Génération conjointe de commandes et d'interfaces de supervision pour systèmes sociotechniques reconfigurables

Alain Bignon

THÈSE / UNIVERSITÉ DE BRETAGNE-SUD
UFR Sciences et Sciences de l'Ingénieur
Sous le sceau de l'Université européenne de Bretagne

Pour obtenir le grade de
DOCTEUR DE L'UNIVERSITÉ DE BRETAGNE-SUD
Mention : STIC

Présentée par

Alain Bignon

Préparée au Lab-STICC (UMR 6285)
Université de Bretagne-Sud

Génération conjointe de commandes et d'interfaces de supervision pour systèmes sociotechniques reconfigurables

Thèse soutenue le 10 juillet 2012
devant le jury composé de :

Nidhal REZG
Professeur / Université Paul Verlaine de Metz / *président*

Armand TOGUYENI
Professeur / École centrale de Lille / *rapporteur*

Jean Jacques LOISEAU
Directeur de recherche CNRS / IRCCyN / *rapporteur*

Khalid KOUISS
Maître de conférences / IFMA / *examinateur*

Pierre DENNEULIN
Directeur de pôle / SEGULA / *invité*

Pascal BERRUET
Professeur / Université de Bretagne-Sud / Directeur de thèse

André ROSSI
Maître de conférences / Université de Bretagne-Sud / Co-directeur de thèse

À ma Grand-mère.

Remerciements

Je souhaite exprimer ici toute ma gratitude envers celles et ceux qui ont contribué, de près ou de loin, à ce que ces travaux soient réalisés.

Je remercie tout d'abord Pascal Berruet et André Rossi d'avoir accepté d'encadrer scientifiquement ce projet. Ils ont été là avant même le début, pour construire avec moi ce sujet. Ils ont su, m'apporter rigueur et ouverture pour ces travaux.

Je remercie également messieurs Armand Toguyeni et Jean-Jacques Loiseau pour avoir accepté de relire ces travaux ainsi que messieurs Nidhal Rezg et Khalid Kouiss pour avoir accepté de les examiner.

Je remercie tout le personnel du **Lab-STICC** de Lorient, doctorants, chercheurs, personnels administratifs et techniques qui m'ont accueilli pendant la moitié de la durée de cette thèse. La bonne humeur était souvent au rendez-vous des pauses et je remercie tout particulièrement Florence et Virginie pour cela.

Cette thèse n'aurait pu avoir lieu sans le concours de **SEGULA Technologies** et tout particulièrement sans l'accord et l'appui de Dominique Burgot, Emmanuel Huser et Pierre Denneulin. Qu'ils en soient pour cela vivement remerciés.

Avec eux, je pense également à tout le personnel de l'agence de Lorient (passé, présent et futur) qui sait si bien vous faire redescendre sur terre quand on s'en éloigne un peu. Un merci tout particulier à Isabelle pour son aide et son soutien. Et un autre à Djamal et Innocent à qui je dois énormément pour la concrétisation de ces travaux. Je remercie également tous ceux qui ont participé à l'enquête.

Comme la recherche ne s'arrête pas à la porte du travail, j'en profite pour remercier tous mes amis avec qui il est si facile de décrocher même si certains d'entre eux ont vécu la même expérience. Merci à Didine, Johann, JM, Soraja, JB, Marie, Julien et Lena pour tous les moments, plus rares aujourd'hui, passés ensemble.

J'aimerais également remercier mes parents et ma famille qui à l'heure actuelle ne réalise pas encore tout à fait ce que je fais, mais qui n'ont jamais douté. Ou alors si peu !

Merci également à Lena qui a su me trouver et me garder à un moment qui n'est pas le plus facile pour cela. Avec elle, on n'a pas une minute pour s'ennuyer et ça fait du bien quand on veut « sortir la tête du guidon ! »

Et enfin, un grand merci à tous les oubliés, ceux que j'ai croisés durant cette période si particulière qu'est la thèse et avec qui les rapports furent aussi divers qu'enrichissants.

Cette thèse est le fruit d'une collaboration entre l'entreprise SEGULA Technologies, acteur majeur dans les métiers de l'ingénierie et le Lab-STICC, pôle de référence en recherche sur les systèmes communicants.

