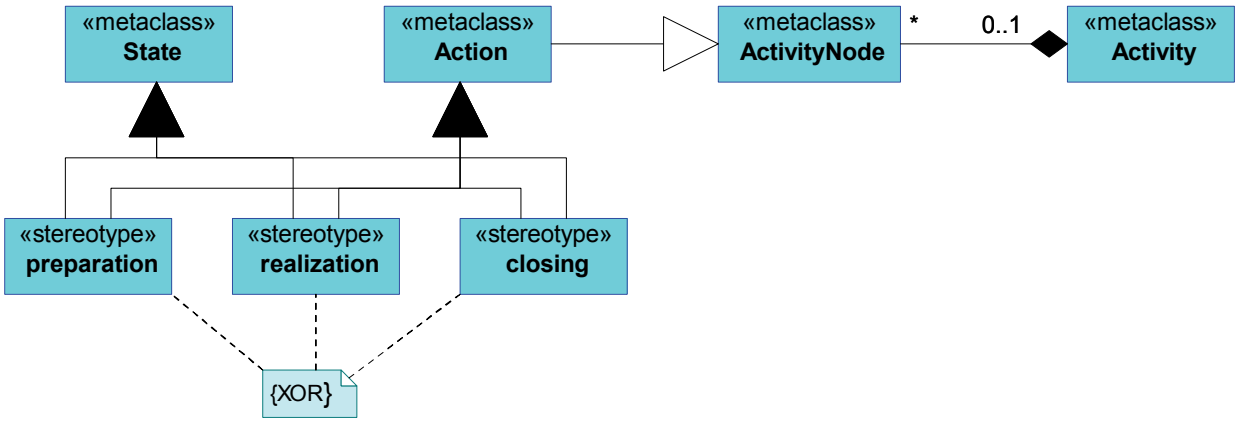


Figure 2.12 : Stéréotypes pour la modélisation des actinomies

2.2.4 Conception de l'architecture organique avec SysML

Le rôle des constituants systèmes est de réaliser les fonctions qui ont été définies auparavant.

En effet, les fonctions peuvent être rassemblées dans des constituants logiques. En SysML, ce rassemblement peut être fait à l'aide d'objets de type « ActivityPartition » (partitions) dans les diagrammes d'activités. Comme le montre la Figure 2.13, à chaque partition nous pouvons assimiler un bloc SysML représentant un constituant du système. Ainsi, l'activité 1.1 et l'activité 1.2 sont affectées au constituant 1 et l'activité 1.3 est affectée au constituant 2.

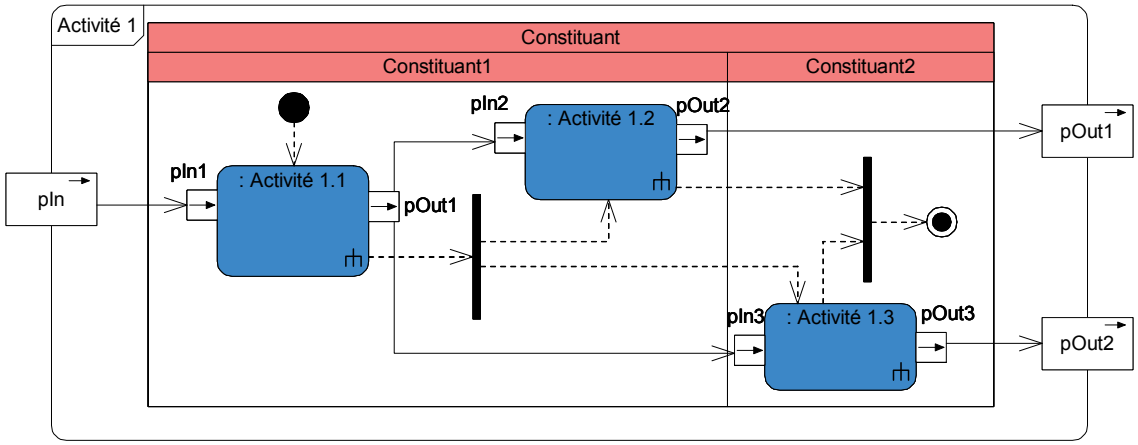


Figure 2.13 : Rassemblement des fonctions dans des constituants à l'aide des partitions

Les constituants ainsi définis peuvent être regroupés dans des constituants d'un niveau supérieur et mis en relations pour modéliser une décomposition des constituants dans un diagramme BDD (Figure 2.14).

Le problème qui se pose est de pouvoir dire explicitement la relation entre les constituants et les fonctions, étant donné que les partitions ne permettent que de montrer cette relation graphiquement. Ainsi, il est nécessaire de faire des allocations des activités vers les blocs.

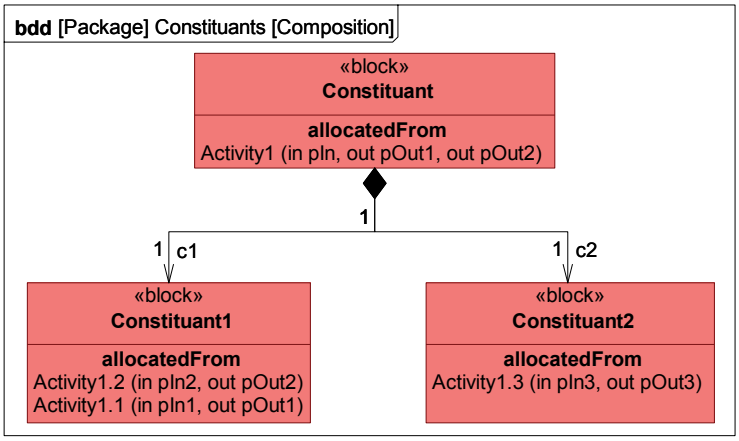


Figure 2.14 : Composition des blocs avec allocation des activités

L'architecture des constituants est réalisée à l'aide du diagramme interne de bloc IBD qui rassemble les constituants composants dans le constituant composé (Figure 2.15).

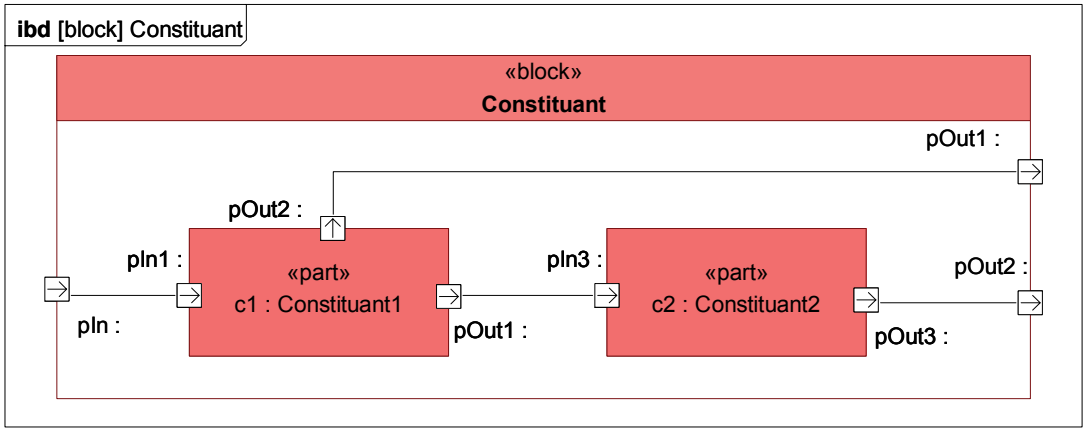


Figure 2.15 : Exemple d'architecture organique

Le problème qui se pose ici est la cohérence entre les flux d'entrée/sortie des blocs et des activités. En effet, les ports des blocs doivent être de même type que les paramètres des activités qu'ils réalisent, mais les spécifications SysML ne fournissent aucune information sur leur mise en relation. C'est pour cela qu'il est nécessaire d'ajouter les ports d'E/S des blocs en suivant les paramètres des activités. La difficulté consiste à avoir des ports d'E/S qui permettent de faire une connexion avec d'autres blocs, ce qui n'est pas toujours évident (dans notre exemple, le constituant 1 réalise deux fonctions et nous avons ajouté que les ports correspondants aux échanges avec les autres blocs).

Comme certains éléments du contexte du système sur lesquels le système doit agir existent avant le système, ils peuvent être décrits dès le début du processus de modélisation. En effet, la description de l'environnement se base sur une approche ascendante, par composition des constituants existants en vue de décrire l'architecture du contexte.

La décomposition des constituants du système représente quant à elle une approche descendante, qui part du système étudié par le bloc système actuel et qui va jusqu'aux constituants

qui le composent. Nous proposons pour la définition de cette architecture d'utiliser le patron de conception de type composite (Figure 2.16).

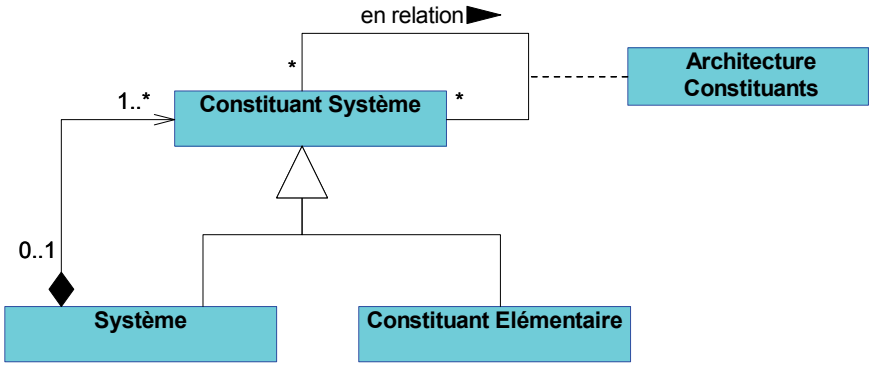


Figure 2.16 : Patron de conception de la structure d'un système

Cette décomposition s'approche de la vision orientée objets, dans laquelle l'accent est d'abord mis sur la définition des objets qui composent le système et après sur les opérations (ou fonctions) que les objets vont réaliser. La décomposition s'arrête à des constituants élémentaires, qui peuvent être des COTS[29]. De plus, cette décomposition doit prendre en considération les exigences systèmes et doit s'arrêter quand toutes les exigences sont satisfaites (critère de complétude).

2.3 Exécution de la spécification pour la Vérification et la Validation

L'exécution de la spécification dès la première itération du procédé de spécification est très importante pour vérifier sa correspondance aux exigences et pour la valider en regard des attentes des clients. Comme cela à été présenté, dans SysML les scénarios de vérification des exigences sont décrits à l'aide de diagrammes comportementaux dans des « TestCase ». Du fait que SysML n'a pas de sémantique opérationnelle, la vérification doit être faite par d'autres mécanismes/méthodes, tels que le model checking ou la simulation/émulation.

La différence entre simulation et émulation est présentée dans la littérature. Ainsi, pour Pannequin (2007) le concept d'émulation est défini par rapport à sa finalité : il s'agit de reproduire l'interaction du système à représenter avec son environnement. Le modèle d'émulation doit reproduire la réponse du système réel à des séquences d'entrées, afin d'être utilisé dans un système plus vaste. En comparaison, la finalité de la simulation est d'observer l'évolution des états internes du modèle placé dans une situation prédéfinie.

Albert (2009) considère qu'une simulation est un modèle, ou un ensemble de modèles, soumis à un environnement d'exécution qui anime ce modèle dans le temps. Un des problèmes qu'il

[29] COTS = Components off-the-shelf : composants sur étagère, c'est-à-dire disponibles sans développement particulier

étudie est la validité du modèle et la correction de l'environnement d'exécution, i.e. le simulateur, qui ajoute des contraintes additionnelles et des erreurs d'implémentation à la validité du modèle.

Pour nous, la différence entre une émulation et une simulation est une différence de vue du modèle exécuté par un espace de problème et un espace de solution. Ainsi, étant intéressé par les liens entre les entrées/sorties d'un système qui vont impacter l'environnement de l'espace de problème, ce dernier perçoit le modèle exécutable de la spécification comme une émulation qui fait une abstraction du système. À l'opposé, l'espace de la solution va essayer de donner des solutions pour réaliser et satisfaire ce lien entrées/sorties en spécifiant la manière dont s'exécute l'intérieur du système (structure et comportement) et donc ce qui est simulé. Du fait que notre procédé de modélisation système prescrit une solution à des besoins des différentes parties prenantes, nous allons nous intéresser à l'exécution de la spécification par simulation avec pour but sa vérification que nous présentons dans les paragraphes suivants, en nous limitant cependant aux travaux liées au langage SysML.

Il existe plusieurs approches de simulation. Par exemple, Isermann (2005) définit trois types de simulations dans le domaine de la mécatronique (Figure 2.17), qui prennent en compte la nature réelle ou simulée du système de contrôle (Partie Commande : PC) et du système opérant (Partie Opérative : PO), séparation utilisée aussi dans le domaine de l'automatique :

1. Un modèle du système de contrôle peut être appliqué à un système opérant réel, mais plus simple que le système final. Ce mode de simulation permet le « prototypage du système de contrôle ».
2. Un système de contrôle réel peut être appliqué au système opérant simulé. C'est la simulation « harware-in-the-loop ».
3. Le système opérant virtuel peut être opéré par le système de contrôle simulé. Ce dernier mode de simulation est désigné comme « software-in-the-loop ».

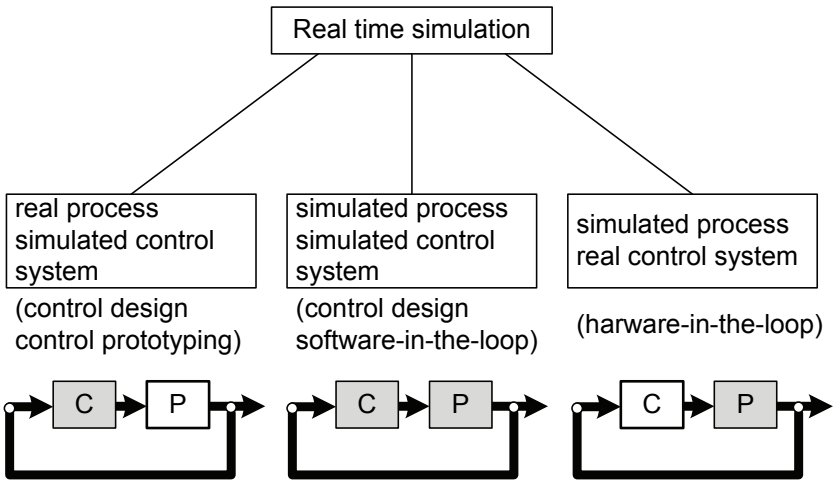


Figure 2.17 : Classification des approches de simulation temps-réel, dans le domaine de la mécatronique (Isermann, 2005)

On retrouve aussi la simulation « model-in-the-loop » (Saint-Germain, 2010). En effet, conforme à la typologie donnée par Isermann (2005), le « model-in-the-loop » est une particularisation de la simulation « software-in-the-loop ». Ceci consiste à avoir un seul modèle qui simule le système de contrôle et le système opérant comme un tout. Par exemple, Modelica (2010) permet de réaliser ce type de simulation. Ainsi, on peut associer à la simulation « software-in-the-loop » deux modèles différents, un pour le système opérant et un pour le système de contrôle, réalisés par des outils spécialisés (par exemple Modelica pour la PO et ControlBuild pour la PC) et qui sont mis ensemble afin de simuler leur comportement émergeant.

Du fait que SysML permet de modéliser le système comme un tout, la simulation « model-in-the-loop » permet de retrouver l'ensemble des modèles SysML dans ces modèles de simulation. Ainsi, se pose le problème de la synchronisation entre le modèle SysML et le modèle de la simulation. C'est ici que la transformation de modèle intervient pour maintenir la signification du modèle dans le langage qui permet une évaluation qualitative et quantitative.

Le problème de la vérification et de la validation des modèles réalisés avec des langages dits de « haut niveau », tels que SysML, AADL et eFFBD[30], habituellement utilisés en IS, fait l'objet des travaux de Seidner (2009). Elle montre la difficulté de vérifier formellement des propriétés, décrites aussi dans un langage de « haut niveau » (ex. langage naturel), sur ces modèles. Ces langages ont leur propre syntaxe et sémantique, qui leur permet de modéliser un univers de discours, mais il leur manque un système de preuves pour vérifier formellement l'exactitude des modèles proposés.

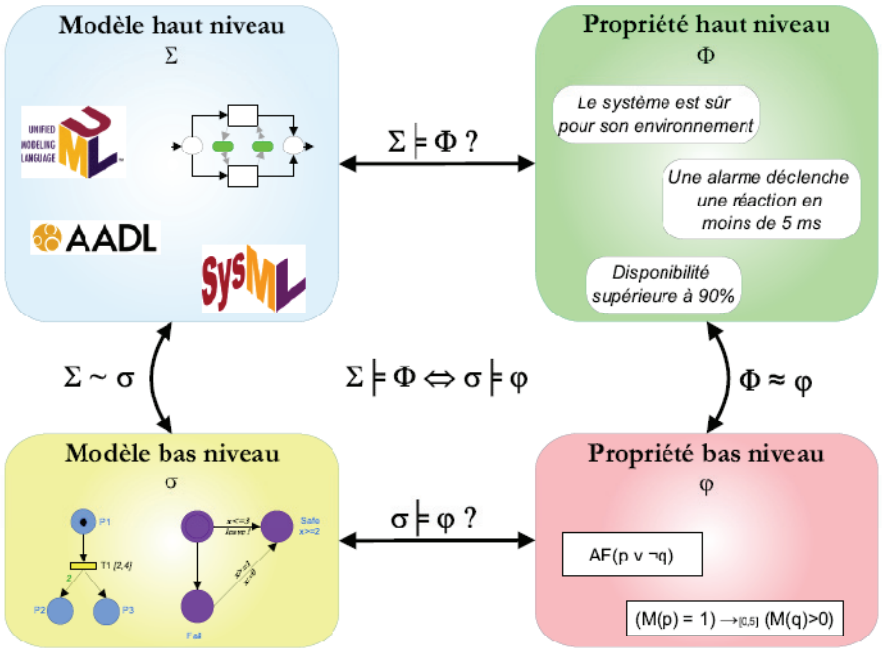


Figure 2.18 : Démarche générale de spécification et vérification des propriétés d'après Seidner (2009)

[30] enhanced Functional Flow Block Diagram

Comme solution, Seidner propose de passer de ces langages de modélisation de « haut niveau » vers des langages formels dits de « bas niveau » (Figure 2.18), tels que les Réseaux de Petri Temporels, qui supportent la vérification formelle des propriétés. Ce passage peut être réalisé par des transformations de modèles et permet de vérifier formellement, par l'exécution du modèle de « bas niveau », les propriétés définies.

Etant intéressés par l'utilisation de SysML comme langage de spécification en IS, nous avons cherché dans la littérature des moyens de transformations des différents diagrammes proposés par ce langage. L'intérêt de ces transformations au niveau Meta (Figure 2.19) est d'assurer l'équivalence entre les modèles SysML (de « haut niveau ») et le modèles de « bas niveau », équivalence valable aussi que pour les propriétés dans le cas de Seidner, pour garantir la validité des résultats obtenus. La transformation des modèles (Sendall, et al., 2003; Czarnecki, et al., 2006; Bondé, 2006) est une activité importante dans l'Ingénierie Dirigée par les Modèles (IDM).

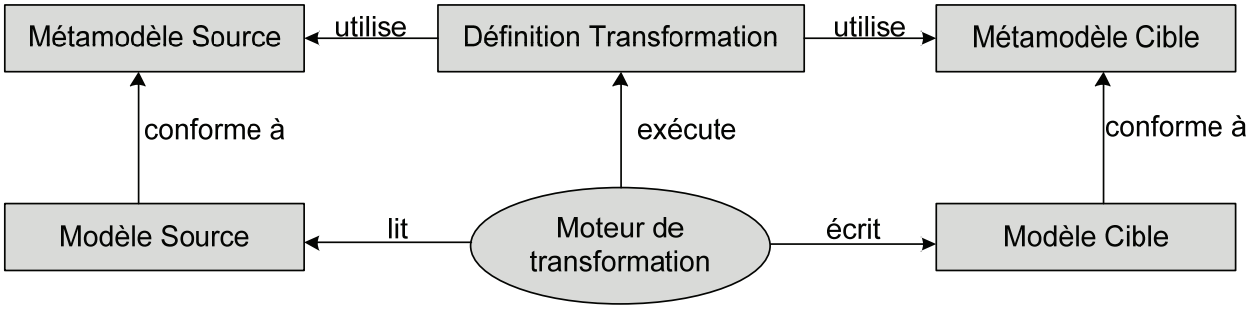


Figure 2.19 : Concepts de base des transformations de modèles

Ainsi, nous avons pu trouver des transformations des diagrammes paramétriques (vues dans la section §2.1), des diagrammes de définition de blocs ou des diagrammes d'activité.

Les diagrammes d'activité sont transformés par Bonhomme (2008) vers une représentation basée sur le formalisme HiLes[31]. Ce formalisme fournit une représentation de la commande en Réseaux de Petri, avec une vérification fonctionnelle de cette commande par un outil de vérification associé à HiLes (TINA[32]), une représentation simulable des caractéristiques logico-temporelles du système (à chaque fonction est associé un délai de réalisation qui anticipe les performances du système réel), et la possibilité d'une traduction semi-automatique en VHDL-AMS de cette représentation logico-temporelle, prête à s'enrichir des choix technologiques pour aboutir « sans faute » à la solution terminale physique.

La méthodologie d'IS basée sur SysML et MDA proposée par Turki (2008) comporte des générateurs de code Modelica permettant de simuler certains sous-ensembles du modèle global. Il définit pour cela un profile Bond Graph pour SysML qui, avec la bibliothèque BondLib de Modelica, permet de faire un mapping entre les deux langages afin de transformer les modèles SysML/Bond Graphs vers des modèles Modelica. Johnson (2008) fait le passage entre le méta-modèle partiel (autour la notion de « block ») de SysML et le méta-modèle de Modelica. Pop

[31] HiLes Designer : http://www.laas.fr/toolsys/hiles.htm
[32] TINA - TIme petri Net Analyzer: http://www.laas.fr/TINA

(2008) définit quant à lui un profile pour UML/SysML, ModelicaML, qui ajoute des diagrammes spécifiques aux simulations Modelica, le diagramme des équations et le diagramme de simulation.

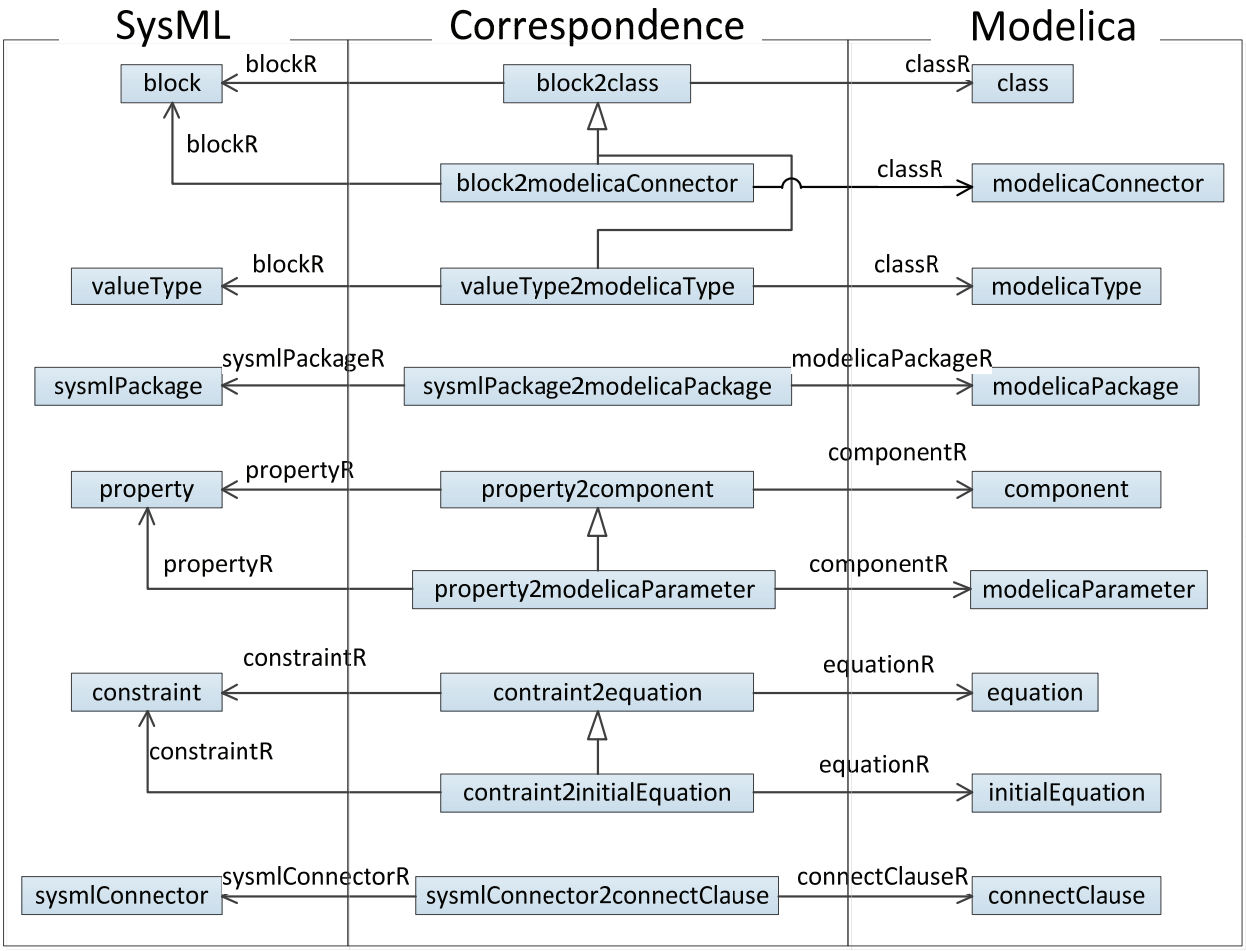


Figure 2.20 : Correspondance entre objets de modélisation SysML et Modelica (Johnson, 2008)

Paredis, et al. (2008) ont vu l'opportunité de vérifier et valider les modèles SysML par simulation avec le langage multidomaine Modelica et ont proposé à l'OMG la définition d'un groupe de travail sur l'intégration de SysML et de Modelica qui a débuté en 2009 et qui a publié en décembre 2009 ses premières spécifications : SysML-Modelica Transformation Specification (2010). Ces spécifications portent sur la définition d'une extension de SysML pour supporter les objets de modélisation de Modelica, SysML4Modelica qui permet de faire des transformations dans les deux sens entre les deux langages.

On peut citer d'autres travaux sur la transformation des modèles SysML pour la vérification et la validation des modèles comme Prevostini et Zamsa (2007) (SysML vers du code SystemC), Wang et Dagli (2008) (SysML vers Réseaux de Petri Colorés), Giese, et al. (2009) (SysML vers AUTOSAR), Shah, et al. (2009) (SysML vers ECAD).

Dans tous ces travaux de vérification et validation à l'aide de la transformation de modèles nous pouvons observer l'utilisation d'outils génériques de modélisation système jusqu'à

l'utilisation d'outils de conception métier. Les résultats de la vérification et de la validation doivent être remontés afin d'optimiser les exigences prescrites précédemment.

Ainsi, il reste la question de qui fournit la connaissance métier par rapport aux modèles développés. Cela infère nécessairement une collaboration système/métiers. Se pose alors le problème du système d'abstraction suffisant du modèle métier pour exécuter le modèle de spécification système.

2.4 Conclusions

Dans ce chapitre nous avons étudié SysML comme langage de spécification dans l'Ingénierie Système. D'une part nous avons ainsi posé les artéfacts du langage SysML par rapport à ceux que nous avons présentés dans le premier chapitre, de même que des artéfacts connexes. D'autre part, nous avons mis en évidence que ce langage ne propose pas de méthode de modélisation. C'est pour cela que nous proposons une méthodologie de modélisation système basée sur l'approche systémique et les bonnes pratiques de l'IS.

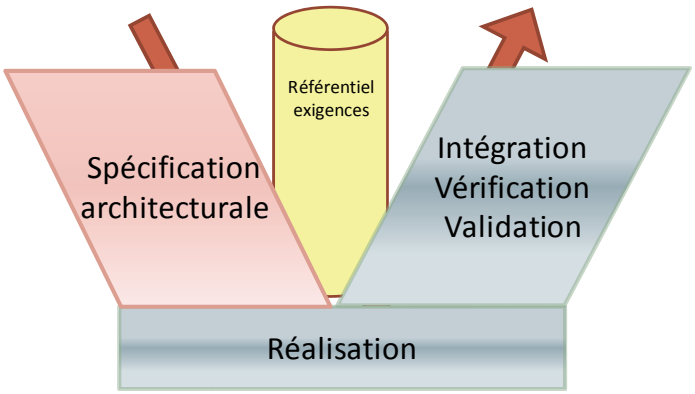
Le bloc système qui est à la base de notre procédé de modélisation permet de spécifier les exigences, le comportement et la structure d'un système. Cette approche de modélisation par bloc système passe par les étapes suivantes :

- Définition des besoins et prescription des exigences réalisées à l'aide des diagrammes d'exigences, diagrammes de cas d'utilisation et diagrammes de séquence ;
- Modélisation des fonctions à l'aide des diagrammes d'activité et diagrammes de blocs ;
- Modélisation de la structure organique à l'aide des diagrammes d'activité, diagrammes de blocs et des diagrammes de bloc internes.

Pour chaque phase nous avons détaillé la démarche à adopter et nous avons proposé les types de diagrammes SysML à utiliser. Des extensions nécessaires à la mise en correspondance des artéfacts sont introduites sous la forme de stéréotypes et des patrons d'architecture spécifiques (patron composite pour la structure et patron de séquence pour le comportement). Chaque spécification doit de plus être vérifiée et validée à chaque niveau.

PARTIE II :
Système à Faire

Chapitre 3
Spécification de l'architecture système du Système Interactif d'Aide à la Conduite



Dans le premier chapitre nous avons introduit des artéfacts pour la spécification en l'Ingénierie Système. Ces artéfacts sont supportés dans le deuxième chapitre par le langage de modélisation système SysML qui, avec des mécanismes de spécification, nous permet de définir une méthodologie basée sur l'approche systémique de l'IS. Cette méthodologie de spécification est illustrée dans cette deuxième partie du mémoire par une application industrielle liée à la conduite des grands systèmes à risques.

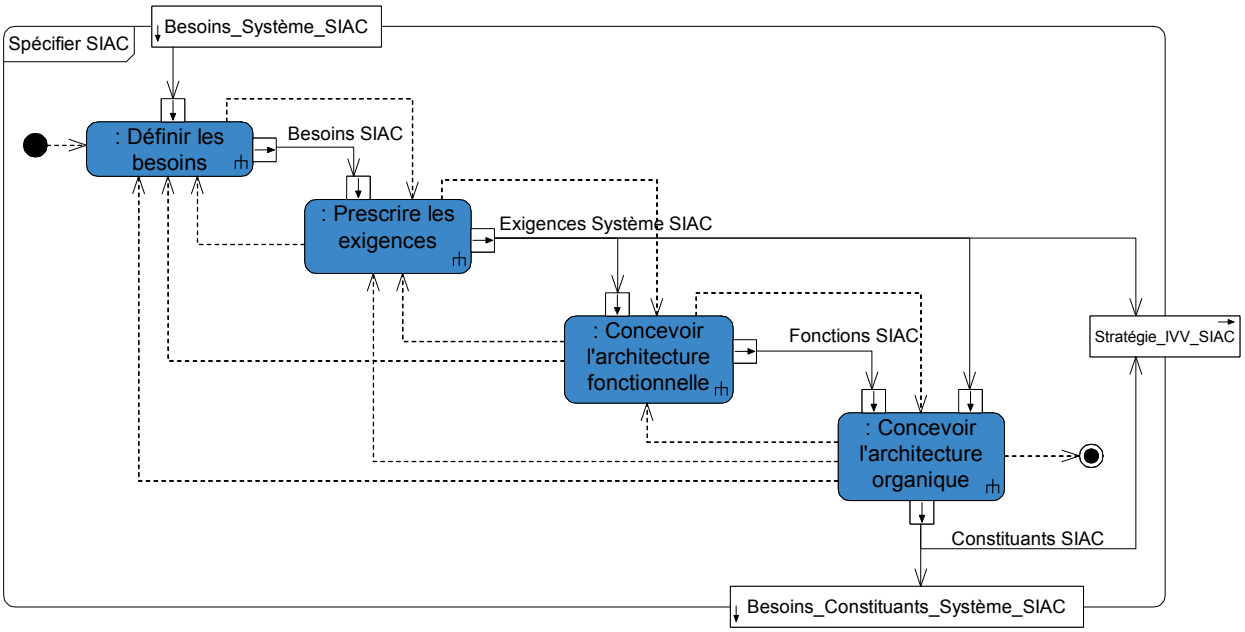


Figure 3.1 : Phase de spécification pour le bloc système SIAC

Ce chapitre introduit le contexte et la problématique de la conduite de systèmes complexes et, en particulier, la conduite de ces systèmes en local par des opérateurs spécialisés, les rondiers.

Nous proposons ensuite un Système Interactif d'Aide à la Conduite (SIAC) pour apporter des solutions à certains des problèmes liés à la conduite de ces systèmes. La spécification (Figure 3.1) du bloc système SIAC est réalisée à l'aide de SysML en suivant la méthodologie proposée dans le chapitre précédent. Les phases de réalisation et d'intégration sont présentées dans le chapitre suivant.

3.1 Contexte et enjeux industriels

La conduite de grands systèmes industriels à risque est un ensemble de processus complexes couvrants des modes opératoires variés (en production, en arrêt, en démarrage) adaptés à la criticité des modes d'exploitation rencontrés (conduite normale, conduite incidentelle et conduite accidentelle) selon des constantes de temps différentes (conduite en temps réel, maintenance hors ligne).

Aujourd'hui, ces processus reposent sur des interactions entre différents opérateurs et des systèmes propriétaires, hétérogènes et généralement limités à la phase de production normale (Gauthereau, et al., 2005; Carvalho, et al., 2008). Dans ce contexte, l'apparition d'alarmes ou de défauts provoque souvent le basculement d'une conduite dite « normale » vers une conduite dite « accidentelle » visant essentiellement à préserver la sécurité des personnels, à conserver le système de production et à limiter les rejets sur l'environnement. La mise en œuvre d'une conduite « incidentelle », limitant ce basculement systématique en conciliant les objectifs de production et de sécurité, constitue un gisement de productivité important.

La conduite mise sur la capacité du système d'information à fournir aux opérateurs les moyens leur permettant d'analyser la situation d'état dans laquelle se trouve le processus, de choisir des procédures appropriées à ces situations dans des conditions de sécurité acceptables, de décliner ces procédures sous la forme de plans d'actions et d'assurer le suivi et la surveillance du système de production. Ceci suppose notamment :

- la mise à disposition des opérateurs de conduite, en temps réel, d'informations synthétiques, fiables et directement interprétables émanant des algorithmes de surveillance, de détection, voire de pronostic des équipements et matériels,
- l'élaboration, à partir de ces informations, d'indicateurs relatifs à la disponibilité des systèmes et sous-systèmes dans lesquels interviennent les équipements et matériels surveillés en tenant compte de la structuration hiérarchique utilisée par les opérateurs,
- la coordination des différents corps de métiers en exploitation (conduite, maintenance, gestion technique) intervenant sur les équipements, quelle que soit leur localisation (en salle de commande ou sur site).

Dans la même idée, la plupart des systèmes industriels sont étudiés comme des systèmes socio-techniques (Galara D. , 2006; Reiman, 2007; de Bruijn, et al., 2009), qui intègrent des systèmes

physiques et techniques complexes, ainsi que des réseaux d'acteurs interdépendants, sans lesquels ces systèmes ne pourraient pas fonctionner correctement (Figure 3).

La conduite de tels systèmes sociotechniques est réalisée par l'interaction entre les trois systèmes : humain, CMMS (Système de Conduite, Maintenance et Gestion Technique) et processus physique. Cette interaction est guidée par des procédures de conduite qui décrivent les actions et les conditions qui doivent être remplies pour réaliser les fonctions du procédé. Nous nous intéressons à l'interaction directe entre les opérateurs de conduite et les équipements de terrain composant le processus physique, pour exécuter des actions de conduite dans le cadre d'un modèle d'architecture CMMS. Ainsi, nous allons commencer par décrire ces deux systèmes, processus physique et humain, qui composent un système industriel de production, pour expliquer ensuite l'interaction nécessaire à l'exécution des fonctions du procédé.

3.1.1 Le processus dans les systèmes de production

Le cycle de vie d'un système de production, après sa conception et réalisation et avant son démantèlement, comporte quatre phases (Figure 3.2) : démarrage, production, arrêt et maintenance. Notons que des actions manuelles en local, par exemple les consignations, sont plus fréquentes dans les phases de maintenance, voire de démarrage et d'arrêt, et beaucoup moins fréquentes en phase de production. Nous nous proposons d'étudier l'exploitation du processus en local.

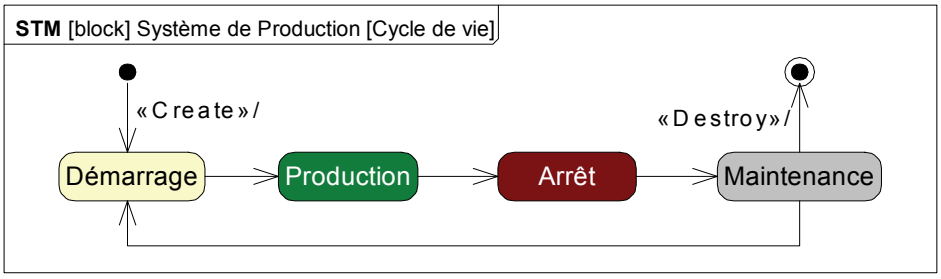


Figure 3.2 : Cycle de vie d'un système de production

La finalité d'un processus est de fournir des produits finis (biens de consommation ou services). Ainsi, nous pouvons décrire sa mission (Figure 3.3) comme la transformation en temps, espace et forme (Feliot, 1997) des matières premières en produits finis. Cette transformation utilise de l'énergie et peut produire des déchets.

Figure 3.3 : La transformation des matières premières en biens de consommation par un processus

Un processus désigne deux aspects d'une installation de production, l'un descriptif, l'autre matériel (Grout, et al., 2009) :

- Aspect descriptif : décrit l'ensemble des opérations détaillées permettant l'élaboration, selon un procédé déterminé, d'un produit fini ; celui-ci doit posséder des caractéristiques imposées et être dans les limites des tolérances fixées ;
- Aspect matériel : représente l'installation proprement dite, comprenant tous les appareils nécessaires à la transformation des matières premières.

La mission est réalisée par le processus physique à travers les équipements de terrain (vannes, pompes, cuves, …) que nous appelons Objets Matériels Techniques (OMT). Ce sont ces OMT qui représentent l'aspect matériel du processus. La transformation est guidée par les opérations fournies par l'aspect descriptif du processus, appelé aussi procédé. Cette section étudie l'aspect matériel du processus, l'aspect descriptif étant étudié dans la section §3.1.3.

Ces éléments sont décrits dans des « schémas mécaniques » souvent désignés sous le nom P&ID (Process and Instrumentation Diagram) qui se traduit par « schéma de tuyauterie et instrumentation du processus » ou *schéma TI*. La description des schémas du processus est donnée par les normes ANSI/ISA 5.1 (1984) et ISO 10628 (2002).

Nous nous intéressons plus particulièrement à deux types d'OMT :

- OMT instrumentés (OMTi) : les vannes de régulation, les pompes, qui sont actionnées par les opérateurs de conduite à l'aide du système de conduite depuis la salle de commande ; les capteurs et transmetteur de niveaux et débit font aussi partie de ces OMTi ;
- OMT non instrumentés (que nous appelons passifs : OMTp) : les vannes manuelles, qui doivent être manipulés localement par un opérateur humain.

L'instrumentation des OMT peut ne pas être uniforme dans les systèmes industriels de production. Ainsi, les OMTp peuvent représenter jusqu'à 80% du total des OMT. La plupart d'entre eux appartiennent à des systèmes non-critiques. Cette répartition de l'instrumentation dans le système de production est justifiée par l'utilisation des OMTp, plutôt dans les phases de démarrage et d'arrêt du processus, qui peuvent n'avoir lieux que quelques fois par an. Galara (2006) précise en effet que ces OMTp pourront être manipulés par l'humain parce que le coût de l'instrumentation n'est pas justifié.

Ce manque d'instrumentation réduit l'observabilité et la contrôlabilité directes du processus depuis la salle de commande. Ceci impose un travail en local des rondiers, qui doivent observer visuellement et actionner manuellement ces OMTp. Cependant, les actions exécutées et les mesures prises sur ces OMTp peuvent s'avérer parfois erronées à cause des différents facteurs d'influence (difficultés d'identification et localisation des OMTp, manque d'informations dans les procédures, manque de connaissances sur le système, manque d'expérience, conditions environnementales, …). Afin d'améliorer l'observabilité de ces OMTp, Dionis, et al. (2007) et Dang, et al. (2008) proposent des solutions d'instrumentation ambiante incluant des technologies sans fil.

Afin de récréer à l'échelle réduite ces problèmes d'instrumentation et de valider différents travaux de recherche, le laboratoire CRAN a construit, au sein du service plateformes, une plateforme expérimentale, appelé CISPI (Conduite Interactive et Sûre de Procédés Industriels : Figure 3.4), à l'élaboration de laquelle nous avons participé. Cette plateforme a été financée dans le cadre du centre d'innovation et de démonstration des technologies sûres SAFETECH du CPER MISN[33].

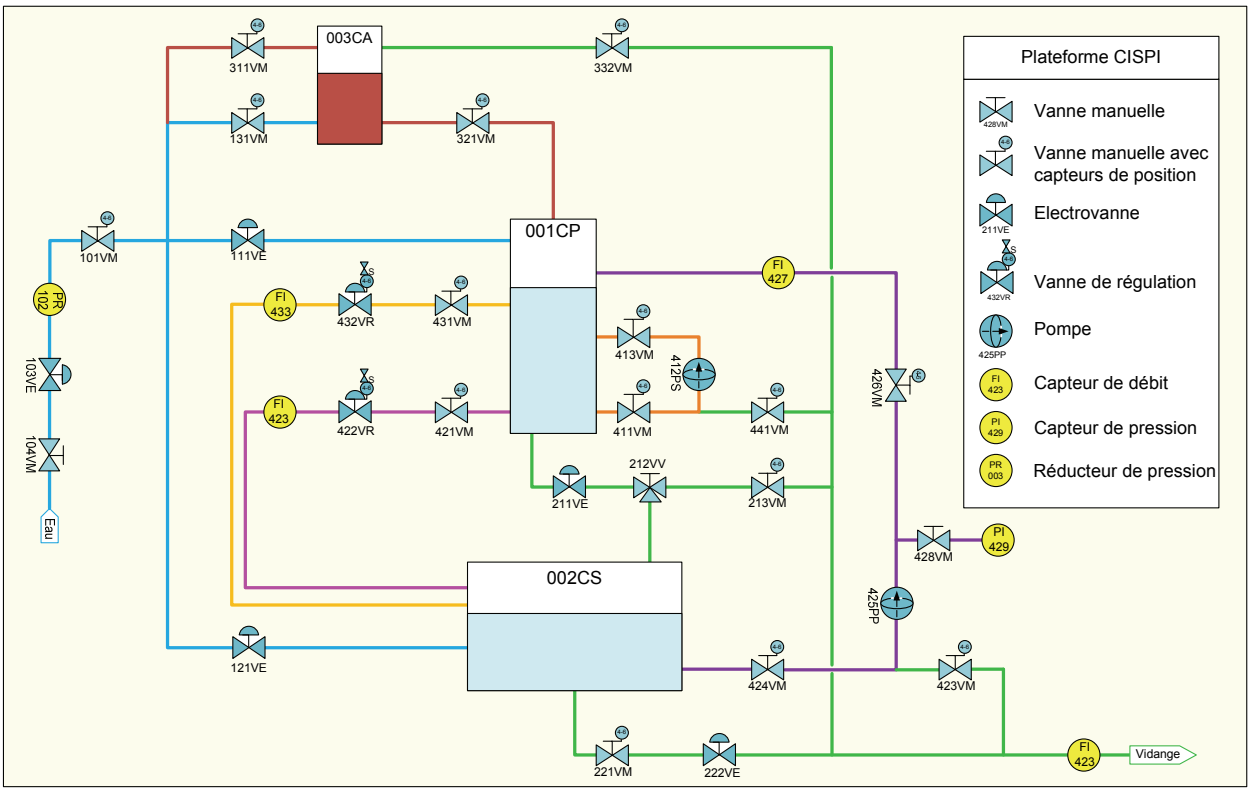


Figure 3.4 : Schéma TI du système élémentaire CISPI

Pour pallier au manque d'observabilité du processus, aussi que pour réduire le risque d'erreur sur l'exécution des actions sur les OMT passifs, nous proposons d'intégrer dans chaque OMTp, d'une part, des fonctions de gestion des actions (validation/autorisation de l'action à exécuter, vérification et validation de l'action exécutée) et, d'autre part, des fonctions de gestion de son état pour augmenter son observabilité. Ces fonctions transforment l'objet passif en un objet actif afin de l'intégrer dans un canal d'informations auquel les opérateurs auront accès par le système de conduite, maintenance et gestion technique (CMMS) (Figure 5).