SEGULA Technologies
B.P. 50256
56602 Lanester CEDEX

www.segula.fr

Laboratoire des Sciences et Techniques de l'Information, de la Communication et de la Connaissance (Lab-STICC) - UMR n° 6285
Université de Bretagne-Sud
Centre de recherche Christiaan Huygens, BP 92116
56321 Lorient cedex, France

www.lab-sticc.fr

"Rien ne naît ni ne périt, mais des choses déjà existantes se combinent, puis se séparent de nouveau."

Anaxagore (500 – 428 av. J.-C.)

Sommaire

Liste des figures

Liste des définitions

Liste des abréviations

ACIM	:	Automation Component Implementation Model (concept du projet MEDEIA)
AFNOR	:	Association Française de NORmalisation
ATEX	:	ATmosphère EXplosive (réglementation)
ATL	:	Atlas Transformation Language
Aux	:	Auxiliaires (en références aux systèmes auxiliaires du navire)
BTP	:	Bâtiment, Travaux Publics
BU	:	Business Unit
CAO	:	Conception Assistée par Ordinateur
CIM	:	Computational Independant Model (concept de la recommandation MDA)
COTS	:	Commercial Off-The-Shelf
DN	:	Diamètre Nominal (normalisé suivant ISO 6708:1995)
DSL	:	Domain Specific Language
DSV	:	Domain Specific View (concept du projet MEDEIA)
ECDIS	:	Electronic Charts Display Information System (norme OMI A817(19))
EdS	:	système de production de stockage et de distribution de l'Eau douce Sanitaire (un système auxiliaire du navire)
EMF	:	Eclipse Modeling Framework
ERP	:	Enterprise Resource Planning
FAST	:	Functional Analysis System Technic (normalisé suivant NF EN 1325-1)
FB	:	Function Block
FBD	:	Function Block Diagram (langage de programmation normalisé IEC 61131)
GED	:	Gestion Electronique de Documents
GEMMA	:	Guide d'Etude des Modes de Marche et d'Arrêt (proposé par l'Agence nationale pour le Développement de la Production Automatisé)
HVAC	:	Heat, Ventilation and Air Conditioning
IDE	:	Integrated Devlopment Environnement
IDM	:	Ingénierie Dirigée par les Modèles
IHM	:	Interface Homme Machine
IMCS	:	Integrated Management Control System
IPMS	:	Integrated Platform Management System (système de conduide de la plateforme propulsée du navire)
LMJ	:	Laser Mega Joule
MAC	:	Modèle d'Analyse des Configurations
MDA	:	Model Driven Architecture (recomendation de l'OMG)
MOF	:	Meta-Object Facility (méta-métamodèle défini par l'OMG)
OCL	:	Object Constraint Language (langage proposé par l'OMG)
OMG	:	Object Management Group
OS	:	Operating System
PDM	:	Platform Dependant Model (concept de la recommandation MDA)
PF	:	PlateForme (en référence au domaine de la platforme propulsée du navire)
PGI	:	Progiciel de Gestion Intégré
PID	:	Piping and Instrumentation Diagram (suivant norme ANSI/ISA 5.1)
PIM	:	Platform Independant Model (concept de la recommendation MDA)
PLC	:	Progammable Logic Controller
PLM	:	Product Lifecycle Management
POU	:	Program Organization Unit (concept de l'IEC 61131)

PSM	:	Platform Specific Model (concept de la recommandation MDA)
PSV	:	Platform Specific View (concept du projet MEDEIA)
SADT	:	Structured Analysis and Design Technic
SCADA	:	Supervisory Control and Data Acquisition
SFC	:	Sequential Function Shart (langage de programmation normalisé IEC 61131)
SoS	:	System of Systems
ST	:	Structured Text (langage de programmation normalisé IEC 61131)
TAAF	:	Terres Australes et Antarctiques Françaises
TI	:	schéma de Tuyauterie et Instrumentation
UDFB	:	User Defined Function Block (concept associé au langage FBD de l'IEC 61131)
UML	:	Unified Modeling Language (langage proposé par l'OMG)
URI	:	Uniform Resource Identifier
XML	:	eXtensible Markup Langage

Introduction Générale

Depuis le XIXème siècle et la révolution industrielle, l'amélioration continue des méthodes de fabrication a permis une croissance importante de la productivité (Siegel 2008). Comme le montre la figure 1, l'évolution de la productivité aux États-Unis s'est accentuée au cours du XXème siècle.

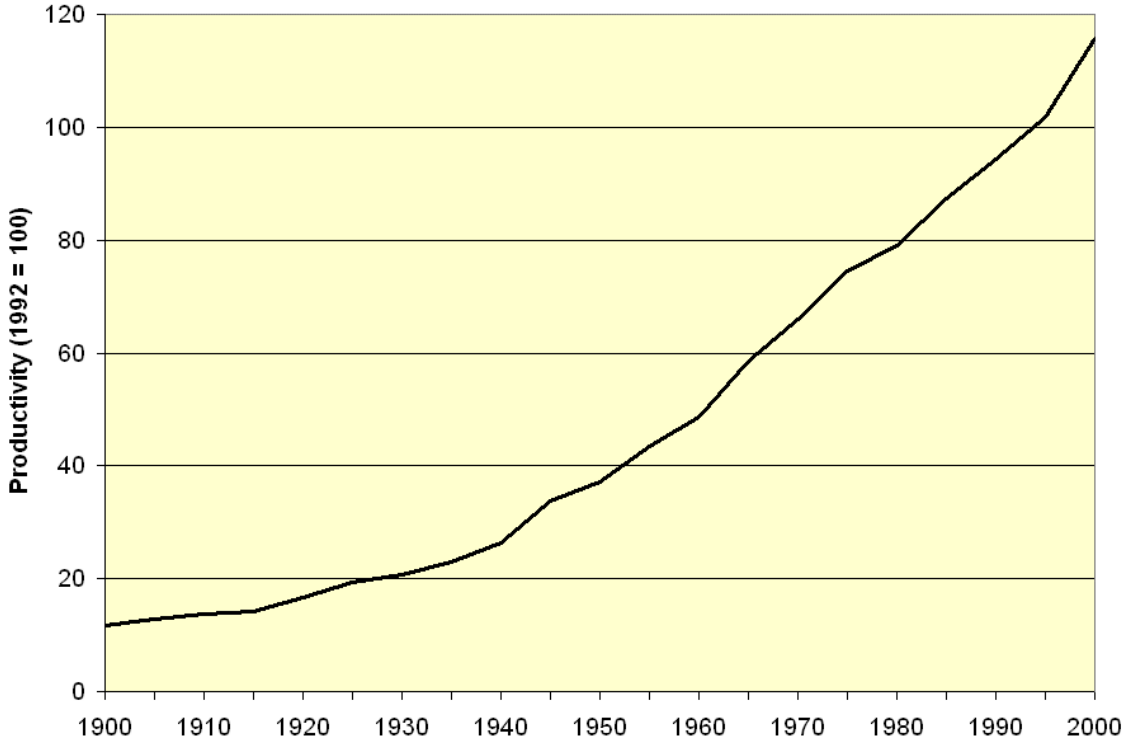


Fig. 1 Productivité américaine (Siegel 2008)

Cependant, tous les secteurs n'ont pas démarré leur progression à la même époque. Le secteur de l'informatique n'a, quant à lui, émergé que dans la deuxième moitié du XXème siècle. Ainsi les activités de développement de logiciel ne sont pas encore rentrées dans une ère industrielle. En comparaison, les méthodes de production du logiciel peuvent être qualifiées d'assez artisanales (Sverdlov, Kopec et al. 2005).

Par ailleurs, l'informatique tend aujourd'hui à devenir ubiquitaire (Berry 2008; Sperandio 2008) aussi bien dans l'environnement privé que dans l'environnement professionnel. Ce besoin continuellement croissant de nouveaux produits, et donc de nouvelles applications, nécessite un grand nombre de programmeurs qui doivent, de surcroit, travailler de plus en plus avec des experts d'autres domaines, au sein d'équipes de travail hétérogènes.

Des méthodes de développement plutôt artisanales et un effort de production important entrainent une qualité de la production dégradée, avec de nombreux bugs. Pour s'en convaincre, il suffit de noter la fréquence des mises à jour pour les logiciels récents (257 mises à jour Microsoft® entre le 6 mars 2010 et le 11 mai 2011 sur un ordinateur personnel).

Ce décalage de maturité avec d'autres secteurs d'activité, tels que l'ingénierie mécanique ou électrique, entraîne nécessairement des difficultés de communication au sein des équipes de conception, surtout lorsque celles-ci font appel à des champs de conception très éloignés les uns des autres. La conception de grands systèmes sociotechniques relève pleinement de cette problématique.