Ceci nous amène à faire le lien avec le concept d'objet actif, développé au sein du CRAN depuis quelques années. Le concept d'objet actif a été introduit pour la première fois par Gershenfeld (1999), et décliné au niveau du produit « intelligent » par McFarlane, et al. (2002). Il est admis dans le milieu scientifique des systèmes automatisés de fabrication comme la fusion d'objets physiques et virtuels, principalement d'informations, définis par des caractéristiques élémentaires qui ont été enrichies à plusieurs reprises par différents contributeurs :

[33] Contrat Plan État Région Modélisation, Information et Systèmes Numériques

- Possède une identification unique (Wong, et al., 2002),
- Est capable de communiquer avec son environnement (Kintzig, et al., 2002),
- Peut mémoriser et gérer des informations propres,
- Dispose d'un langage de dialogue et d'échange d'informations et d'états,
- Participe aux processus de prise de décisions concernant son état,
- Peut surveiller et contrôler son environnement,
- Peut réaliser des interactions à travers les services qu'il offre (Bajic E. , 2005).

Les activités de recherche sur les objets actifs se concentrent principalement sur la spécification d'infrastructures de services et sur le prototypage d'objets actifs autonomes (Bajic, et al., 2008), services qui puissent être invoqués de façon automatique et transparente dès l'interaction physique avec cet objet. Ainsi, des architectures de services ont été mises en place pour permettre aux produits d'intégrer des services (Cea Ramirez, 2006) : identification du produit, localisation, conditions de stockage … Dans cet ordre d'idées, le concept d'objet actif a évolué de l'objet informationnel vers l'objet fournisseur de services.

La notion d'objet actif a donné aussi naissance au produit actif développé dans le cadre de nos travaux au sein du CRAN depuis 2006 (Dobre, et al., 2009). Ainsi, le produit a les capacités citées et il met en avant ses caractéristiques de réaction aux phénomènes environnementaux sous une forme de surveillance active de son contexte et des autres produits, dans une collaboration Objet-à-Objet (O2O). Ces travaux réalisées au début de cette thèse ont posé les premières briques de l'étude de l'interaction entre un opérateur et des produits passifs, en mettant l'accent sur l'intégration de services dans les produits afin d'assister l'opérateur pendant leur manipulation. Nous nous intéressons dans la section suivante aux acteurs de conduite.

3.1.2 Les acteurs de la conduite d'un système de production

Les acteurs de la conduite des systèmes de production que nous étudions peuvent être regroupés en deux catégories :

- l'équipe des opérateurs qui travaillent dans la salle de contrôle-commande ;
- l'équipe des rondiers qui travaillent en local, sur le terrain.

L'opérateur de conduite[34] pilote et surveille à distance les installations de production et en particulier les OMTi. Pour ce faire, il maîtrise et applique tout un ensemble de procédures, les Modes Opératoires.

Le rondier surveille et contrôle en permanence l'ensemble du matériel (tableaux électriques, circuits d'eau, de vapeur, …) pour détecter toute anomalie de fonctionnement. Ses missions principales sont :

[34] Appelé simplement « opérateur » dans le reste du document

- Relever, lors de ses rondes, les informations sur l'état de santé du matériel et les consigner sur des comptes-rendus (CR), habituellement en format papier ;
- Signaler à l'opérateur de conduite les éventuelles anomalies détectées et, à sa demande, effectuer des manœuvres sur les vannes, les moteurs, les pompes, en particulier sur les OMTp, qui ne peuvent être manipulés que manuellement. A ce titre, le rondier est l'œil, l'oreille et le bras de l'opérateur qui pilote le système de production depuis la salle de commande.

La conduite étant réalisée par le biais d'une collaboration entre les opérateurs en salle de commande et les rondiers sur le terrain, il est important d'avoir une bonne communication et une bonne synchronisation entre ces deux métiers afin d'assurer le bon fonctionnement de l'ensemble de processus. Cependant, le langage naturel utilisé lors de la communication est parfois source d'une mauvaise compréhension (Carvalho, et al., 2008; Kim, et al., 2008) et peut entrainer l'exécution d'un ordre qui n'est pas celui demandé. De plus, les rondiers manquent de moyens informatiques permettant de réduire le risque de notations mal-écrites et mal-comprises (Pirus, 2006).

Dans la conduite des systèmes de production, les compétences (Belkadi, et al., 2007) et les connaissances (Grundstein, et al., 2003) de l'ensemble des opérateurs de conduite sont très importantes (O'Connor, et al., 2008). C'est pour cela que le recrutement, la qualification et la formation du personnel sont essentiels et nécessitent des moyens humains, techniques (simulateurs de formations) et financiers importants.

En ce sens, une étude sur les problèmes d'exploitation des systèmes industriels liés à l'humain a été mené par Léger (2009). Elle s'est intéressée particulièrement aux :

- caractéristiques du collectif : la délégation, l'expérience, la formation, la gestion collective et dynamique de groupe,
- outils et procédures utilisés : les aides, la possibilité de respect du cahier des charges, le contrôle et l'atteinte des objectifs, le retour d'expérience, les facteurs contextuels,
- facteurs organisationnels : la faiblesse de la culture organisationnelle de sûreté, la défaillance dans la gestion quotidienne de la sûreté, la faiblesse des organismes de contrôle, le mauvais traitement de la complexité organisationnelle, la difficulté à faire vivre un retour d'expérience, les pressions de production, et l'absence de réexamen des hypothèses de conception.

Endsley (1995) énonce les facteurs qui conditionnent la connaissance de la situation par l'humain. D'une part, il y a les facteurs internes à l'homme : l'attention, la perception, la mémoire de travail, la mémoire à long terme, l'automatisme des opérations réalisées, les objectifs. D'autre part, il y a les facteurs liés à la tâche et au système : le design du système, le design de l'interface, le stress (physique : bruit, vibrations, chaleur/froid, lumière, conditions atmosphériques ; psychologiques : peur ou anxiété, importance ou conséquence des événements, charge mentale, pression sur le temps, …), la charge de travail, la complexité du système, l'automatisation du système. Ces facteurs peuvent être pris en considération dans la spécification d'un sys-

tème d'aide à la conduite pour permettre d'améliorer certains facteurs comme la perception du processus, la mise à disposition des informations permettant d'améliorer la mémoire de travail et en intégrant des fonctions automatiques remplaçant certaines fonctions habituellement exécutées par l'humain.

On retrouve aussi des mesures pour la performance humaine, c'est-à-dire la réussite d'une tâche par un opérateur humain ou par une équipe d'opérateurs humains. Ces mesures de la performance (Gawron, 2008) sont classifiées en six catégories : mesure de la précision/exactitude, mesure de la durée (temps), mesures de l'ensemble des tâches (pour des tâches exécutées en série ou en parallèle), mesures dépendantes du domaine, mesures des incidents critiques, mesure de la performance de l'équipe.

Les opérateurs de conduite ont un rôle très important dans l'exploitation des systèmes de production. C'est pour cela que nous les considérons et nous défendons leur intégration comme des constituants systèmes essentiels dans la conduite de ces systèmes complexes et non pas comme des acteurs externes qui utilisent un système de conduite. Par exemple, le rôle des rondiers est très important dans la boucle cybernétique (Figure 3.5) parce qu'une partie de l'actionnement et de la mesure est réalisée par eux. L'utilisation des opérateurs pour exécuter des actions de conduite est justifiée par leurs capacités de jugement, d'analyse de la situation et de réaction immédiate dans des situations non prévues.

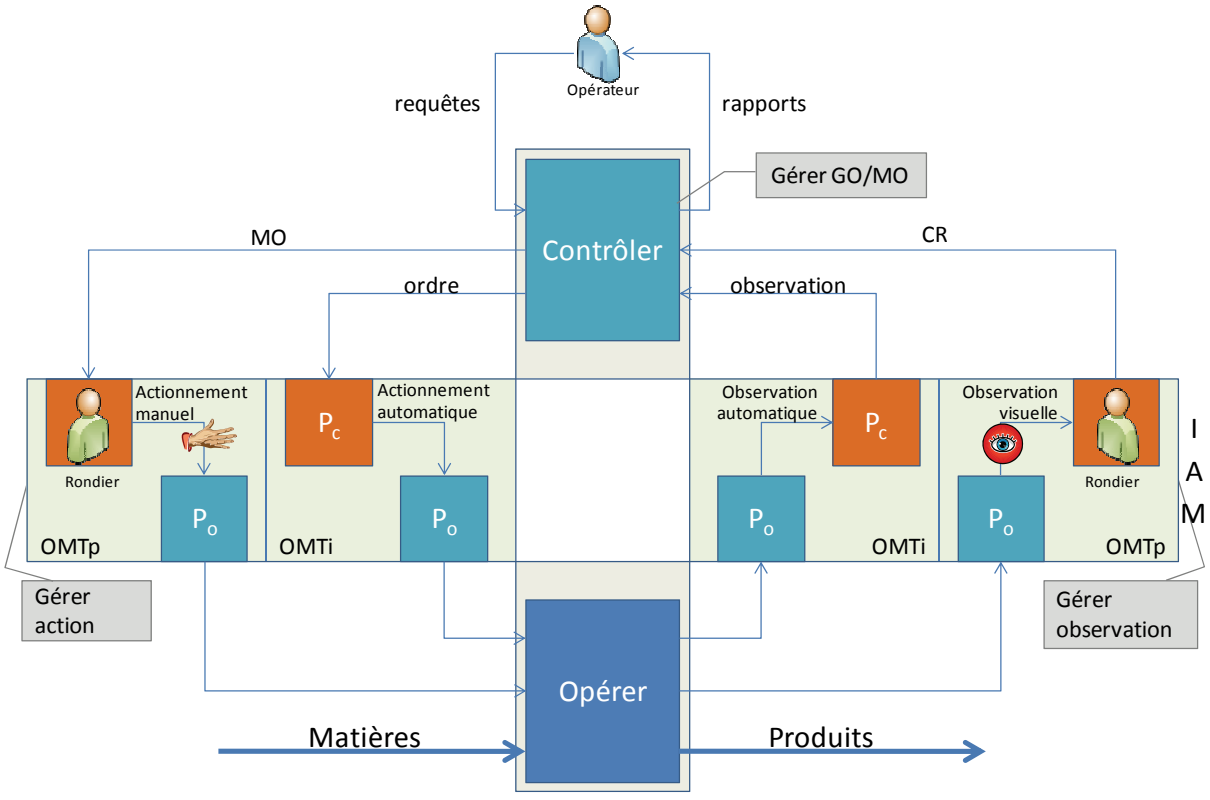


Figure 3.5 : Le rôle des acteurs de la conduite dans la boucle cybernétique

Cependant, l'homme commet des erreurs. Le dicton « *Errare humanum est, perseverare diabolicum* »[35] est tout à fait valable dans la conduite des systèmes industriels. Pour Reason (2000), l'erreur humaine est inéluctable, et peut être liée à la perte de contrôle cognitif de la situation (surcharge cognitive ou mauvaise interprétation) (Amalberti, 2001). Comme on ne peut ni remplacer l'humain totalement par de l'automatisme, ni changer la nature humaine, on peut envisager de changer les conditions dans lesquelles les hommes travaillent. C'est pour cela que nous proposons une assistance à l'aide d'un système interactif d'aide à la conduite afin de diminuer le risque d'erreur de conduite, principalement à destination des rondiers.

Par exemple, les mesures de la performance humaine citées plus tôt pourront être utilisées afin d'évaluer les performances d'un système d'aide à la conduite. Par ailleurs, il est important de considérer les interfaces fournies et requises par un opérateur humain (Figure 3.6), afin de lui fournir les meilleures façons d'interagir avec ce système d'aide à la conduite, comme le montre aussi Hall et Rapanotti (2005).

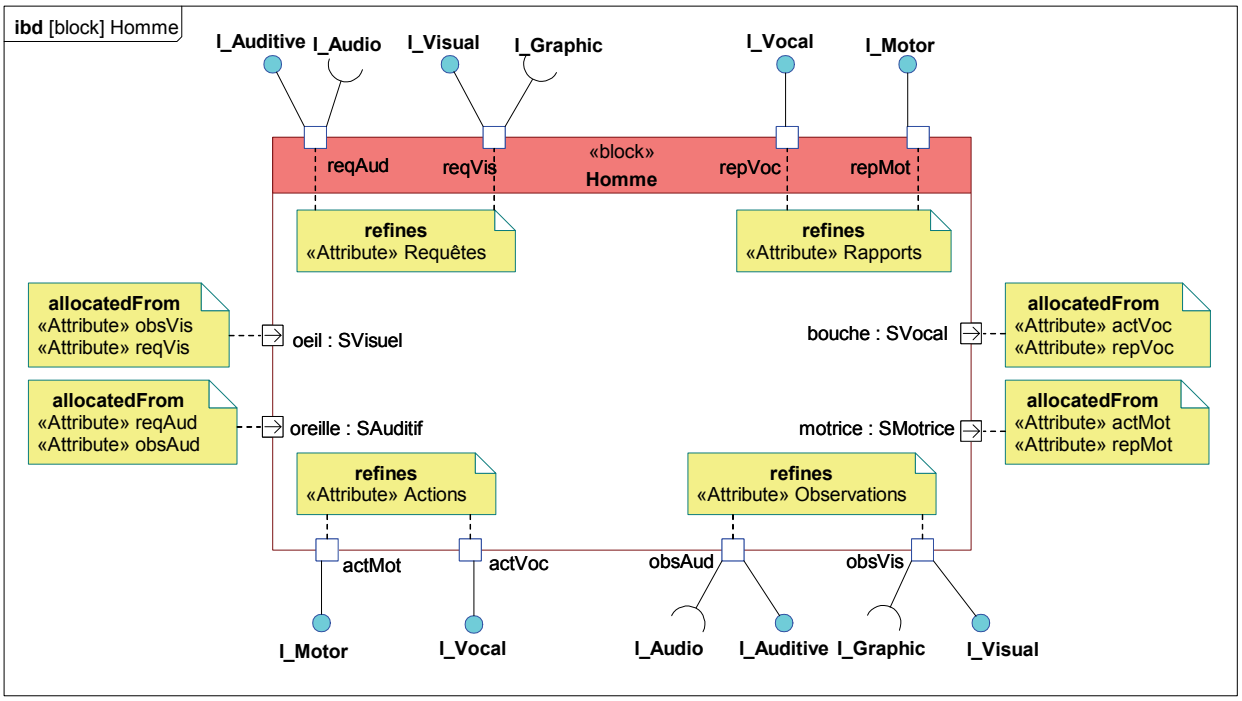


Figure 3.6 : Les interfaces fournies et requises par un opérateur humain (après Kieras et Meyer (1997))

La nature humaine des opérateurs de conduite a une forte influence sur les problèmes de conduite. Ces problèmes sont étudiés par les facteurs humains et les sciences sociales et ne rentrent pas dans le cadre de cette thèse. Notre intérêt et de fournir aux opérateurs des services d'aide à la conduite. Pour cela, il faut d'abord comprendre la conduite des systèmes de production.

[35] « L'erreur est humaine, persévérer est diabolique »

3.1.3 La conduite des systèmes de production

Une entreprise peut être schématisée par différentes activités complémentaires (Grout, et al., 2009) représentées à la Figure 3.7 :

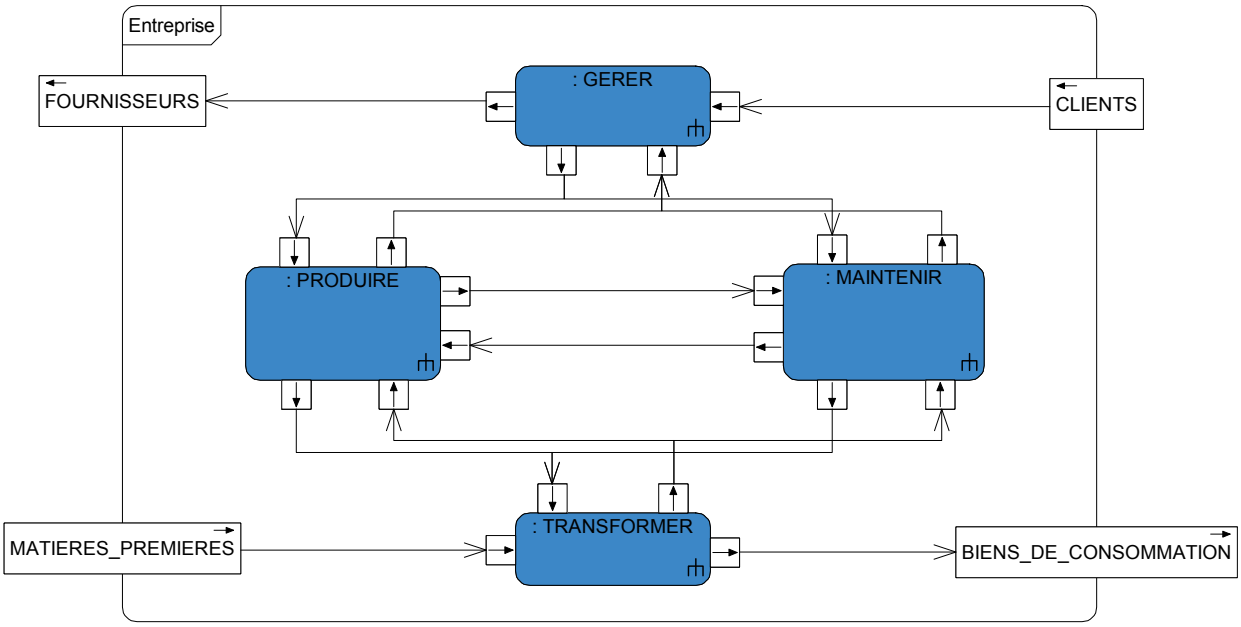


Figure 3.7 : Schéma fonctionnel d'une entreprise (inspiré de Grout et Salaün (2009))

- L'activité *Produire* représente l'utilisation des moyens de production qui vont *transformer* les matières premières et de l'énergie en biens de consommation, en respectant les spécifications des produits.
- L'activité *Maintenir* consiste à assurer la meilleure disponibilité de l'outil de production par des actions de maintenance curative et préventive à court, moyen et long terme.
- L'activité *Gérer* comporte deux activités distinctes :
    - *Gérer techniquement* : consiste à gérer les configurations système en vue d'optimiser la quantité de matières premières et d'énergie à utiliser pour assurer la production, et apprécier la disponibilité de l'outil de production afin de maîtriser les coûts de production et d'entretien.
    - *Gérer financièrement* : consiste à trouver les marchés et les moyens à mettre en œuvre au sein de l'entreprise afin d'assurer la rentabilité et la pérennité de l'entreprise.

La conduite est une des fonctions d'un système CMMS. Elle est directement liée à l'activité *Produire* et réalisée à l'aide de l'instrumentation (partie opérative d'un système de production) et du contrôle (partie contrôle d'un système de production).

Les systèmes de conduite sont utilisés par des opérateurs depuis la salle de commande afin de contrôler l'ensemble des OMTi et pour gérer la situation d'état du processus. Des solutions d'assistance à la conduite de ces OMTi ont été proposées à travers le temps : d'une part, des études sur le design et l'ergonomie de la salle de commande ont été réalisées (Vicente, et al.,

1992; O'Hara, et al., 2002; O'Hara, et al., 2004; Galara, et al., 2007; Santos, et al., 2008) ; d'autre part, afin de réduire les risques d'erreurs des opérateurs, des méthodes de structuration des systèmes de Contrôle/Commande (Berruet, et al., 2005) ont été proposées, comme par exemple le module fonctionnel d'automatisme (Galara D. , 1986; Belhimeur, 1989), les filtres de comportement (Lhoste, 1994; Marangé, 2008) ou les systèmes de contrôle multi-agents (Wang, et al., 1997; Dounis, et al., 2009). Un seul problème : ces solutions nécessitant la remontée d'informations du processus, elles ne sont applicables qu'aux OMTi.

Certains de ces travaux (Pétin J.-F. , 1995) ont donné naissance aux systèmes d'Actionnement et de Mesure Intelligents (IAMS) qui ont été intégrés dans le CMMS. « *L'Actionnement et la Mesure Intelligents comprennent l'ensemble complet des fonctions Intelligentes d'Actionnement (IA) et des fonctions Intelligentes de Mesure (IM) nécessaires pour supporter l'automatisation d'un système industriel dans un contexte CMMS. Habituellement, une partie des fonctions d'Actionnement (IA) est implémentée dans les actionneurs intelligents, le reste étant implémenté dans les systèmes de Conduite, Maintenance et Gestion technique. La situation est identique pour les fonctions de Mesure (IM) et les transmetteurs intelligents.* »

Le Module Fonctionnel d'Automatisme (MFA) permet d'intégrer une intelligence technique dans les composants instrumentés (Figure 3.8). Il est composé de quatre modules: validation des objectifs, validation des observations, élaboration des rapports et élaboration des actions. Nous considérons le MFA comme un Filtre de Comportement permettant de filtrer les entrées/sorties du module.

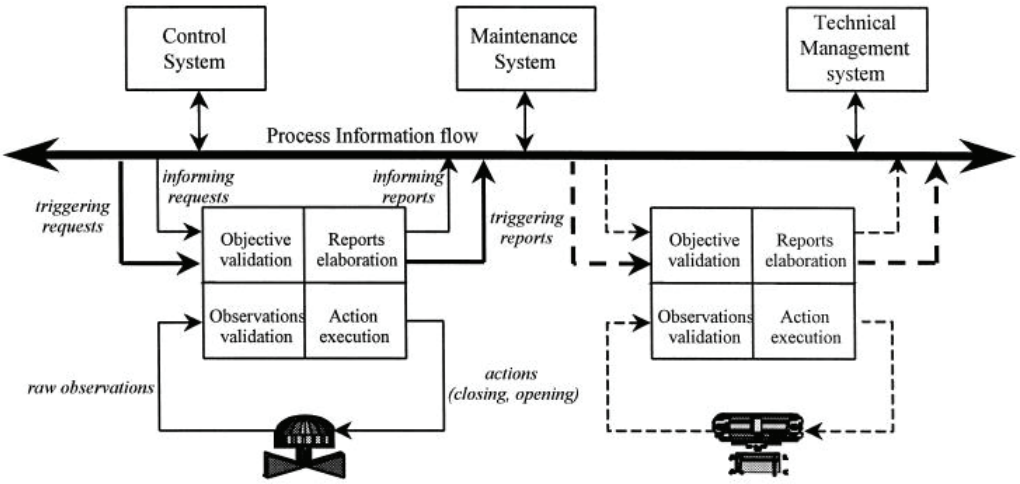


Figure 3.8 : Le Module Fonctionnel d'Automatisme dans le cadre C.M.M.S. (Pétin, et al., 1998)

Étant intéressés par ce domaine de l'automatisation, nous proposons d'utiliser le Filtre de Comportement (MFA) comme patron de conception architectural particulier à ce domaine (Figure 3.9), car il permet de mieux structurer la partie commande d'un système automatisé.

Les OMT sont contrôlés, maintenus et gérés à l'aide du système CMMS. Pour Pétin (1995), « *la distribution de l'intelligence dans les équipements de terrain ne peut se limiter à une distribution des traitements de Conduite, Maintenance ou de Gestion technique induisant une coopération*

entre les équipements de terrain et le processus d'application, mais doit prendre en compte la distribution des informations en dotant chaque équipement d'un système d'information sur lequel reposent les traitements qu'il met en œuvre. » Une telle distribution conduit aux actionneurs et capteurs intelligents, au sens d'Albus (1999), c'est-à-dire coopérant à la réalisation d'un objectif commun dont il est possible d'évaluer la réalisation (Figure 3.10), sur la base d'une image virtuelle de leur environnement.

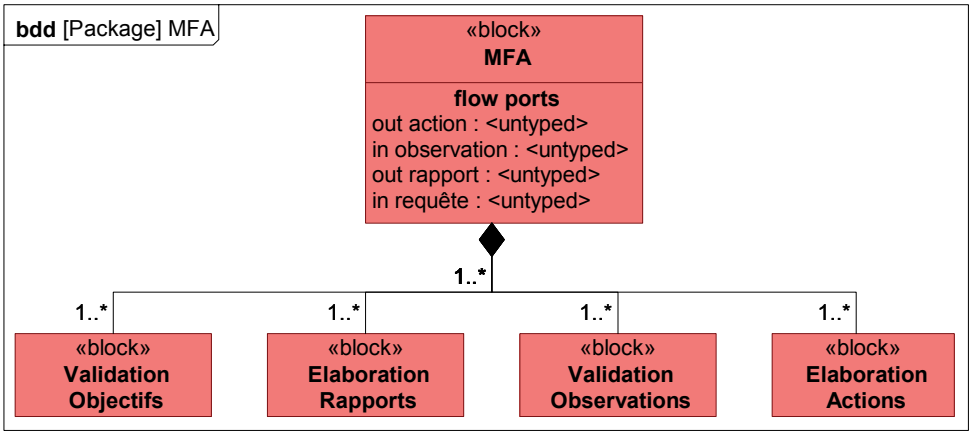


Figure 3.9 : Patron de conception pour le domaine de l'automatisation : le MFA

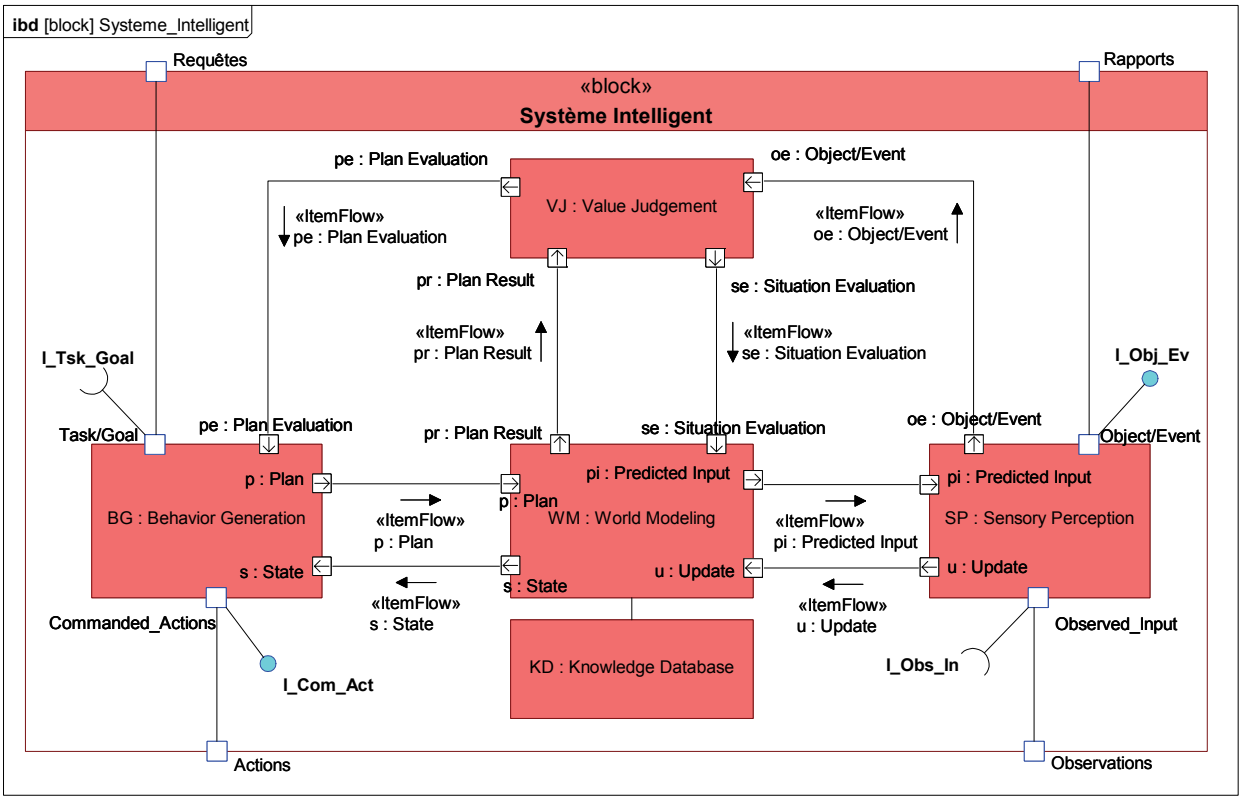


Figure 3.10 : Eléments constituants un système intelligent (d'après Albus (1999))

L'IAMS a été étudié pour être intégré dans les équipements de terrain instrumentés. Mais comme nous l'avons vu, dans les grands systèmes de production, une partie des équipements de terrain sont non-instrumentés (OMTp) et cela pose des problèmes d'interactivité. Afin que

toutes les conditions de sûreté soient remplies par les différents systèmes de conduite, il faut que ces systèmes aient la connaissance de la situation du processus, ce qui inclut l'état des OMTp. Ainsi, il faut s'assurer qu'il n'y a pas de différences entre la situation réelle et la situation logique, c'est-à-dire des écarts entre les valeurs attendues et les valeurs relevées, ce qui pourrait engendrer des actions de conduite erronées.

La manière de conduire, dans des systèmes de production complexes, est décrite dans des Guides d'Exploitation (GE : Figure 3.11). Un GE représente le sommaire ou l'index des procédures opératoires ou Modes Opératoires (MO).

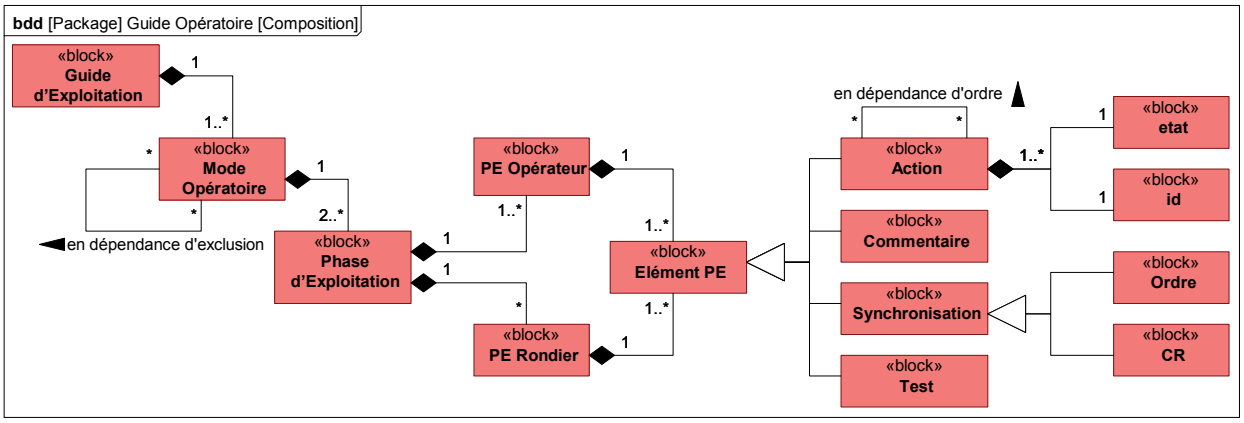


Figure 3.11 : Composition d'un Guide Opératoire

Un MO représente la procédure utilisée pour conduire une partie d'un système de production afin d'exécuter une fonction du procédé. Il recrée le comportement désiré des objets matériels du flux physique (OMF), représentant le flux matériel qui est transformé par le processus. Il est décrit par une liste d'actions (ouverture, fermeture, …) qui vont être exécutées séquentiellement par le personnel sur les OMT qui transforment l'OMF pour réaliser par exemple des opérations de lignage. La liste des actions est complétée par le schéma PID du système sur lequel les actions seront appliquées. Ainsi, un MO est exécuté en deux ou plusieurs Phases d'Exploitation (PE) sous la forme de mise en service (MES) et de mise hors service (MHS) du procédé. Ce sont ces PE qui présentent les actions à exécuter.

De par la physique du procédé, les actions décrites dans les PE doivent être exécutées dans un ordre spécifique, ce qui définit une *dépendance d'ordre* d'actionnement. De même, dans le cas où un OMT est actionné par deux ou plusieurs PE dans des états différents (par exemple une qui ouvre une vanne et une autre qui la ferme), ces PE ne doivent pas s'exécuter en même temps, ce qui implique des *dépendances d'exclusion*. Ces dépendances d'exclusion entre MO précisent ce qu'il est possible à faire, et donc ce qui est interdit pour des raisons matérielles, de sûreté, ou de retour d'expérience. Il donc est important de vérifier les conditions d'exclusions entre MO et les conditions de dépendances entre actions lors de leur exécution.

Nous pouvons pour cela définir trois états d'un MO (Figure 3.12) :

- Disponible dans le cas où il n'est pas exécuté ou qu'aucun autre MO avec lequel il est en exclusion n'est en exécution,
- En exécution quand le MO s'exécute (après le début de MES et avant la fin de MHS),
- En exclusion ou bloqué quand un MO avec lequel il est en exclusion est en exécution.

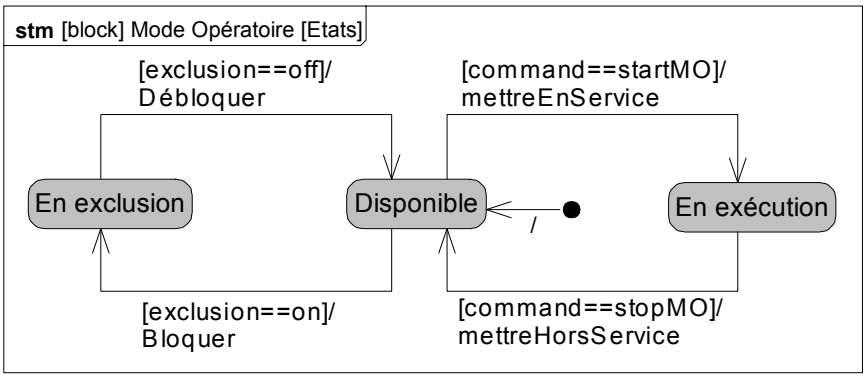


Figure 3.12 : Etats d'un MO

Pour la plateforme CISPI, nous avons proposé un ensemble de procédures de conduite en situation normale, qui prennent en considération la physique du procédé (par exemple on ne peut pas faire un remplissage et un vidage en même temps). Ces procédures sont groupées dans quatre catégories : MO pour le démarrage, MO pour la production, MO pour l'arrêt et MO pour la maintenance, et définissent des dépendances d'exclusion entre elles. Les MO ainsi que la grille des exclusions sont présentés dans l'Annexe B.

La conduite des processus continus à base de Modes Opératoires est vue comme une évolution d'un état stable initial vers un état stable final, à travers le temps (Figure 3.13). Cette évolution est réalisée par des actions exécutées de façon discrète. Nous pouvons distinguer la vue diachronique (évolution du système à travers le temps) de la vue synchronique (le changement d'état à un moment donné).

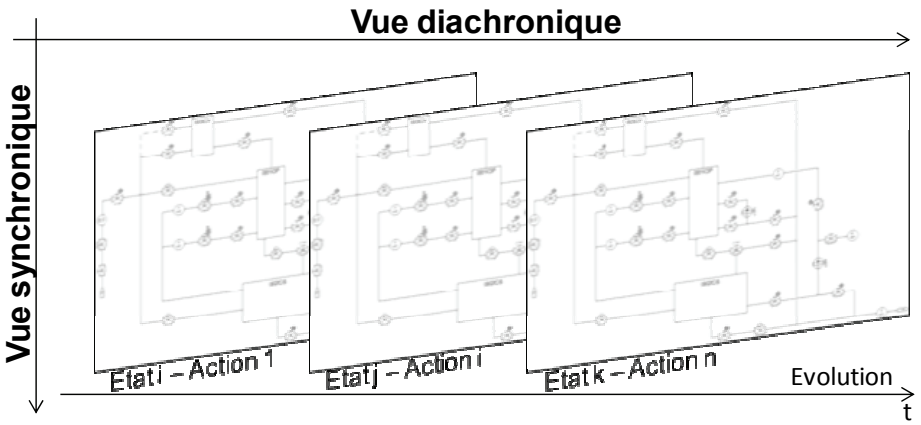


Figure 3.13 : L'évolution d'un processus continu est réalisée par des procédures qui décrivent des actions réalisées de façon discrète (d'après Galara et Hennebicq (1999))

Cherchant à formaliser le comportement des MO, plusieurs éléments doivent être considérés :

- les différentes dépendances d'exclusion entre les MO et de précédence entre les actions doivent être vérifiées avant l'exécution des actions,
- il doit y avoir une logique d'exécution des actions décrites dans le MO.

Afin de tenir compte de ces éléments, notre proposition est de reproduire au mieux le comportement du processus avec de l'information encapsulée dans des objets logiques. Leur rôle est d'assister l'opérateur de conduite pour la vérification et la validation du suivi des actions réalisées sur le processus, et la vérification des dépendances d'exclusions qui peuvent apparaître. Le concept d'objet logique nous permet de définir un canal de communication permettant de faire passer les informations entre objets. Ce canal de communication peut être celui utilisé par le CMMS/IAMS, ce qui permettrait l'interopérabilité entre ces objets logiques et le système de conduite, maintenance et gestion technique.

Afin de définir une logique d'exécution des actions, nous proposons de formaliser l'ensemble des actions décrites dans les MO en suivant les opérations antagonistes proposées par le patron de séquence. Ainsi, les actinomies nous permettent de définir la MHS d'un MO comme une exécution inverse de la MES.

Dans un contexte industriel fortement automatisé, la priorité a été mise depuis longtemps sur les fonctions automatiques de pilotage. Néanmoins l'interaction nécessaire des opérateurs humains implique la mise en œuvre des nouvelles relations entre les opérateurs, les équipements et le système de conduite. De par l'intégration dans les OMTp de fonctions avancées de gestion des actions ainsi que de par le reflet du procédé dans des objets logiques, il faut fournir aux opérateurs de conduite des interfaces interactives afin de pouvoir interagir et échanger des informations avec ces systèmes via le canal d'informations.

3.2 Besoin d'un Système Interactif d'Aide à la Conduite

Le paragraphe précédent pose le contexte et présente des problèmes liés à la conduite des systèmes industriels. Nous avons vu que l'interaction entre les opérateurs de conduite (principalement le rondier) et le processus est essentielle pour bien exécuter les fonctions de ce procédé. C'est pour cela que nous nous intéressons à l'interaction entre les rondiers et le système physique dans le but de conduire localement l'ensemble des composants non-instrumentés (Figure 3.14).

Même si de nombreuses mesures de sécurité sont prises en considération pour la conduite des grands systèmes à risque, il s'avère que des problèmes peuvent encore apparaître. A partir d'informations recueillies par le biais d'interviews de personnes impliqués dans la conduite de grands systèmes à risques, nous avons pu établir un diagramme cause-effet de type 6M présenté à la Figure 3.15. De cette manière, nous avons classé les différentes causes de problèmes liés à la conduite dans 6 catégories :

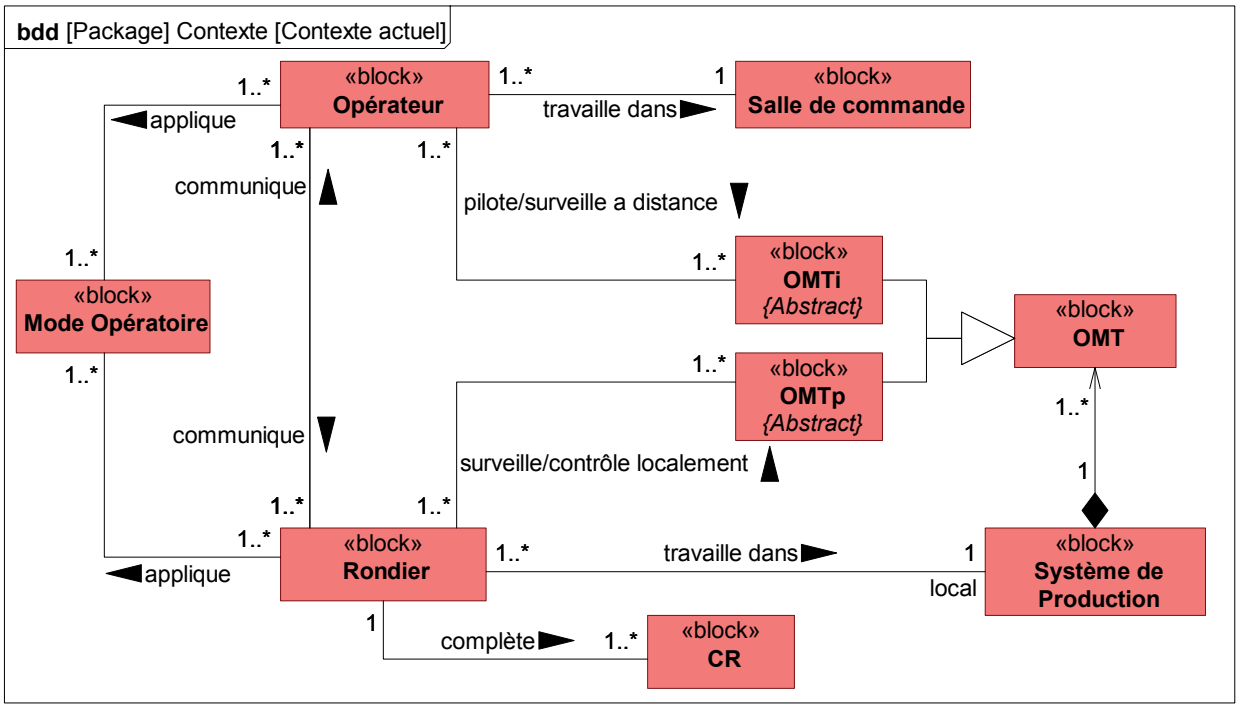


Figure 3.14 : Contexte actuel de la conduite des processus industriels

- main d'œuvre : causes liées aux opérateurs de conduite ou rondiers, ou à la nature humaine ;
- moyens financiers : problèmes liées au manque d'instrumentation sur la majorité des composants et d'équipement pour le rondier ;
- machines : problèmes causés par les OTB ;
- méthode : problèmes causés par la méthode de travail, c'est-à-dire la conception et l'exploitation du mode opératoire ;
- milieu : problèmes engendrés par l'environnement ;
- management : problèmes causés par des facteurs organisationnels.

Notre intérêt et l'objectif de cette application est de proposer un Système Interactif d'Aide à la Conduite (SIAC) qui pourrait amener des solutions à certains de ces problèmes de conduite.

Notre proposition initiale (Figure 5) du SIAC consiste à ajouter autour du processus un tuyau d'informations dans lequel s'intègrent des objets logiques communicants qui réfléchissent les OMT, les flux physiques (limités aux modes opératoires) et les rondiers. Ainsi, nous proposons au personnel de conduite d'accéder à ces informations utiles pour la conduite du processus par un canal de services.