Un système sociotechnique peut être défini comme un système avec une grande composante technique mais avec également une interaction forte avec l'humain. L'Abeille Bourbon (Fig. 2) est une illustration d'un tel système.

Fig. 2 Un grand système sociotechnique : l'Abeille Bourbon

L'Abeille Bourbon est un remorqueur de haute mer affecté à la sécurité du rail d'Ouessant et basé à Brest. Navire de force dimensionné pour naviguer en mer formée, on imagine aisément les contraintes mécaniques auxquelles il est soumis. On peut supposer que les études de conception de ce navire ont eu une forte dominante mécanique.

Pourtant, lorsque l'on sait que son équipage ne comprend que quatorze marins et que l'on voit la passerelle du navire (Fig. 3) bardée de systèmes de contrôle/commande, on comprend alors que bien d'autres aspects du navire ont dû faire l'objet d'études poussées :

- En électrotechnique par exemple, quand on sait qu'une des grandes craintes d'un marin est le black out du système en mer.
- En ergonomie aussi pour permettre à seulement quatorze personnes de maîtriser ce navire de 80m en toute sécurité.

L'équipe de conception a donc été constituée d'experts en mécanique, en électrotechnique, en automatisme et en ergonomie, devant travailler ensemble pendant toute la durée du projet. Il semble assez aisé d'imaginer les difficultés qu'ils ont pu rencontrer pour communiquer, avec leurs connaissances différentes, leurs terminologies technologiques propres, leurs exigences parfois concurrentes et des méthodologies de travail qui n'ont pas la même maturité.

Il apparaît alors intéressant de rechercher des pistes permettant de réduire le décalage de maturité dans les méthodologies de conception. Notre objectif consiste ainsi à assurer une communication plus simple entre les intervenants d'un projet, afin de permettre à chacun d'entre eux de se concentrer sur les tâches de son corps de métier à forte valeur ajoutée, plutôt que sur la traduction de ses exigences aux autres corps de métier. Nous souhaitons ainsi minimiser les risques consécutifs à ces différences de cultures techniques.

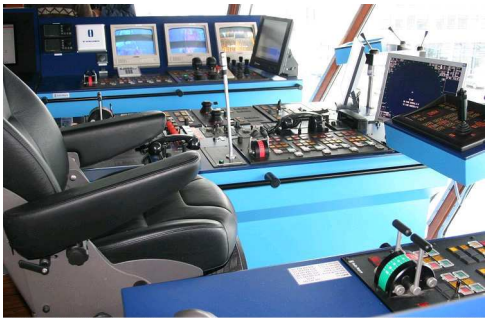


Fig. 3 Passerelle de l'Abeille Bourbon (Y. Le Bris)

Nous appliquons cette recherche à la conception de programmes de commande et d'interface de supervision. Cette partie de la conception fait intervenir, outre un automaticien, un mécanicien qui seul dispose d'une connaissance approfondie du système à contrôler, ainsi qu'un ergonome chargé de vérifier l'adéquation de l'interface de supervision avec son ou ses opérateurs. Nous savons par expérience que le dialogue entre ces trois intervenants n'est pas toujours facile.

Afin de valider l'applicabilité de nos travaux et dans un souci de confronter notre démarche à un cas industriel, nous avons défini un support d'étude (voir annexe) aussi proche que possible de la réalité. Il prend la forme d'un système de production et de distribution d'eau embarqué sur un navire. Ce système est une composante des systèmes auxiliaires de la plate-forme propulsée et est contrôlable à distance depuis la passerelle du navire.

Outre cette introduction générale, ce manuscrit est divisé en trois grandes parties suivies par une conclusion.

La première partie amène le lecteur de l'analyse du besoin à une proposition de solutions. Le chapitre I.1 introduit un état de l'art des pratiques industrielles dans la conception de grands systèmes sociotechniques. Mené sous la forme d'une enquête réalisée auprès de concepteurs, il nous permet d'établir précisément notre contexte d'étude ainsi que les attentes des professionnels.

Le chapitre I.2 propose une définition de la problématique, posée à la lumière de notre retour d'expérience industriel.

Dans le chapitre I.3, nous présentons un état de l'art académique des approches, proposant des solutions partielles à notre problématique.

Nous terminons cette première partie par le chapitre I.4 qui offre une proposition de solutions, synthèse de nos états de l'art académique et industriel. Cette proposition prend la forme d'un flot de conception.

La seconde partie détaille l'implémentation de notre proposition. Elle reprend le flot de conception étape par étape, et pour chacune d'entre elles, elle présente les concepts associés et leur implémentation. Notre flot comporte quatre étapes, les quatre premiers chapitres de cette partie leur sont dévolus.

Cette seconde partie se termine par le chapitre II.5 qui présente l'application de notre flot de conception sur un exemple concret, issu du support de l'étude présentée en annexe. Nous y présentons l'outil développé pour implémenter le flot de conception proposé, ainsi que le résultat de toutes les étapes menées sur notre cas d'illustration.

La troisième partie de ce manuscrit propose une extension de notre proposition, destinée à la prise en compte du système dans son ensemble ainsi que de son caractère reconfigurable.

Le chapitre III.1 présente les concepts nécessaires à cette prise en compte.

Le chapitre III.2 précise les modifications apportées à notre démarche et détaille leur implémentation. Le déroulement de la démarche implémentée est testé sur un nouveau cas d'illustration toujours issu de notre support d'étude.

Nous terminons cette troisième partie par le chapitre III.3 qui offre une proposition d'extension de notre flot, constituant les perspectives à court terme de nos travaux.

Nous concluons enfin ce manuscrit par un rappel de notre contribution, ainsi que par des propositions de perspectives.

Partie. I Du besoin à une proposition

Cette partie a pour double objectif de présenter l'origine des travaux de cette thèse, d'une part, et de positionner ces travaux dans un contexte à la fois industriel et académique d'autre part.

Le premier chapitre de cette partie présente notre analyse du **besoin** réalisée sous la forme d'une enquête menée auprès d'industriels concernant leurs outils et méthodes de travail. Les résultats de cette enquête, illustrés par des exemples, mettent en évidence les attentes des industriels notamment en termes de standardisation. Ils nous renseignent également sur les habitudes des concepteurs, en particulier sur l'usage des schémas.

Le second chapitre de cette partie pose la **problématique** de nos travaux à la lumière de notre retour d'expérience industriel. Nous évoquons les difficultés de communication au sein des équipes de projets hétéroclites ainsi que les conséquences de ces difficultés.

Le troisième chapitre présente l'état de l'art académique en rapport avec la problématique et le contexte de nos travaux. Nous y comparons les différents **paradigmes** d'approches orientées composants. Nous constatons qu'appliquées individuellement ces approches ne permettent pas de couvrir l'ensemble d'un processus de conception.

Le quatrième chapitre introduit succinctement notre contribution sous la forme d'une **proposition** de flot de conception unifié, dont le but est de répondre au besoin et à la problématique.

I.1 Conception dans l'industrie

Nous avons souhaité ancrer nos travaux dans un fort contexte industriel. Aussi, nous avons cherché à établir un état de l'art des méthodes et des outils de conception utilisés dans l'industrie.

Une enquête, menée auprès d'experts de SEGULA Technologies (Fig. 4) intégrés dans les équipes de conception de différents clients, nous a permis, d'une part d'établir une démarche de conception générique, et d'autre part d'identifier les principaux problèmes entravant le bon déroulement de la conception.

N°	BU	Domaine	Date
1	Naval	Études fonctionnelles	13/03/2009
2	Naval	Études automatismes	21/04/2009
3	Industrie	Études fonctionnelles	21/04/2009
4	Sud-ouest	Études automatismes	03/06/2009

Fig. 4 Liste des entretiens

Après une rapide analyse des différents outils permettant la réalisation d'une enquête (ANFH ; de Singly 2008), nous avons choisi de réaliser un entretien semi directif (Bignon 2010). Cette méthode offre plus de souplesse dans la discussion qu'un simple questionnaire mais permet tout de même de cadrer les échanges. On assure ainsi l'enchaînement et l'exhaustivité des thèmes abordés.

L'entretien aborde successivement trois sujets :

- Le système conçu, afin de s'assurer de la cohérence des entretiens entre eux.
- Le processus de conception, dont la synthèse nous permet de mettre en place une démarche générique à suivre.
- Les moyens mis en œuvre, qui nous permettent de mettre en évidence les éventuels problèmes de formalisme ou d'outillage actuels.

Ce chapitre reprend ces trois thèmes pour en faire la synthèse.

I.1.1 Le contexte

Nous nous plaçons dans un contexte de grand projet pluridisciplinaire aboutissant dans notre cas à la conception de systèmes sociotechniques complexes.