Afin d'identifier de manière précise les besoins auxquels devra répondre le SIAC, nous commençons par identifier les parties prenantes, intégrées ou ayant un intérêt dans l'environnement d'exploitation pour ensuite structurer leur besoin.

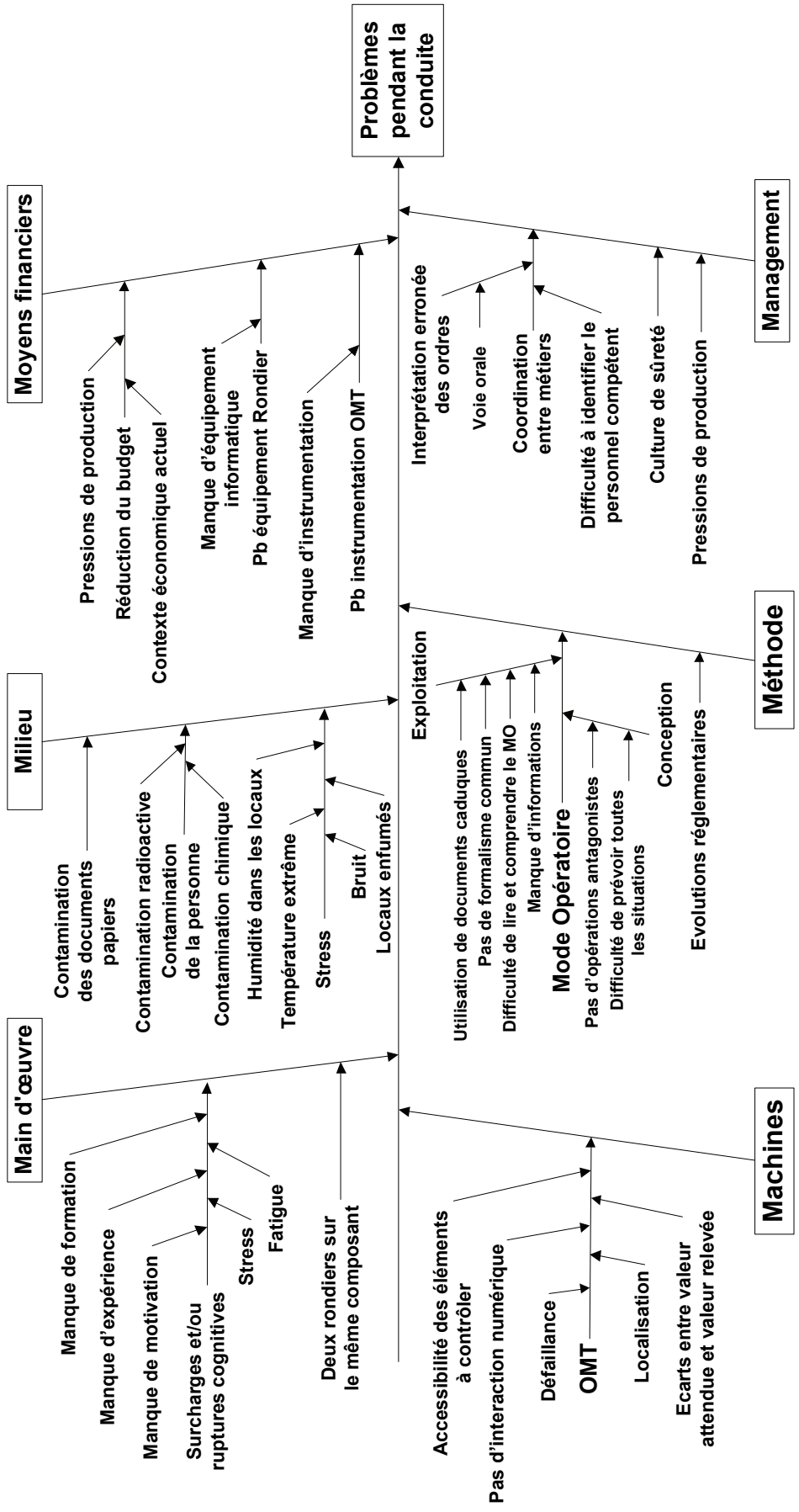


Figure 3.15 : Diagramme cause-effet pour les problèmes de conduite d'un système de production

Par exemple, les parties prenantes réalisatrices (PPR) sont représentées dans le diagramme de bloc à la Figure 3.16. Normalement, ce diagramme avec les PPR se construit au fur et à mesure de l'évolution du projet et de l'émergence de nouveaux espaces de solutions relatifs à des besoins de conception ou de réalisation. Le nombre de PPR montre ici la dimension du travail à réaliser.

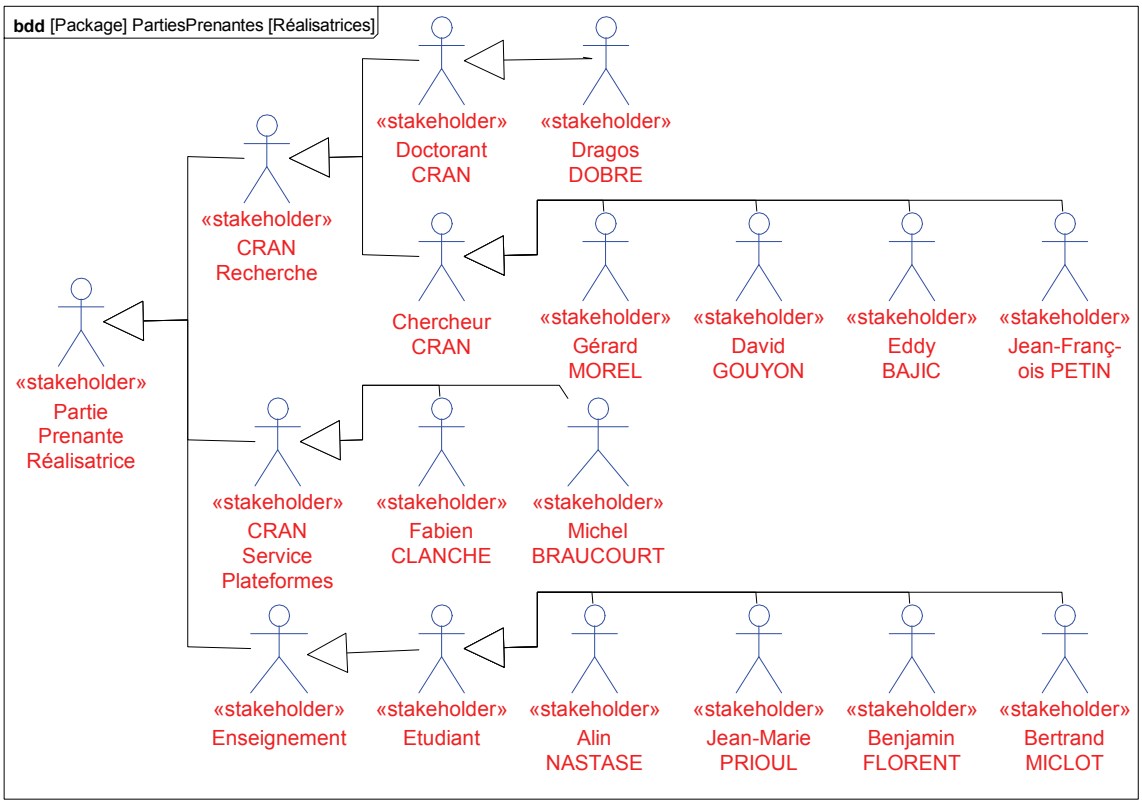


Figure 3.16 : Diagramme de bloc représentant des parties prenantes impliquées

Différentes études sur l'assistance à l'opérateur ont été réalisées (Alengry, 1985; Cahour, et al., 1991). D'après Boy (1988), l'assistance à l'opérateur a trois objectifs :

- la conception de systèmes à bases de connaissances en environnements réactifs,
- la prise en compte de l'acquisition incrémentale des connaissances,
- le développement d'interfaces « intelligences », conçues comme des prolongements de la mémoire à court terme de l'opérateur.

De plus, Pirus (2006) illustre trois catégories de besoins pour l'assistance des rondiers :

- Avoir accès, en temps réel, à une représentation détaillée de l'état physique du processus ;
- Besoin de communication : orale, visuelle et de données ;
- Besoin de soutien pour améliorer la qualité des activités :
    - Améliorer la fiabilité des valeurs enregistrées ;
    - Vérifier que toutes les actions requises soient réalisées ;

- Aider les rondiers à détecter plus rapidement les différences entre les valeurs réelles et les valeurs attendues ;
- Encourager une attitude prudente et interrogative ;
- Aider les opérations atypiques ;
- Accélérer la localisation des équipements ;
- Aider les activités sensibles avec des calculs automatisés ;
- Accéder rapidement aux informations complémentaires ;
- Éliminer les documents papier, en particulier dans les zones radioactives.

Un exemple de décomposition des besoins fonctionnels exprimés par Pirus est donné à la Figure 3.17. La liste des besoins identifiés figure dans l'Annexe A.

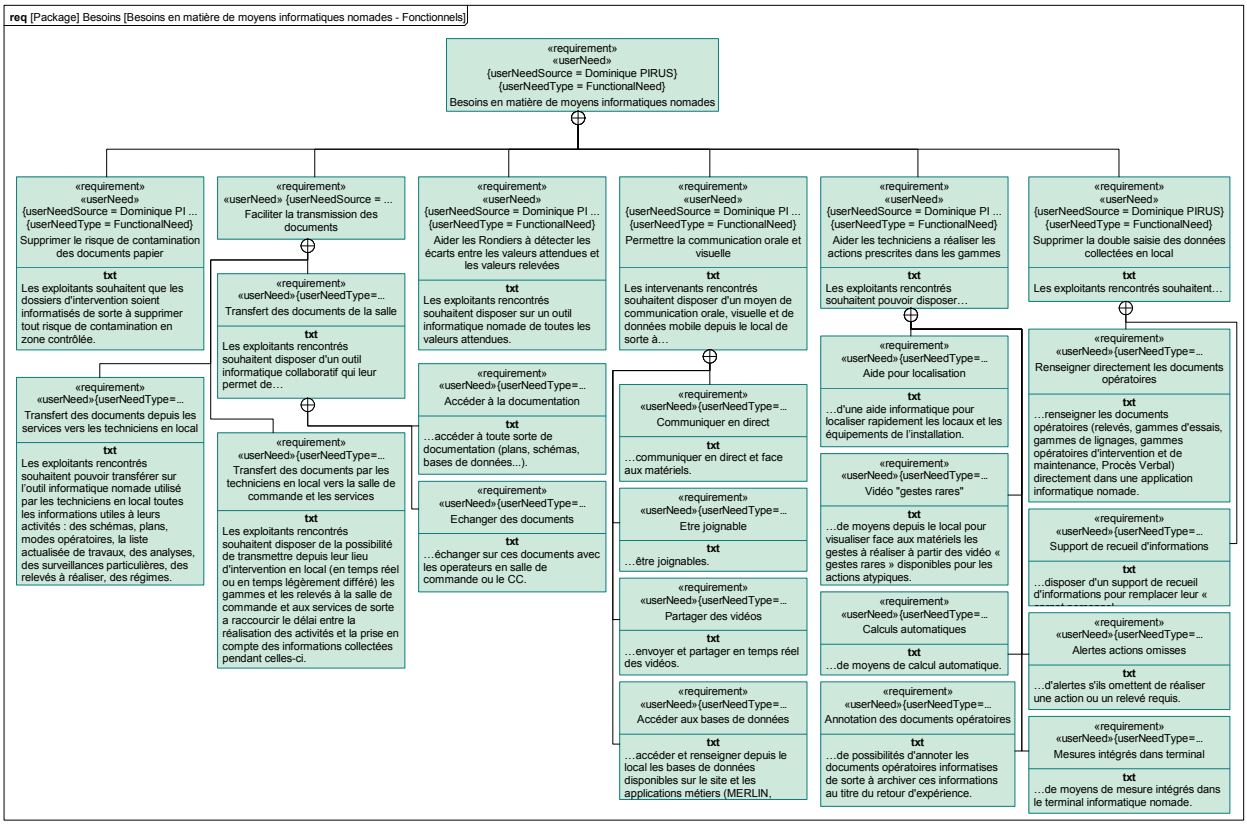


Figure 3.17 : Exemple structuré de besoins en matière de moyens informatiques nomades

Etant donnée la complexité de la solution théorique proposée et à des fins de lisibilité, ce chapitre va se focaliser sur l'étude et la réponse à l'un des besoins : « Aligner le processus en local » (Figure 3.18). En effet, du fait que les problèmes posés viennent plutôt de la conduite en local, le besoin de lignage du processus devient un point de départ clé dans l'analyse du SIAC.

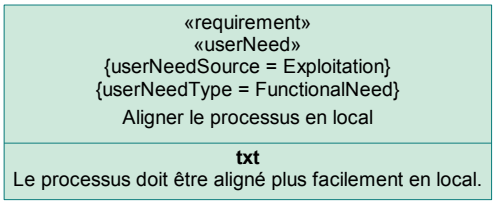


Figure 3.18 : Le besoin fonctionnel qui est traité dans ce chapitre

3.3 Analyse des exigences

La première phase de l'analyse des exigences est la description de la hiérarchie du contexte du SIAC. Ce dernier sera intégré sur la plateforme CISPI du CRAN, son environnement (Figure 3.19) se compose donc des acteurs de conduite (opérateur et rondier), du CMMS (conduite, maintenance et gestion technique) et de la plateforme CISPI existante (nous nous intéressons plus particulièrement aux vannes manuelles).

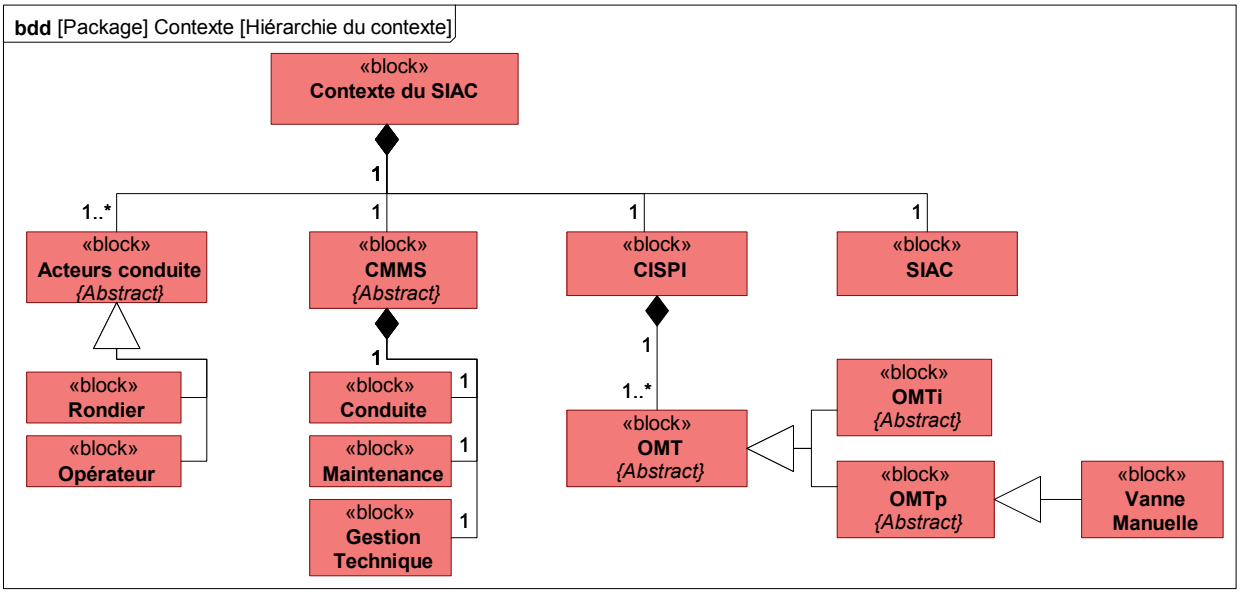


Figure 3.19 : Hiérarchie du contexte du SIAC

Le modèle de la structure interne du contexte (Figure 3.20) présente l'environnement direct du système et donne les premières informations sur les relations entre les éléments de l'environnement et le système. Dans cette structure, le SIAC est vu comme une boite noire qui assiste l'opérateur et le rondier pendant la conduite des procédés, rend compte au système de conduite et ajoute des capacités d'acquisition, de traitement, de stockage et de communication de l'information des vannes manuelles, afin de permettre une meilleure interaction entre : rondier et CMMS, rondier et OMT, et les OMT entre eux.

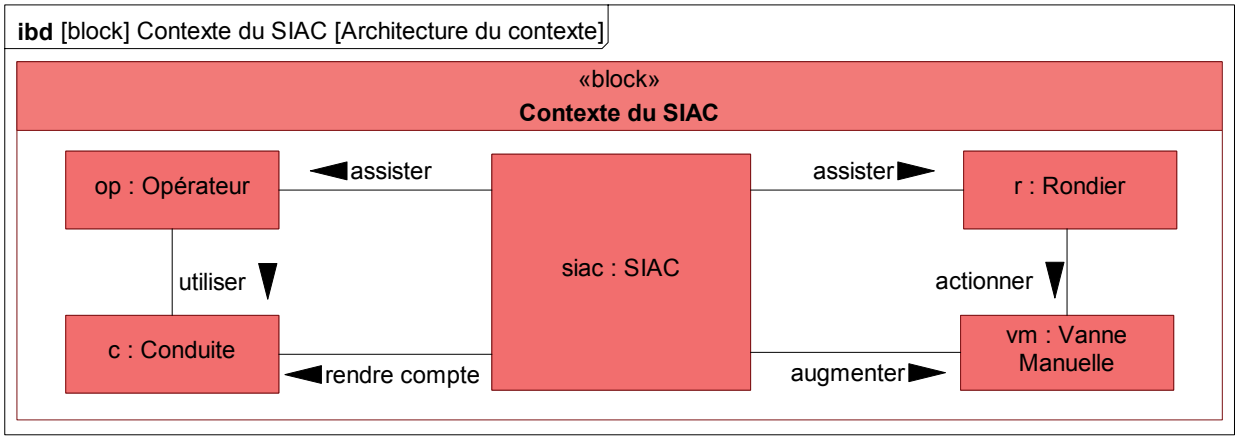


Figure 3.20 : Architecture du contexte du SIAC

Le besoin fonctionnel initial, « *Aligner le processus en local* », est dérivé pour obtenir l'exigence fonctionnelle « *Faciliter le lignage en local* » (Figure 3.21). Cette dérivation est réalisée à l'aide de la règle de passage d'un besoin vers une exigence.

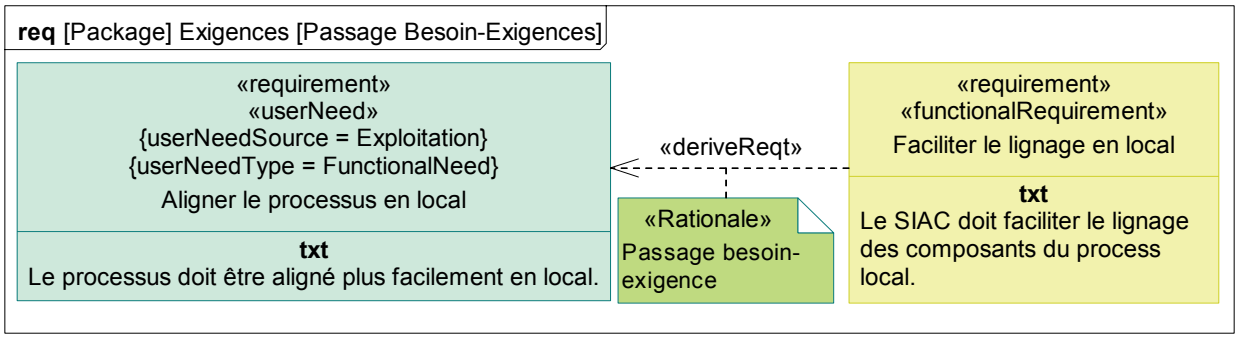


Figure 3.21 : Dérivation du besoin fonctionnel en exigence fonctionnelle

La première question que nous nous sommes posée est : « qu'est ce que le lignage en local ? ». La réponse nous a été donnée par une connaissance métier (une exigence indicative) : « *Le lignage des composants du process en local est réalisé par l'exécution des actions en local.* » C'est cette connaissance qui nous guide pour la dérivation de la première exigence vers l'exigence « *Faciliter l'exécution des actions en local* » (Figure 3.22).

Les exigences, et en particulier les exigences fonctionnelles, peuvent être raffinées avec des cas d'utilisation, qui nous permettent de décrire des scénarios opérationnels[36]. C'est pour cela que nous avons choisi d'expliciter l'exigence « *Faciliter l'exécution des actions en local* » à l'aide d'un scénario opérationnel décrit par le cas d'utilisation du même nom.

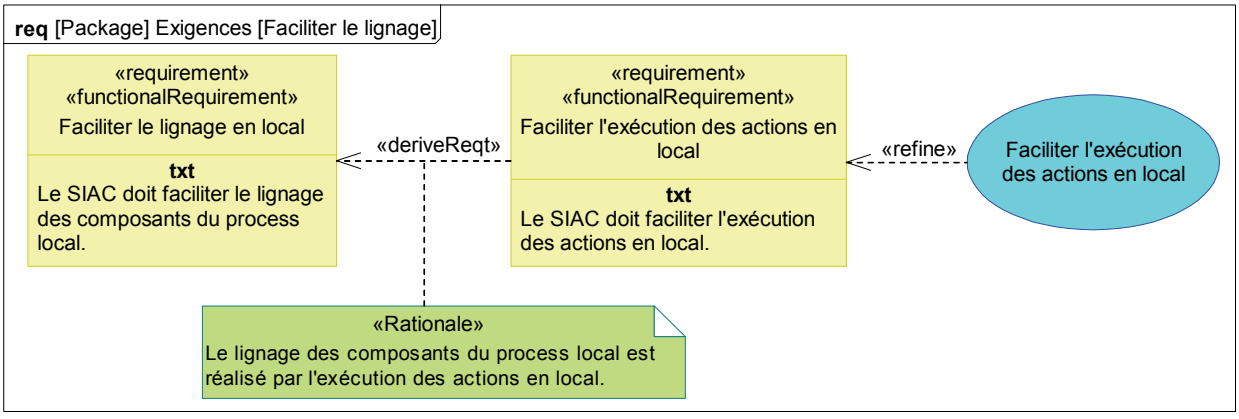


Figure 3.22 : Dérivation de l'exigence fonctionnelle et son raffinement par un cas d'utilisation

Les diagrammes de cas d'utilisation nous permettent de relier les cas d'utilisation avec les éléments de l'environnement qui participent à la réalisation du scénario ainsi décrit. Le cas d'utilisation « Faciliter l'exécution des actions en local » représente une fonction (mode de

[36] Les scénarios opérationnels doivent être validés par les parties prenantes intéressés avant de passer à leur réalisation.

fonctionnement) que le SIAC réalise et qui fait appel aux éléments du contexte (Figure 3.23) : le rondier, l'opérateur de conduite, la conduite et les vannes manuelles.

Du fait que dans un diagramme de cas d'utilisation on ne peut utiliser que des objets de type « Acteur », nous avons choisi d'allouer les acteurs aux blocs correspondants car ce sont eux qui décrivent les vrais constituants du contexte. Ceci nous permet d'utiliser dans le scénario les objets de type bloc du contexte et les opérations associées. Ceci est aussi dû à notre considération des opérateurs comme composants systèmes et donc représentés par des blocs SysML.

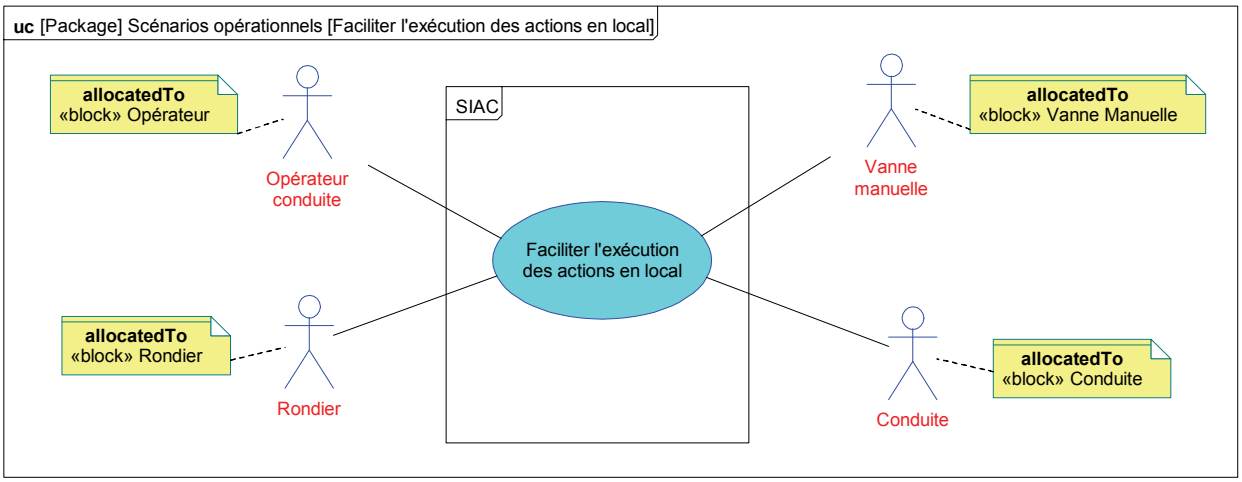


Figure 3.23 : Cas d'utilisation « *Faciliter l'exécution des actions en local* »

Nous décrivons ensuite les cas d'utilisation avec des diagrammes de séquence. Les scénarios opérationnels ainsi définis nous permettent, dans un premier temps, d'identifier les messages qui sont échangés entre le SIAC et les éléments de l'environnement (Figure 3.24).

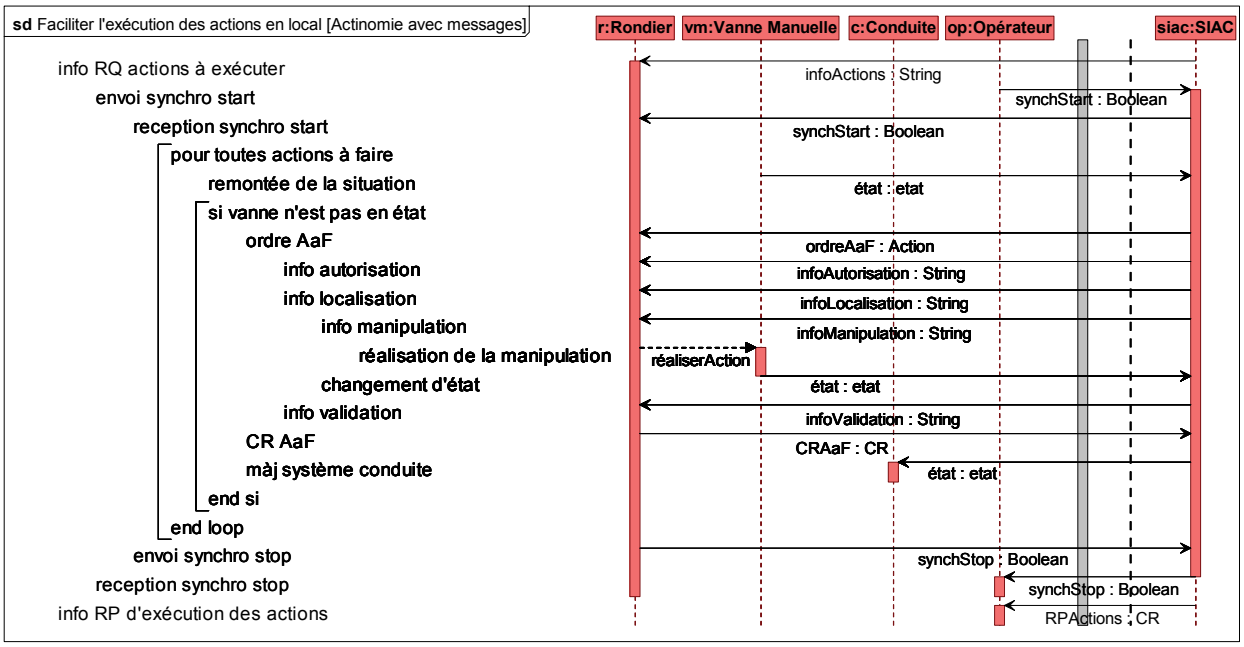


Figure 3.24 : Scénario pour faciliter l'exécution des actions en local

Ainsi, nous montrons dynamiquement l'interaction entre le système et son environnement dans le cadre d'un contrôle local. Pour structurer les messages envoyés nous utilisons le patron de séquence. Nous pouvons observer qu'à la réception d'une requête d'action à exécuter, le système doit envoyer un rapport sur les actions qui ont été exécutées.

Comme précisé précédemment, nous avons décidé de structurer le SIAC en utilisant le patron d'architecture de type Filtre de Comportement. Nous raffinons le diagramme de contexte défini initialement en ajoutant ainsi les ports du patron d'architecture utilisé et les messages échangés dans le scénario opérationnel (Figure 3.25).

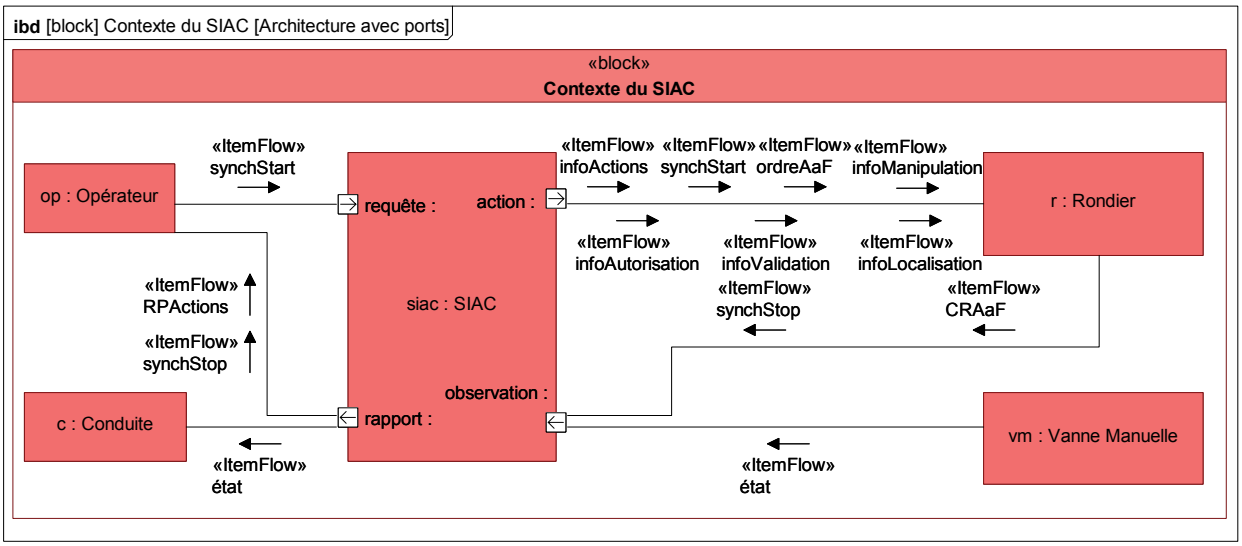


Figure 3.25 : Architecture du contexte avec ports et messages échangés

Cela nous permet d'identifier, d'une part, les exigences d'interface définies par les échanges et, d'autre part, les exigences fonctionnelles que le SIAC doit satisfaire. Par exemple, chaque message fourni par le SIAC doit être le résultat d'une fonction qui est exécutée par celui-ci. C'est pour cela que nous retournerons vers la définition des exigences afin de l'enrichir en ajoutant ces nouvelles exigences induites.

Nous avons choisi de reprendre certaines des fonctions qui sont exécutées normalement par le rondier et de les intégrer dans le SIAC (Figure 3.26), afin d'alléger sa charge, notamment dans le cas de fonctions ne dépendant pas de la capacité de jugement. Parmi ces fonctions, que nous retrouvons sous la forme d'exigences fonctionnelles, nous avons :

- la transmission des actions de conduite en local et la transmission des rapports vers la salle de commande ;
- la transmission des messages de synchronisation entre le rondier et l'opérateur ;
- la remontée de la situation en local et vers le poste de conduite ;
- la gestion des actions à faire avec l'affichage de l'action courante ;
- l'autorisation des actions à exécuter, et en particulier des autres actions, non liées à un mode opératoire, proposées par exemple dans le cas de la maintenance ;
- la localisation des vannes à manipuler avec confirmation ;

- l'assistance de la manipulation ;
- la validation de l'action exécutée ;
- la gestion des comptes rendus.

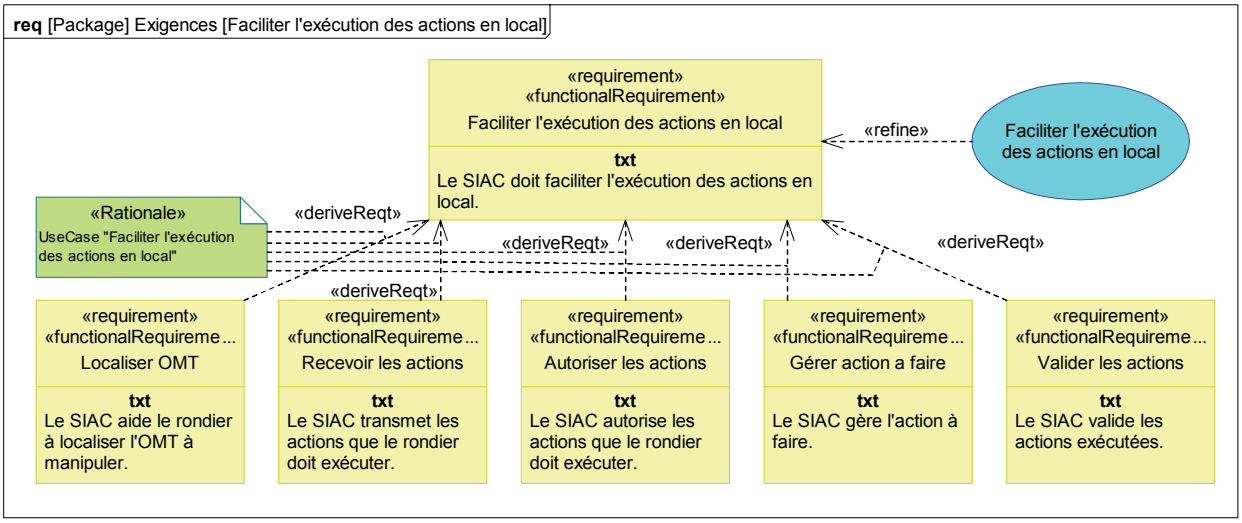


Figure 3.26 : Modèle partiel des exigences fonctionnelles induites par le scénario opérationnel

Étant donné que nous avons à faire avec des entités de natures différentes, il est judicieux de définir les interfaces qui sont requises et fournies par le SIAC (Figure 3.27).

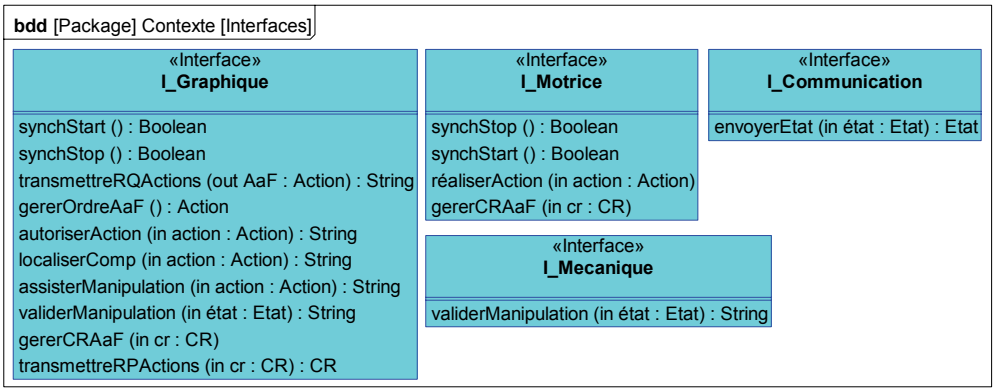


Figure 3.27 : Interfaces du SIAC

En effet, ce sont ces interfaces qui définissent les fonctions qui sont lancées par ou à destination de l'environnement (Figure 3.28).

D'une part, nous avons des interfaces pour interagir avec l'humain : graphiques, tactiles. D'autre part il y a des interfaces qui permettent d'échanger avec d'autres éléments de l'environnement, comme la vanne manuelle via une interface mécanique, ou avec le système de conduite par l'intermédiaire d'une interface de communication.

Ces fonctions sont de plus introduites dans le scénario opérationnel pour obtenir la séquence représentée à la Figure 3.29.

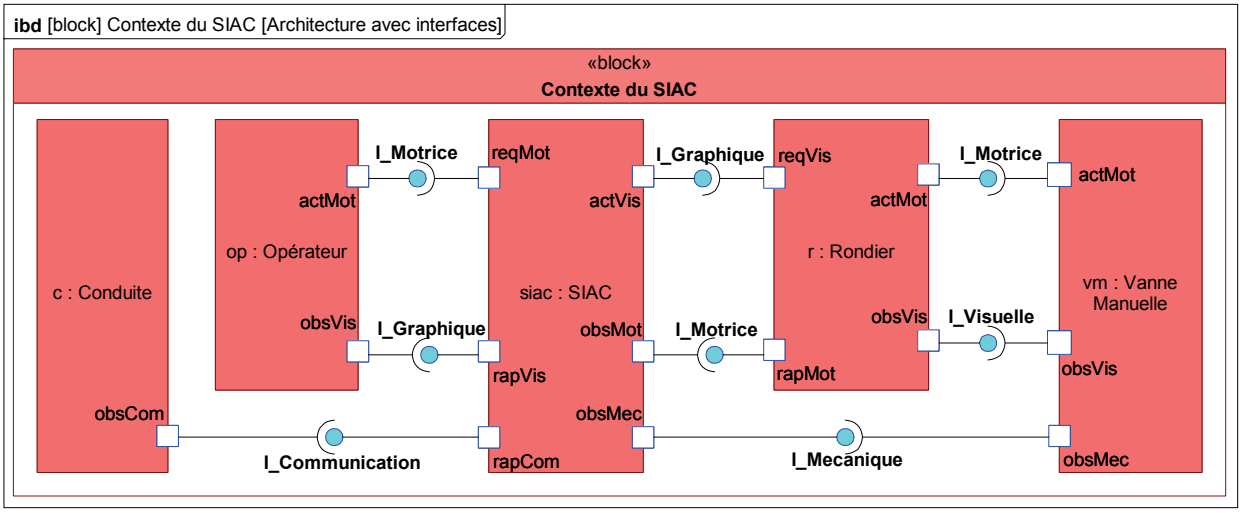


Figure 3.28 : Le contexte du SIAC avec les interfaces fournies et requises

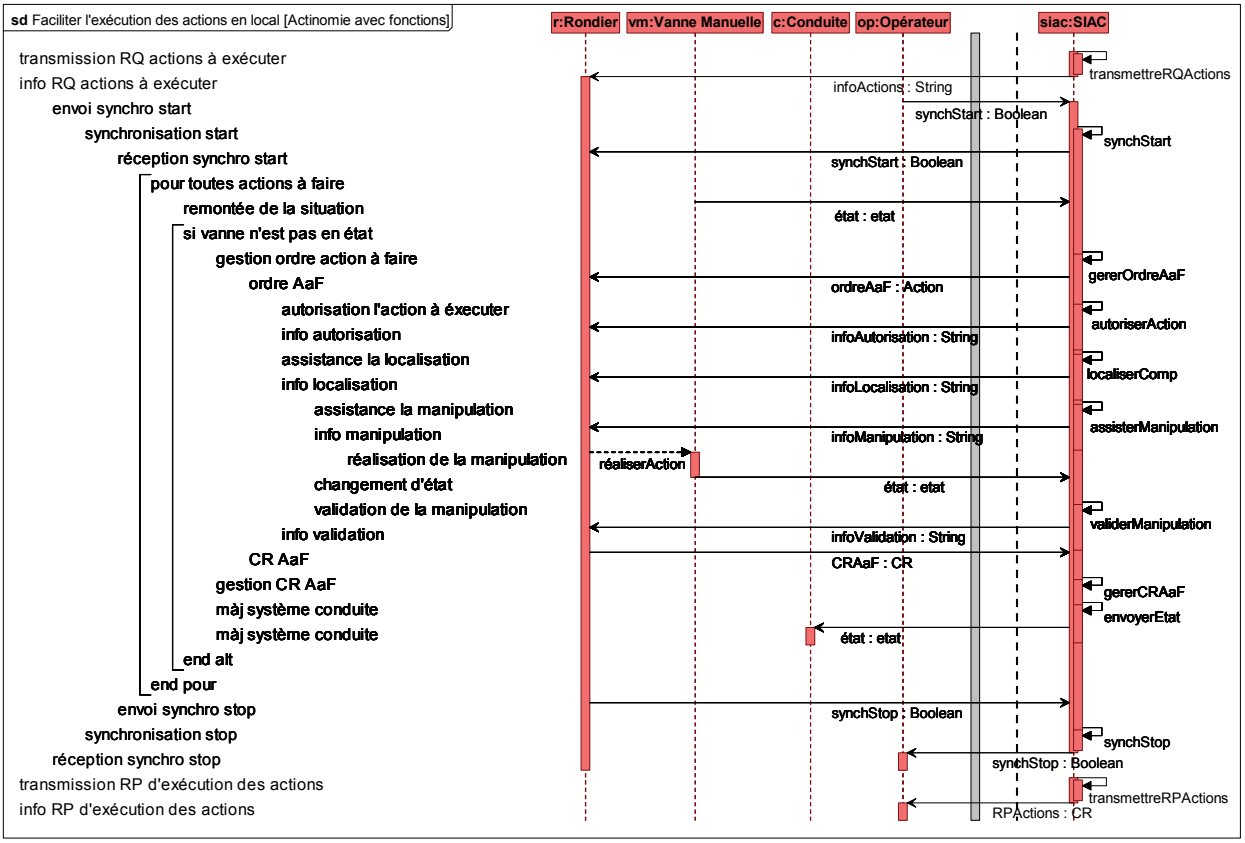


Figure 3.29 : Scénario opérationnel montrant l'exécution des fonctions du SIAC

Il est à noter que dans ce diagramme, ces fonctions sont des opérations réalisées par le bloc SIAC. Chaque opération peut de plus être décrite par une activité qui modélise l'algorithme ou le comportement de l'opération, ce qui va dans le sens de notre choix de représenter les fonctions à l'aide d'activités.

3.4 Conception fonctionnelle

Du point de vue fonctionnel, chaque exigence fonctionnelle est satisfaite par une fonction modélisée sous la forme d'une activité SysML. Ainsi, nous avons explicité un ensemble de fonctions que nous devons architecturer, afin d'arriver à générer un comportement qui permettra au futur système de répondre aux besoins. Ce sont ces fonctions représentées sous la forme d'activités qui sont reliées par un lien de type « satisfy » aux exigences qu'elles satisfont (Figure 3.30).

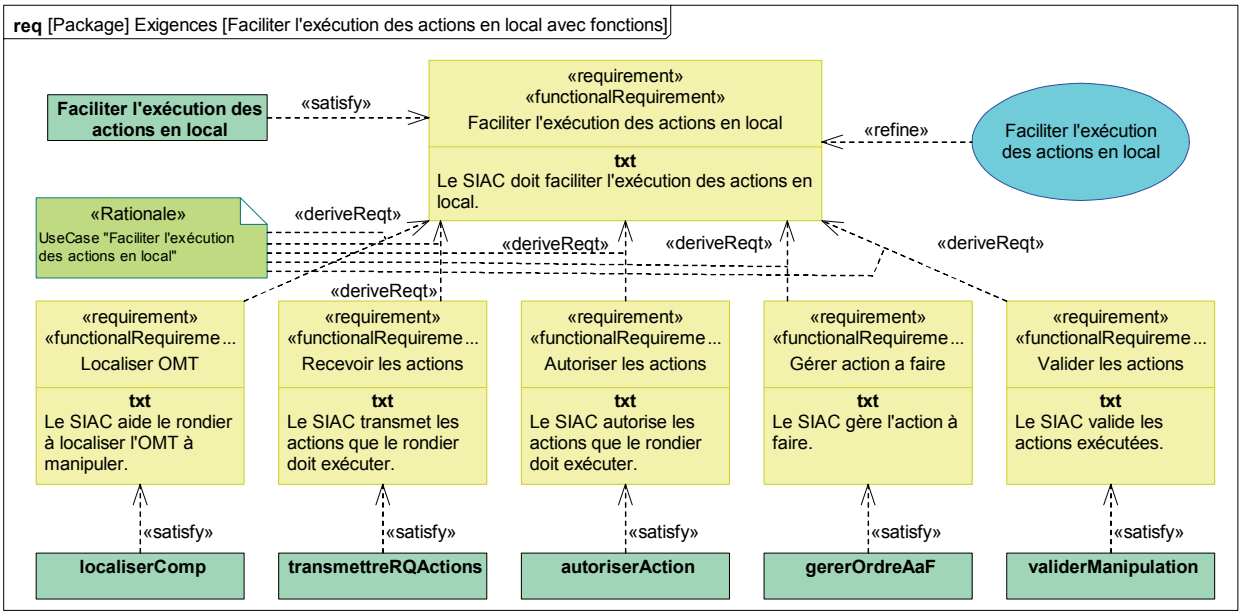


Figure 3.30 : Liens de satisfaction des activités vers les exigences fonctionnelles

Comme les exigences fonctionnelles sont dérivées d'une exigence fonctionnelle principale, nous allons d'une manière similaire composer l'activité qui satisfait l'exigence principale (« Faciliter l'exécution des actions en local ») par des activités qui satisfont les exigences dérivées. Cela nous donne le modèle structuré et hiérarchisé des activités décrit à la Figure 3.31, réalisé à l'aide d'un diagramme de bloc.

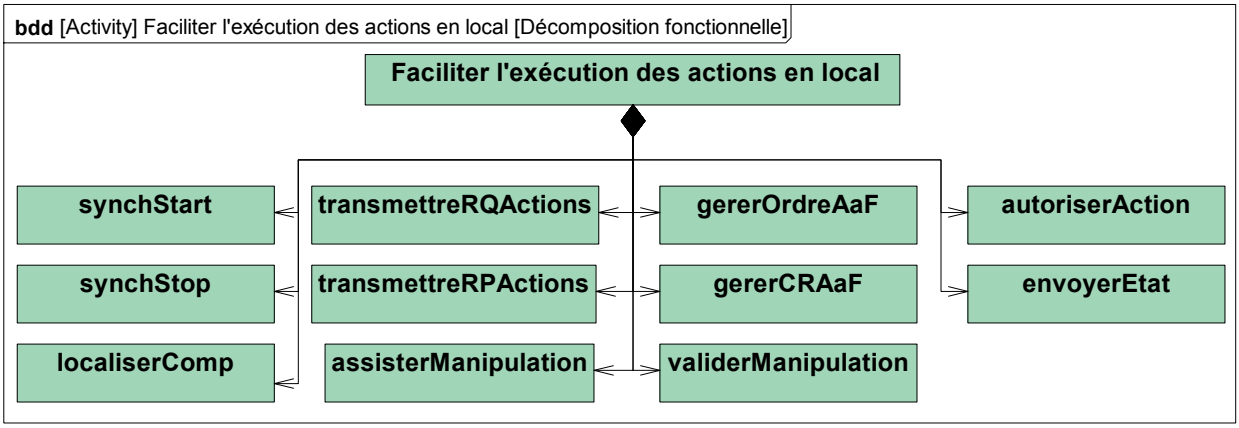


Figure 3.31 : Décomposition fonctionnelle de l'activité « Faciliter l'exécution des actions en local »

L'étape suivante est de rassembler ces activités dans une structure fonctionnelle. Chaque activité peut comporter des paramètres d'entrées, de sortie ou, comme en informatique, peut retourner des valeurs. Ces paramètres sont représentés dans les diagrammes d'activités sous la forme de ports. En effet, un diagramme d'activités modélise la structure interne d'une activité en structurant les autres activités qui rentrent dans sa composition. Dans notre cas, l'activité « Faciliter l'exécution des actions en local » est modélisée à la Figure 3.32.

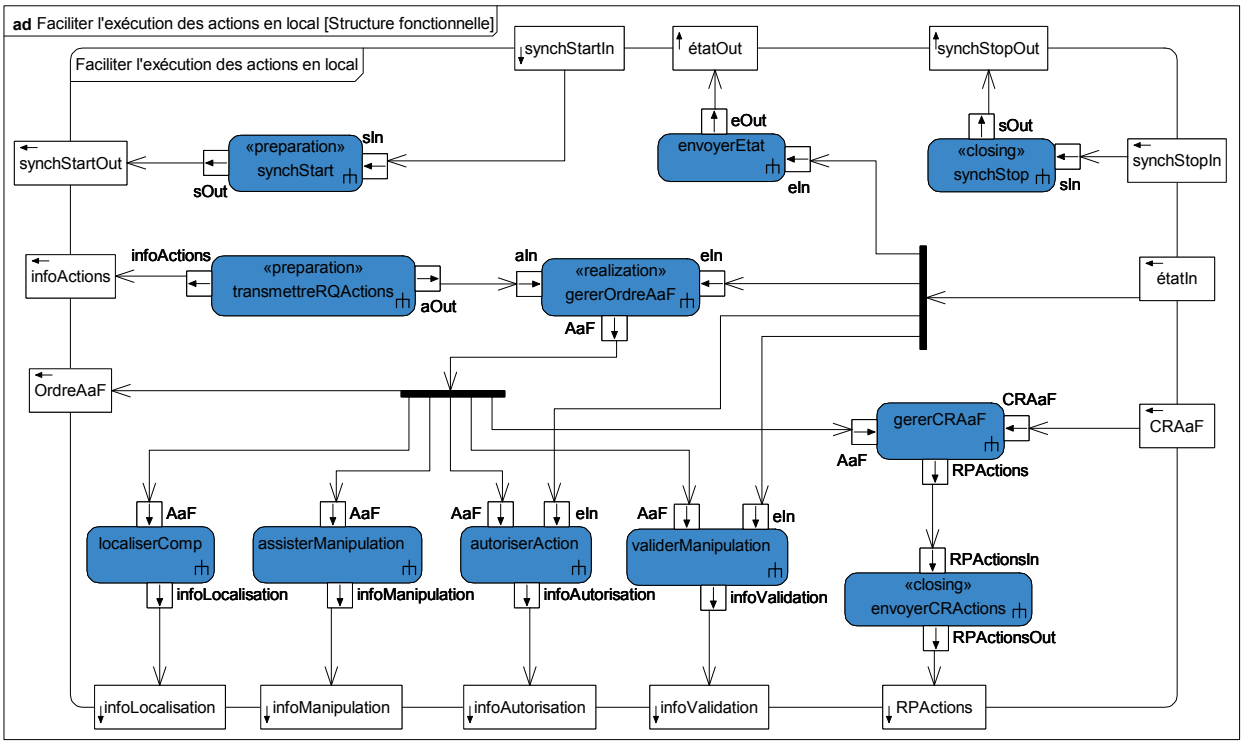


Figure 3.32 : Structure fonctionnelle de l'activité « Faciliter l'exécution des actions en local »

Sur la frontière de l'activité principale nous avons représenté les paramètres d'entrées/sorties de cette activité. Se sont ces paramètres qui sont échangés avec l'environnement. Parmi eux, nous pouvons citer l'action à exécuter qui est transmise au rondier, des informations concernant l'autorisation, la localisation, la manipulation, le résultat de validation, …

A l'intérieur de l'activité principale nous avons structuré les activités avec leurs paramètres en montrant les flux qu'elles s'échangent. Par exemple, l'activité « gererOrdreAaF » permet de récupérer l'action courante, en fonction de la situation sur le terrain, et de la retransmettre au rondier et aux autres activités, qui auront besoin de savoir quelle est l'action que le rondier doit exécuter pour l'autorisation, localisation, vérification, …

La structure fonctionnelle représentée à la Figure 3.32 ne modélise pas un comportement, mais uniquement les échanges entre les différentes actions réalisées à l'intérieur de l'activité. Pour modéliser le comportement que la fonction principale exécute, on doit ajouter le flux de contrôle qui définit l'enchainement des actions à travers le temps (Figure 3.33).

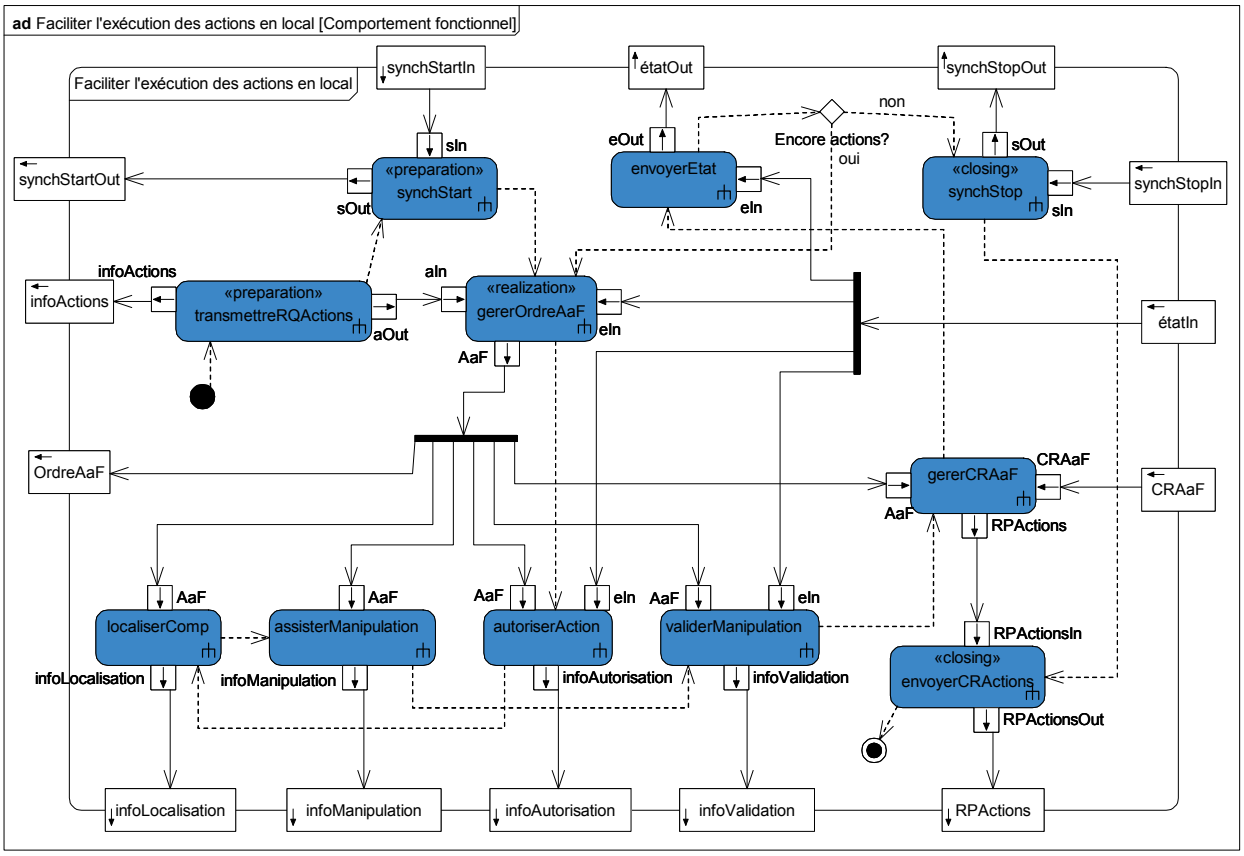


Figure 3.33 : Comportement fonctionnel de l'activité « Faciliter l'exécution des actions en local »

La conception fonctionnelle s'arrête à la définition du comportement fonctionnel du système sans allocation de ces fonctions à des composants, ce qui est fait lors de la conception organique.

3.5 Conception organique

Dans la conception organique nous allouons les fonctions (activités) à des constituants logiques avec comme objectif de définir une architecture organique du SIAC. La partition est réalisée à l'aide de diagrammes d'activités.

Comme cela est proposé dans la solution théorique représentée à la Figure 5, nous avons définis dans la section précédente différentes fonctions du SIAC. Ces fonctions, allouées à des Objets Logiques (OL), doivent pour certaines être intégrées au plus près du processus (si elles concernent l'évolution du processus), pour d'autres au plus près du rondier (si elles concernent les services rendus au rondier) ou encore au plus près du poste de conduite (si elles concernent les services rendus à l'opérateur). C'est pour cela que nous allons, dans une première « passe » d'ingénierie, partitionner notre solution en trois types de constituants organiques :

- l'Objet Logique Technique (OLT) qui va être intégré à la vanne manuelle. Il va réaliser les fonctions dépendantes des attributs statiques (identifiant, localisation, …) ou dynamiques (état) de la vanne. Par exemple, l'autorisation de l'action à exécuter est dépen-

dante au moins de l'état de la vanne et la vérification de l'action doit prendre en considération l'état de la vanne ainsi que l'action à exécuter.

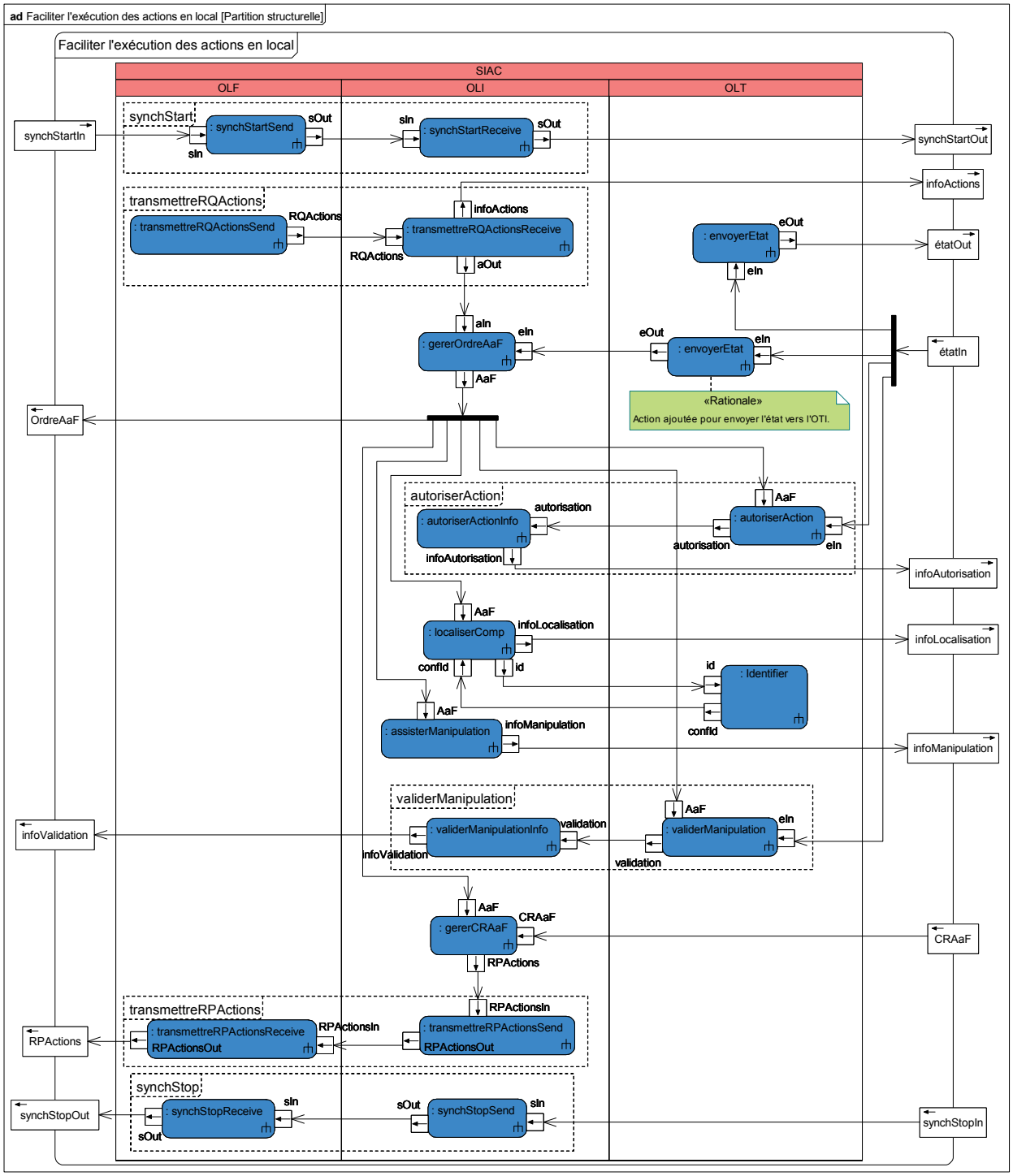


Figure 3.34 : Allocation des fonctions aux Objets Logiques

- l'Objet Logique de Flux (OLF), représente le comportement des objets matériels du flux physique (OMF[37]), avec le niveau d'abstraction correspondant aux modes opératoires. Par exemple, l'exécution des modes opératoires nécessite des informations de synchro-

[37] Ce qui est transformé par le processus physique ; dans notre cas le fluide qui circule dans le système

nisation telles que l'envoi de requêtes et la réception de comptes-rendus concernant les opérations à exécuter. La mise en services et hors service de ces MO étant réalisées par l'opérateur en salle de commande, les OLF seront intégrés au plus près de l'opérateur de conduite.

- l'Objet Logique Interactif (OLI), qui permet au rondier d'accéder aux services d'aide à la conduite et d'interagir avec les autres objets logiques. Par exemple, l'OLI permet au rondier de connaitre l'identifiant d'un OMT par l'intermédiaire de son OLT.

Par conséquent, toutes les fonctions qui concernent la vanne manuelle (remontée de la situation par transmission de l'état, autorisation de l'action à exécuter et vérification de l'action exécutée) vont être intégrées dans les OLT, toutes les fonctions qui concernent les MO sont intégrées dans les OLF, et les autres qui concernent le rondier vont être fournies par les OLI. La Figure 3.34 nous montre les fonctions réalisées par les différents types d'OL.

Cependant, du fait que nous répartissons les fonctions dans trois constituants, nous avons trouvé des inconsistances. Par exemple, l'allocation de la fonction « gererOrdreAaF » à l'OLI pose des questions sur l'accessibilité à l'état de la vanne. C'est pour cela que nous avons décidé d'utiliser la fonction de transmission d'état (« envoyerEtat ») comme action de l'OLT pour envoyer l'état à l'OLI. De même, la fonction de localisation doit permettre la confirmation que le rondier a localisé la bonne vanne. Ainsi, c'est le rôle de la fonction « Identifier » de confirmer que le rondier a trouvé (identifié) la bonne vanne. Comme cette confirmation est faite par rapport à la vanne recherchée, elle doit être faite par l'OLT attaché à la vanne concernée.

L'étape suivante est de rassembler les constituants ainsi définis dans le système principal SIAC (Figure 3.35).

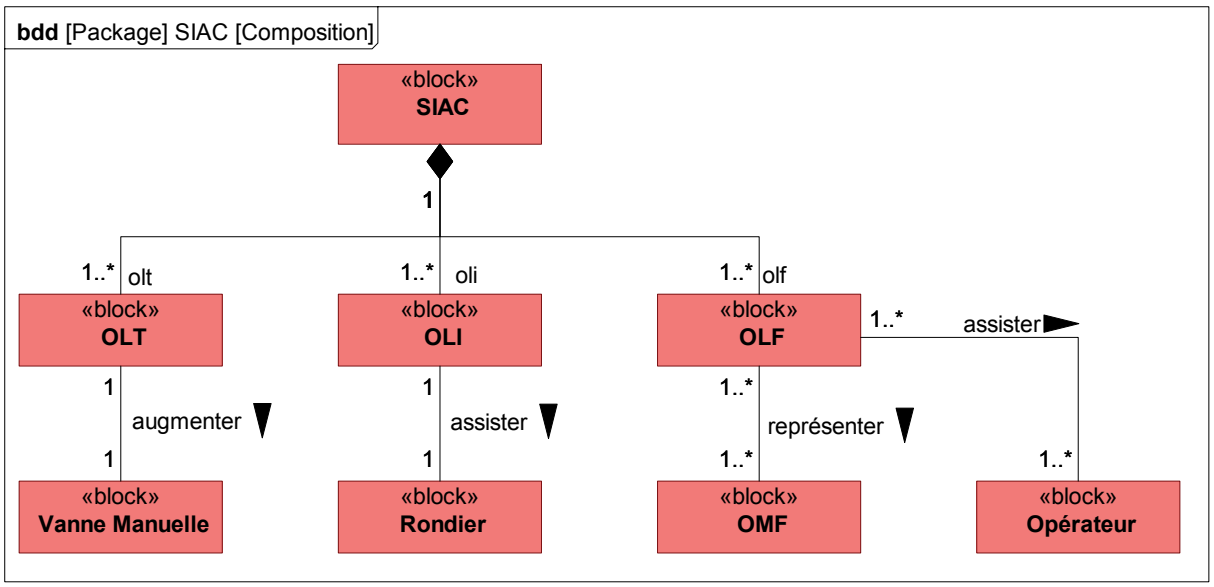


Figure 3.35 : Définition d'une structure hiérarchique par la composition du SAC

L'OLI est utilisé par le rondier et l'OLF doit assister l'opérateur à la mise en œuvre du MO qu'il représente. De la même façon, l'OLT est proposé pour augmenter les capacités passives des

vannes manuelles. Etant donné que dans le contexte du système nous disposons de plusieurs vannes manuelles et de plusieurs rondiers, nous allons avoir un OLT pour chaque vanne et un OLI pour chaque rondier.

Nous proposons de modéliser ces objets logiques sous la forme de Filtres de Comportement comme cela est fait pour le SIAC. Pour cela nous avons spécialisé ces constituants afin d'hériter les ports et les propriétés du Filtre de Comportement (Figure 3.36).

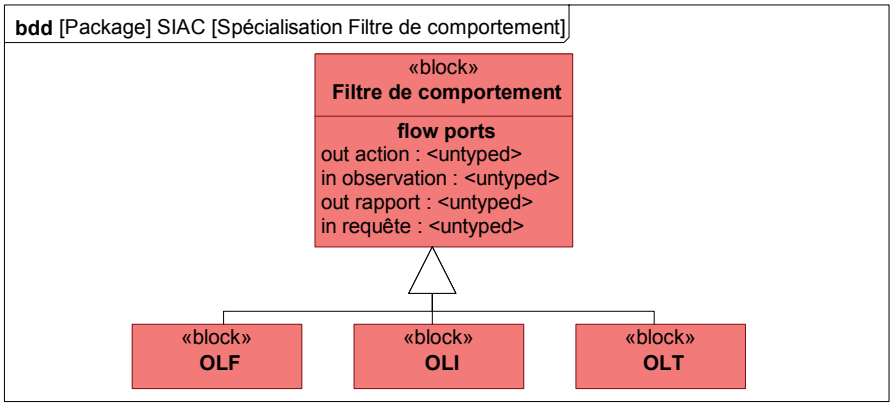


Figure 3.36 : Spécialisation des constituants sous la forme de Filtre de Comportement

Ces constituants (les blocs) sont ensuite rassemblés dans une architecture de constituants que l'on voit à la Figure 3.37 représentée sous la forme d'un IBD. Ceci nous permet de visualiser les ports d'entrées/sorties hérités de chaque bloc avec les connecteurs. En effet, ce diagramme reprend la partition des fonctions définies (Figure 3.34) et ajoute les flux échangés entre les fonctions appartenant à des constituants différents comme des flux échangés directement entre ces constituants.

Cette vision boîte blanche du SIAC nous permet de raffiner le scénario opérationnel défini à la Figure 3.29 pour spécifier les messages et opérations qui sont échangés par ces trois constituants. Ainsi, si on se limite à l'exécution d'une action de conduite en local, une évolution correcte doit passer par plusieurs étapes successives :

1. **L'affichage des informations concernant l'action à exécuter** (Figure 3.38)
   L'OLI affiche, parmi toutes les actions à réaliser en local, l'action à exécuter par le rondier. Ainsi, les informations utilisées pour la réalisation de l'action sont : l'identifiant de la vanne à manipuler, l'état dans lequel il faut mettre la vanne et sa localisation par rapport à la partie du processus à aligner.

2. **La demande d'autorisation** (Figure 3.39)
   L'OLI envoie à l'OLT une demande d'autorisation de l'action à réaliser par le rondier afin de s'assurer que toutes les conditions sont bien respectées et qu'aucune contrainte n'est violée.
   L'autorisation des actions à faire par le rondier est réalisée sous plusieurs conditions :

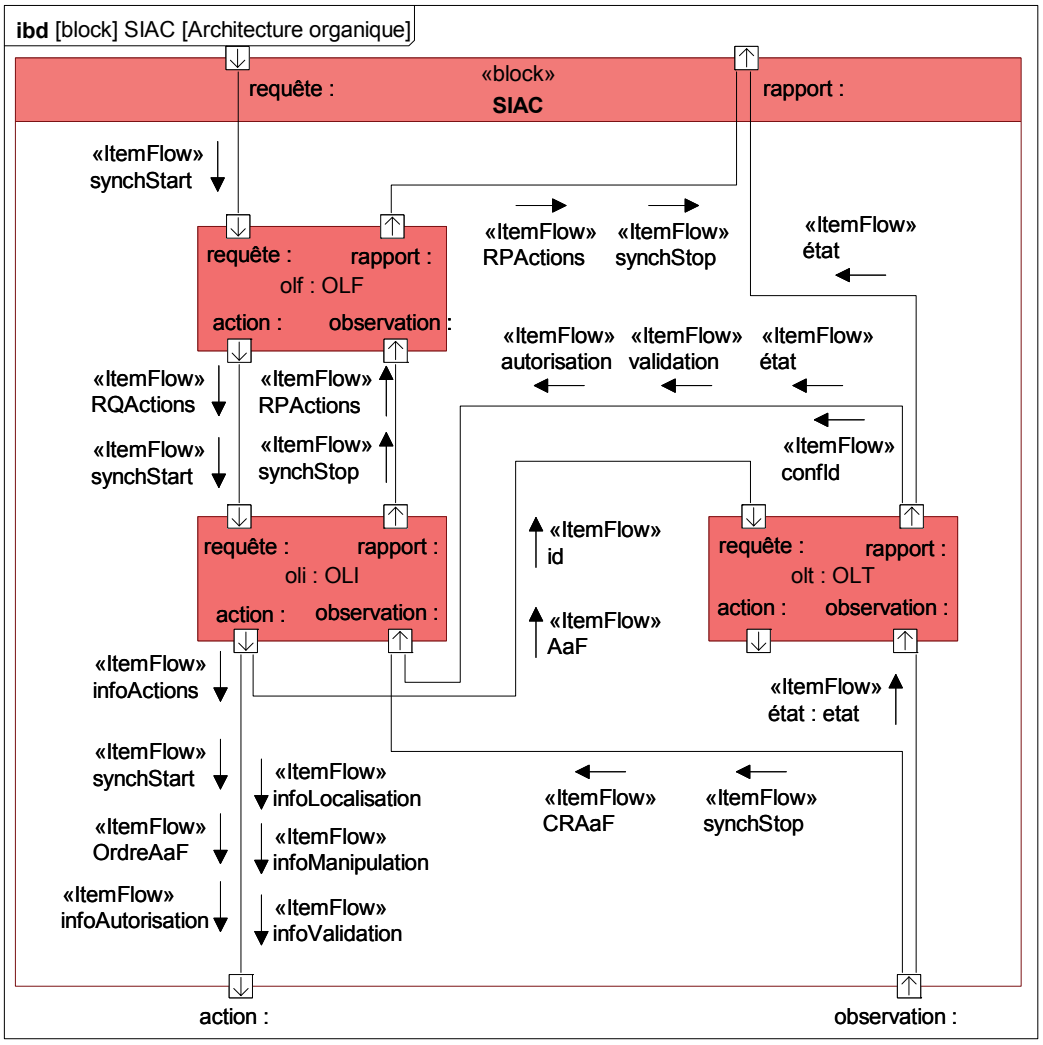


Figure 3.37 : Architecture organique du SAC

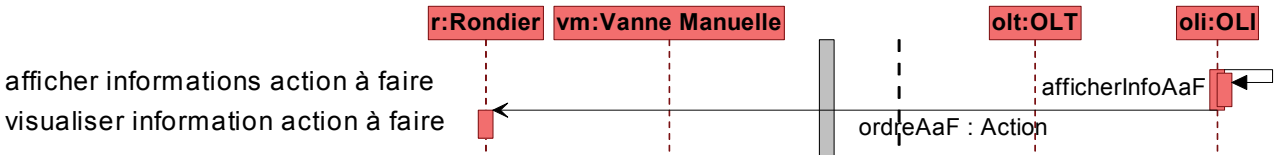


Figure 3.38 : L'affichage des informations concernant l'action à exécuter

- **Limitation des actions :** Autorisation de l'action par rapport aux actions possibles réalisables sur la vanne manuelle : toute autre action que « Ouvrir » et « Fermer » n'est pas autorisée ;
- **Contrôle de cohérence :** Autorisation par rapport à l'état courant de la vanne : une action d'ouverture n'a pas de sens sur une vanne déjà ouverte ;
- **Contrôle de dépendance :** Autorisation de l'action par rapport aux dépendances d'ordre d'actionnement des autres constituants présents dans le processus : une vanne doit être ouverte avant une autre, par exemple.

La réception de l'autorisation a pour effet l'affichage des informations au rondier.

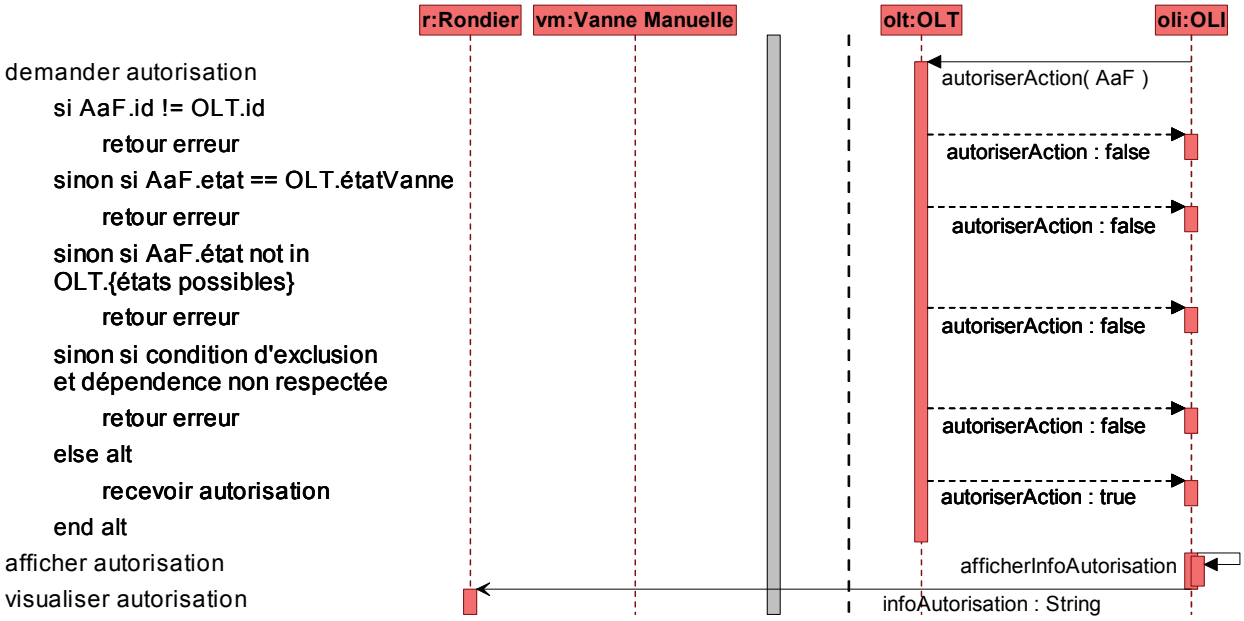


Figure 3.39 : La demande d'autorisation

3. **La demande de localisation** (Figure 3.40)

L'OLI demande au rondier de localiser la vanne en lui affichant des informations de position, voire le chemin pour arriver à la vanne concernée. Dès l'arrivée du rondier à la position indiquée il doit demander, par l'intermédiaire de l'OLI, la confirmation de la localisation à l'OLT. Celui-ci, en utilisant l'identifiant qui est envoyé par l'OLI, s'auto identifie et confirme ou pas la bonne localisation. C'est l'assistant ensuite qui confirme au rondier la validité de la vanne à manipuler en comparant son identifiant avec celui associé à l'action à réaliser.

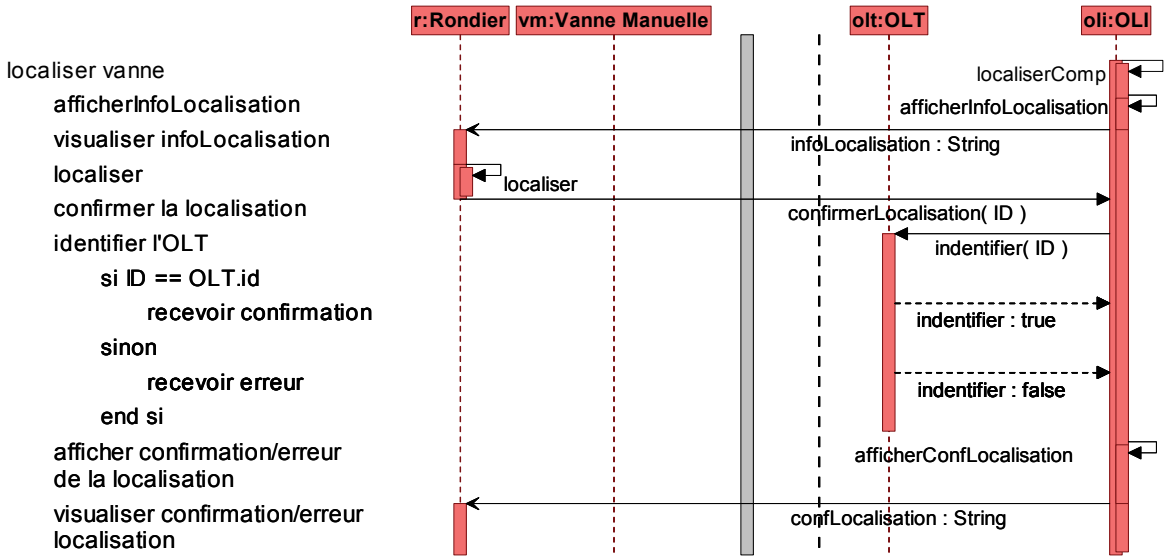


Figure 3.40 : La demande de localisation

4. **La demande de manipulation et la validation de l'action exécutée** (Figure 3.41)
   L'OLI demande au rondier de manœuvrer la vanne localisée et identifiée, par l'affichage des informations de manipulation, voire d'aide à la manipulation comme, par exemple, une animation montrant le sens de rotation de la poignée de la vanne manuelle. Dès réalisation de la manipulation, l'OLT perçoit le changement d'état de la vanne et, en comparant avec la valeur de l'état décrite dans l'action à réaliser, il peut envoyer le message de validation afin de la confirmer au rondier.

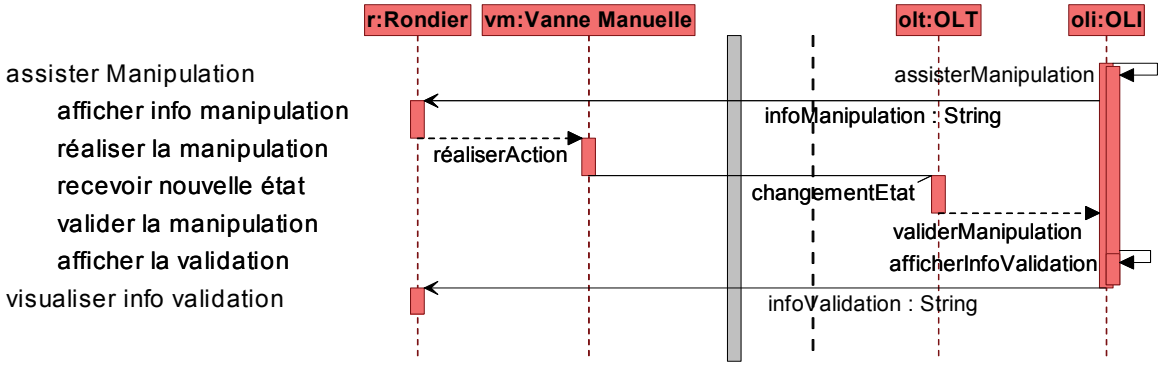


Figure 3.41 : La manipulation de la vanne

5. **Le compte rendu de la manipulation** (Figure 3.42)
   Dès confirmation de la bonne réalisation de l'action demandée, le rondier va devoir clore l'action courante en validant le CR créé automatiquement par l'OLI et en ajoutant, si nécessaire, des informations sur les observations faites pendant l'exécution de cette action.

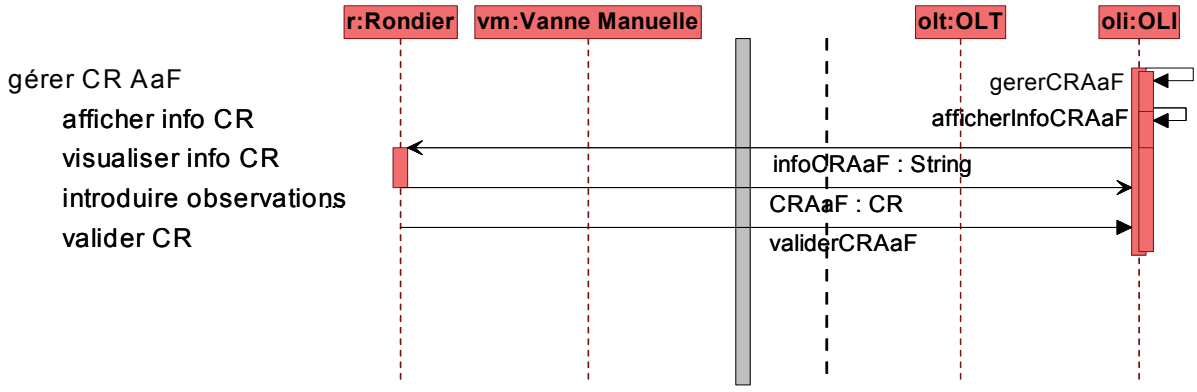


Figure 3.42 : Le compte rendu de la manipulation

Dans cette évolution nous pouvons observer que, dans la plupart des cas, l'OLI est à l'initiative de la demande des actions de la part du rondier et l'OLI se charge de confirmer/valider que l'action a bien été réalisée (localisation, manipulation).

Étant donnée la différence de nature entre le SIAC et les éléments de son environnement, il est préférable de définir les exigences d'interface. Par exemple, la Figure 3.43 montre deux exigences d'interface avec le rondier, une interface visuelle et une interface tactile, et une exigence pour la communication des actions sur le terrain.

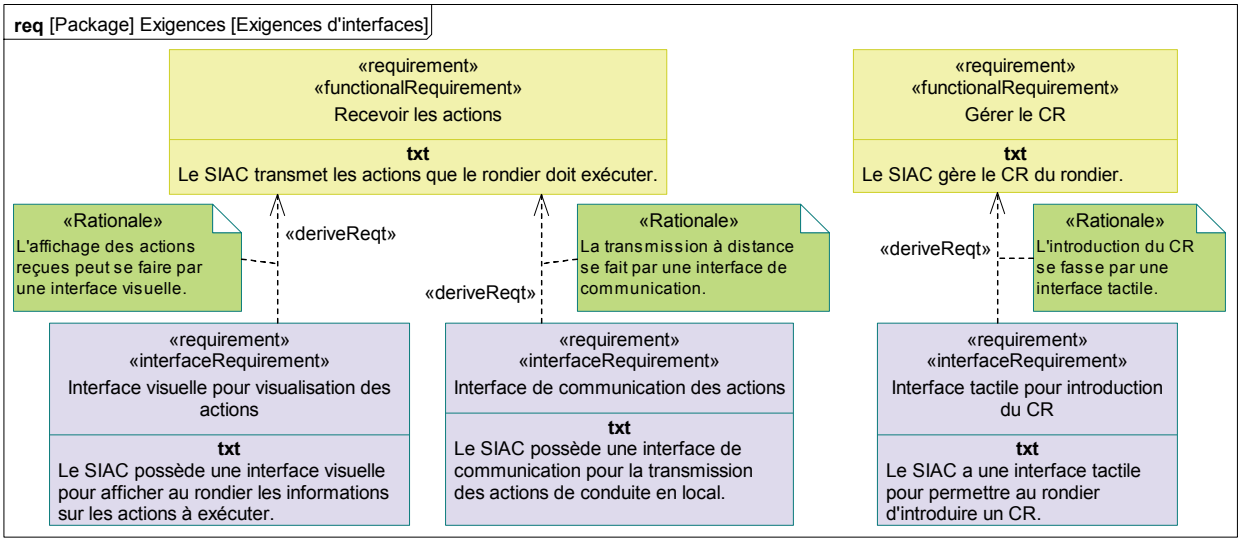


Figure 3.43 : Exemple d'exigences d'interface

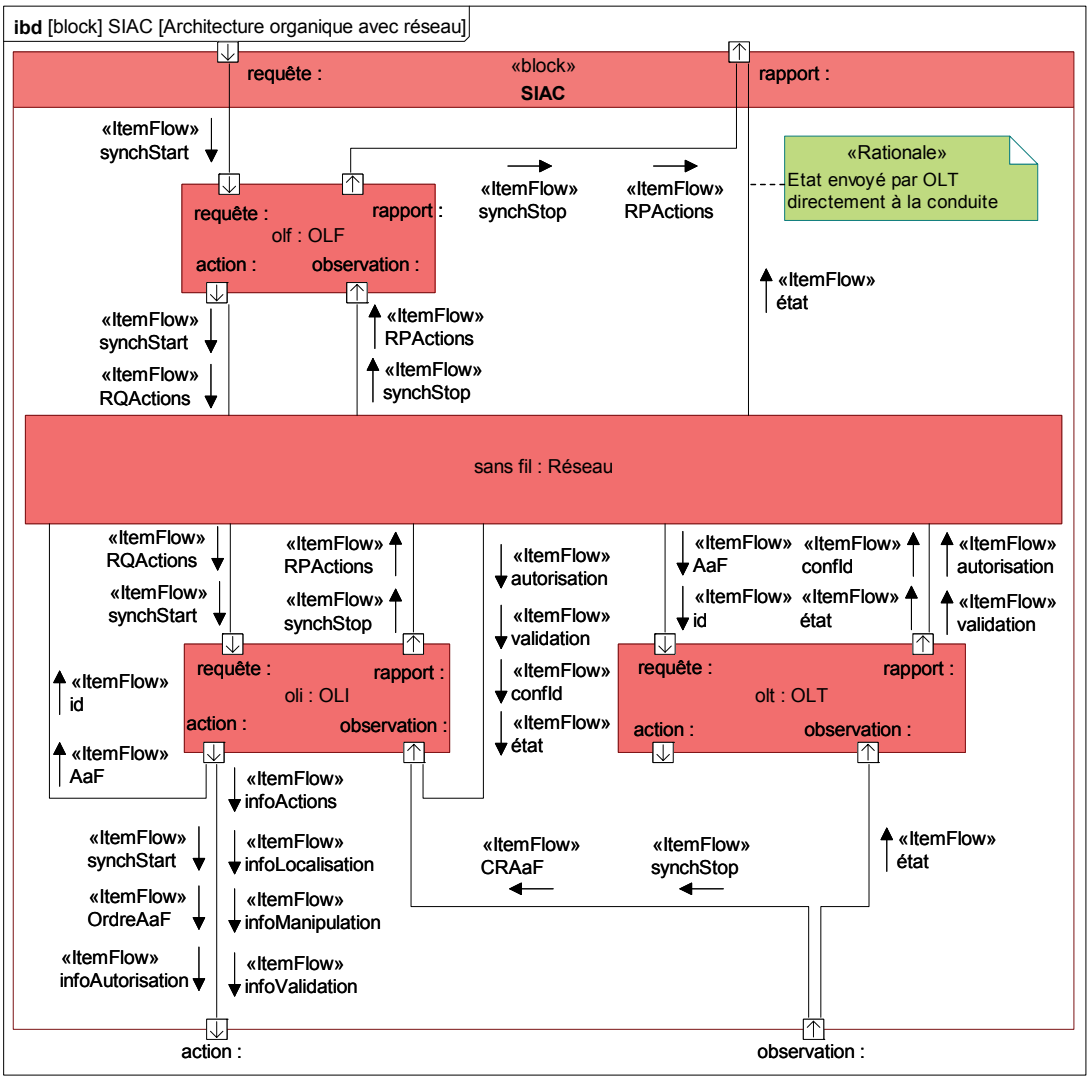


Figure 3.44 : Raffinement de l'architecture organique avec l'ajout du réseau

Ces exigences vont définir les interfaces effectives ou vont ajouter des nouveaux constituants avec le rôle d'interface. Ceci correspond à une deuxième phase d'ingénierie qui permet, dans le deuxième cas, de définir un bloc réseau permettant de transmettre les actions à exécuter qui peut être ajouté entre l'OLF et l'OLI du rondier (Figure 3.44).