Pour l'AFNOR[1] (AFNOR 1991), « un projet se définit comme une démarche spécifique qui permet de structurer méthodiquement une réalité à venir. Un projet est défini et mis en œuvre pour élaborer la réponse au besoin d'un utilisateur, d'un client ou d'une clientèle et il implique un objectif et des actions à entreprendre avec des ressources données. » De par sa taille

[1] AFNOR : Association Française de NORmalisation

et sa nature pluridisciplinaire, ce dernier fait intervenir jusqu'à plusieurs centaines de concepteurs sur des durées de plusieurs années.

Le produit de ces projets est généralement ce qu'on peut appeler un **système sociotechnique** (c'est-à-dire un système en interaction forte avec l'humain). On peut citer en exemple de grands projets, la construction d'un navire, le développement d'une nouvelle motrice ferroviaire, la construction d'une centrale électrique, ou d'un hôpital. Les exigences de sécurité et de productivité entraînent de plus en plus souvent la multiplication des matériels intégrés sans que le potentiel reconfigurable du système global soit pleinement exploité. Ces systèmes peuvent également être qualifiés de systèmes de systèmes du fait de leur taille et de la possibilité de distinguer des entités plus ou moins dépendantes qui les composent.

Les maîtres d'œuvre de tels projets sont généralement de grandes entreprises (STX, Alstom, Areva, Bouygues, etc.) qui offrent un cadre culturel général au processus. Non pas que la démarche soit radicalement différente dans une PME, mais le nombre limité d'intervenants et la durée plus courte des projets rendent moins problématique la communication entre les concepteurs et la traçabilité des évolutions. En conséquence, la formalisation du processus y est moins pratiquée.

Pour donner une idée de la complexité des ces projets, on peut faire un parallèle avec la conception de système logiciel telle que la décrit Brooks dans le « Mythe du Mois-Homme » (Brooks 1996). Il prétend notamment que le coût d'un logiciel est multiplié par neuf entre un logiciel prototype et un logiciel s'intégrant dans un système logiciel livrable. Il estime que pour passer d'un logiciel prototype à un logiciel livrable, le coût est multiplié par trois ; ne serait-ce que pour rendre le logiciel, d'une part compatible avec un maximum d'environnements et d'autre part utilisable par le plus grand nombre. Il estime également que le coût pour passer d'un logiciel prototype indépendant à un logiciel prototype intégré dans un système de logiciel est multiplié par trois.

Ce coût est essentiellement justifié par la nécessité de débuguer les interactions entre les différents logiciels du système. Cette affirmation n'a, à notre connaissance, jamais été démontrée formellement mais elle semble communément admise.

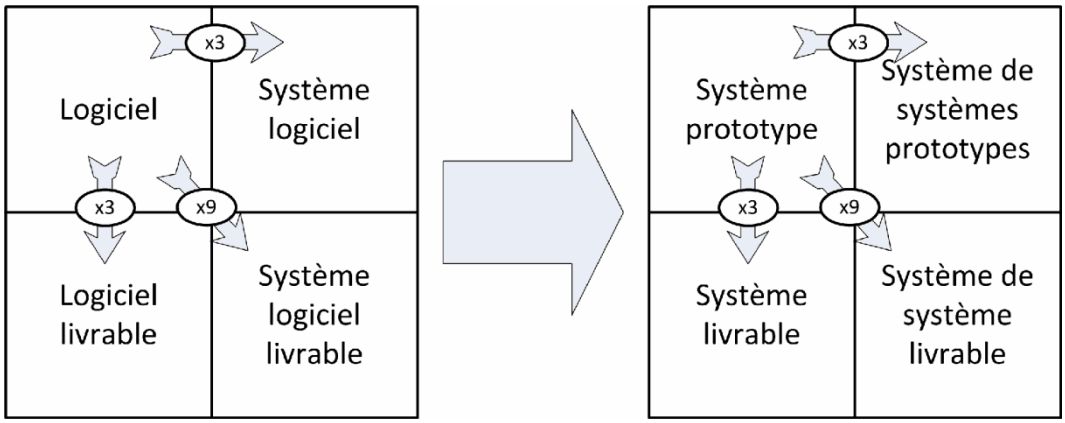


Fig. 5 Parallèle logiciel et systèmes

Ce parallèle (Fig. 5) nous donne une idée des enjeux liés à la conception de système de systèmes. Il faut savoir par exemple que le coût de fabrication d'un navire est de l'ordre d'une centaine de millions d'euros.