Aussi, étant donné que la vanne manuelle sur laquelle le module OLT doit être intégré est imposée, il faut que ce module puisse s'intégrer avec celle-ci. C'est pour cela qu'il est nécessaire de réaliser des architectures d'intégration dans l'environnement d'exploitation, comme par exemple celle de la Figure 3.45.

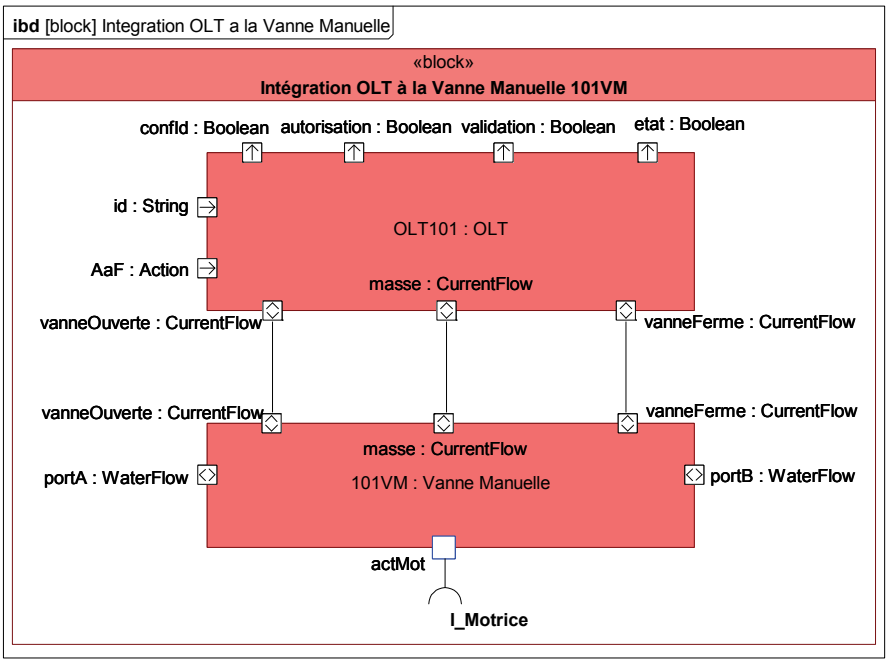


Figure 3.45 : Architecture d'intégration de l'OLT à la vanne manuelle

3.6 Exécution des modèles de spécification

L'objectif de l'exécution des modèles de spécification est de permettre la vérification des solutions proposées. Il est ainsi possible de corriger certaines erreurs de spécification système qui peuvent être retrouvées pendant l'exécution des modèles, et de mettre à jour les modèles de spécification du point de vue exigences, fonctions et constituants.

Nous utilisons deux types de vérifications des modèles de spécifications :

- par simulation de scénarios pour vérifier le respect des exigences ;
- par model checking pour vérifier automatiquement et de manière exhaustive le respect de propriétés sur l'ensemble des situations possibles.

3.6.1 Vérification par simulation de scénarios de test

Afin de vérifier l'ensemble des spécifications du système d'aide à la conduite, nous proposons de les exécuter en utilisant la simulation. Nous avons défini des cas de test (« testCase » en SysML) afin de modéliser les scénarios qui doivent être exécutés sur les modèles de spécification (Figure 3.46).

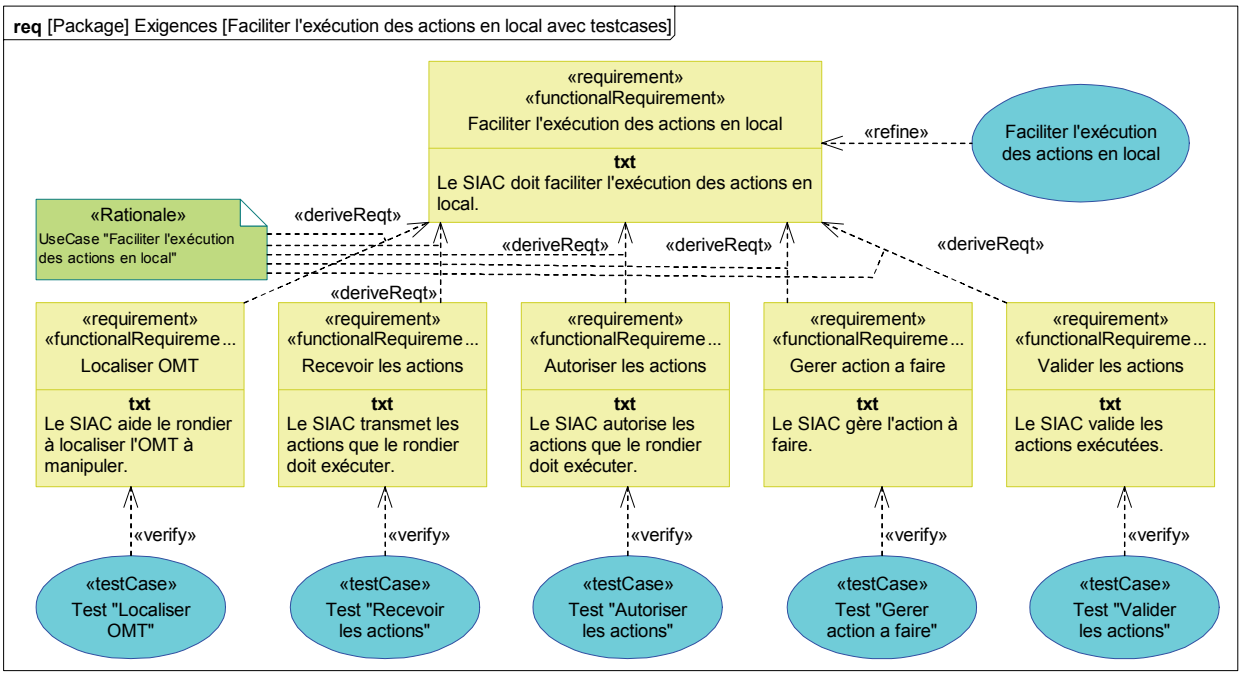


Figure 3.46 : L'ajout des cas de test pour la vérification des exigences fonctionnelles du SIAC

Nous proposons de décrire le scénario de test sous la forme d'un diagramme d'activités (Figure 3.47). Ceci nous permet de définir les actions qui doivent être exécutés et de définir les conditions de vérification du respect de l'exigence.

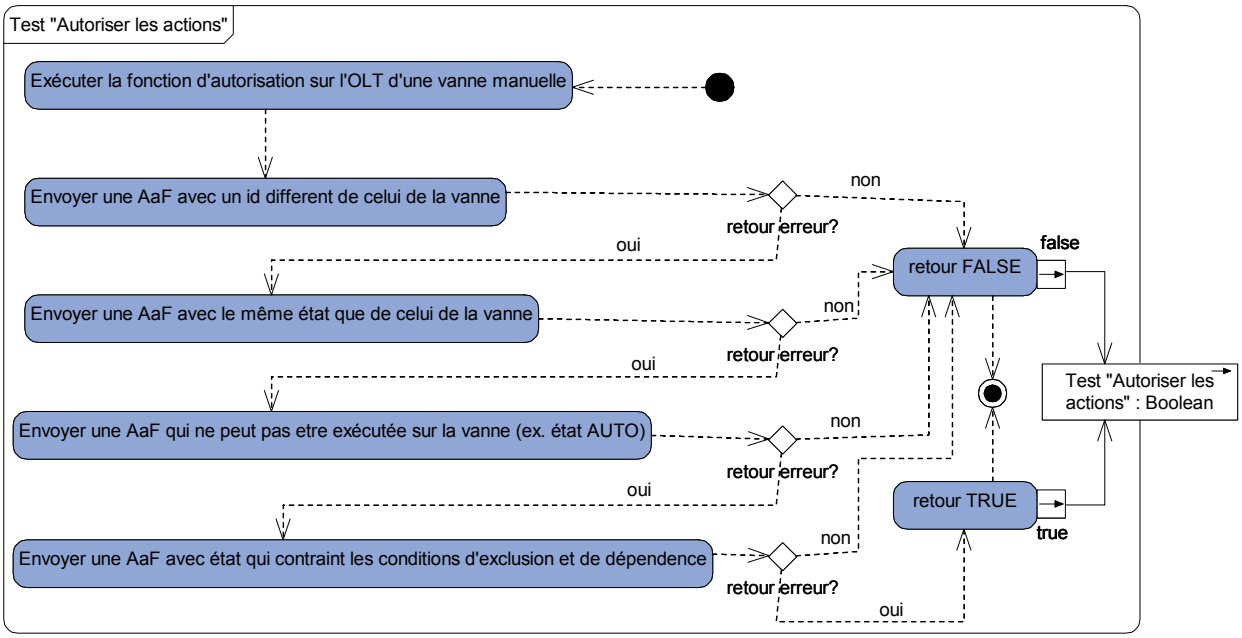


Figure 3.47 : Cas de test pour la vérification de la satisfaction de l'exigence fonctionnelle d'autorisation

Pour illustrer, nous testons la fonction d'autorisation sur l'OLT selon deux aspects : autorisation en fonction de l'identifiant de la vanne qui doit être manœuvrée et autorisation en fonction de l'action à exécuter. Dans ce deuxième cas nous vérifions que l'OLT envoie bien des messages d'erreur quand l'état de la vanne est le même que celui demandé par l'action, quand l'action ne peut pas être exécutée sur la vanne ou quand l'action est contrainte par une condition d'exclusion ou de dépendance.

Afin de mettre en œuvre ces cas de test, il est nécessaire d'avoir un modèle de l'installation CISPI sur lequel la spécification du SIAC peut être exécutée. Nous avons choisi d'utiliser pour cela le langage Modelica, et de simuler avec l'outil Dymola (2010). La bibliothèque utilisée pour la réalisation du modèle est Modelica Fluid (Casella, et al., 2006), ce qui nous permet d'intégrer dans le modèle la physique du procédé (Figure 3.48).

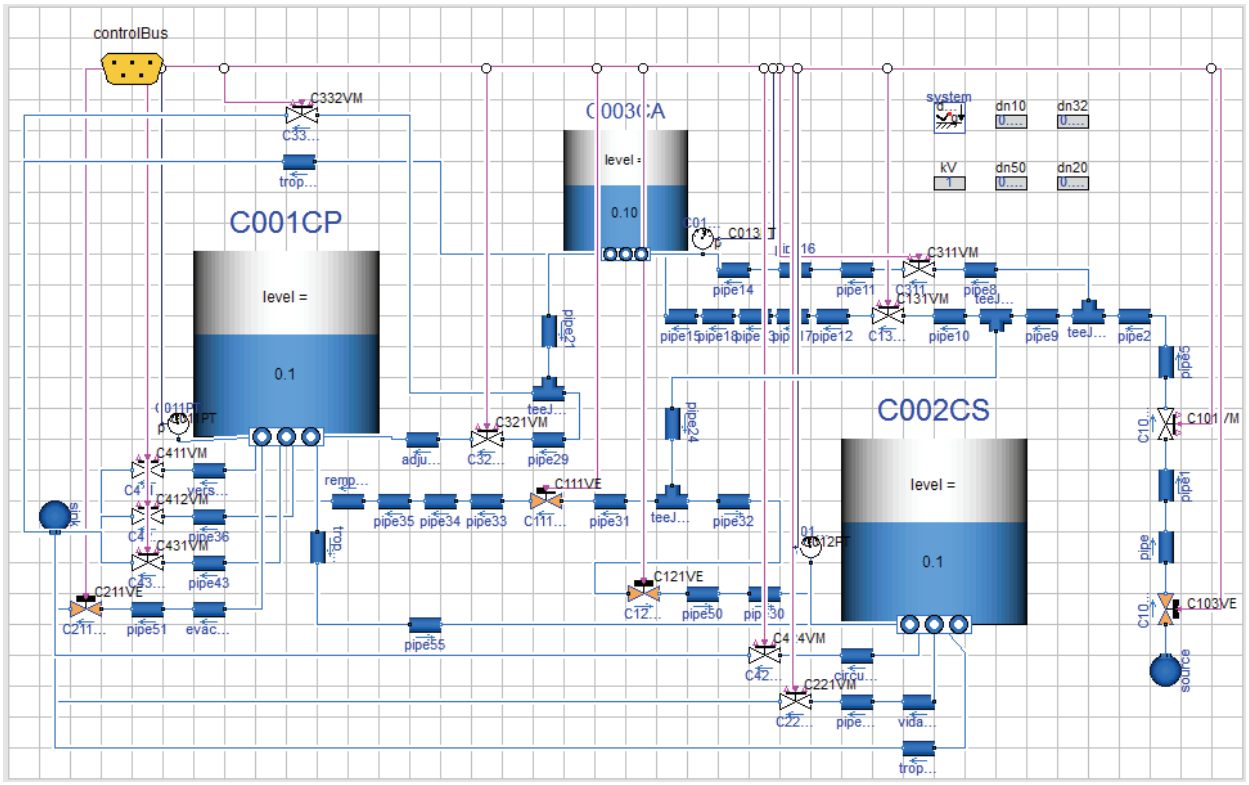


Figure 3.48 : Modèle Modelica de la partie opérative du CISPI pour les fonctions de remplissages

Avant d'utiliser ce modèle couplé aux spécifications du SIAC, nous simulons son comportement afin de nous assurer de la correspondance entre le fonctionnement du modèle et celui de la plateforme réelle. Pour cela nous avons réalisé un modèle avec lequel nous pouvons interagir dynamiquement par le biais d'une interface graphique en temps réel (Figure 3.49).

La spécification du SIAC telle que présentée dans ce chapitre est sous la forme de modèles SysML. Il est donc nécessaire de transformer ces modèles de spécification dans le langage Modelica, et plus particulièrement ceux des constituants logiques. Même si des travaux sur des transformations automatiques existent comme nous avons vu dans la section §2.3, nous avons opté dans une première approche pour une transformation manuelle des modèles SysML vers Modelica. Le principal problème de ce type de transformation manuelle reste la synchronisa-

tion entre les différentes versions des deux modèles, SysML et Modelica, où la transformation automatique s'avère plus pertinente.

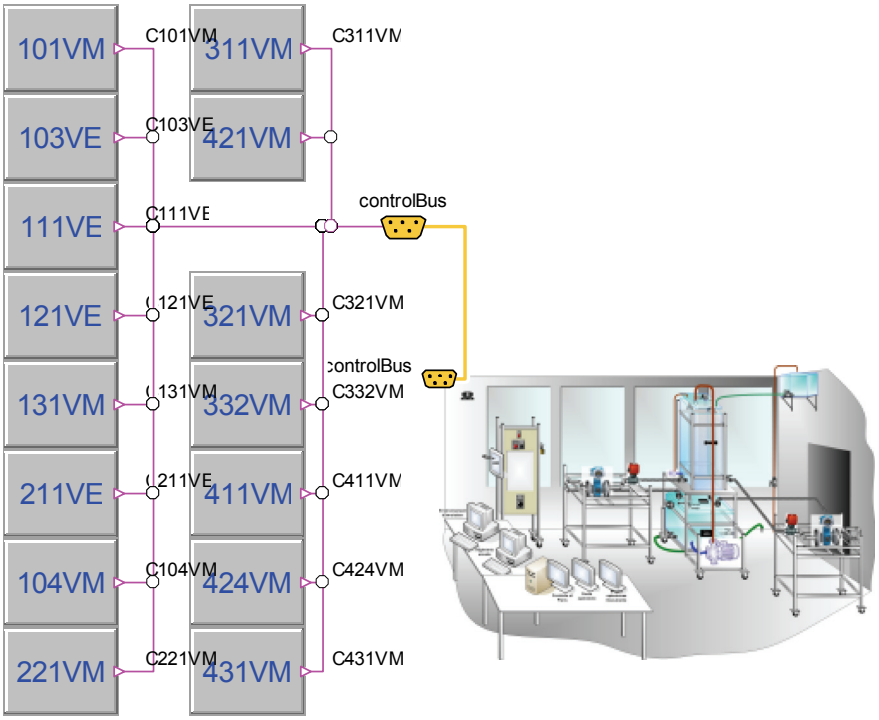


Figure 3.49 : Modèle de simulation « joueur » de la plateforme CISPI

Dans un premier temps nous avons fait une transformation du point de vue structurel. Afin d'illustrer le principe et dans un souci de lisibilité, nous nous limitons ici au modèle d'une vanne couplée à un module OLT. Ainsi, nous disposons dans le diagramme de bloc interne de la Figure 3.45 d'un bloc pour la vanne manuelle et un bloc pour l'OLT. Chaque bloc dispose des ports d'entrée/sortie. Les deux blocs sont liés entre eux avec des connecteurs qui permettent d'échanger l'état de la vanne. Le bloc de la vanne intègre aussi une interface pour permettre son actionnement en local.

Les deux blocs SysML correspondent en Modelica à deux modèles, C101VM correspondant à une vanne discrète que nous avons modifiée pour ajouter les capteurs de fin course, et OLT101 que nous avons modélisé en correspondance avec les spécifications SysML. Les ports des blocs SysML correspondent en Modelica à des ports. Nous avons typés ces derniers en fonction des types SysML avec des types similaires ou les plus proches possibles si des types similaires n'existent pas (*ex.* du fait que Modelica ne travaille pas avec des variables de type string, l'identifiant qui est envoyé à l'OLT est codé dans un type entier). Les connecteurs SysML correspondent à des connecteurs Modelica qui permettent de relier les ports des deux blocs. De plus, nous avons ajouté la possibilité d'interagir avec le modèle Modelica en utilisant des éléments définis dans la bibliothèque UserInteraction de Dymola. Le résultat de la transformation est disponible à la Figure 3.50.

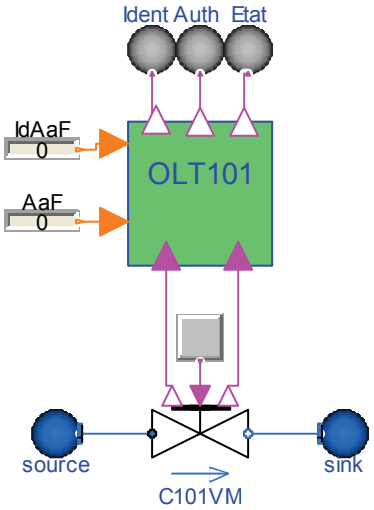


Figure 3.50 : Modèle exécutable issu de la conception organique

Étant au niveau de spécification du SIAC et non de ses constituants, le comportement simulé de l'OLT correspond aux règles générales de définition (venant des exigences de type scénarios définis par la Figure 3.39) et non au comportement détaillé ultérieurement dans le Chapitre 4. Ainsi, le scénario définit le comportement les sorties du bloc OLT en fonction de ses entrées (vue boite noire). Nous proposons de représenter ces règles en utilisant des blocs logiques. Cela nous donne le comportement Modelica définit à la Figure 3.51.

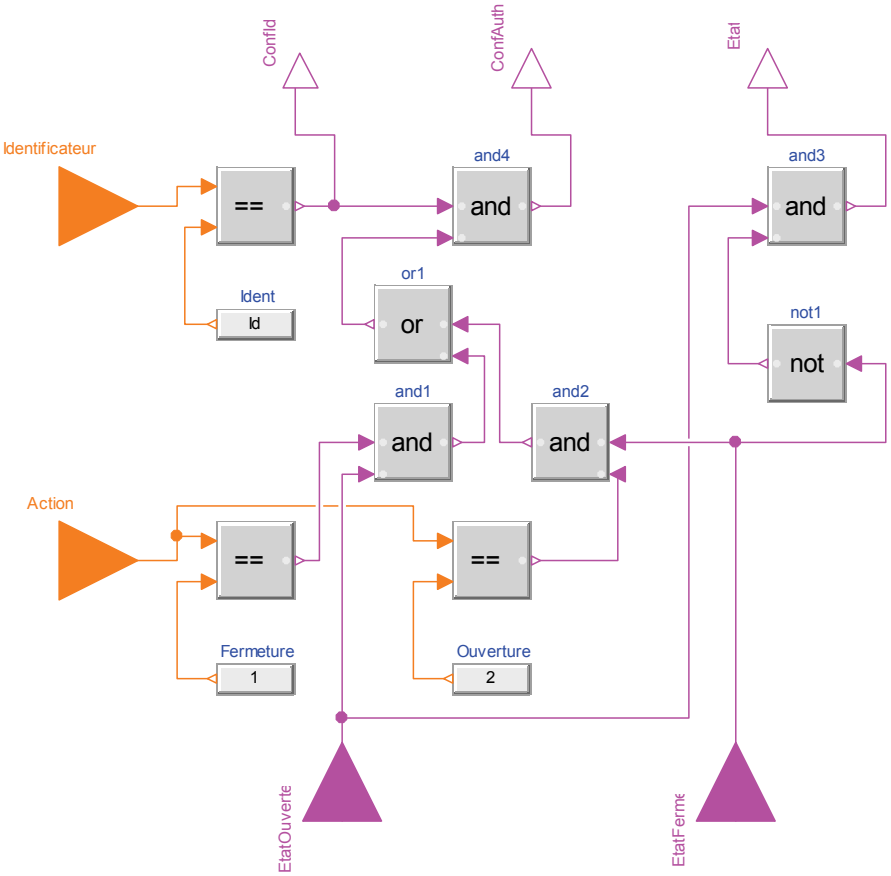


Figure 3.51 : Modèle exécutable de la spécification comportementale de l'OLT (issue des règles)

Ces différents modèles sont exécutés conformément au scénario décrit par le cas de test « autoriser » (Figure 3.47), permettant ainsi la vérification de la spécification SysML correspondante. Ainsi, la Figure 3.52 nous montre entre autres que :

- lorsqu'une requête de fermeture est présente (actionner101VM.Value à 1) et que la vanne est déjà fermée (etat101VM.y à 0) alors l'action n'est pas autorisée (auth101VM.y à 0) ;
- lorsqu'une requête d'ouverture est présente (actionner101VM.Value à 2) et que la vanne est fermée alors l'action est autorisée (auth101VM.y à 1) ;
- avec l'exécution de l'action (etat101VM.y à 1), tant que la requête d'ouverture est présente, l'action n'est plus autorisée (auth101VM.y à 0).

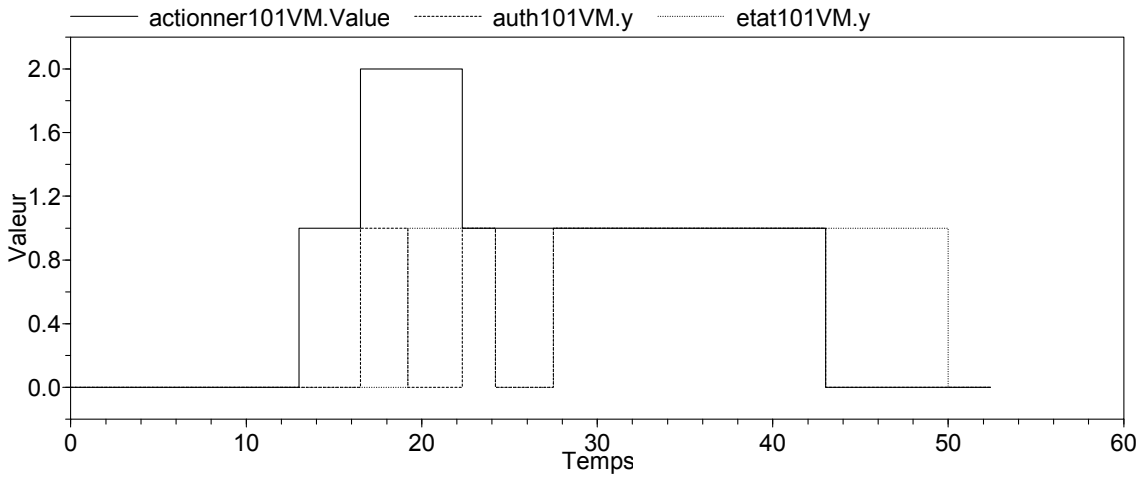


Figure 3.52 : Résultat de la simulation du couplage OLT-vanne

Les résultats de l'exécution de la simulation permettent de vérifier d'une part la spécification ainsi définie avant son envoi au niveau du bloc système de l'OLT et d'autre part le comportement désiré de l'OLT qui va devoir être validé à la réception de l'OLT réalisé.

3.6.2 Vérification des conditions d'exclusion et d'ordre d'actionnement

Dans le contexte de la conduite en local par des opérateurs et des rondiers, certaines de ces actions exécutées doivent être interdites afin de ne pas générer de problèmes de conduite et de préserver l'ensemble du processus. Ces interdictions sont basées sur le savoir-faire métier d'experts du processus, de sa conduite et de sa sécurité. Ainsi, des contraintes sur l'exécution de ces actions sont utilisées « en ligne » par les OLT (Figure 3.39) afin d'empêcher des situations interdites, principalement liées aux dépendances d'exclusions et d'ordre d'actionnement. Ces contraintes doivent être vérifiées par l'OLT d'une part pendant l'autorisation des actions à exécuter, et d'autre part pendant la validation de l'action exécutée. Voici deux exemples de dépendances :

- « L'électrovanne 103VE ne doit pas être ouverte si la vanne manuelle 101VM n'est pas ouverte » : en d'autres termes, cela signifie que l'opérateur ne doit pas ouvrir la vanne 103VE lorsque la vanne 101VM est fermée et que le rondier ne doit pas fermer la vanne 101VM lorsque 103VE est ouverte.
- « L'électrovanne 111VE ne doit pas être ouverte en même temps que l'électrovanne 121VE » : il y a différentes conditions à respecter, par exemple, l'opérateur ne doit pas ouvrir la vanne 111VE lorsque 121VE est ouverte, l'opérateur ne doit pas enclencher le mode automatique sur la vanne 111VE lorsque 121VE est ouverte.

Ces dépendances, liées à des besoins utilisateurs, sont des propriétés qui doivent être garanties par le système. Pour cela, nous proposons d'insérer à la spécification du SIAC des contraintes relatives à ces propriétés, et qui seront implantées dans les OLT.

Pour qu'une utilisation « en ligne » de ces contraintes permette d'éviter toutes les situations interdites, il est nécessaire de procéder « hors ligne » avant leur implantation à la vérification de leur complétude. Nous proposons de vérifier la complétude des contraintes en utilisant le model checking, afin de tenir compte de manière exhaustive de l'ensemble des situations possibles. En nous basant sur les travaux de Marangé (2008), nous utilisons les automates temporisés et l'outil de model checking UPPAAL.

Pour cela nous modélisons dans un premier temps l'environnement d'exécution (Figure 3.53a). Nous proposons que son comportement soit du type observation/contrôle/actionnement correspondant à une boucle cybernétique (Figure 3.5) :

- L'observation est réalisée par la lecture des états capteurs ;
- Le contrôle est réalisé par les choix d'actions possibles ;
- L'actionnement est réalisé par la vérification des contraintes et l'écriture des ordres à destination des actionneurs.

Au niveau opérationnel nous ajoutons des modèles représentant les actionneurs. Pour représenter le fait que le système est dans un état lors du démarrage, une première évolution est exécutée automatiquement pour initialiser les modèles des actionneurs. Les exemples ci-dessous montrent le modèle de comportement discret d'une vanne manuelle avec des états ouverte/fermée (Figure 3.53b), et le modèle d'un choix d'actionnement d'une telle vanne avec les actions d'ouverture ou de fermeture (Figure 3.53d).

Les contraintes sont écrites sous forme de conditions logiques. Par exemple :

- « L'opérateur ne doit pas ouvrir la vanne 103VE lorsque la vanne 101VM est fermée » : (I102VE.aO==1) AND (S101VM.cF==1) == 0
- « L'opérateur ne doit pas ouvrir la vanne 111VE lorsque 121VE est ouverte » : (I111VE.aO==1) AND (s121VE.cO==1) == 0

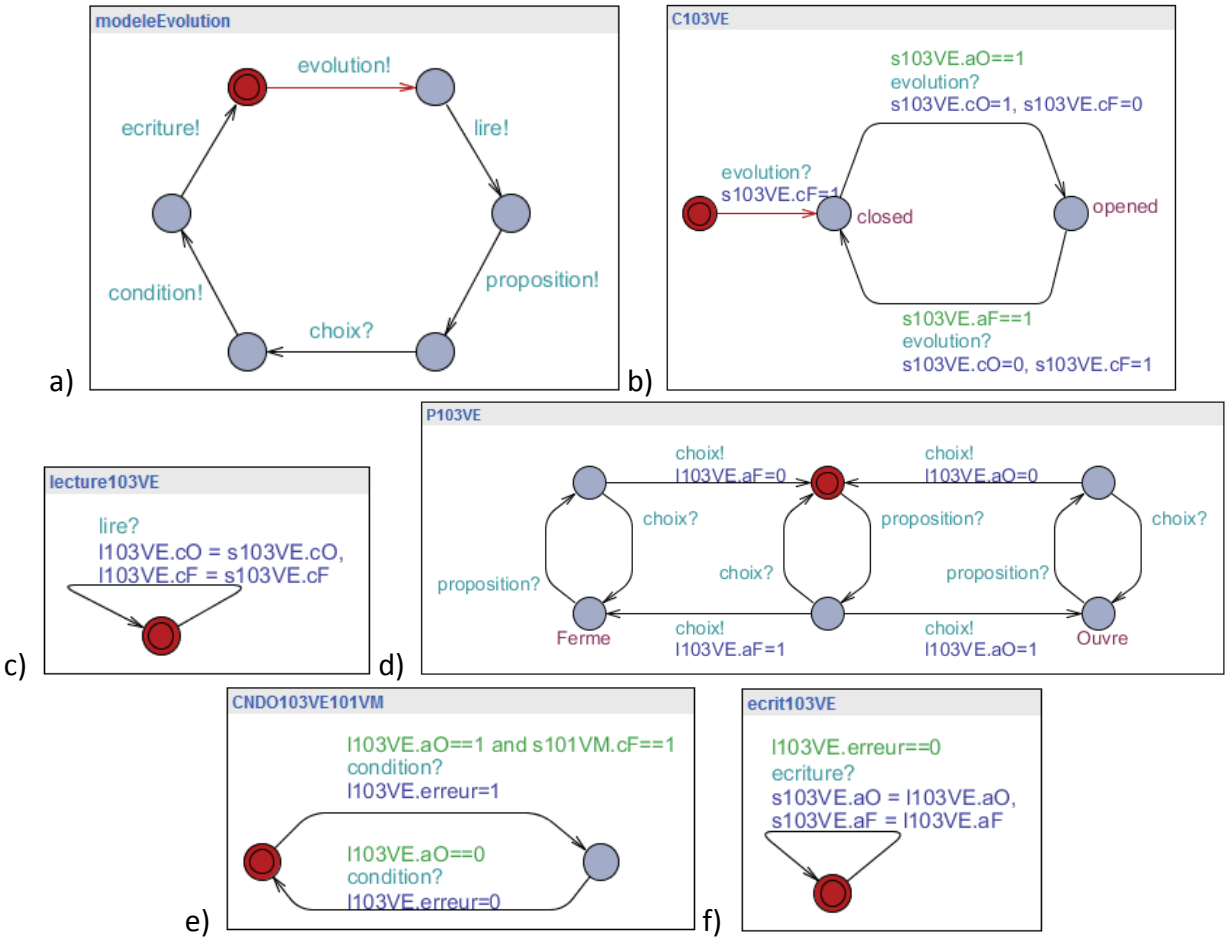


Figure 3.53 : Modèles UPPAAL pour la vérification des conditions : a) modèle de l'environnement d'exécution, b) évolution de l'état pour une vanne à deux états, c) lecture de l'ancienne valeur, d) proposition d'action pour une vanne à deux états, e) vérification des conditions, f) écriture de la nouvelle valeur

De la même manière, des règles, relatives aux propriétés à respecter (exclusion, précédence), et que l'outil de model-checking va utiliser, sont décrites par des prédicats ; par exemple :

- « L'électrovanne 103VE ne doit pas être ouverte si la vanne manuelle 101VM n'est pas ouverte» : A[] not (C103VE.opened and C101VM.closed)
- « L'électrovanne 111VE ne doit pas être ouverte en même temps que l'électrovanne 121VE » : A[] not (C111VE.opened and C121VE.opened)

Le model checking consiste dans notre cas à prendre une situation courante des capteurs et actionneurs, à la mémoriser, à envisager un choix d'action, et à vérifier que l'exécution de cette action, tout en respectant les contraintes, respecte bien les règles d'exclusion et/ou de précédence. Dans l'hypothèse où lors de l'évolution des modèles ces contraintes ne sont pas respectées, la variable « erreur » de l'actionneur correspondant est mise à 1. Dans le cas contraire, la proposition d'action est transmise sous forme d'ordre à l'actionneur dont le modèle va évoluer. En parallèle, l'outil de model checking teste en permanence le non blocage des différents modèles et le respect des règles décrites sous forme de prédicats. Si l'une de ces règles vient à ne

pas être respectée, l'outil arrête le model checking et fournit la trace de l'évolution qui en est la cause, facilitant ainsi la mise à jour des contraintes à respecter.

De manière synthétique, ce model checking passe cycliquement par les étapes suivantes :

1. Mémorisation de la situation des capteurs (Figure 3.53c) ;
2. Choix d'une action à exécuter sur l'un des constituants (Figure 3.53d) ;
3. Vérification du respect des contraintes d'exclusion et de dépendance (Figure 3.53e) ;
4. Si les contraintes sont respectées (pas d'erreur), transmission du choix sous forme d'ordre aux actionneurs (Figure 3.53f) ;
5. Evolution des modèles d'actionneurs (Figure 3.53b).

La vérification par model checking des dépendances d'exclusion et d'ordre d'actionnement nous permet de retrouver des contraintes que le système doit garantir. Ces contraintes peuvent être vues comme des exigences que le SIAC doit satisfaire et donc qui doivent être vérifiés « en ligne » pendant l'exécution des actions de conduite.

3.7 Conclusions

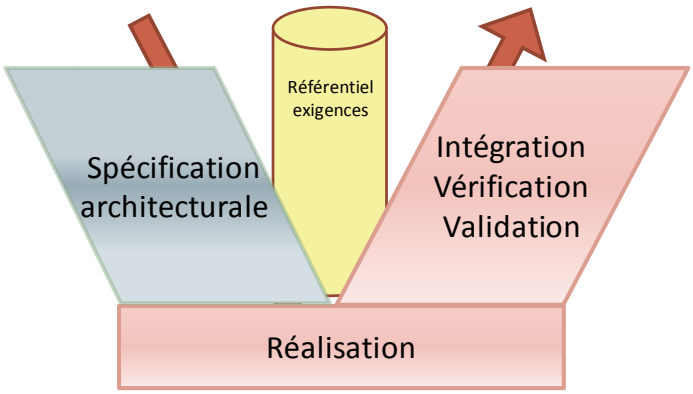
Dans ce chapitre nous avons spécifié un Système Interactif d'Aide à la Conduite qui apporte des solutions à certains problèmes de conduite des grands systèmes à risques. Pour cela nous avons appliqué la méthode proposée dans le premier chapitre et outillée avec SysML dans le deuxième chapitre.

Le bloc système SIAC permet, à partir des besoins des parties prenantes, de prescrire les exigences que le SIAC doit satisfaire. Ces exigences sont ensuite raffinées dans des cas d'utilisation explicités avec des scénarios opérationnels. Ainsi, des fonctions sont recensées en vue de concevoir l'architecture fonctionnelle du SIAC. Ces fonctions architecturées sont ensuite partitionnées dans trois types de constituants, les Objets Logiques Techniques, les Objets Logiques de Flux et les Objets Logiques Interactifs, composant le SIAC. Ces réflexions sur les OL nous amènent à envisager la possibilité de généraliser le patron de conception d'OL à d'autres systèmes sociotechniques faisant interagir l'humain et le système.

Cette passe d'ingénierie du SIAC nous amène à la réalisation les constituants architecturés, ce qui fait l'objet du chapitre suivant.

Si les phases remontantes du cycle en V d'intégration, de vérification et de validation du SIAC doivent être réalisées dans le même bloc système que sa spécification, et ce à réception des constituants réalisés, elles sont présentées, à des fins de structuration du mémoire, également dans le chapitre suivant.

Chapitre 4
Réalisation et Intégration du Système Interactif d'Aide à la Conduite



Ce chapitre est consacré à la réalisation de trois constituants du Système Interactif d'Aide à la Conduite (SIAC) et à leur intégration, vérification et validation dans le cadre de la plateforme expérimentale CISPI.

Au cours du Chapitre 3 nous avons spécifié le SIAC. A partir de cette spécification logique et en respectant le cycle de développement du système par bloc système, trois nouveaux domaines sont instanciés pour réaliser ces constituants du SIAC. Cette phase de réalisation se situe en dehors de l'Ingénierie Système (IS) et doit permettre de caractériser et de choisir les solutions technologiques afin de satisfaire les exigences système et les fonctionnalités des composants logiques. A ce stade, une analyse est nécessaire afin de permettre un choix parmi les technologies ambiantes permettant d'intégrer des comportements et fonctions « intelligentes » dans les équipements de terrain (OLT), et parmi les technologies d'assistance opérateur, permettant au rondier d'interopérer avec les composants (OLI). D'autre part, il nous faut définir et réaliser un système de gestion des modes opératoires permettant de gérer l'ensemble des procédures de conduite, et de faciliter le transfert des documents de conduite entre la salle de commande et le terrain (OLF).

De manière complémentaire, ces constituants réalisés doivent être ensuite intégrés, vérifiés et validés dès leur réception par le domaine métier du bloc système qui a spécifié le SIAC.

4.1 Architecture initiale de la plateforme CISPI

Au cours de nos travaux de recherche, nous avons proposé une plateforme expérimentale qui reproduit à l'échelle réduite des fonctionnalités que nous pouvons trouver dans des systèmes de production. Bien que nous ayons participé activement à la conception et à la réalisation de la plateforme CISPI, nous l'avons considérée dans la présentation de nos travaux comme une architecture initiale et existante, afin d'aborder la spécification architecturale du SIAC comme un système contributeur avec un ensemble de contraintes imposées par la physique d'un procédé existant. Le plan PID de la plateforme est présenté dans la Figure 3.4.

La plateforme CISPI est composée de trois stations : une station d'élaboration du produit et deux stations de régulation de débit. Nous disposons de plus d'une station de commande constituée d'une armoire d'automatisme. Nous disposons d'une part d'actionneurs manuels ou instrumentés, de capteurs et d'automates qui gèrent les capteurs et les actionneurs instrumentés, et d'autre part de postes pour la conduite et la supervision de la plateforme.

Représentée selon une vue d'intégration des constituants (Figure 4.18), la plateforme CISPI comporte un ensemble d'éléments technologiques qui sont détaillés dans le Tableau 4.1.

Tableau 4.1 : Equipements de la plateforme CISPI

Type	Matériel	No.	Composants dans système
Actionneurs	Vannes TOR manuelles	17	101VM, 104VM, 131VM, 213VM, 221VM, 223VM, 311VM, 321VM, 332VM, 411VM, 413VM, 421VM, 424VM, 426VM, 428VM, 431VM, 441VM
	Vannes linéaires de régulation	2	422VR, 432VR
	Electrovannes	5	103VE, 111VE, 121VE, 211VE, 222VE
	Vanne trois voies	1	212VV
	Clapets anti-retour	2	210VA, 420VA
	Pompe de régulation	1	425PP
	Pompe de brassage	1	412PS
	Cuve principale d'élaboration	1	001CP
	Cuve de stockage	1	002CS
	Cuve adjuvant	1	003CA
Capteurs	Capteurs de débit (ANA)	4	200FT, 423FT, 427FT, 433FT
	Capteurs de niveau (ANA)	3	001PT, 002PT, 003PT
	Capteurs de pression (TOR)	1	429PT
	Détecteur d'inondation	1	001DI
Automates	Automate S7 315F-2 PN/DP	1	
	Boitier d'entrées/sorties déportées ET200S	2	

	Boitier d'entrées/sorties déportées ET200S-F	1	
	Client Wifi Scalance S W744	3	
	Point accès Wifi W788	1	
Conduite et Supervision	Ecran tactile SIEMENS MP377		
	Poste conduite 3 écrans 20" et un écran TV 32"		
	Poste système d'information avec MSSQL Server 2008		
	Poste serveur OPC		

Une attention spéciale doit être portée sur les automates de sécurité équipés de modules PROFISafe, du fait que leur programmation spécifique, axée sécurité, est différente d'un automate classique. Le Tableau 4.2 nous montre une comparaison des variables utilisées dans un programme automate standard et dans un programme automate de sécurité.

Tableau 4.2 : Accessibilité des variables par programmes automate standard et de sécurité

	DB	EN	SN	DBF	EF	SF	Mémentos
Programme standard	R/W	R	W	R	R	X	R/W
Programme de sécurité	X	R	X	R/W	R	W	R
DB : Bloc de données, EN : Entrée Normale, SN : Sortie Normale, DBF : Bloc de données de sécurité, EF : Entrée d'un module PROFISafe, SF : Sortie d'un module PROFISafe, Mémentos : variable d'accès globale							

Un module d'Entrées/Sorties PROFISafe apporte de nombreux avantages pour la sécurité de commande des installations, mais aussi des contraintes inhérentes aux systèmes de sécurité. Il permet une série de contrôles programmables comme la redondance, l'anti-valance, le test de court-circuit et d'autres fonctions configurables dans les options de chaque module. Lors de la compilation du projet programme automate, lorsque le mode de sécurité est activé, une série de Blocs de Données (DBF) est créée pour représenter l'état de chaque module (défaut global, réarmement, diverses configurations et défaut voie par voie). Il est donc important de bien nommer chaque module PROFISafe afin de retrouver le nom du DBF le représentant.

Cependant, tous les modules de sécurité devraient uniquement rester accessibles par le programme de sécurité (groupe d'exécution F). Selon les définitions du constructeur SIEMENS sur les organes de sécurité, seul un programme de sécurité peut "écrire" dans les Blocs de Données F (DBF) et dans les sorties des modules. A cela s'ajoute la contrainte que le programme de sécurité ne peut accéder qu'aux données présentes dans un DBF, E/S de module PROFISafe et les mémentos (variables configurables depuis la table des mnémoniques).