De par la nature complexe du produit, les nombreux concepteurs sont nécessairement issus de cultures techniques très différentes. Des mécaniciens doivent alors apprendre à dialoguer avec des automaticiens, des ingénieurs structure avec des électroniciens, ou bien encore des intégrateurs avec des experts des facteurs humains. Chacun d'entre eux arrivant avec ses méthodes, son vocabulaire et son expérience.

Bran Selic (Selic 2008) apporte un éclairage intéressant sur les différences de culture entre les informaticiens et les ingénieurs électriciens ou mécaniciens. Il fait très justement remarquer que l'inertie est inhérente aux grands projets car chaque solution proposée par les concepteurs doit faire l'objet de validations successives avant de lancer une fabrication, sur laquelle il serait très coûteux de revenir. Les programmeurs, quant à eux, sont plus habitués à rentrer très vite dans la « fabrication » du logiciel qu'ils font évoluer au fur et à mesure de l'avancement, fonctionnant ainsi plus sur le mode des méthodes agiles.

Cette plus faible inertie dans la conception logicielle que dans la conception mécanique fait que les programmeurs interviennent en général tardivement sur les projets, alors que le système est figé physiquement. Ainsi les problèmes constatés lors de la vérification du programme, qui peuvent avoir leurs sources dans des défauts de conception du système, sont très onéreux à résoudre.

I.1.2 Le processus

I.1.2.1 Généralités

Le processus auquel nous nous attachons est un processus de réalisation au sens de l'AFNOR. C'est-à-dire qu'il définit l'enchaînement des activités liées à la conception d'un produit. Il s'inscrit donc au sein du cycle de vie global de ce dernier. Rappelons que le modèle de cycle de vie le plus utilisé actuellement reste le cycle de vie en V, même si ce dernier ne reflète pas la réalité des activités menées au cours du projet. Par exemple les activités de vérification et de validation commencent en fait dès la phase de spécification. Cependant la représentation d'un modèle complet s'avère très complexe (Fig. 6).

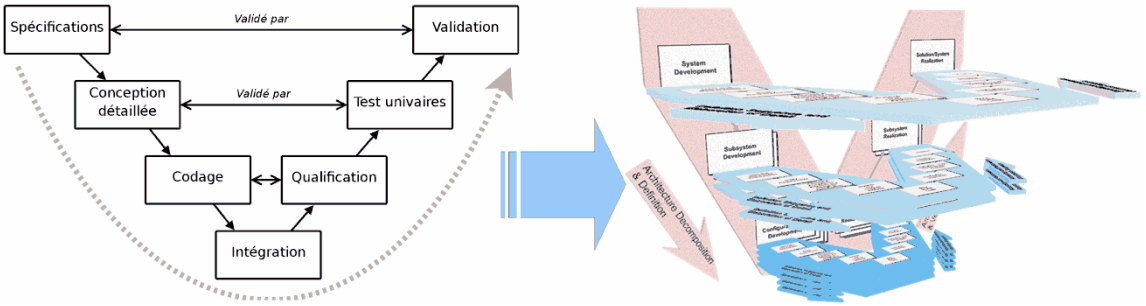


Fig. 6 Cycle en V et « Architecture et Entités, Intersection des V » (Forsberg and Mooz 2006)

Le modèle est parfois simplement enrichi pour prendre en compte un aspect particulier comme les interactions homme-machine (Kolski and Ezzedine 2003). Il existe aussi d'autres modèles basés sur une conception itérative (Pomian, Pradère et al. 1997)

I.1.2.2 Processus générique

La représentation du processus ci-dessous (Fig. 7) est volontairement simplifiée ; les retours d'information ainsi que les étapes de validation n'apparaissent pas. Nous présentons simplement une démarche que chaque concepteur partage au sein du projet.