Une dernière contrainte est liée à l'intégrité des variables qui sont vérifiées dès le début du lancement du cycle du programme de sécurité (groupe d'exécution F). Ainsi, une image totale est faite sur les E/S et les mémoires accessibles, puis le programme est lancé. Si durant ce cycle il est constaté qu'une des variables a changé autrement que par le programme de sécurité, une interruption est levée avec une erreur. Pour cette raison, il est judicieux et conseillé de ne créer qu'un seul et unique programme de sécurité (il est effectivement possible de créer plusieurs groupes d'exécution F) afin de prévenir le recouvrement de l'exécution d'un cycle par un autre.

Dans cet ordre d'idées, lors des travaux menés sur la plateforme (Prioul, 2009), nous avons défini un principe général décrit dans la Figure 4.1 pour la structuration des programmes implantés dans les automates de sécurité.

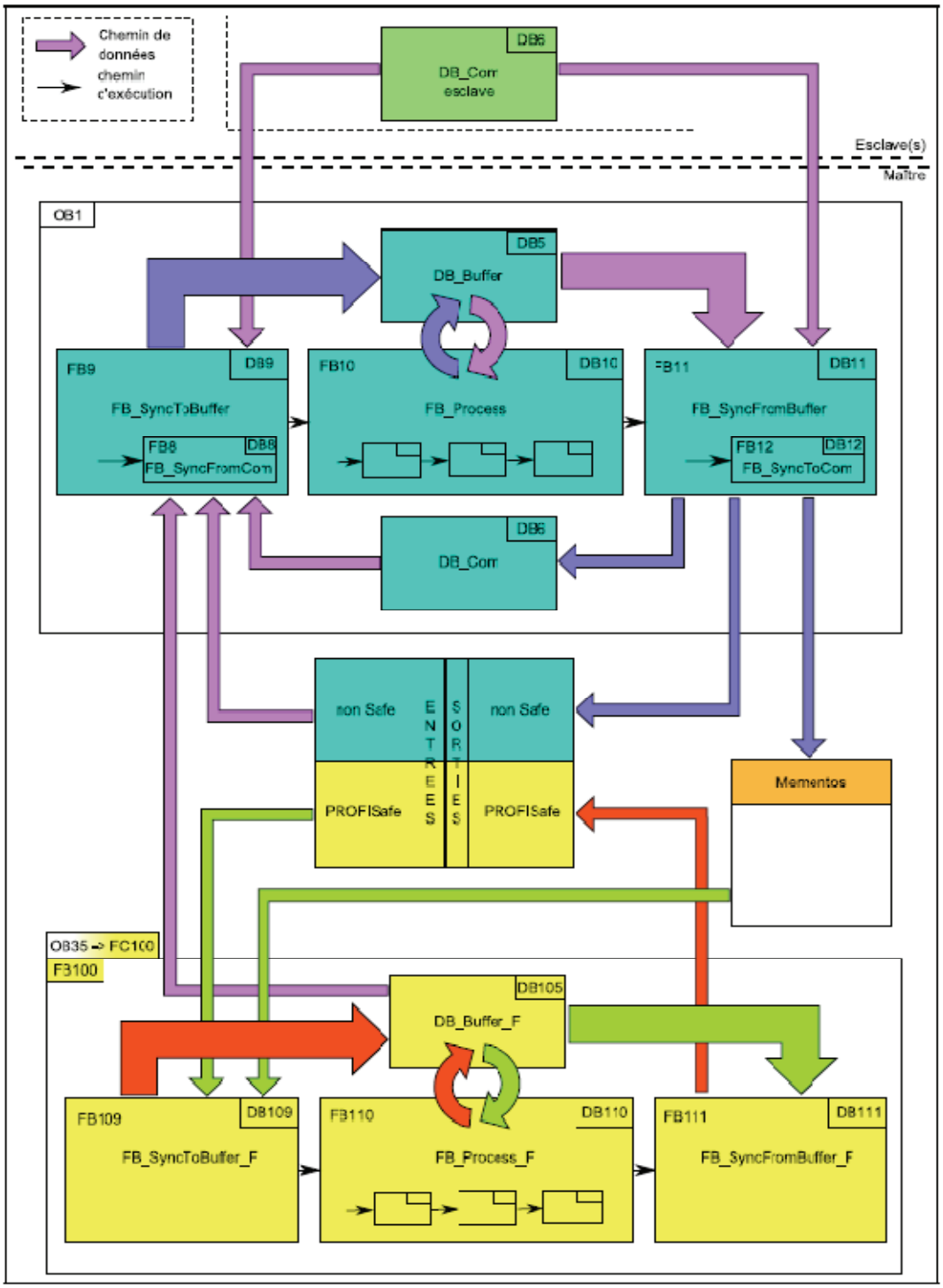


Figure 4.1 : Principe général de la structure de référence des programmes implantés dans l'automate de sécurité

Trois principes sont réunis à l'intérieur de ce schéma : l'utilisation de tampons de travail pour la synchronisation des entrées (FB9 et FB109) et sorties (FB111 et FB11) du programme de sécuri-

té et classique, l'utilisation d'un système de communication pour synchroniser les données du maître avec l'esclave (DB8) et les adaptations aux contraintes PROFISafe.

Nous avons choisi, de même, d'appliquer le principe des Filtres de Comportement aux modules de contrôle/commande des constituants instrumentés pour structurer l'architecture de commande de la plateforme. Ces modules sont initialement définis et structurés dans ControlBuild puis programmés en Step7 pour exécution dans l'automate. La Figure 4.2 présente le Filtre de Comportement d'une vanne de régulation de la plateforme CISPI.

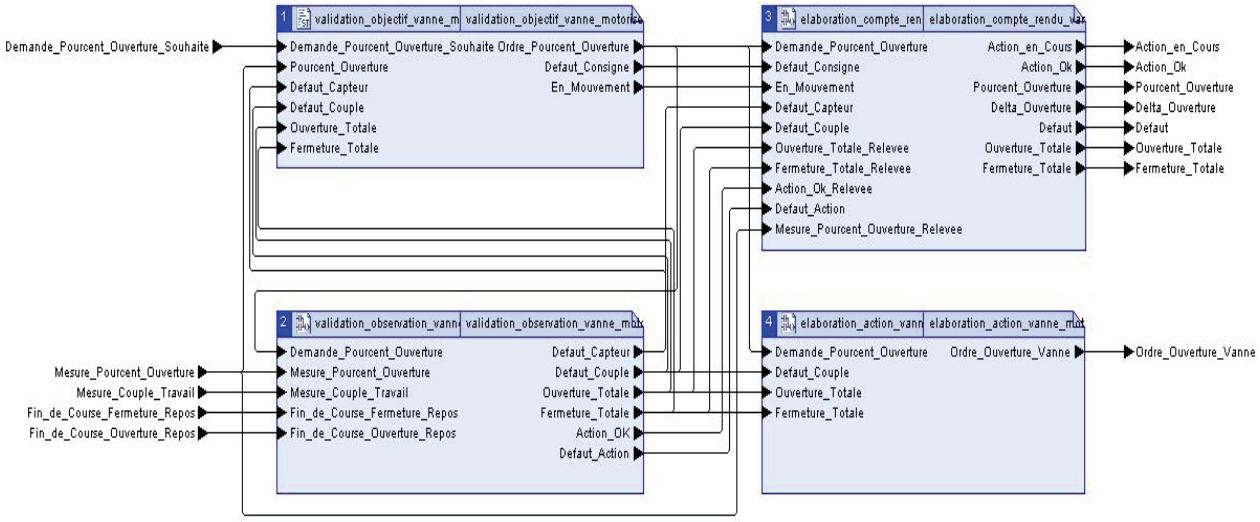


Figure 4.2 : Exemple de Filtre de Comportement d'une vanne de régulation réalisé avec ControlBuild

Après l'étape de spécification architecturale développée au Chapitre 3, nous allons décrire la réalisation des composants du SIAC devant s'intégrer dans l'architecture de la plateforme existante CISPI.

4.2 Réalisation des constituants du système interactif d'aide à la conduite

L'architecture organique du SIAC (Figure 3.37) que nous avons proposée dans le chapitre précédent comporte trois constituants : l'Objet Logique Technique (OLT), l'Objet Logique Interactif (OLI) et l'Objet Logique de Flux (OLF). Cette architecture ainsi que l'ensemble des exigences prescrites, représentent le point d'entrée de notre réalisation. En effet, chaque constituant pourrait être développé indépendamment par un métier spécifique (OLT par le métier informatique/réseau, OLI par le métier informatique/ergonomie, OLF par le métier informatique/planification/systèmes d'informations), ce qui nous amène à présenter leur développement dans des sections séparées.

Pour chaque constituant nous justifierons nos choix technologiques pour la réalisation, et en définirons le comportement. Dans certains cas, nous représenterons une structure physique montrant le lien avec les éléments de l'environnement.

Étant données l'utilisation et les caractéristiques de la plateforme CISPI installée au laboratoire, le choix des technologies que nous devrons faire, et qui devra nous permettre de tester et de vérifier la faisabilité des solutions proposées, peut être détaché des contraintes environnementales industrielles sévères du domaine d'utilisation visé, sans remettre en cause notre démarche de recherche.

Nous commençons avec la réalisation de l'OLT qui va être intégré directement sur la vanne manuelle afin d'augmenter ses capacités fonctionnelles intrinsèques.

4.2.1 Objet Logique Technique : module augmenté pour les vannes manuelles

Nous avons vu que pour exécuter des actions de conduite en local, il est nécessaire d'autoriser d'abord l'action à faire, puis d'identifier la bonne vanne à manipuler, de valider l'action réalisée et enfin d'envoyer l'état des vannes d'une part au système de conduite pour mettre à jour la situation, et d'autre part au rondier afin de gérer la liste des actions restant à réaliser. Au niveau de la spécification de l'architecture du SIAC, nous avons choisi d'affecter ces fonctions (autorisation, identifier, validation, envoi d'état) à l'OLT, qui devra à la fois surveiller l'état de la vanne manuelle, et communiquer avec le système de conduite et l'OLI du rondier.

Les technologies permettant d'augmenter les capacités d'un objet technique dit passif, tel qu'une vanne manuelle, sont liées au domaine de l'Intelligence Ambiante. Des travaux réalisés au début de cette thèse (Cea Ramirez, et al., 2006; Dobre, et al., 2009) pour étudier le concept d'objet actif ont mis en exergue les points forts de la RFID[38] et des réseaux de capteurs sans fil (WSN[39]).

La technologie choisie pour l'OLT devra être reliée à l'OMT (vanne manuelle) dans une structure physique évoluée supportant les nouvelles fonctions demandées par l'architecte système. Les nouvelles fonctions d'intelligence ambiante attendues doivent permettre à l'OLT de disposer d'un comportement interne avancé offrant des capacités de communication et de décision afin d'augmenter les capacités standard de l'OMT.

4.2.1.1 Contexte et choix de la technologie ambiante

L'objet actif est un élément d'un environnement à l'Intelligence Ambiante qui permet à un objet physique d'incorporer des capacités de stockage, de surveillance, de communication avec son environnement proche, et de réaction dans un espace de décision défini.

De nombreuses technologies existent sur le marché offrant des caractéristiques distinctes pour implémenter un objet actif à différents niveaux de complexité. Une représentation du panorama des technologies ambiantes est proposée à la Figure 4.3, prenant en compte deux axes :

[38] Radio Frequency Identification
[39] Wireless Sensor Network

autonomie d'action et intelligence technique (Bajic, et al., 2008). En partant du code à barre qui permet d'identifier un type de produit en utilisant une lecture optique en vue directe, les technologies évoluent pour gagner en capacités de sauvegarde d'information dans une mémoire non-volatile telle qu'une puce RFID. A présent, nous disposons de capacités de calcul, de décision et de communication (Gouyon, et al., 2009) pour fournir un traitement automatique de l'information échangée à l'aide des SmartCards et WSN. Le Tableau 4.3 présente une analyse des avantages et inconvénients de ces technologies.

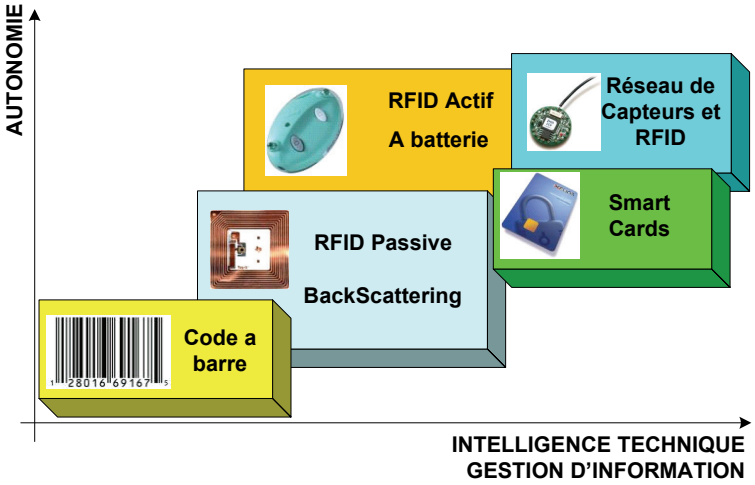


Figure 4.3 : Technologies ambiantes utilisées pour le développement d'un objet actif

Tableau 4.3 : Comparaison des technologies ambiantes

Technologie	Avantages	Désavantages
Code à barre	• Code visible	• Distance de lecture réduite • Vue directe pour la lecture
RFID Passive	• Ne nécessite pas de batterie • Mémoire pour les données • Dimensions réduites	• Distance de lecture réduite • Taille mémoire réduite
RFID Actif	• Distance de lecture assez grande • Mémoire pour les données • Peut intégrer des capteurs	• Nécessite une batterie • Dimensions assez grandes
Smart Card	• Programmable • Mémoire pour le program/données • Communication sécurisé	• Distance de lecture petite ou • Nécessite le contact pour la lecture
Réseau de Capteurs sans Fil	• Programmable • Microcontrôleur puissant • Mémoire programme+données • Communication à distance entre capteurs et avec un nœud base • Inclut des capteurs et circuits d'acquisitions	• Coût • Nécessite une batterie

Nous avons testé les différentes technologies présentées ci-dessus pour connaitre leurs caractéristiques dans le cadre du développement du concept d'objet actif. Ainsi, une des premières implémentations d'une architecture matérielle de produit actif (Cea Ramirez, et al., 2006) utili-

sait la technologie RFID permettant de sauvegarder les informations propres au produit. Associée à une infrastructure de services UPnP[40] nous avons pu attribuer à chaque produit physique des services personnalisés qui peuvent être invoqués depuis n'importe quel point de l'architecture réseau comme une image virtuelle du produit physique. La nécessité d'avoir un ordinateur intermédiaire afin d'interagir en continu avec les objets étiquetés RFID est un inconvénient structurel de cette architecture. De plus, l'ordinateur nécessite une connexion réseau pour permettre la communication entre objets actifs. Ainsi, l'objet actif représente une entité virtuelle (Dispositif UPnP) d'un objet physique capable de communiquer, d'interagir et de prendre des décisions concernant son état et son usage. L'utilisateur peut interagir avec celui-ci à partir d'un point de contrôle UPnP, caractérisé par une machine informatique intégrant une interface utilisateur dédié à la surveillance et au contrôle de ces objets actifs.

Afin d'explorer les capacités des capteurs sans fil, nous avons réalisé une plate-forme pour la gestion de la sécurité des biens et des personnes dans les activités de stockage et manipulation des produits chimiques dangereux (Dobre, et al., 2009). En attachant un capteur sans fil, le produit chimique « actif » recueille des données de l'environnement (température, chocs, autres produits qui l'entourent, …) et les traite. Sa réaction automatique à des menaces de l'environnement (température trop élevée, produit incompatible dans sa proximité, …) est traduite par une alarme audio et/ou visuelle, qui alerte le personnel qui le manipule, et diffuse l'information en réseau. Un point important de cette approche est le stockage et le traitement des connaissances associées au produit dans le produit lui-même. Ainsi, le produit actif peut communiquer avec d'autres produits pour prendre des décisions sur un environnement en évolution surveillée ou une intrusion d'un produit.

Afin de réaliser le module OLT de la vanne, notre choix s'oriente vers l'intégration de 2 solutions technologiques : réseau de capteurs sans fil et RFID.

Choix de réseau de capteurs sans fil

La solution doit permettre la communication sans fil entre les opérateurs et les composants, ainsi que l'intégration avec des vannes manuelles qui comportent des capteurs de fin course. Notre choix c'est orienté vers le capteur sans fil MicaZ[41], exploitant un protocole de communication basé sur la couche physique IEEE 802.15.4 et pouvant récupérer les états des vannes manuelles à l'aide du circuit d'acquisition MDA300CA (Figure 4.4).

Les réseaux de capteurs sans fil sont exploités dans de nombreux projets de recherche et industriels, dont notamment le projet OCARI (Dang, et al., 2008) pour instrumenter des centrales de production d'électricité ou des stations navales. Au niveau application, le projet OCARI a choisi le protocole ZigBee afin de fournir une compatibilité maximale avec les applications ZigBee non limitées au domaine industriel, contrairement à la spécification ISA100.11a basée sur la couche physique IEEE 802.15.4 (2006) quand ZigBee est basée sur IEEE 802.15.4 (2003).

[40] Universal Plug and Play
[41] Les MicaZ sont commercialisés par MEMSIC : http://www.memsic.com

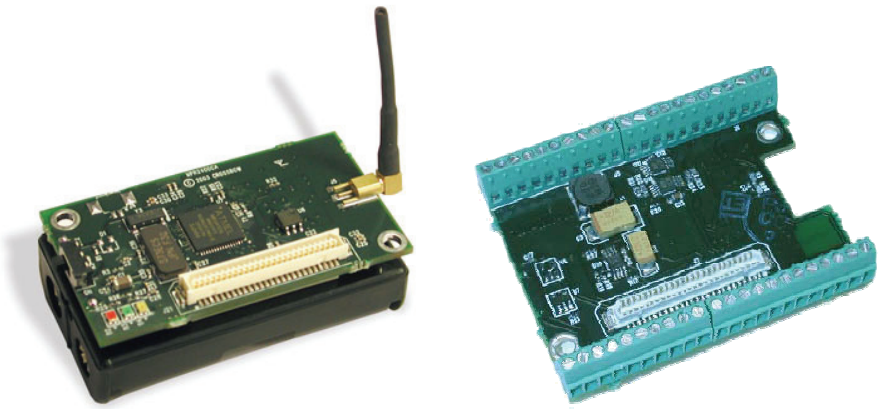


Figure 4.4 : Capteur sans fil MicaZ et circuit d'acquisition MDA300CA

Dans nos travaux, nous avons privilégié la couche application fournie par la solution Crossbow (Memsic), plutôt que les niveaux application ZigBee ou ISA100.11a. Ce choix est justifié par des coûts d'acquisition et de développement inférieurs pour la solution Crossbow (Memsic) tout en offrant des fonctionnalités suffisantes en regard à nos contraintes d'application qui sont : une communication point à point ou globale. Toutefois, la solution Zigbee pourrait nous apporter des fonctionnalités supplémentaires intéressantes, comme notamment la disponibilité sur le marché d'assistants personnels de type PDA intégrant la technologie ZigBee.

Un axe de recherche très développé dans la communauté scientifique concerne la gestion de l'énergie dans les réseaux de capteurs sans fil, assurée par batteries, dans l'objectif d'optimiser la durée de vie de ces types de systèmes. Le comportement interne du capteur ainsi que le protocole de communication participent fortement à la consommation de l'énergie embarquée. Nous n'avons pas considéré cette contrainte dans le test de nos solutions, mais ceci est à prendre en compte dans les perspectives futures de nos travaux.

Afin de faciliter l'intégration du réseau de capteurs sans fil dans l'infrastructure de réseau de la plateforme existante, nous avons choisi d'utiliser la passerelle Stargate Netbridge. Cette passerelle permet aux applications clients de recevoir et d'envoyer des messages de et vers les capteurs sans fil, au travers d'un serveur TCP/IP appelé XServe. Ce serveur assure la structuration des messages échangés entre les clients et les capteurs sans fil dans des objets XML qui facilitent leur utilisation.

Choix de l'identification par RFID

L'identification de façon sûre et unique d'une vanne parmi un environnement physique varié et difficile d'accès, comme peut l'être un processus industriel complexe, nécessite une technologie de proximité offrant ce service. Pour cela, nous proposons un marquage RFID de chaque vanne afin de pouvoir identifier précisément le composant et accéder à diverses autres informations associées telles que : fabriquant, date d'installation, dernière intervention, … Les informations étendues peuvent être accédées en liaison avec un système de gestion technique éventuel. La solution RFID pour l'identification de composants physiques s'impose comme un

standard de fait à ce jour et offre de nombreuses applications de référence en industrie comme en recherche (Kou, et al., 2006).

Nous avons choisi une technologie RFID I-Code Philips (Philips, 2005), de type HF SLI ISO 15693 avec 1048 bits mémoire, fonctionnant dans la bande HF à 13.56 MHz.

La technologie choisie comme support de réalisation du module vanne nécessite une intégration physique et fonctionnelle avec la vanne matérielle. Cette intégration est présentée dans les paragraphes suivants.

4.2.1.2 Modèle structurel de la vanne augmentée

Afin de pouvoir augmenter les capacités des vannes manuelles, un certain nombre de fonctions et de traitements sont nécessaires, allant de la connexion physique des signaux de la vanne jusqu'à la communication sans fil des informations associées. La Figure 4.5 indique les différents niveaux et étapes de traitement entre la vanne pris comme composant physique, et la communication sans fil.

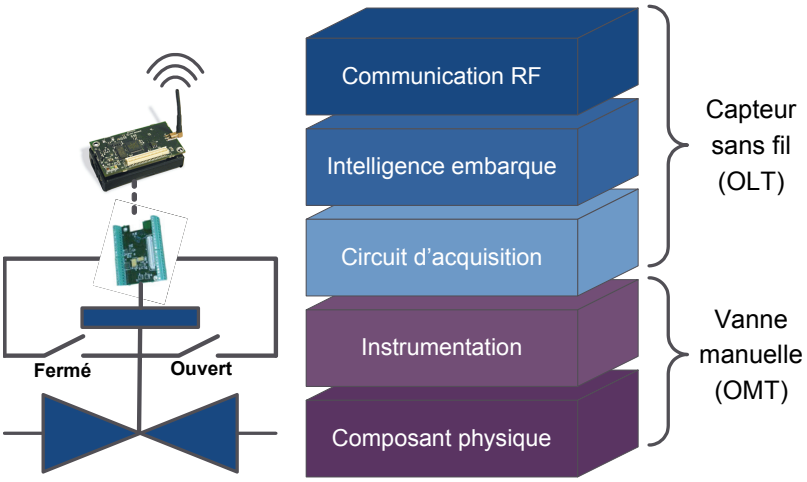


Figure 4.5 : Vue physique du lien entre le capteur sans fil et la vanne manuelle

Le niveau bas de la connexion physique nécessite de relier le circuit d'acquisition du capteur à la vanne manuelle. Dans la Figure 4.6 nous montrons les connecteurs du circuit d'acquisition MDA300CA qui permettent de se connecter à la vanne manuelle dotée des capteurs de fin de course, ainsi que son schéma interne. Parmi les connecteurs nous disposons de 7 entrées analogiques, 6 entrées/sorties digitales, 2 entrées relais.

Du coté de la vanne manuelle de type Georg Fischer, nous disposons de deux interrupteurs jouant le rôle de capteurs de fin course, et de trois connecteurs : un pour la masse et deux autres pour chaque interrupteur. Ainsi, nous avons décidé d'utiliser les ports GND, D0 et D1 du circuit d'acquisition afin de relier la vanne manuelle au capteur sans fil (Figure 4.7).

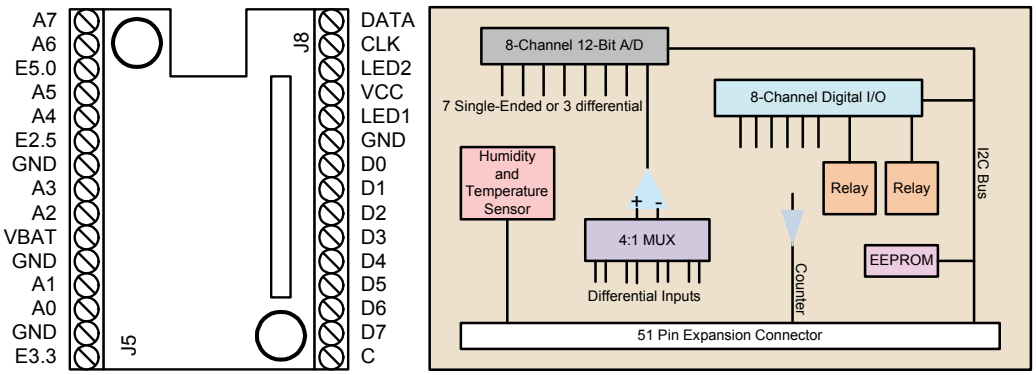


Figure 4.6 : Les entrées/sorties du circuit d'acquisition MDA300CA et son schéma interne

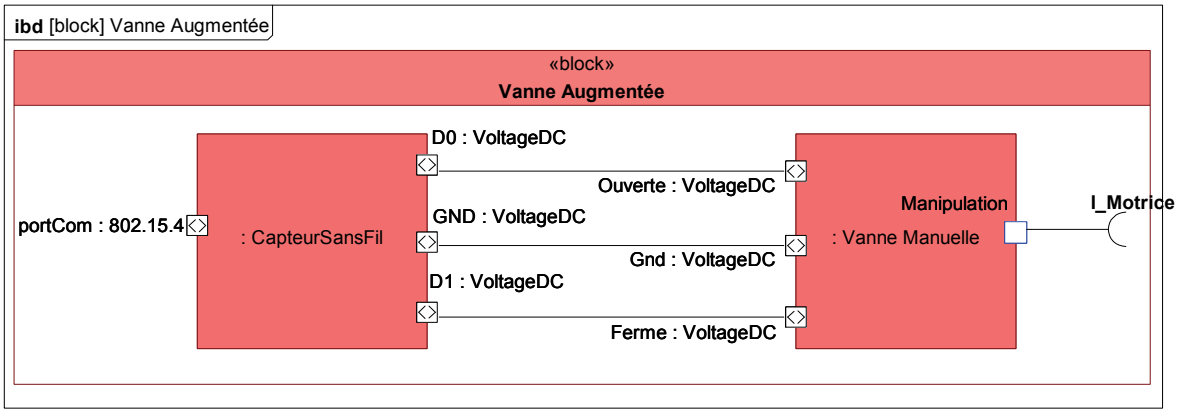


Figure 4.7 : Liaison d'une vanne manuelle à un capteur sans fil

Étant donné que la fermeture des contacts de fin course fait le lien entre la masse et une des entrées digitales, le fonctionnement du capteur est mis sur une logique inverse, c'est-à-dire qu'une entrée avec un contact fermé aura la valeur 0 et avec un contact ouvert, la valeur 1. Cette logique inverse doit être prise en considération pendant la réalisation du comportement du capteur sans fil, en tant qu'exigence induite du choix de la technologie.

La connexion entre la vanne manuelle et le capteur sans fil, à laquelle s'ajoute la puce RFID est montrée à la Figure 4.8.

4.2.1.3 Modèle comportemental de la vanne augmentée

Le rôle de la vanne augmentée est d'assurer une sécurité intrinsèque en vue de garantir une bonne manipulation des actions de conduite de la part des rondiers.

Les trois principales fonctions que les modules des vannes doivent exécuter sont l'autorisation de l'action à exécuter, la validation de l'action exécutée et la remontée de l'état de la vanne vers l'assistant du rondier et vers le système de conduite, pour synchroniser la situation logique relativement à la situation physique. Cette synchronisation est nécessaire afin d'éviter une prise de décision erronée de la part des opérateurs de conduite, qui se réfèrerait à une situation observée (logique) différente de la situation réelle (physique). A ces trois fonctions s'ajoute

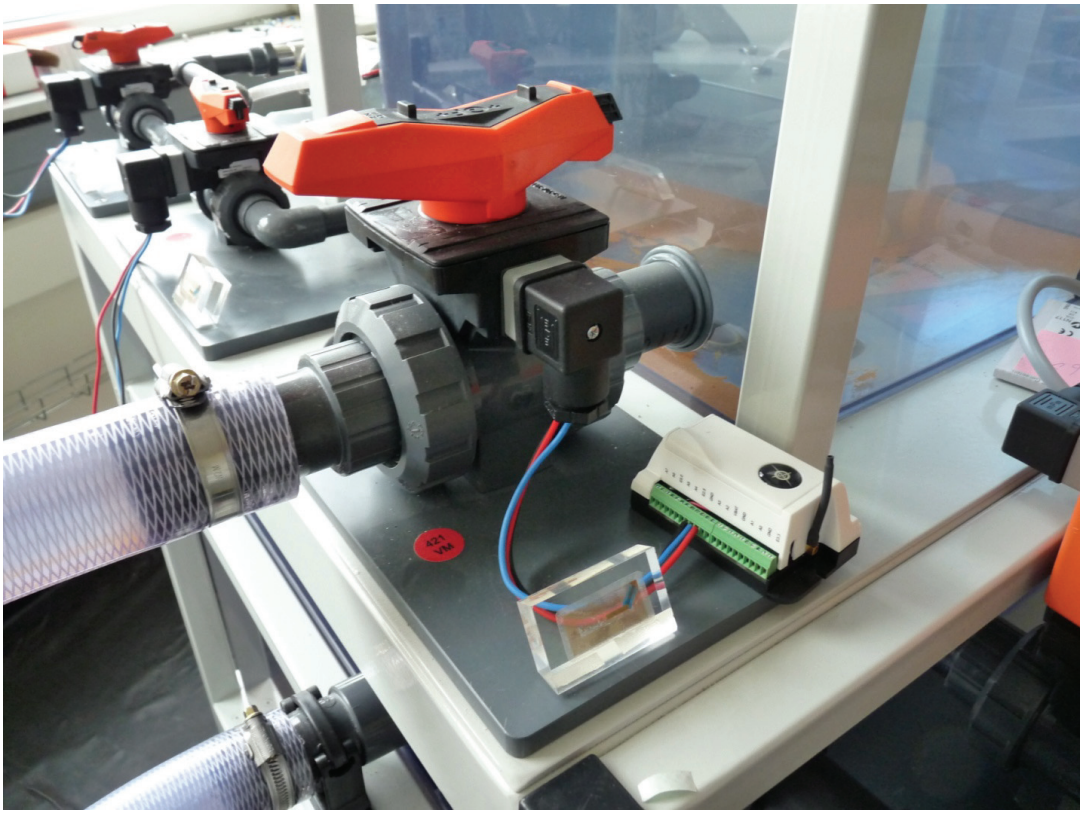


Figure 4.8 : Vanne manuelle avec capteur sans fil et puce RFID

la fonction d'identification qui en effet est réalisée par l'OLI, ceci à cause du caractère passif de la puce RFID choisi pour remplir cette fonction.

Le modèle comportemental du capteur sans fil entrant dans la composition de la vanne augmentée doit ajouter au comportement normal de la vanne manuelle des états supplémentaires introduits par les fonctions réalisées. La Figure 4.9 présente les deux diagrammes d'états des comportements de la vanne manuelle et du capteur sans fil, montrant clairement les états supplémentaires liés à l'autorisation (*AutorisationOuverture*, *AutorisationFermeture*) et à la validation (*ValidationOuverture, ValidationFermeture*) des actions à faire. Ainsi, pour une exécution correcte des actions de conduite en local, le chemin doit passer par l'autorisation (préparation), la réalisation par le rondier de l'action et la validation (clôture). Tout autre chemin qui passe directement entre l'état Ouverte et Fermée (action réalisée sans autorisation préalable ou une fausse manipulation de la mauvaise vanne) va générer des messages d'erreur.

Le contrôle de dépendance des actions nécessite une interaction entre les vannes, par le réseau de capteurs, ou avec les autres constituants instrumentés classiquement à l'aide des capteurs/actionneurs gérés par des automates programmables, afin de disposer des états physiques des composants. La logique d'autorisation doit être sauvegardée dans le capteur comme une contrainte intrinsèque d'actionnement.

La validation est faite en comparant l'état de la vanne avec l'action à faire, reçue lors de la demande d'autorisation. Ainsi, nous avons convenu, en tenant compte d'une procédure réaliste dans un environnement industriel, que si l'état de la vanne ne change pas dans une période de

30 min après l'autorisation ou si l'état change sans avoir une autorisation préalable, alors une erreur est générée.

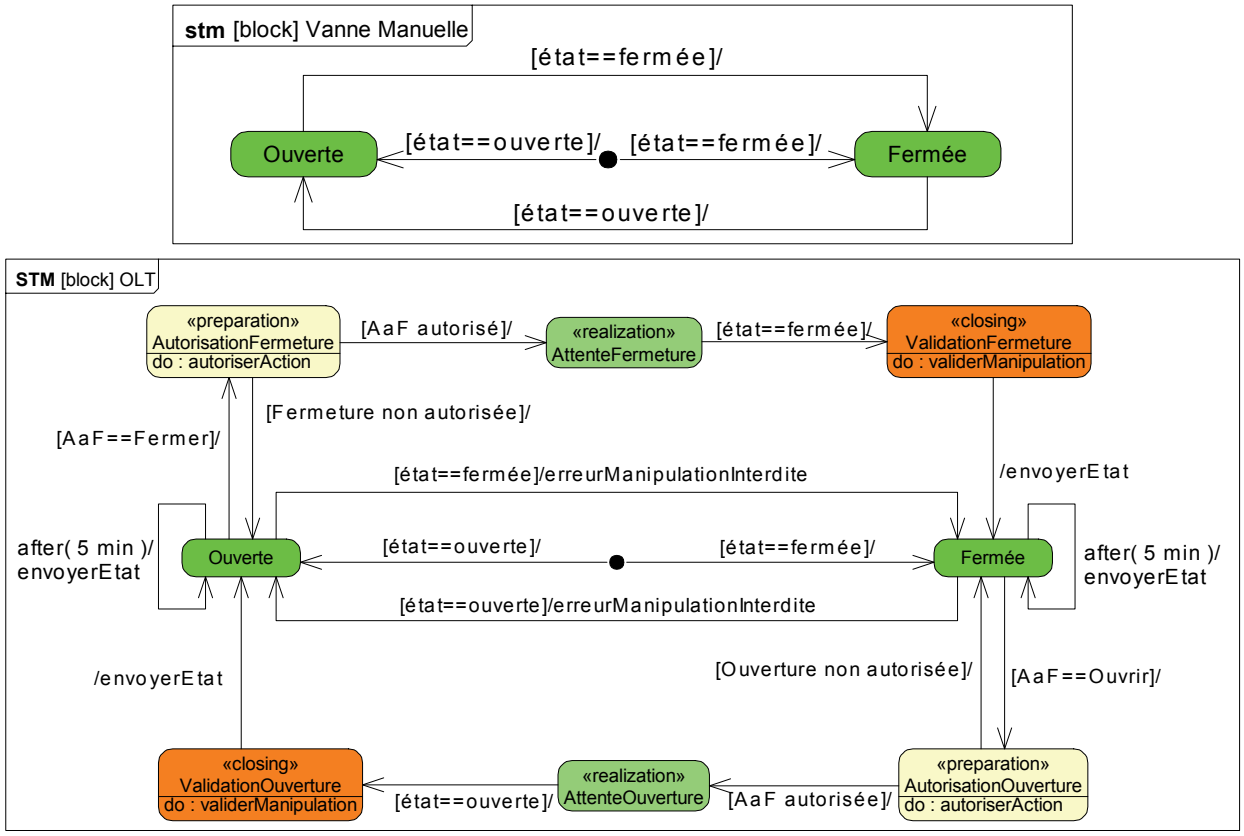


Figure 4.9 : Comparaison des diagrammes d'état du comportement de la vanne manuelle et du comportement du capteur sans fil

De plus, les capteurs sans fil doivent pouvoir interagir avec l'assistant du rondier afin de répondre à l'appel de celui-ci pour l'autorisation et la validation des actions réalisées. Un autre élément est la mise à jour du système de conduite qui, dans la spécification système, était faite directement par la vanne augmentée. Pour cela, une interface entre le réseau de capteurs et le serveur OPC permettra de mettre à jour des variables automates avec l'état des vannes manuelles. Ceci permet aussi aux constituants instrumentés de prendre en considération la situation du système physique afin de prendre des décisions concernant l'évolution de leurs situations.

4.2.1.4 Interface d'échange avec la vanne augmentée

Il est nécessaire d'assurer une communication transparente entre la vanne et le rondier afin de pouvoir interopérer par l'échange de messages d'autorisation et de validation des actions à réaliser. L'accès à la vanne se fait au travers d'une passerelle Stargate, intégrant le serveur XServe. L'assistant du rondier communique avec cette passerelle sur la base des messages respectant le format XML avec lequel nous avons spécifié le format des messages échangés.

L'API XCommand fournit par XServe est présenté comme un ensemble de fonctions XML RPC[42]. XML RPC est un cadre simple qui permet aux clients d'exécuter des procédures à distance en utilisant XML. Dans un échange typique, un client envoie un document XML au port XServe de commande spécifié (9003) et XServe retransmet le message dans le réseau de capteurs sous le format du protocole XMesh utilisé par les MicaZ. En réponse, même s'il n'y a pas de réponse de capteurs sans fil, XServe envoie au client une confirmation d'exécution de la commande dans le même formalisme XML.

Une fois configuré pour utiliser l'API XCommand, XServe s'attend à ce que tous les clients envoient et reçoivent des demandes en utilisant ce protocole. Un exemple de document XCommand XML RPC est donné ci dessous (Exemple 4.1).

Exemple 4.1 Document XCommand XML RPC

```
<?xml version="1.0" encoding="UTF-8"?>
<methodCall>  <methodName>xmesh.sleep</methodName>
    <params>
     <param>
      <value>
        <struct>
          <member>
            <name>seqNumber</name>
            <value>
              <int>108</int>
            </value>
          </member>
          <member>
            <name>destAddress</name>
            <value>
              <int>0</int>
            </value>
          </member>
          <member>
            <name>groupId</name>
            <value>
              <int>145</int>
            </value>
          </member>
        </struct>
      </value>
   </param>
 </params>
</methodCall>
```

Les informations qui sont envoyées vers XServe dans un document XML RPC sont le nom de la méthode à exécuter et ses paramètres.

[42] Appels de procédures à distance (Remote Procedure Calls)

L'OLI utilisé par le rondier utilise ce format de commande XML pour interopérer avec l'ensemble des OLT de son voisinage assurant ainsi un haut niveau de généricité des messages échangés.

4.2.2 Objet Logique Interactif : assistance au rondier par un environnement nomade

De l'architecture système, nous avons extrait le besoin d'un constituant représentant l'OLI utilisé par le rondier pendant l'exécution des actions de conduite. Cet assistant de type nomade doit permettre au rondier de circuler dans l'ensemble de l'infrastructure industrielle et entrer facilement en interaction avec les constituants de l'architecture physique. De nombreux produits d'informatique nomade sont disponibles sur le marché (TabletPC, IPad, SmartPhone, PDA) et pour certains, utilisés dans un environnement industriel, comme les PDA et les PocketsPC Industriels.

Mais le choix technologique doit tenir compte des exigences définies au niveau système, ainsi que des exigences induites par la réalisation de l'OLT.

4.2.2.1 Contexte et choix de l'OLI

Le choix de la technologie pour l'OLI doit prendre en considération plusieurs éléments : le type de communication sans fil, l'ergonomie, l'intégration d'un lecteur RFID, …

Le support d'assistance du rondier doit pouvoir disposer de fonctions de communication sans fil normalisées et RFID correspondant aux choix technologiques réalisés pour le module de la vanne : MicaZ et RFID. Etant donné l'utilisation de la passerelle Stargate Netbridge, la communication sans fil peut utiliser la technologie standard IEEE 802.11 et la communication RFID s'appuyer sur la norme ISO 15693, largement répandue en applications industrielles.

L'ergonomie est un facteur important dans l'efficacité de la conduite en local. Ainsi, l'assistant doit disposer d'une ergonomie adaptée pour le rondier, d'une interface graphique, d'une interface tactile et d'interface audio. L'interface graphique doit permettre une bonne visualisation (résolution élevée) et une bonne compréhension (graphisme adapté) des actions de conduite dans des conditions de faible et forte luminosité. L'interface tactile doit permettre l'introduction des observations du rondier dans le CR et la signature du rapport qui doit être envoyé vers l'opérateur de conduite. L'interface audio doit permettre la communication directe avec l'opérateur en salle de commande et doit intégrer un microphone et des haut-parleurs qui peuvent être utilisés dans un environnement bruyant (écouteurs).

L'ergonomie et le packaging du système doit permettre, entre autres, une grande liberté d'usage sans contraintes fortes de manipulation afin de permettre au rondier de disposer d'une liberté maximale pour des opérations physiques (Pirus, 2006).

De nombreuses applications ont été développées en exploitant des technologies d'interface homme-machine à base de réalité virtuelle, commandes vocales, …, mais le transfert dans le monde industriel n'est pas encore à ce jour totalement effectif pour diverses raisons : manque de maturité de certaines technologies, coût, fragilité, …

D'autres contraintes doivent être considérées comme la durée de vie de l'OLI, la compatibilité électromagnétique, l'existence d'une plateforme ouverte de développement.

Notre choix s'est porté alors sur un assistant de type PDA industriel, le WORKABOUT PRO S[43] (Figure 4.10), avec un module de communication WiFi 802.11a/b/g et un lecteur RFID à la fréquence de 13,56 MHz.

Figure 4.10 : Le PDA industriel WORKABOUT PRO S avec lecteur RFID intégré

L'intégration de cet assistant ne nécessite pas un modèle structurel spécial. Celui-ci doit être simplement intégré dans l'infrastructure de réseau sans fil existante dans la plateforme.

4.2.2.2 Comportement de l'assistant du rondier

L'assistant du rondier doit pouvoir exécuter des fonctions qui sont normalement exécutées par le rondier, et fournir à celui-ci des informations concernant le résultat de l'exécution de ces fonctions. Étant donné que l'assistant interagit avec d'une part les OLF et d'autre part avec les OLT, nous pouvons repartir ces fonctions en trois catégories :

- Fonctions partagées avec l'OLF :
  - La réception des actions de conduite en local ;
  - La transmission du rapport vers l'opérateur ;
  - La synchronisation entre l'opérateur et rondier ;
- Fonctions partagées avec l'OLT :
  - La demande d'autorisation vers les vannes concernées par les actions ;
  - La validation de l'action exécutée ;
- Fonctions partagées avec le rondier :

[43] Le WORKABOUT PRO S est commercialisé par Psion Teklogix : http://www.psion.com

- La gestion de l'action à faire qui fournit au rondier des informations concernant l'action qui est actuellement à l'exécution ;
- L'assistance à la localisation et manipulation de la vanne ;
- La gestion du CR de l'action.

Ces fonctions nous permettent de définir deux types d'interactions :

- interaction entre l'assistant, le rondier et la vanne augmentée pour exécuter les actions demandées par l'opérateur ;
- interaction entre l'assistant et les OLF pour communiquer les actions à exécuter.

Dans le premier cas, le rôle de l'assistant est de fournir au rondier des informations suffisantes pour que celui-ci exécute l'action dans de bonnes conditions. Ainsi, nous avons choisi de donner à l'assistant un comportement proactif dans l'évolution du rondier pour l'exécution des actions de conduite en local. Pour cela, l'assistant prend l'initiative de la demande des actions de la part du rondier et se charge de confirmer/valider que l'action a bien été réalisée (localisation, manipulation).

Du côté de l'interaction avec la salle de commande, l'assistant doit interagir avec l'OLF qui est en train de s'exécuter afin de recevoir la liste d'actions à exécuter, d'envoyer le rapport sur l'exécution des actions de conduite et de faciliter la synchronisation entre l'opérateur et le rondier.

Nous avons pour cela choisi d'intégrer au cœur de l'assistant du rondier un agent de communication pour le rendre compatible avec l'infrastructure multi-agents qui définit les OLF. Cet agent, réalisé à l'aide de la bibliothèque JADE SHARP Add-On, a pour rôle de recevoir toute demande émisse par le OLF et d'envoyer toute information vers l'OLF. Il constitue une interface de communication pour l'application d'assistance au rondier développée sur le PDA. Ce choix sera justifié dans la section §4.2.3.

L'intégration des deux principes d'interaction nous permet de définir le comportement interne de l'OLI comme représenté à la Figure 4.11.

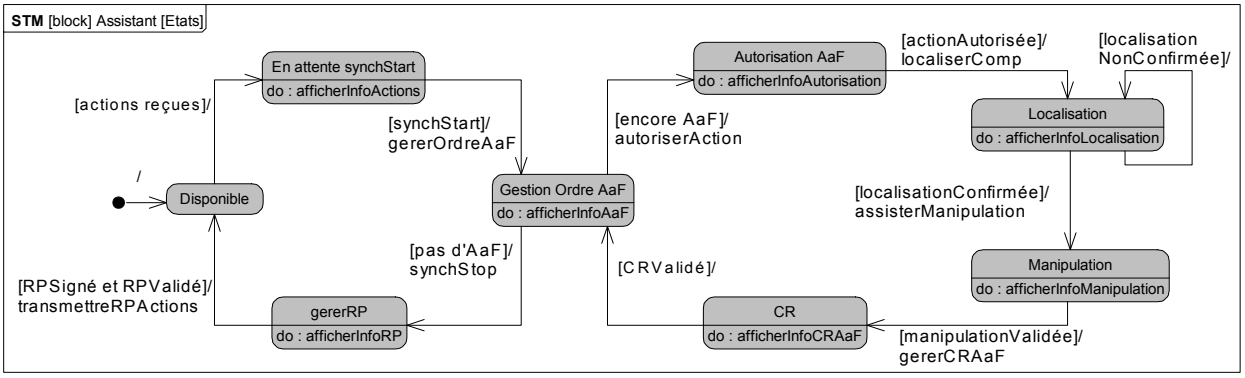


Figure 4.11 : Diagramme d'état du fonctionnement interne de l'assistant du rondier

4.2.2.3 Interfaces d'échanges

Du fait que l'OLI doit communiquer avec les OLT et les OLF, nous avons défini 2 interfaces d'échanges.

La première permet d'interfacer l'OLI avec les vannes augmentées. Celle-ci est réalisée en utilisant le réseau sans fil Wifi qui permet de connecter le PDA à la passerelle Stargate. Comme nous avons configuré le XServe pour utiliser le formalisme XML RPC, l'assistant du rondier doit pouvoir utiliser le même formalisme que XServe. Ainsi, entre le PDA et le XServe, tous les messages échangés sont en format XML et respectent le format spécifique du document XCommand XML RPC, présenté dans la section §4.2.1.4.

Entre l'OLI et OLF, les messages échangés sont conformes au protocole FIPA ACL (FIPA, 2010) qui spécifie un langage standard de messagerie qui définit le codage, la sémantique et l'utilité du message.

4.2.3 Objet Logique de Flux : Système de gestion des Modes Opératoires

Comme nous l'avons détaillé dans le chapitre précédent, les modes opératoires présentent des listes d'actions qui suivent la logique du procédé, réalisées à distance par l'opérateur en salle de commande et en local par les rondiers. Ces MO présentent différentes contraintes d'exécution comme par exemple les dépendances d'exclusions. Des besoins émergent pour la transmission et la réception des informations concernant le MO vers et depuis le local.

Conformément aux spécifications architecturales, les OLF doivent permettre une gestion automatisée des modes opératoires. Ainsi, chaque OLF doit permettre à l'opérateur la mise en service et mise hors service des MO, l'affichage des actions de conduite à distance, la transmission des actions de conduite vers le rondier en local et la réception par l'opérateur en salle de commande du rapport concernant les actions effectuées.

Pour la réalisation de ce système, deux problèmes se posent. Le premier est directement lié à la numérisation des MO. Pour cela il convient de définir un système d'information générique qui permettra de sauvegarder toute information concernant les MO.

Le deuxième problème est lié à la gestion d'exécution de MO qui doit vérifier les dépendances d'exclusions, gérer les synchronisations entre l'opérateur et le rondier, assister l'opérateur et communiquer avec l'assistant du rondier.

4.2.3.1 Système d'information pour la gestion technique des MO

Afin de numériser les modes opératoires, nous avons décidé de mettre en place un système d'information. Celui-ci doit pouvoir sauvegarder des informations concernant le personnel de conduite (rondiers et opérateurs), les équipements du processus, les MO avec les actions, les

dépendances d'exclusions, les dépendances d'ordre d'actionnement, … Notre choix s'est orienté sur un système de gestion technique de type MES (Manufacturing Execution Systems) standardisé ISO/IEC 62264 (2001). Ainsi, le « Personnel Model » permet de sauvegarder des informations concernant le personnel de conduite, l'« Equipement Model » est quant à lui dédié à la sauvegarde des informations sur les équipements du processus. Les MO sont instanciés en utilisant le « Process Segment Model » (Figure 4.12).

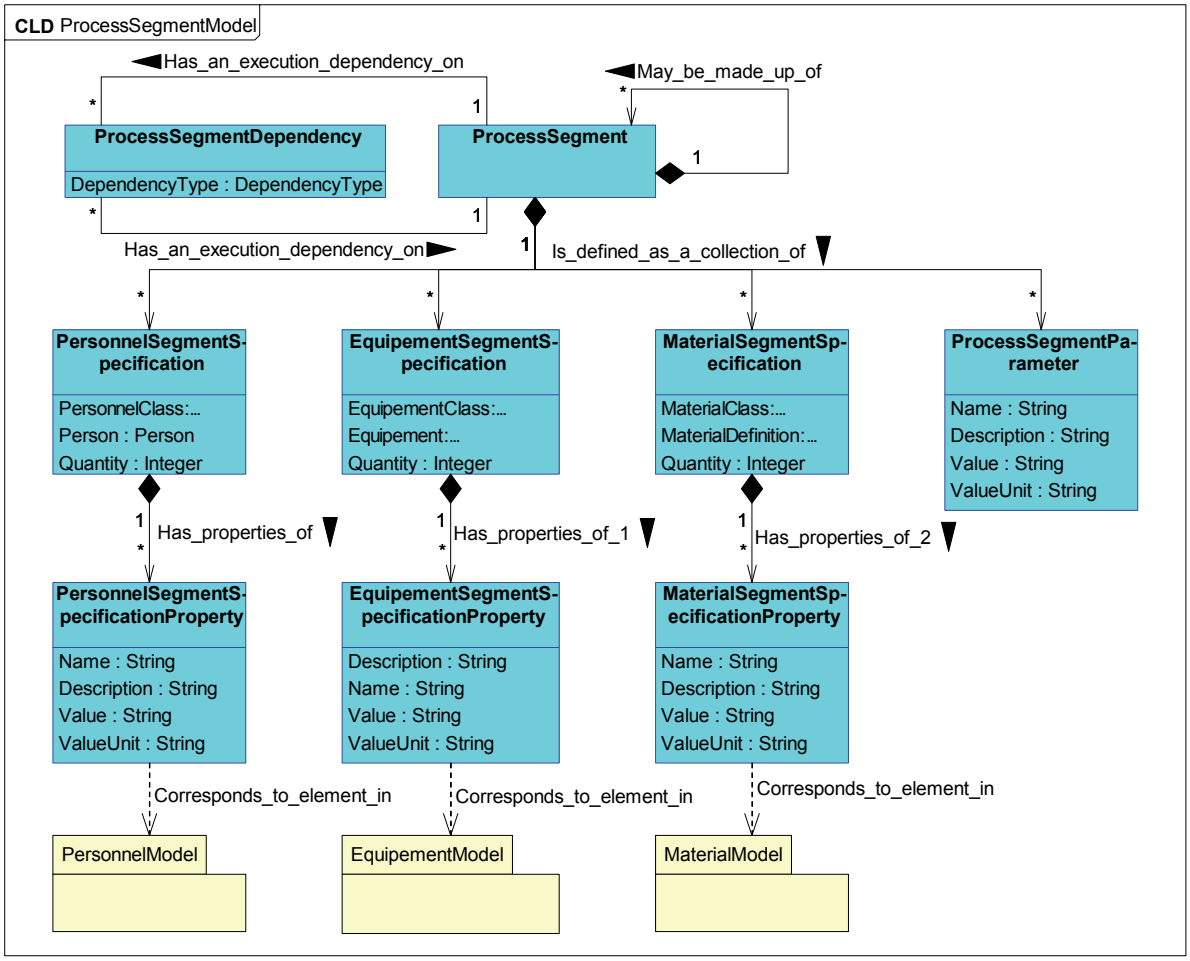


Figure 4.12 : Le Process Segment Model (ISO/IEC 62264, 2001)

Comme cela est montré à la Figure 3.11, un Guide d'Exploitation est une composition de MO composés d'un enchainement de Phases d'Exploitation structurées par des actions. Le lien de composition « May_be_made_up_of » est très approprié pour représenter ces compositions. Ainsi, le GE, les MO, les PE et les actions sont codées à l'aide de l'objet « ProcessSegment ». Ce sont ensuite les actions qui vont prendre en considération le personnel qui doit exécuter l'action (opérateur ou rondier) avec des propriétés spécifiques, et l'équipement sur lequel l'action doit être exécutée. Nous avons de plus défini les types de dépendances d'exclusion entre les MO et les dépendances d'ordre entre les actions (Figure 4.13).

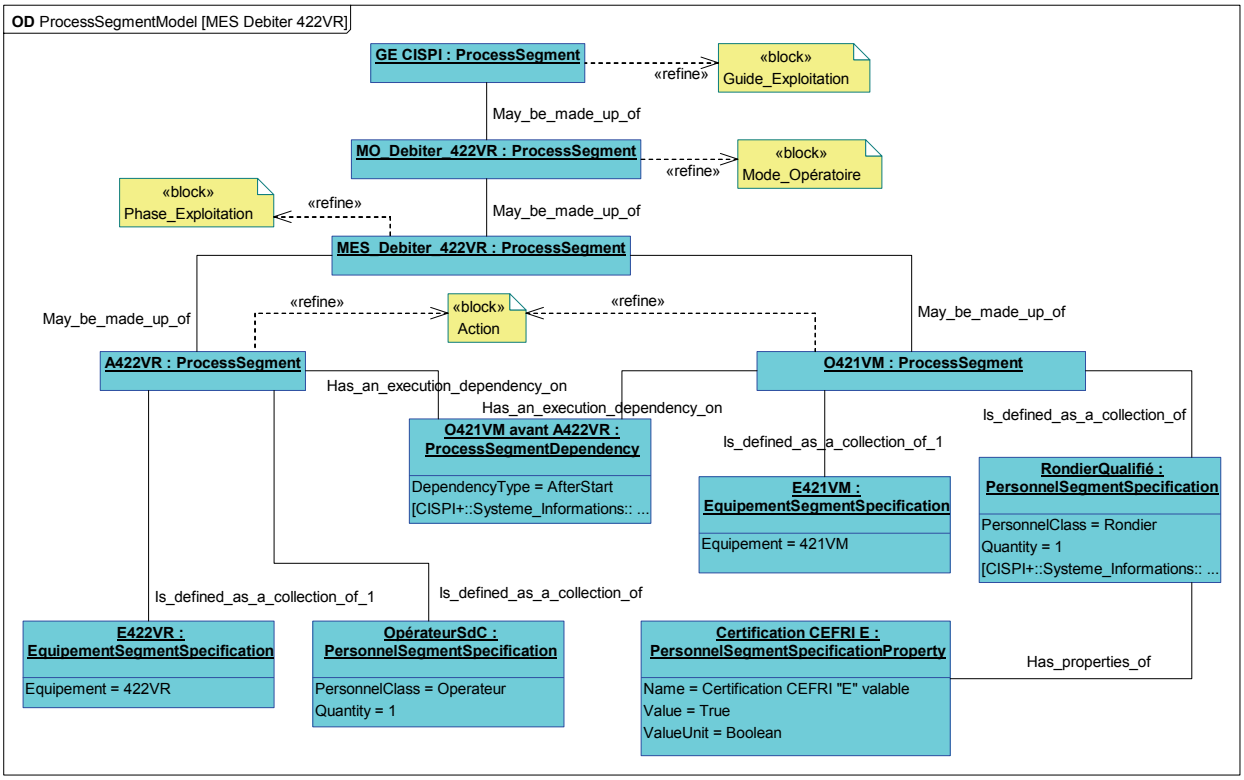


Figure 4.13 : Model partiel du « Process Segment Model » représentant le GE de la plateforme CISPI

Pour l'implémentation du standard MES ISO/IEC 62264 nous avons choisi d'utiliser B2MML[44] *(2008)* qui définit un modèle relationnel à partir du modèle conceptuel proposé par le standard. Le modèle physique de données de la base utilisée est représenté à la Figure 4.14.

4.2.3.2 Exécution et contrôle des MO

Les OLF doivent permettre la gestion des exclusions entre les MO, la transmission de la liste des actions au rondier et l'affichage des actions à exécuter par l'opérateur. Les systèmes multi-agents permettent de définir des entités logicielles capables de sauvegarder des informations, d'interagir entre eux pour un but commun et d'avoir un comportement propre. Ainsi, nous proposons de réaliser les OLF à l'aide d'un système multi-agents et de les implémenter à l'aide de la bibliothèque multi-agents JADE.

Dans l'architecture du SIAC nous avons prévu que les OLF représentent les MO. Étant donnée la multitude des MO dont un système de production pourrait disposer, nous avons ajouté un agent qui rassemble les MO. C'est ainsi que nous avons définis deux types d'agents :

- Un agent Guide d'Exploitation qui permet à l'opérateur de démarrer les MO ;
- Des agents MO qui reprennent chacun la connaissance relative à un MO.

Comme présenté dans le principe d'interaction entre l'assistant et le SGMO, un troisième type d'agent doit être définit pour faciliter l'échange des messages entre MO et assistant (OLI) du rondier.

[44] Business To Manufacturing Markup Language

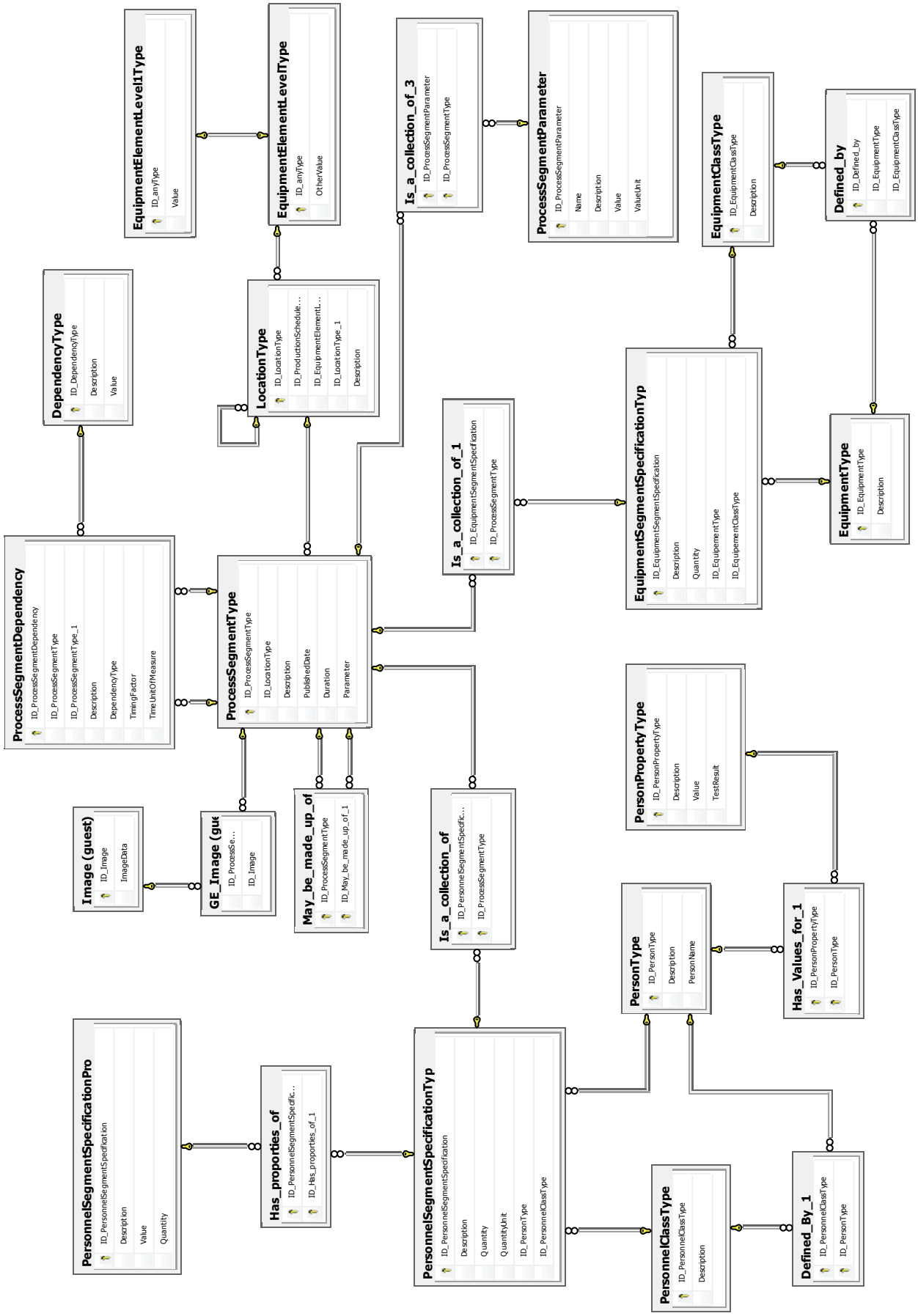


Figure 4.14 : Modèle physique de données de la base utilisée

Ainsi, nous définissons une architecture multi-agents représentée à la Figure 4.15. Nous avons défini les états d'un agent identiques à ceux des MO (Figure 3.12), c'est-à-dire disponible, en exécution, en exclusion.

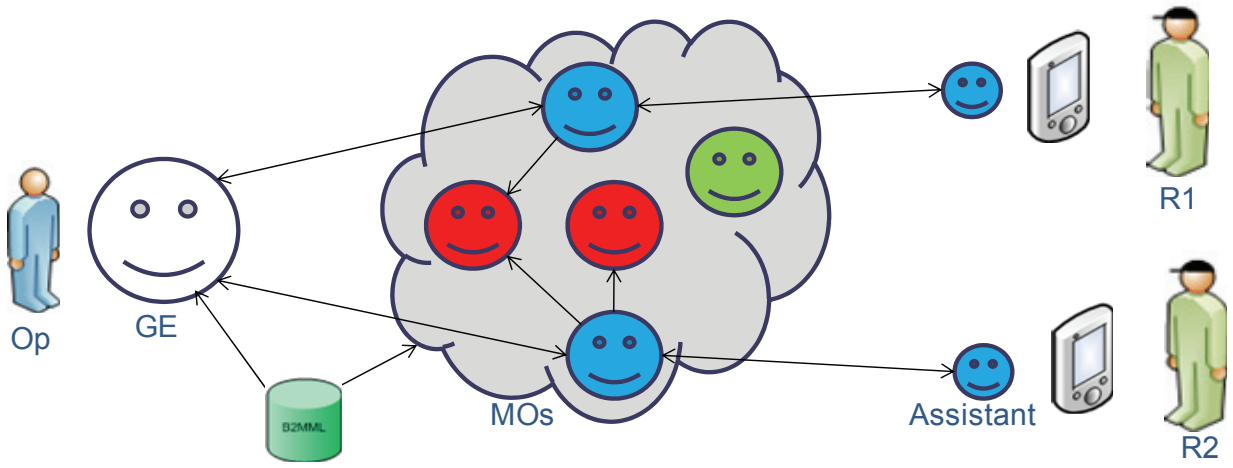


Figure 4.15 : Architecture du système multi-agents (vert = agent disponible, bleu = agent en exécution, rouge = agent en exclusion)

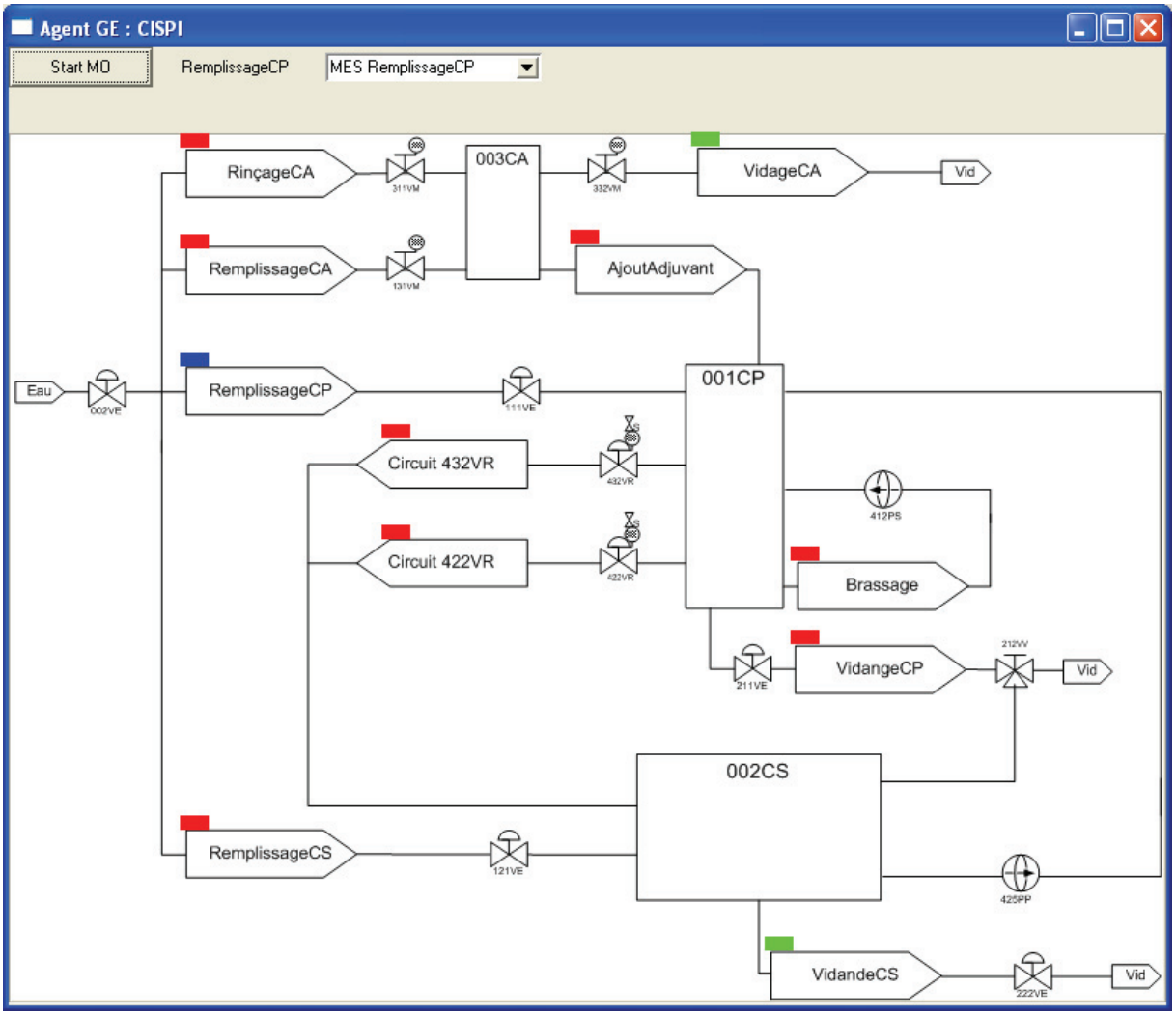


Figure 4.16 : Interface graphique de l'agent Guide d'Exploitation

Les agents réalisés sont génériques du fait qu'ils s'instancient en fonction des informations stockées dans le système de gestion technique B2MML. Le premier instancié est l'agent GE, qui va ensuite créer les agents MO. Cette instanciation à l'exécution facilite la mise à jour des informations de conduite (les MO) par la mise à jour du système de gestion technique sans reprogrammation des agents logiciels.

L'agent GE permet à l'opérateur en salle de commande de démarrer les MO. En effet, cet agent comporte une interface graphique (Figure 4.16) sauvegardée dans la base de données, qui présente la vue de l'ensemble des MO avec leur état représenté par la couleur associée. Pour démarrer un MO, l'opérateur doit choisir le MO et appuyer sur « Start ». Si le MO est bloqué, l'opérateur ne peut pas l'exécuter.

En exécution, l'agent de type MO envoie la liste des actions à un rondier disponible et affiche à l'opérateur la liste des actions que celui-ci doit exécuter (Figure 4.17). Il permet de faire la synchronisation entre l'opérateur et le rondier par l'envoi des messages de synchronisation.

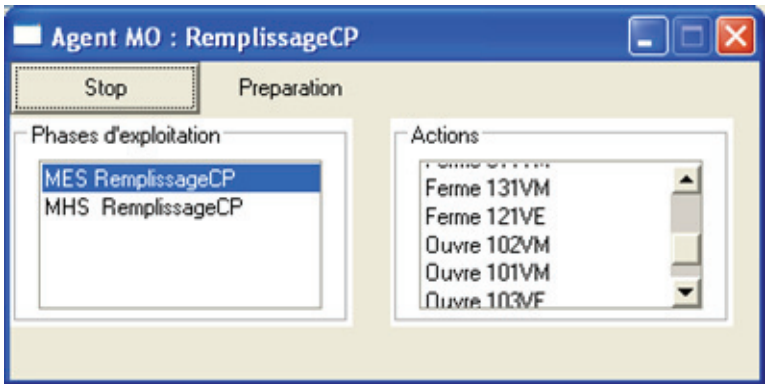


Figure 4.17 : Interface graphique de l'agent de type MO

On peut s'interroger sur l'intégration de ces OLF dans le système de conduite existant. Ainsi, nous avons choisi de ne pas les intégrer directement, mais de les exécuter en parallèle avec le système de conduite, l'intégration directe étant une des perspectives de cette thèse.

4.3 Intégration des constituants du SIAC

Après la réalisation des constituants du système d'aide à la conduite, il faut les rassembler dans une architecture et les intégrer dans la plateforme CISPI. L'architecture des constituants (Figure 3.44) qui est le point d'entrée de la réalisation va être utilisée pour guider la méthode avec laquelle les trois constituants avec leurs interfaces doivent être interconnectés.

4.3.1 Architecture matérielle

Dans la section précédente nous avons réalisé trois constituants qui entrent dans la composition du SIAC. Ainsi, l'OLT est réalisé pour être connecté avec la vanne manuelle, l'OLI est réalisé

pour être utilisé par le rondier pendant l'exécution du MO et l'OLF est utilisé par l'opérateur en salle de commande pour gérer les MO.

Ces trois constituants interagissent entre eux pour offrir les fonctions demandées par les parties prenantes intéressées. En effet, c'est leur intégration dans l'environnement de la plateforme CISPI qui permet de fournir ces fonctions qui assistent le rondier et l'opérateur à bien exécuter l'ensemble d'actions de la plateforme.

L'architecture matérielle de la plateforme CISPI avec le SIAC intégré est présentée à la Figure 4.18.

Intégration des OLT

Nous avons intégré les capteurs sans fil aux vannes manuelles qui sont dotées de capteurs de fin de course. Ces capteurs sans fil sont reliés entre eux avec la passerelle Stargate Netbridge par le réseau 802.15.4. Cette dernière est connectée au réseau filaire du VLAN Manip. C'est sur cette passerelle que l'outil XServer s'exécute.

Nous avons de plus attaché une puce RFID sur chaque OTB, utilisable pendant la conduite, voire pendant la maintenance.

L'intégration des OLF

Le système de gestion des modes opératoires (les OLF) est intégré dans le poste de conduite. Le système multi-agents s'exécute sur le même poste que le système de supervision et contrôle à distance (WinCC). En effet, un des écrans du poste de conduite est réservé pour le SMA qui affiche le GE et les MO en exécution.

Le système d'information reposant sur B2MML a été intégré dans une base de données SQL Server 2008 hébergé par un ordinateur réservé pour le système d'information. Le SMA doit donc se connecter à ce Système de Gestion Technique pour récupérer les informations qui leur permettent de s'initialiser à leur démarrage.

Intégration des OLI

Le PDA utilisé par le rondier est connecté via le réseau sans fil à la passerelle WiFi, qui permet de s'intégrer au même VLAN dédié aux plateformes expérimentales. Étant donné que le PDA est sur le même réseau que la passerelle, il peut se connecter pour interagir avec les capteurs sans fil. Une fois connecté, la communication avec les capteurs sans fil est transparente et vue comme une communication point-à-point. La lecture des puces RFID ne nécessite aucune infrastructure, mais requière une courte distance entre le PDA et la puce (OMT en particulier).

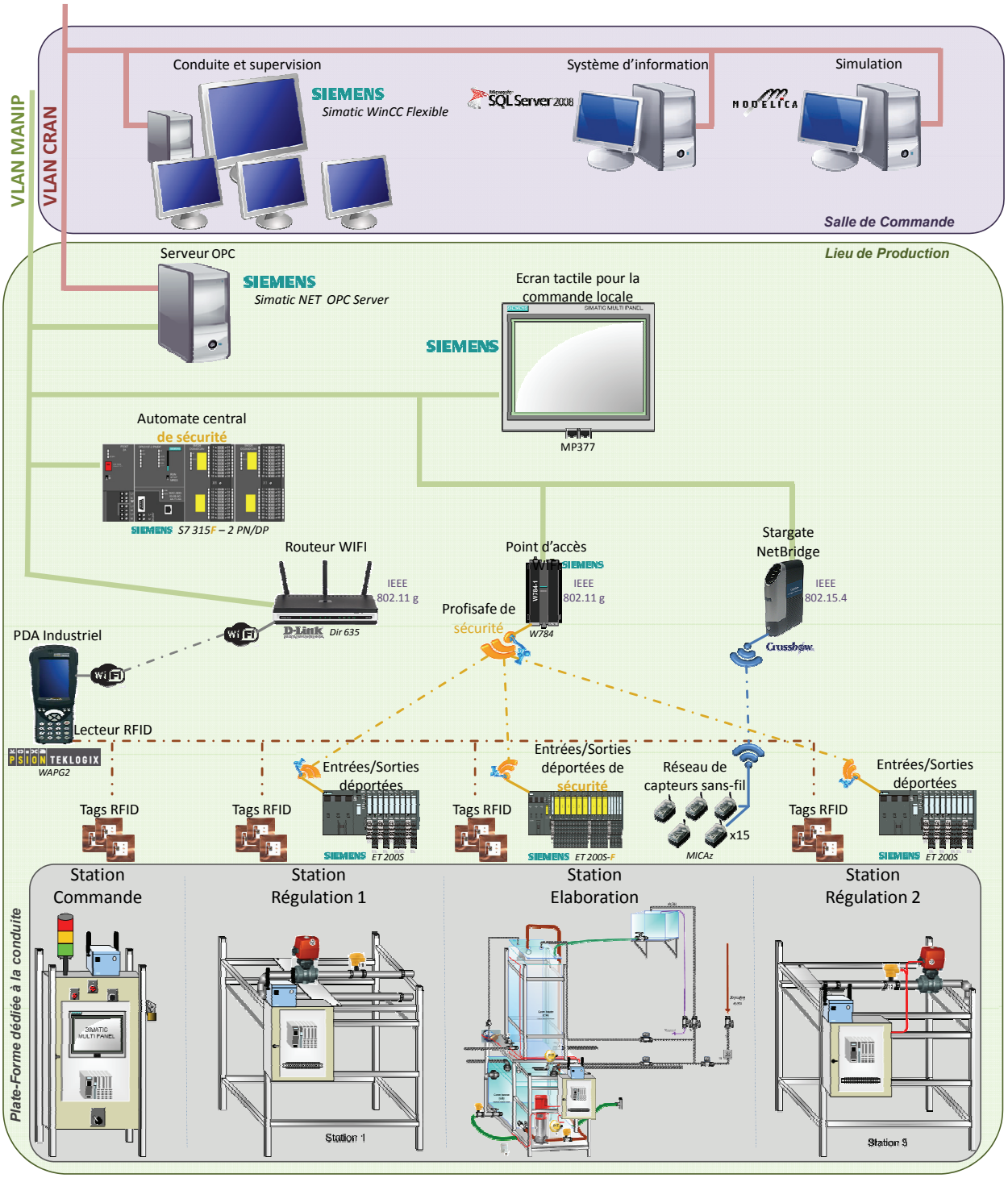


Figure 4.18 : Architecture matérielle de la plateforme CISPI avec le SIAC intégré

Après l'intégration des trois constituants du SIAC dans la plateforme CISPI, il nous faut vérifier et valider l'ensemble des fonctions que ces constituants réalisent dans une situation réelle de conduite en local.

4.3.2 Vérification de l'architecture matérielle

Pour vérifier l'ensemble des constituants réalisés et intégrés, nous avons utilisé les mêmes cas de test défini lors de la spécification système (section §3.6). De cette manière, les constituants réels ont également été vérifiés.

4.4 Conclusions

Ce chapitre présente la réalisation des différents constituants du SIAC et leur implantation sur la plateforme expérimentale CISPI.

Les OLT sont réalisés à l'aide de capteurs sans fil attachés aux vannes manuelles. Les capacités des capteurs sans fil permettent d'augmenter les capacités passives des vannes manuelles en ajoutant des services d'autorisation des actions à exécuter et de validation des actions exécutées. Pour la fonction d'identification, des puces RFID passives sont implantées.

Les OLF sont réalisés à l'aide d'un système multi-agents JADE permettant de gérer l'ensemble des informations et contraintes d'exclusions des Modes Opératoires. Un système d'information de type ISO 62264 est utilisé et implanté dans une base de données B2MML.

L'OLI est réalisé à l'aide d'un PDA industriel permettant au rondier d'interagir via une passerelle avec les capteurs sans fil et via un client JADE avec le système multi-agents.

La vérification des trois constituants est réalisée dans le cadre de la plateforme CISPI à l'aide des scénarios de test qui ont été définis et testés auparavant sur les modèles simulables. Les technologies choisies ont montré la faisabilité du concept proposé d'un Système Interactif d'Aide à la Conduite.

Nous avons développé les éléments présentés dans ce chapitre en collaboration avec différents étudiants en projet et/ou stage avec des durées entre 2 et 6 mois. Certains de ces travaux restent encore à finaliser comme le système multi-agent pour vérifier la faisabilité de la gestion des informations des modes opératoires et des exclusions en ligne.

Conclusion et Perspectives

Les travaux présentés dans ce mémoire s'inscrivent, d'une part, dans le cadre de la conduite des systèmes industriels complexes, et d'autre part, dans le cadre de l'ingénierie des systèmes complexes.

Le domaine de la conduite d'un procédé industriel nous a amené à dépasser une approche par trop technologique initialisée dans le domaine de l'« Intelligence Ambiante » (Cea Ramirez, et al., 2006; Dobre, et al., 2007a; Dobre, et al., 2007b; Dobre, et al., 2008a; Bajic, et al., 2008; Dobre, et al., 2009). Notre formation antérieure en informatique nous a alors amené naturellement à nous intéresser au langage SysML, mais qui est trop descriptif en l'absence d'un procédé de modélisation. Nous avons alors étudié le domaine de l'Ingénierie Système pour en extraire les artéfacts clés essentiels. Ces artéfacts clés issus des bonnes pratiques industrielles soufrent encore d'un défaut de formalisation. Ceci nous a amené à nous intéresser aux travaux de Jackson pour formaliser la relation de spécification dans notre procédé itératif d'IS afin de spécialiser SysML (Panetto, et al., 2010).

Nous avons alors disposé d'un outil-méthode pour définir, développer et déployer une solution pour le Système Interactif d'Aide à la Conduite (SIAC) que nous proposons (Dobre, et al., 2008b; Dobre, et al., 2010).

Chacune des deux parties de ce mémoire, le Système Pour Faire et le Système A Faire, aurait pu faire l'objet de développements plus complets. La difficulté de cette étude dans le temps imparti nous laisse un certain nombre de problèmes ouverts pour lesquels nous proposons des perspectives d'étude.

Système pour faire

Nos travaux ont conduit à la proposition d'un procédé de spécification dans l'Ingénierie Système qui partition tout domaine en espace problème / espace solution. Dans ce procédé, l'exigence est considérée comme le « moteur » de raisonnement avec une vision compositionnelle d'une partie optative et d'une partie indicative. Ce procédé permet de tracer les exigences en séparant ce qui est spécifié de ce qui a contribué à cette spécification. Il reste cependant à distinguer parmi les exigences satisfaites celles qui peuvent être prouvées dans un domaine d'application spécifique.

Cela amène à une forme d'Ingénierie Système Basée sur des Modèles (ISBM) mais Guidée par des Processus (ISGP). En effet, nous avons constaté qu'il est d'usage de manipuler des diagrammes dans une ISBM basée sur SysML sans procédé explicite de modélisation. Nous avons donc proposé un procédé de modélisation basé sur des bonnes pratiques de l'IS formalisées sous forme de processus.

Comme nous avons mis l'accent dans ce mémoire sur les processus d'ingénierie d'un bloc système, appliqués avec SysML, il nous reste à étudier plus en détail les processus d'intégration, de vérification et de validation. Du fait que les modèles SysML doivent faciliter l'intégration (par l'architecture organique), la vérification et la validation (par les cas de test), la difficulté de cette étude est de retrouver le lien entre les constituants réalisés et les modèles SysML.

Du fait que SysML est un langage semi-formel, il est possible d'envisager la formalisation du procédé de modélisation proposé sur les bases des travaux de Jackson. Par exemple, les travaux de Seater (2009) proposent des éléments pour la formalisation des transformations en « Problem Frame » pour vérifier et valider le passage du besoin initial vers la spécification de la machine en vue.

Le modèle de référence pour la spécification d'un système sociotechnique de Hall et Rapanotti (2005) nous offre une autre vision sur la spécification d'un système vu comme un TOUT, qui ne sépare pas l'humain du système technique. Ainsi, des travaux sur la spécification d'un système de conduite doivent être réalisés pour prouver une meilleure répartition des fonctions de conduite entre l'humain et le système technique de soutien.

Le modèle de référence de Hall et Rapanotti (2005) permet d'aborder avec une vision « Système » la spécification d'un système sociotechnique comme un TOUT dans un premier temps, pour ensuite allouer les exigences à des composants techniques ou à un ... « Humain ». Il s'agit d'une vision Ingénierie Système qui fait encore l'objet de débats et controverses par rapport à une approche Facteurs Humains qui aborde plutôt ce type de spécification de l' « Humain » vers ... le système technique (Rasmussen, 1983). Nous avons considéré ce modèle de référence comme un artefact-clé de notre procédé de spécification d'un système sociotechnique, bien que nous n'ayons pas encore abordé la définition des capacités (K) pour spécifier plus formellement les interactions de l'Humain avec le système technique (UI) dans le domaine d'intérêt (I). Plus précisément, c'est la complétude des spécifications $S \wedge UI \wedge I$ qui fait l'objet de travaux complémentaires (Fass, et al., 2009) sur la base de la théorie de la physiologie intégrative (Chauvet, 1993) qui pourrait offrir l'avantage d'intégrer les aspects essentiels des Facteurs Humains (Perception-Cognition-Action) dans la boucle de Décision. En ce sens, nos travaux ne sont que très préliminaires et nécessitent d'être développés au passage à l'échelle de problématiques industrielles (Galara D. , 2010) relatifs à la spécification d'un système de conduite pour assurer une répartition cohérente, d'un point de vue système, des fonctions de conduite entre l'humain et le système technique de soutien.

A d'autres égards, des travaux de mise en relation des artéfacts d'IS dans des bases de connaissances, comme les ontologies, doivent être réalisés pour consolider les vues qu'un Architecte Système doit avoir dans la phase de spécification d'un système. Ainsi, une sémantique de construction doit être définie autour de SysML (avec notamment des règles de type OCL), en prenant comme fil conducteur l'approche systémique de l'IS, pour guider le modélisateur dans son processus de spécification.

Ainsi, des nombreuses questions sur l'utilisation de SysML pour supporter le procédé d'IS restent encore ouvertes :

- Quel procédé mettre en place ?
- Vérification par transformation de modèles ?
- Quelle sémantique il faut utiliser par rapport à un métier ?

A l'inverse, se pose la question sur l'extension du langage métier à la modélisation système comme Modelica par des artéfacts SysML pour supporter l'ingénierie des exigences ou d'autres artéfacts de modélisation (d'ingénierie) système.

Système à faire

Nos travaux ont conduit à la proposition d'un concept d'architecture de Système Interactif d'Aide à la Conduite (Figure 5) dont nous avons proposé une spécification et un développement partiel. Si ces travaux à l'échelle réduite de la plateforme de laboratoire CISPI ont été réalisés, il reste à faire la preuve du concept à une échelle industrielle, ce qui nécessite une complétude de la spécification par des parties prenantes et qui dépend d'une collaboration industrielle.

Ces travaux prospectifs, sur la spécification d'un système interactif d'aide à la conduite, ouvrent des pistes de réflexions sur la prise en compte du domaine de l'humain et plus précisément les capacités dont il doit disposer pour pouvoir conduire d'une façon pertinente l'ensemble des constituants systèmes composant un procédé industriel complexe. La version actuelle du SIAC se limite aux informations concernant les modes opératoires, sans pouvoir pour l'instant réagir aux changements au niveau du processus devant être encore surveillés par les opérateurs et les rondiers. Ainsi, l'encapsulation des connaissances de l'humain dans le canal d'informations par une spécification intégré du système humain, vu comme composant système alloué au cours du procédé de l'IS, et du système technique pourrait créer un système réactif aux changements. Par exemple, le système multi-agents n'aura pas qu'une fonction de gestion des modes opératoires, mais pourra aussi, de la même façon que dans la planification tactique et opérationnelle, adopter un comportement flexible en fonction de la situation sur le terrain, pour prendre des décisions et réagir aux changements (*ex*. une opération non prévue qui pourrait mettre en danger l'installation industrielle, les personnes et/ou l'environnement) en fonction de la disponibilité des équipements et des rondiers sur le terrain.

Afin de valider la pertinence du SIAC, il sera intéressant de comparer les performances en termes de fiabilité de la conduite classique et de la conduite interactive. De plus, il est nécessaire d'étudier la validité et la qualité des informations traitées et communiquées par le SIAC. En autres, les informations gérés par le SIAC doit satisfaire les exigences fournies par la loi informatique et libertés de CNIL concernant l'identité humaine, les droits de l'homme, la vie privée et les libertés.

Pour que les bénéfices potentiels du SIAC soient pérennes dans le temps, certains verrous technologiques, comme par exemple l'autonomie des capteurs sans fil, sont encore à lever. En

ce sens, les travaux de la communauté des Réseaux de Capteurs Sans Fil tendent à réduire leur consommation en proposant de nouveaux protocoles de communication (Akkaya, et al., 2005).

Remarquons aussi que nos travaux ne tiennent pas compte du comportement intrinsèque du réseau par rapport au comportement attendu de l'ensemble du système. En ce sens, les travaux de la communauté des systèmes contrôlés en réseaux (Habib, et al., 2009) cherchent à évaluer l'influence du comportement des réseaux de communication sur la commande des systèmes automatisés.

Nos propres perspectives de travaux de post-doctorat dans le projet FerroCOTS nous permettent dès maintenant d'envisager d'appliquer notre procédé de modélisation avec SysML. D'une part, vu le niveau de sûreté envisagé par ce type d'application (SIL4), l'utilisation d'un langage plus formel doit être privilégiée. Cela imposera la formalisation des différents passages entre les blocs systèmes et à l'intérieur de chaque bloc système afin d'assurer le raffinement de bout en bout des propriétés définies par les besoins et prescrites par les exigences. Pour cela, des mécanismes de vérification et de validation de ces propriétés doivent être définis pour assurer la conformité avec des modèles SysML moins formels. D'autre part, des transformations entre SysML et l'outil métier ControlBuild, considéré pour la conception de la partie commande des trains, doivent être étudiées afin de réaliser une synchronisation entre une bibliothèque de composants de type COTS et les constituants spécifiés avec SysML.

Annexe A
Besoins identifiés pour un système d'aide à la conduite

Tableau A.1 : La liste des besoins fonctionnels

Id	Nom	Description
BF1	Eviter les conflits entre MO	Eviter d'exécuter des actions contradictoires sur le même composant.
BF2	Eviter les conflits entre actions	Les actions doivent suivre la logique du procédé pour éviter les conflits entre ces actions.
BF3	Exécuter des actions en local	Le rondier doit …
BF3.1	Autoriser l'action à faire	… pouvoir autoriser l'action qu'il veut exécuter.
BF3.2	Localiser le composant à actionner	… pouvoir trouver l'OMT à actionner.
BF3.3	Exécuter l'action demandée	… pouvoir exécuter l'action demandée.
BF3.4	Valider l'action exécutée	… pouvoir valider l'action exécutée.
BF4	Besoins en matière de moyens informatiques nomades	Besoins fonctionnels pour l'utilisation d'outils informatiques « nomades » pour supporter les taches réalisées en local.
BF4.1	Supprimer le risque de contamination des documents papier	Les exploitants rencontrés souhaitent que les dossiers d'intervention soient informatisés de sorte à supprimer tout risque de contamination en zone contrôlée.
BF4.2	Faciliter la transmission des documents	Les exploitants rencontrés souhaitent …
BF4.2.1	Transfert des documents depuis les services vers les techniciens en local	… pouvoir transférer sur l'outil informatique nomade utilisé par les techniciens en local toutes les informations utiles à leurs activités : des schémas, plans, modes opératoires, la liste actualisée de travaux, des analyses, des surveillances particulières, des relevés à réaliser, des régimes.
BF4.2.2	Transfert des documents par les techniciens en local vers la salle de commande et les services	… disposer de la possibilité de transmettre depuis leur lieu d'intervention en local (en temps réel ou en temps légèrement différé) les gammes et les relevés à la salle de commande et aux services de sorte à raccourcir le délai entre la réalisation des activités et la prise en compte des informations collectées pendant celles-ci.
BF4.2.3	Transfert des documents de la salle de commande vers les techniciens en local	… disposer d'un outil informatique collaboratif qui leur permet de…
BF4.2.1.1	Accéder à la documentation	… accéder à toute sorte de documentation (plans, schémas, bases de données, ...).
BF4.2.1.2	Echanger des documents	… échanger sur ces documents avec les opérateurs en salle de commande ou le C/C.
BF4.3	Aider les Rondiers à détecter les	Les exploitants rencontrés souhaitent disposer sur

	écarts entre les valeurs attendues et les valeurs relevées	un outil informatique nomade de toutes les valeurs attendues.
BF4.4	Aider les techniciens à réaliser les actions prescrites dans les gammes	Les exploitants rencontrés souhaitent pouvoir disposer …
BF4.4.1	Aide pour localisation	… d'une aide informatique pour localiser rapidement les locaux et les équipements de l'installation.
BF4.4.2	Vidéo "gestes rares"	… de moyens depuis le local pour visualiser face aux matériels les gestes à réaliser à partir des vidéos « gestes rares » disponibles pour les actions atypiques.
BF4.4.3	Calculs automatiques	… de moyens de calcul automatique.
BF4.4.4	Mesures intégrés dans terminal	… de moyens de mesure intégrés dans le terminal informatique nomade.
BF4.4.5	Annotation des documents opératoires	… de possibilités d'annoter les documents opératoires informatises de sorte à archiver ces informations au titre du retour d'expérience.
BF4.4.6	Alertes actions omisses	… d'alertes s'ils omettent de réaliser une action ou un relevé requis.
BF4.5	Permettre la communication orale et visuelle	Les intervenants rencontrés souhaitent disposer d'un moyen de communication orale, visuelle et de données mobile depuis le local de sorte à …
BF4.5.1	Communiquer en direct	… communiquer en direct et face aux matériels.
BF4.5.2	Etre joignable	… être joignables.
BF4.5.3	Partager des vidéos	…envoyer et partager en temps réel des vidéos.
BF4.5.4	Accéder aux bases de données	… accéder et renseigner depuis le local les bases de données disponibles sur le site et les applications métiers.
BF4.6	Supprimer la double saisie des données collectées en local	Les exploitants rencontrés souhaitent …
BF4.6.1	Renseigner directement les documents opératoires	… renseigner les documents opératoires (relevés, gammes d'essais, gammes de lignages, gammes opératoires d'intervention et de maintenance, Procès Verbal) directement dans une application informatique nomade.
BF4.6.2	Support de recueil d'informations	… disposer d'un support de recueil d'informations pour remplacer leur « carnet personnel».

Tableau A.2 : La liste des contraintes

Id	Nom	Description
BC1	Besoins liés à la recherche	Etudier l'intérêt et la faisabilité des différents concepts étudiés au sein du CRAN, tels que …
BC1.1	Utiliser le concept de Filtre de Comportement	… les Filtres de Comportement.
BC1.2	Utiliser le concept d'AmI	… l'Intelligence Ambiante.
BC2	Contraintes en matière de moyens informatiques nomades	Contraintes pour l'utilisation d'outils informatiques « nomades » pour supporter les taches réalisées en local.
BC2.1	La portabilité du matériel	

BC2.1.1	Outil léger	L'outil doit être léger.
BC2.1.2	Eviter risques d'accrochage	L'outil ne doit pas être porté en bandoulière pour éviter les risques d'accrochage.
BC2.1.3	Mains libres	L'outil doit laisser les utilisateurs libres de leurs mouvements.
BC2.1.4	Outil format poche	Un outil type « PDA » qui peut être placé dans une poche serait d'un format pratique.
BC2.2	Sa facilité d'utilisation	
BC2.2.1	Utilisation double paire de gants	L'outil doit être utilisable avec une double paire de gants (coton et latex).
BC2.2.2	Dialogues simples et intuitifs	Les dialogues doivent être le plus simple et le plus intuitif possible.
BC2.2.3	Transfert de données simples et intuitives	Le transfert des données doivent être le plus simple et le plus intuitif possible.
BC2.2.4	Flexibilité tâches	L'outil doit donner la plus grande flexibilité dans la réalisation des tâches pour permettre la meilleure autonomie aux Intervenants.
BC2.3	L'optimisation des tâches	
BC2.3.1	Supprimer documents papier	L'outil doit permettre de supprimer tous (ou presque tous) les documents papier emportés en local. Continuer à emporter en local des documents papier volumineux et/ou nombreux en plus de l'outil nomade réduirait de beaucoup l'intérêt de ce dernier.
BC2.3.2	Saisie automatique dans applications tierces	Les informations relevées en local sur l'outil nomade ne doivent pas être saisies manuellement dans une application tierce.
BC2.4	La traçabilité	Pour satisfaire aux exigences Qualité, les utilisateurs doivent pouvoir signer informatiquement les relevés, les gammes, etc., renseignés à l'aide de l'outil nomade.
BC2.5	La sûreté	L'utilisation d'un outil informatique nomade ne doit pas provoquer d'interférence dommageable pour la sûreté de l'installation.
BC2.6	La sécurité	L'outil doit intégrer une fonction « homme mort ». Certains interviewés pensent que cette fonction pourrait suffire à motiver des techniciens à utiliser l'outil.
BC2.7	La compatibilité du dispositif nomade avec les applications utilisées sur site	Les applications embarquées dans l'outil nomade doivent pouvoir dialoguer avec les applications génériques et les applications spécifiques aux métiers.
BC3	Coût du système	Le coût du système doit être plus bas qu'une instrumentation classique avec des capteurs et/ou actionneurs.

Annexe B
Les Modes Opératoires de la plateforme CISPI

La plateforme CISPI présente 14 modes opératoires, structurés en 4 groupes :

- MO pour démarrage de CISPI :
  - Remplissage Cuve Principale ;
  - Remplissage Cuve Secondaire ;
  - Remplissage Cuve Adjuvant ;
- MO pour production :
  - Ajout adjuvant ;
  - Brassage ;
  - Débiter par 422VR ;
  - Débiter par 432VR ;
- MO pour arrêt de CISPI:
  - Evacuation de la Cuve Principale dans Cuve Secondaire ;
  - Evacuation de la Cuve Principale au Vidage ;
  - Evacuation de la Cuve Secondaire ;
  - Vidange CA ;
- MO pour la maintenance du CISPI :
  - Rinçage de la Cuve Adjuvant ;
  - Vidange de la Pompe Principale ;
  - Vidange de la Pompe Secondaire.

Les MO sont organisés sur les colonnes du Tableau B.1. Sur les lignes nous avons mis les actionneurs de la plateforme. A l'intersection des lignes avec les colonnes nous avons les actions exécutes dans le MO sur l'actionneur concerné.

Pour chaque MO, chaque action est numérotée pour suivre l'ordre d'exécution. Aussi, la couleur de l'action représente l'action à exécuter :

- En vert : une action de fermeture de l'actionneur ;
- En rouge : une action d'ouverture de l'actionneur ;
- En jaune : une mise en mode automatique contrôlé par un automate programmable.

Tableau B.1 : Les modes opératoires pour la plateforme CISPI

	Remplissage CP	Remplissage CS	Rinçage CA	Remplissage CA	Evacuation CP/CS	Evacuation CP/VID	Evacuation CS	Vidange PP	Vidange PS	Ajout adjuvant	Vidange CA	Brassage	Débiter 422VR	Débiter 432VR
101VM	10	10	7	7										
103VE	11	11	8	9										
104VM	9	9	6	6										
111VE	12	1	4	4	5	5				4		1	1	1
121VE	1	12	5	5			1							
131VM	8	8	9	3						1	1			
211VE	2	2			7	8				5		3	3	3
212VV					6	6	2							
213VM						7								
221VM		5					6							
222VE							7							
223VM								4					5	5
311VM	7	7	3	8						2	2			
321VM	6		1	1	4	4				9	3	2	2	2
332VM			2	2						3	4			
411VM	5				1	1			2	8		8	4	4
412PS									1			9		
413VM									3			7		
421VM	4	4			2	2	3			6		4	6	
422VR													10	
424VM		6					5	2					7	7
425PP								1					11	11
426VM								3					8	8
428VM													9	9

	Remplissage CP	Remplissage CS	Rinçage CA	Remplissage CA	Evacuation CP /CB	Evacuation CP/VID	Evacuation CS	Vidange PP	Vidange PS	Ajout adjuvant	Vidange CA	Brassage	Débiter 422VR	Débiter 432VR
431VM	3	3			3	3	4			7		5		6
432VR														10
441VM									4			6		

Légende : **Fermer** **Ouvrir** **Auto**

Le Tableau B.2 présente les exclusions entre les MO de la plateforme CISPI (cellules en rouge). En effet, c'est une matrice carrée qui a sur les lignes et colonnes les MO. Etant donné que cette matrice d'exclusion est symétrique par rapport à la diagonale principale, nous n'avons affiché que les exclusions supérieures à la diagonale.

Tableau B.2 : Les exclusions entre les MO de la plateforme CISPI

Remplissage CP	Remplissage CS	Rinçage CA	Remplissage CA	Evacuation CP CB	Evacuation CP/VID	Evacuation CS	Vidange PP	Vidange PS	Ajout adjuvant	Vidange CA	Brassage	Débiter 422VR	Débiter 432VR	
	■	■	■	■	■				■		■	■	■	Remplissage CP
		■	■	■	■	■						■	■	Remplissage CS
			■						■	■				Rinçage CA
									■	■				Remplissage CA
					■	■					■	■	■	Evacuation CP CB
											■	■	■	Evacuation CP/VID
												■	■	Evacuation CS
												■	■	Vidange PP
											■			Vidange PS
										■	■	■	■	Ajout adj
														Vidange CA
												■	■	Brassage
														Débiter 422VR
														Débiter 432VR

Liste des figures

Liste des tableaux

Bibliographie

AADL. (2009). *Architecture Analysis & Design Language.* SAE International,Version 2.0.

AFIS. (2009). *Découvrir et Comprendre l'Ingénierie Système* (éd. 3). Association Française d'Ingénierie Système.

AFIS. (2005). *Modèle de données AFIS.* Association Française d'Ingénierie Système.

Akkaya, K., & Younis, M. (2005). A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks , 3* (3), 325-349.

Albert, V. (2009). *Evaluation de la validité de la simulation dans le cadre du développement des systèmes embarqués.* Thèse de doctorat, Université Paul Sabatier - Toulouse III.

Albus, J. S. (1999). The Engineering of Mind. *Information Sciences , 117* (1-2), 1-18.

Alengry, P. (1985). *RR-0437 - Evaluation d'un dispositif d'assistance a l'opérateur dans le secteur industriel. Ergonomie du dialogue, processus de traitement d'information, organisation socio-technique.* INRIA.

Alexander, C., Ishikawa, S., & Silverstein, M. (1977). *Pattern Language: Towns, Buildings, Construction.* Oxford University Press.

Amalberti, R. (2001). *La Conduite de systèmes à risques* (éd. 2). Presses Universitaires de France.

ANSI/EIA-632. (1999). *Processes for engineering a system.* American National Standards Institute.

ANSI/ISA5.1. (1984). Instrumentation Symbols and Identification.

Auzelle, J.-P. (2009). *Proposition d'un cadre de modélisation multiéchelles d'un système d'information en entreprise centré sur le produit.* Université Henri Poincaré, Nancy 1.

B2MML. (2008). *Business To Manufacturing Markup Language.* World Batch Forum.

Baïna, S. (2006). *INTEROPERABILITE DIRIGEE PAR LES MODELES : Une Approche Orientée Produit pour l'interopérabilité des systèmes d'entreprise.* Thèse de doctorat, Université Henri Poincaré, Nancy 1.

Bajic, E. (2005). Ambient services modelling framework for intelligent products. *Smart Object Systems, UbiComp 2005*, (pp. 83-90). Tokyo, Japan.

Bajic, E., Cea Ramirez, A., & Dobre, D. (2008). Service Modeling for Smart Objects in the Supply Chain using RFID and UPnP Technologies. Dans T. Blecker, & G. Q. Huang (Éds.), *RFID in Operation and Supply Chain Management* (Vol. 6, pp. 91-118). ESV.

Balmelli, L. (2007). An Overview of the Systems Modeling Language for Products and Systems Development. *Journal of Object Technology , 6* (6), 149-177.

Baptiste, P., Bernard, A., Bourrières, J.-P., Lopez, P., Morel, G., Pierreval, H., et al. (2007). Comité d'Experts Productique : prospectives de recherche.

Beeck, M. v. (2002). A structured operational semantics for UML-statecharts. *Software and Systems Modeling , 1* (2), 130-141.

Belhimeur, A. (1989). *Contribution à l'étude d'une méthode de conception des automatismes des systémes de conduite des processus industrels.* Thèse de doctorat, Université des Sciences et Techniques de Lille, Flandres Artois.

Belkadi, F., Bonjour, E., & Dulmet, M. (2007). Competency characterisation by means of work situation modelling. *Computers in Industry , 58* (2), 164-178.

Berruet, P., Lallican, J., Rossi, A., & Philippe, J. (2005). A component based approach for the design of FMS control and supervision. *IEEE International Conference on Systems, Man and Cybernetics*, *4*, pp. 3005-3011.

Bjelkemyr, M., & Lindberg, B. (2007). An engineering systems perspective on system of systems methodology. *Proceeding of 1st Annual 2007 IEEE Systems Conference*, (pp. 1-7). Hawaii, USA.

Bjørner, D. (2009). *From Domains to Requirements: The Triptych Approach to Software Engineering.*

Bjørner, D., & Henson, M. C. (Éds.). (2008). *Logics of Specification Languages.* Springer Berlin Heidelberg.

Blaise, J.-C. (2000). *Apport d'une modélisation de l'information normative à l'intégration des règles de sécurité des machines en conception.* Thèse de doctorat, Université Henri Poincaré, Nancy 1.

Bondé, L. (2006). *Transformations de Modèles et Interopérabilité dans la Conception de Systèmes Hétérogènes sur Puce à Base d'IP.* Thèse de doctorat, Université des Sciences et Technologies de Lille.

Bonhomme, S. (2008). *Méthodologie et outils pour la conception d'un habitat intelligent.* Institut National Polytechnique de Toulouse - INPT.

Bouhours, C. (2010). *Détection, Explications et Restructuration de défauts de conception : les patrons abîmés.* Thèse de doctorat, Université de Toulouse - Toulouse III, Institut de Recherche en Informatique de Toulouse – IRIT UMR 5505 CNRS, Toulouse.

Boy, G. (1988). *Assistance à l'opérateur: une approche de l'intelligence artificielle.* Teknea.

Bruns, F. W. (2006). Ubiquitous computing and interaction. *Annual Reviews in Control , 30* (2), 205-213.

Cahour, B., & Falzon, P. (1991). Assistance à l'opérateur et modélisation de sa compétence. *Intellectica , 2* (12), 159-186.

Canals, A., & Roussel, A. (2010). Mise en Place d'un processus « SysML-CS ». *8ième journée thématique AFIS: « Modélisation Système : SysML Langage universel ? ».* Paris.

Caron, F. (2005). La gestion collaborative des données d'ingénierie système. *Génie logiciel , 75*, 2-6.

Caron, F., Penas, O., & M'Henni, F. (2010). De la modélisation système à la simulation métier: SysML et le pré-dimensionnement dans des projets "Mécatroniques". *8ième journée thématique AFIS: « Modélisation Système : SysML Langage universel ? ».* Paris.

Carvalho, P. V., Santos, I. L., Gomes, J. O., & Borges, M. R. (2008). Micro incident analysis framework to assess safety and resilience in the operation of safe critical systems: A case study in a nuclear power plant. *Journal of Loss Prevention in the Process Industries , 21* (3), 277-286.

Casella, F., Otter, M., Proelss, K., Richter, C., & Tummescheit, H. (2006). The Modelica Fluid and Media library for modeling of incompressible and compressible thermo-fluid pipe networks. *Modelica 2006* (pp. 631-641). The Modelica Association.

Cea Ramirez, A. (2006). *Contribution à la Modélisation et à la Gestion des Interactions Produit-Processus dans la Chaîne Logistique par l'Approche Produits Communicants.* Thèse de doctorat, Université Henri Poincaré - Nancy I.

Cea Ramirez, A., Dobre, D., & Bajic, E. (2006). Ambient services interactions for Smart Objects in the supply chain.

Chauvet, G. A. (1993). Hierarchical functional organization of formal biological systems: A dynamical approach. *Philosophical Transaction in Biological Sciences , 339* (1290), 425-481.

Cloutier, R. J. (2006). *Applicability of Patterns to Architecting Complex Systems.* Thèse de doctorat, Stevens Institute of Technology, Hoboken, NJ .

Cook, D. J., Augusto, J. C., & Jakkula, V. R. (2009). Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing , 5* (4), 277-298.

Czarnecki, K., & Helsen, S. (2006). Feature-Based Survey of Model Transformation Approaches. *IBM Systems Journal, special issue on Model-Driven Software Development , 45* (3), 621-645.

Dang, T., Devic, C., & al. (2008). OCARI: Optimization of Communication for Ad hoc Reliable Industrial networks. *Computational Methods in Applied Mechanics and Engineering*, (pp. 688 - 693).

de Bruijn, H., & Herder, P. M. (2009). System and Actor Perspectives on Sociotechnical Systems. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans , 39* (5), 981-992.

Décret 2003-296. (2003). Décret no2003-296 du 31 mars 2003 relatif à la protection des travailleurs contre les dangers des rayonnements ionisants.

Dick, J., Fanmuy, G., & Thevenet, L. H. (2006). A Requirements Guide For All (REGAL): An INCOSE Initiative. *14th IEEE International Conference on Requirements Engineering.* Mineapolis, USA.

Dionis, F., & Pirus, D. (2007). Ways to improve field operation on NPPs facilities by bringing additional operator aids. *IEEE 8th Human Factors and Power Plants and HPRCT 13th Annual Meeting*, (pp. 42-44). Monterey, CA, USA.

Dobre, D., & Bajic, E. (2008a). Active product modeling for chemical security management based on smart object concept. *7ème Conférence Internationale de Modélisation, Optimisation et Simulation des Systèmes (MOSIM'08).* Paris, France.

Dobre, D., & Bajic, E. (2007b). Produit Actif : Concept et implémentation. *Présentation au Groupe de recherche du CNRS : GdR MACS.* Marseille, France.

Dobre, D., & Bajic, E. (2007a). Smart object design for active security management of hazardous products. *1st International Workshop on Design and Integration Principles for Smart Objects (DIPSO 2007).* Innsbruck, Autriche.

Dobre, D., Bajic, E., & Zouinkhi, A. (2009). Active product modeling based on smart object concept: Application to chemical security management. *Journal européen des systèmes automatisés , 43* (4-5), 561-579.

Dobre, D., Morel, G., & Gouyon, D. (2010). Improving Human-System Digital Interaction for Industrial System Control: Some Systems Engineering Issues. *10th IFAC Workshop on Intelligent Manufacturing Systems (IMS'10).* Lisbonne, Portugal.

Dobre, D., Morel, G., Petin, J.-P., & Bajic, E. (2008b). Improving digital interaction for operator-driven process-plant operation. *9th IFAC Workshop on Intelligent Manufacturing Systems (IMS'08).* Szczecin, Pologne.

Dounis, A., & Caraiscos, C. (2009). Advanced control systems engineering for energy and comfort management in a building environment - A review. *Renewable and Sustainable Energy Reviews , 13* (6-7), 1246-1261.

Dymola. (2010). Récupéré sur Dymola: Multi-Engineering Modeling and Simulation: http://www.3ds.com/products/catia/portfolio/dymola

EDF. (1992). *Recueil des prescriptions au personnel. Activités sur les ouvrages du Service de la Production Thermique.*

Endsley, M. R. (1995). Toward a Theory of Situation Awareness in Dynamic Systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society , 37* (1), 32-64.

Estefan, J. A. (2008). *Survey of Model-Based Systems Engineering (MBSE) Methodologies.* Version B, INCOSE.

Faisandier, A. (2010). *Formation enseignant-chercheur en ingénierie des systèmes.* Toulouse: MAP Système.

Faisandier, A. (2008). *Sensibilisation à l'Ingénierie des Systèmes.* Toulouse: MAP Système.

Fass, D., & Lieber, R. (2009). Rationale for human modelling in human in the loop systems design. *3rd Annual IEEE International Systems Conference, SysCon 2009.* Vancouver, Canada.

Feliot, C. (1997). *Modélisation de systèmes complexes : intégration et formalisation de modèles.* Thèse de doctorat, Université de Lille I.

Feliot, C. (2007). Toward a formal theory of systems. *Colloque d'Automne du LIX 2007 (CAL07), Complex Industrial Systems: Modelling, Verification and Optimization.* Paris.

FIPA. (2010). Récupéré sur Foundation for Intelligent Physical Agents: http://www.fipa.org

Forsberg, K., Mooz, H., & Cotterman, H. (2005). *Visualizing Project Management: Models and Frameworks for Mastering Complex Systems* (éd. 3). Wiley.

Friedenthal, S., & Burkhart, R. (2003). Extending UML From Software To Systems. *Proceedings Of The INCOSE 2003, International Symposium.*

FUML. (2009). *Semantics of a Foundational Subset for Executable UML Models.* Object Management Group, Version 1.0 Beta 2.

Fusaoka, A., Seki, H., & Takahashi, K. (1983). A Description and Reasoning of Plant Controllers in Temporal Logic. *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, *1*, pp. 405-408. Karlsruhe, West Germany.

Galara, D. (1986). *Elaboration d'Informations crédibles pour les Systèmes de Contrôle/Commande Industriels: les Modules Fonctionnels d'Automatismes.* Rapport Interne, EDF / DER, CHATOU.

Galara, D. (2010). L'humain : un composant cognitif pour exploiter un système technique ? *4ème Forum Académie - Industrie de l'AFIS (AFIS'10).* Bordeaux Mérignac.

Galara, D. (2006). Roadmap to master the complexity of process operation to help operators improve safety, productivity and reduce environmental impact. *Annual Reviews in Control , 30* (2), 215-222.

Galara, D., & Hennebicq, J. P. (1999). Process Control Engineering Trends. *Annual Reviews in Control , 23*, 1-11.

Galara, D., & Pirus, D. (2007). Finding the way up to the standardization of Human Machine Interface. *IEEE 8th Human Factors and Power Plants and HPRCT 13th Annual Meeting*, (pp. 133-139). Monterey, CA, USA.

Galara, D., Morel, G., Pétin, J., & Méry, D. (2008). Vers un langage d'expression des besoins en informations pour les métiers d'exploitation. *1er Workshop du Groupement d'Intérêt Scientifique "Surveillance, Sûreté, Sécurité des Grands Systèmes" (3SGS'08).* Troyes, France.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley Professional.

Gauthereau, V., & Hollnagel, E. (2005). Planning, Control, and Adaptation: A Case Study. *European Management Journal , 23* (1), 118-131.

Gawron, V. J. (2008). *Human performance, workload, and situational awareness measures handbook* (éd. 2). CRC.

Gershenfeld, N. (1999). *When Things Start to Think.* Henry Holt & Company.

Giese, H., Hildebrandt, S., & Neumann, S. (2009). Towards Integrating SysML and AUTOSAR Modeling via Bidirectional Model Synchronization. *Model-based Development of Embedded Systems.*

Gouyon, D., & David, M. (2009). Vers l'implantation d'un système contrôlé par le produit à l'aide de réseaux de capteurs sans fil. *Journal Européen des Systèmes Automatisés , 43* (4-5), 537-560.

Grout, M., & Salaün, P. (2009). *Instrumentation industrielle: spécification et installation des capteurs et vannes de régulation* (éd. 2). (L. N. Dunod, Éd.)

Grundstein, M., Rosenthal Sabroux, C., & Pachulski, A. (2003). Reinforcing decision aid by capitalizing on company's knowledge: Future prospects. *European Journal of Operational Research , 145* (2), 256-272.

Gunter, C. A., Gunter, E. L., Jackson, M., & Zave, P. (2000). A Reference Model for Requirements and Specifications. *IEEE Software , 17* (3), 37-43.

Habib, G., Marangé, P., Pétin, J.-F., & Divoux, T. (2009). Evaluation de l'influence d'un réseau de communication sans fil sur la commande d'un SED. *Journal Européen des Systèmes Automatisés , 43* (7-9), 855-870.

Hall, J. G., & Rapanotti, L. (2005). Problem Frames for Socio-Technical Systems. Dans A. Silva, & J. L. Maté (Éds.), *Requirements Engineering for Socio-Technical Systems* (pp. 318-339). Idea Publishing Group.

Halpin, T. (1998). Object-Role Modeling (ORM/NIAM). Dans P. Bernus, K. Mertins, & G. Schmidt (Éds.), *Handbook on Architectures of Information Systems* (pp. 81-104). Berlin: Springer-Verlag.

Hoffmann, H.-P. (2004). *UML 2.0-Based Systems Engineering Using a Model-Driven Development Approach.* White Paper, I-Logix, Andover, MA.

Holt, J., & Perry, S. (2008). *SysML for Systems Engineering.* London, United Kingdom: The Institution of Engineering and Technology.

IEEE 1220. (2005). *IEEE Standard for Application and Management of the Systems Engineering Process.* Institute of Electrical and Electronics Engineers.

IEEE 802.15.4. (2003). *Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs).* The Institute of Electrical and Electronics Engineers.

IEEE 802.15.4. (2006). *Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs).* The Institute of Electrical and Electronics Engineers.

INCOSE. (2010). *Systems Engineering Handbook : a guide for system life cycle processes and activities* (éd. 3.2). International Council on Systems Engineering.

Isermann, R. (2005). Mechatronic Design Approach. Dans R. H. Bishop (Éd.), *MECHATRONICS: AN INTRODUCTION.* CRC Press.

ISO 10303-233. (s.d.). Industrial automation systems and integration -- Product data representation and exchange -- Part 233: Systems engineering data representation.

ISO 10628. (2002). Schémas de procédé pour les unités de fabrication de production - Régles générales. *Organisation internationale de normalisation* .

ISO/IEC 15288. (2008). *Systems and software engineering - System life cycle processes.* International Organisation for Standardization.

ISO/IEC 42010. (2007). *Recommended Practice for Architectural Description of Software-Intensive Systems.* International Organization for Standardization.

ISO/IEC 62264. (2001). *Enterprise-control system integration.* International Organization for Standardization.

Iung, B. (1992). *Contribution à une intelligence distribuée dans les equipements de niveau zero des processus industriels complexes.* Thèse de doctorat, Université Henri Poincaré - Nancy 1.

Jackson, M. (2000). *Problem Frames: Analysing & Structuring Software Development Problems.* Addison-Wesley Professional.

Jackson, M. (1997). The meaning of requirements. *Annals of Software Engineering , 3* (1), 5-21.

Jackson, M., & Zave, P. (1995). Deriving Specifications from Requirements: An Example. *Proceedings of the 17th International Conference on Software Engineering* (pp. 15-24). Seattle, Washington, USA: ACM and IEEE CS Press.

Jarraya, Y., Debbabi, M., & Bentahar, J. (2009). On the Meaning of SysML Activity Diagrams. *16th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2009)*, (pp. 95 - 105). San Francisco, CA.

Johnson, T. (2008). *Integrating Models and Simulations of Continuous Dynamic System Behavior into SysML.* Rapport de Master, Georgia Institute of Technology.

Kieras, D. E., & Meyer, D. E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction , 12* (4), 391-438.

Kim, M. C., Park, J., & Jung, W. (2008). Sentence completeness analysis for improving team communications of safety-critical system operators. *Journal of Loss Prevention in the Process Industries , 21* (3), 255-259.

Kintzig, G., Poulain, G., Privat, G., & Favennec, P. (2002). *Objets communicants.* Hermès Science Publications.

Koestler, A. (1967). *The Ghost in the Machine.* London: Arkana.

Kou, D., Zhao, K., Tao, Y., & Kou, W. (2006). RFID Technologies and Applications. Dans W. Kou, & Y. Yesha (Éds.), *Enabling Technologies for Wireless E-Business* (pp. 89-108).

Kuras, M. L. (2006). *A Multi Scale Definition of a System.* Technical Report, MITRE .

Lamboley, P. (2001). *Proposition d'une méthode formelle d'automatisation de systèmes de production à l'aide de la méthode B.* Thèse de doctorat, Université Henri Poincaré, Nancy I, Centre de Recherche en Automatique de Nancy.

Lavigne, J.-P., Mayer, F., & Lhoste, P. (2003). Category Theory Based Approach for IMS Modelling. *IMS.* Budapest, Hungary.

Le Put, A., & Meinadier, J.-P. (2009). Référentiel Ingénierie Système AFIS. *5ème Conférence Annuelle d'Ingénierie Système (AFIS 2009).* Paris.

Léger, A. (2009). *Contribution à la formalisation unifiée des connaissances fonctionnelles et organisationnelles d'un système industriel en vue d'une évaluation quantitative des risques et de l'impact des barrières envisagées.* Thèse de doctorat, Université Henri Poincaré, Nancy 1.

Leger, J.-B. (1999). *Contribution méthodologique à la maintenance prévisionnelle des systèmes industriels de production : proposition d'un cadre formel de modélisation.* Thèse de doctorat, l'Université Henri Poincaré – Nancy 1.

Lhoste, P. (1994). *Contribution au Génie Automatique: concepts, modèles, méthodes et outils pour le Génie Automatique.* Habilitation à Diriger des Recherches, Université Henri Poincaré, Nancy I.

Liu, Y. (2002). *Contribution à l'automatisation des systèmes intégrés et à intelligence distribuée de Contrôle, Maintenance et Gestion Technique (ICMMS) pour les centrales hydroélectriques.* Thèse de doctorat, Université Henri Poincaré - Nancy 1, Université de HUST, Pékin.

Luzeaux, D. (2009). Vers une nouvelle théorie abstraite des systèmes en vue de leur ingénierie. *5ème Conférence Annuelle d'Ingénierie Système - AFIS 2009.* Paris, France.

Luzeaux, D., & Ruault, J. (2008). *Ingénierie des systèmes de systèmes - concepts et illustrations pratiques.* Hermes Science.

Maier, M. W. (1998). Architecting principles for systems-of-systems. *Systems Engineering , 1* (4), 267–284.

Marangé, P. (2008). *Synthése et filtrage robuste de la commande pour des systémes manufacturiers sûrs de fonctionnement.* Thèse de doctorat, Université de Reims Champagne-Ardenne.

MARTE. (2009). *A UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded systems.* Object Management Group, Version 1.0.

Mayer, F. (1995). *Contribution to manufacturing engineering: Application to pedagogical engineering within a CIME centre.* Thèse de doctorat, Université Henri Poincaré.

McFarlane, D., Sarma, S., Lung, C. J., Wong, C., & Ashton, K. (2002). The intelligent product in manufacturing control and management. *15th Triennial IFAC World Congress.* Barcelona, Spain.

Meinadier, J.-P. (1998). *Ingénierie et intégration des systèmes.* Paris: Hermes Sciences Publications.

Meinadier, J.-P. (2002). *Le métier d'intégration de systèmes.* Paris: Hermes Science Publications.

Modelica. (2010). *Modelica – a unified object-oriented language for physical systems modeling. Language specification,.* Version 3.2, The Modelica Association.

Morel, G., Panetto, H., Mayer, F., & Auzelle, J.-P. (2007). System of Enterprise-Systems Integration Issues: an Engineering Perspective. *IFAC Conference on Cost Effective Automation in Networked Product Development and Manufacturing (IFAC-CEA'07).* Monterrey, Mexico.

NASA. (2007). *NASA Systems Engineering Handbook.* Washington, D.C.: National Aeronautics and Space Administration.

Neunreuther, E. (1998). *Contribution à la modélisation des Systémes Intégrés de Production à Intelligence Distribuée: application à la distribution du contrôle et de la gestion technique sur les équipements de terrain.* Thèse de doctorat, Université Henri Poincaré, Nancy 1.

NPR-7123.1A. (2009). NASA Systems Engineering Processes and Requirements. National Aeronautics and Space Administration.

O'Connor, P., O'Dea, A., Flin, R., & Belton, S. (2008). Identifying the team skills required by nuclear power plant operations personnel. *International Journal of Industrial Ergonomics , 38* (11-12), 1028-1037.

O'Hara, J. M., Pirus, D., & Beltratcchi, L. (2004). Information Display: Considerations For Designing Computer-Based Display Systems. *4th International Topical Meeting on Nuclear Plant Instrumentation Control and Human Machine Interface Technologies.* Columbus, Ohio: Brookhaven National Laboratory.

O'Hara, J., Brown, W., Lewis, P., & Persensky, J. (2002). High-Level Human-System Interface Design Review Principles. *US Nuclear Regulatory Commission* .

Panetto, H., Morel, G., & Dobre, D. (2010). SysML : Langage de spécification en Ingénierie Système ? *8ème Journée thématique AFIS sur Modélisation Système : SysML langage universel? Retours d'expérience.* Paris.

Pannequin, R. (2007). *Proposition d'un environnement de modélisation et de test d'architectures de pilotage par le produit de systèmes de production.* Thèse de doctorat, Université Henri Poincaré, Nancy 1.

Paredis, C., Fritzson, P., & Peak, R. (2008). SysML and Modelica : Opportunities for Synergy. *OMG Santa Clara Meeting* . Santa Clara, CA, USA.

Peak, R., Burkhart, R., Friedenthal, S., Wilson, M., Bajaj, M., & Kim, I. (2007). Simulation-Based Design Using SysML - Part 1: A Parametrics Primer, Part 2: Celebrating Diversity by Example. *INCOSE Intl. Symposium.* San Diego.

Penalva, J.-M. (1997). *La modélisation par les systèmes en situation complexe.* Thèse de doctorat, Université de Paris Sud.

Perronne, J.-M., Thiry, L., & Thirion, B. (2006). Architectural concepts and Design Patterns for behavior modeling and integration. *Mathematics and Computers in Simulation , 70* (5-6), 314-329.

Pétin, J.-F. (1995). *Contribution méthodologique à l'Actionnement et la Mesure Intelligents: Application au projet ESPRIT III - P.R.I.A.M. n° 6188.* Thèse de doctorat, Université Henri Poincaré - Nancy 1.

Pétin, J.-F., Iung, B., & Morel, G. (1998). Distributed intelligent actuation and measurement (IAM) system within an integrated shop-floor organisation. *Computers in Industry , 37* (3), 197-211.

Pétin, J.-F., Morel, G., & Panetto, H. (2006). Formal specification method for systems automation. *European Journal of Control , 12* (2), 115-130.

Philips. (2005). *I-CODE SLI : Standard Label IC SL2 ICS20.* Functional Specification. v3.1.

Pirus, D. (2006). Field Operation Computerization Study. *5th ANS meeting on NPIC and HMIT.* Albuquerque, New Mexico.

Pop, A. (2008). *Integrated Model-Driven Development Environments for Equation-Based Object-Oriented Languages.* Thèse de doctorat, Linköping University, Department of Computer and Information Science.

Prevostini, M., & Zamsa, E. (2007). *SysML Profile for SoC Design and SystemC Transformation.* University of Lugano.

Prioul, J.-M. (2009). *Rapport d'activité sur la conception de la partie commande de la plate-forme CISPI.* Rapport de stage 2ème année, Université Henri Poincaré, Nancy 1, ESIAL.

Rasmussen, J. (1983). Skills, rules and knowledge: signals, signs and symbols and other distinctions in human performance models. *IEEE transactions on systems, man, and cybernetics , 13* (3), 257-266.

Reason, J. (2000). Human error: models and management. *British Medical Journal , 320* (7237), 768-770.

Reiman, T. (2007). *Assessing Organizational Culture in Complex Sociotechnical Systems. Methodological Evidence from Studies in Nuclear Power Plant Maintenance Organizations.* Thèse de doctorat, University of Helsinki.

Robertson, S., & Robertson, J. (2006). *Mastering the Requirements Process* (éd. Second Edition). Addison Wesley Professional.

Saint-Germain, B. (2010). *Distributed coordination and control for networked production systems.* Thèse de doctorat, Katholieke Universiteit, Leuven.

Santos, A. I., Grecco, C. H., Mol, A. C., & Carvalho, P. V. (2008). The use of questionnaire and virtual reality in the verification of the human factors issues in the design of nuclear control desk. *International Journal of Industrial Ergonomics , 39* (1), 1-8.

Santos, M. d., & Vrancken, J. (2009). Including SysML in the 4+1 View Model of Architecture for Software-Intensive Systems. *7th Annual Conference on Systems Engineering Research (CSER 2009).* Loughborough University, UK.

Seater, R. M. (2009). *Building Dependability Arguments for Software Intensive Systems.* Thèse de doctorat, Massachusetts Institute of Technology.

Seidner, C. (2009). *Vérification des EFFBDs : Model-checking en Ingénierie Système.* Thèse de doctorat, Université de Nantes.

Sendall, S., & Kozaczynski, W. (2003). Model Transformation: The Heart and Soul of Model-Driven Software Development. *IEEE Software , 20* (5), 42-45.

Shah, A. A., Schaefer, D., & Paredis, C. J. (2009). Enabling Multi-View Modeling With SysML Profiles and Model Transformations. *International Conference on Product Lifecycle Management.* Bath, UK.

SysML. (2008). *Systems Modeling Language.* Object Management Group, Version 1.1.

SysML-Modelica_Integration. (2010). *SysML-Modelica Transformation Specification .* Object Management Group, Version 1.0.

Topcased. (2010). Récupéré sur TOPCASED, The Open-Source Toolkit for Critical Systems: http://www.topcased.org/

Turki, S. (2008). *Ingénierie système guidée par les modèles: Application du standard IEEE 15288, de l'architecture MDA et du langage SysML à la conception des systèmes mécatroniques.* Thèse de doctorat, Université du Sud Toulon-Var.

van Lamsweerde, A. (2001). Goal-oriented requirements engineering: A guided tour. *Proceedings of the 5th IEEE International Symposium on Requirements Engineering (RE2001)*, (pp. 249-263). Toronto.

van Lamsweerde, A., & Letier, E. (1998a). Integrating obstacles in goal-driven requirements engineering. *Proc. 20th International Conference on Software Engineering*, (pp. 53-62). Redwood City, San Francisco Bay, USA.

van Lamsweerde, A., Darimont, R., & Letier, E. (1998b). Managing conflicts in goal-driven requirements engineering. *IEEE Transactions on Software Engineering , 24* (11), 908-926.

Vicente, K. J., & Rasmussen, J. (1992). Ecological Interface Design: Theoretical Foundations. *IEEE Transactions on Systems, Man, and Cybernetics , 22* (4), 589-606.

Vogel, C. (1988). *Génie cognitif.* Masson.

Wang, H., & Wang, C. (1997). Intelligent agents in the nuclear industry. *Computer , 30* (11), 28-31.

Wang, R., & Dagli, C. (2008). An Executable System Architecture Approach to Discrete Events System Modeling Using SysML in Conjunction with Colored Petri Net. *2nd Annual IEEE Systems Conference*, (pp. 1-8). Montreal, Quebec, Canada.

Weilkiens, T. (2008). *Systems Engineering with SysML/UML: Modeling, Analysis, Design.* Morgan Kaufmann.

Willard, B. (2007). UML for systems engineering. *Computer Standards & Interfaces , 29* (1), 69–81.

Wippler, J.-L. (2010). Les Architectes Systèmes ont-ils les moyens de s'offrir le luxe de SysML ? *8ième journée thématique AFIS: « Modélisation Système : SysML Langage universel ? ».* Paris.

Wong, C., McFarlane, D., Zaharudin, A., & Agarwal, V. (2002). The intelligent product driven supply chain. *IEEE SMC.* Hammamet.

Résumé

Les travaux présentés dans ce mémoire s'inscrivent dans le contexte de l'Ingénierie d'un Système Interactif d'Aide à la Conduite (SIAC) d'un procédé industriel. Nous défendons l'intérêt d'améliorer l'interactivité numérique entre un procédé et un agent (opérateur de conduite, rondier) qui applique des procédures de conduite.

L'originalité du SIAC consiste à encapsuler le système physique par un canal d'objets logiques afin de mieux équilibrer la distribution des rôles entre l'humain et le système technique qu'il conduit. Ce SIAC fournit aux opérateurs et aux rondiers des services d'aide à la conduite tels que la localisation des équipements, l'autorisation des actions à exécuter et la validation des actions exécutées, de même que la gestion des contraintes d'exclusion et de dépendances entre actions, imposées par la physique du procédé.

La spécification de ce système sociotechnique interprète les travaux des « Problem Frames » en génie informatique pour proposer un procédé de modélisation itératif dans le cadre d'une Ingénierie Système Basée sur des Modèles (ISBM). Cette ISBM s'appuie sur le langage SysML, dont la syntaxe et la sémantique sont spécialisées pour supporter le procédé de modélisation proposé. Cette spécialisation met en correspondance les artéfacts clés extraits à la fois des bonnes pratiques de l'Ingénierie Système et des bonnes pratiques de SysML.

Mots-clés

Ingénierie Système, SysML, Spécification, Interaction Numérique, Aide à la Conduite

Abstract

Within the context of the Engineering of an Interactive Aiding System for industrial process Control (SIAC), these works aims to improve the digital interaction between a process and a human operator (control room operator, field operator) which applies control procedures.

The proposed SIAC enhances the digital capabilities of the field operator in order to better balance the role distribution between the technical system and the human system. It provides active control services to field operators such as equipment localisation, action authorization and action validation, as well as management services for the verification of exclusions between procedures constraints and verification of dependencies between actions constraints.

The specification of such socio-technical system interprets the "Problem Frames" from software engineering in order to propose an iterative Model Based System Engineering (MBSE) process. This MBSE is based on SysML modelling language, whose syntax and semantics are specialized to support the proposed methodology. This specialization maps the key artefacts that are extracted from the System Engineering and SysML good practices.

Keywords

Systems Engineering, SysML, Specification, Digital Interaction, Aided Control