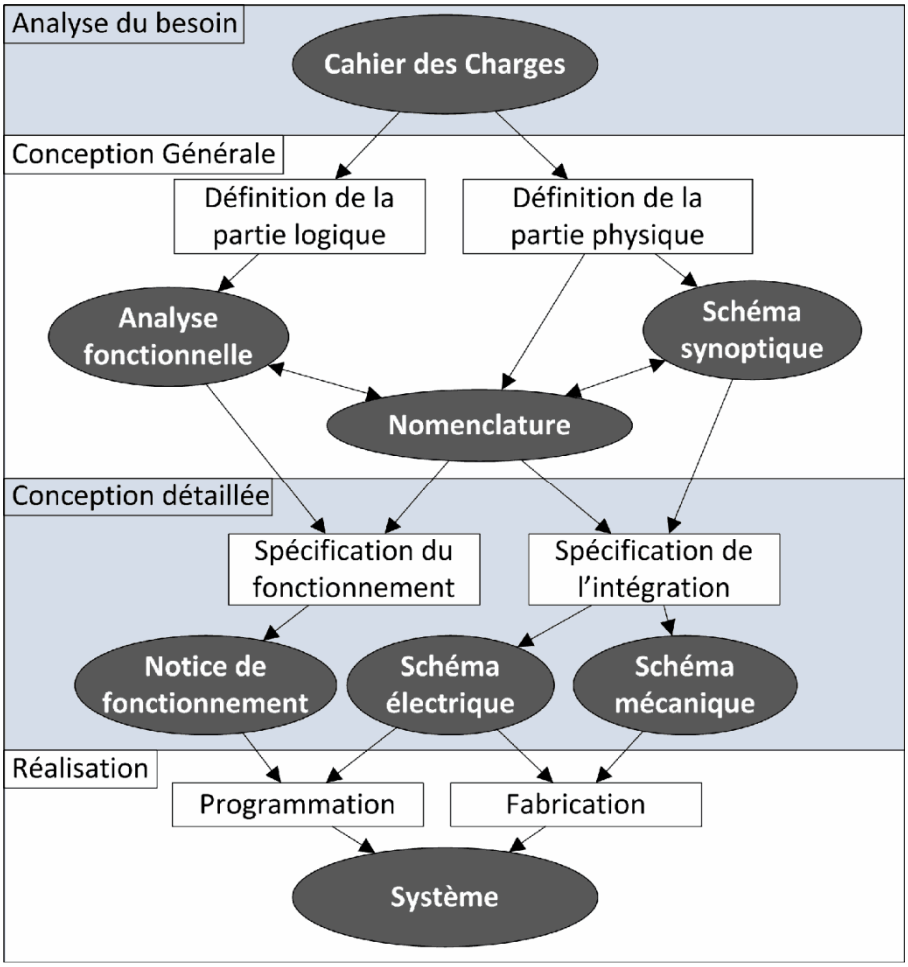


Fig. 7 Processus de conception simplifié (Bignon, Berruet et al. 2010)

Ce processus se décompose en quatre grandes étapes de conception, de l'analyse du besoin à la réalisation du produit et suit deux axes de conceptions parallèles, l'un orienté sur l'architecture du système et l'autre sur ses fonctionnalités.

Il faut cependant comprendre que derrière chaque étape, il y a en général une validation et que la découverte d'un problème en cours d'intégration par exemple, peut remettre en cause l'étape de définition.

I.1.2.2.1. Analyse du Besoin :

A l'origine de tout système, il y a un **cahier des charges**. Il peut être formel si l'utilisateur n'est pas le concepteur (c'est le cas d'un contrat), ou informel si le système est développé pour le concepteur. Il est l'expression du besoin qui justifie la conception du système.

Un cahier des charges formel peut être purement fonctionnel (spécifiant uniquement des performances et des grandes lignes du besoin) ou descriptif (imposant déjà un nombre ou un type de matériels et orientant les choix de conception).

Par extension et pour notre description, le cahier des charges représentera le référentiel complet auquel devra se soumettre le concepteur (exigences du client, normes, contraintes d'interface, etc.). En ce sens il peut comporter des incohérences issues des différentes sources des exigences. Il doit donc pouvoir être corrigé et enrichi au fur et à mesure que la définition globale du projet avance.

C'est au concepteur de mener une analyse critique du cahier des charges et d'en extraire les contraintes pour sa conception.

I.1.2.2.2. Conception générale

La seconde étape de la conception se veut générale. Il s'agit de la définition du système.

On distingue deux définitions complémentaires du système. L'une offre une description fonctionnelle du système en listant ses objectifs (fonctions principales, fonctions contraintes, fonctions complémentaires au sens de l'AFNOR X50-151, (AFNOR 2007)). Elle peut être cadrée par une méthodologie formelle (diagrammes FAST, SADT, etc.) ou être purement littérale. Cette description correspond au début de l'axe fonctionnel de la conception.

Attardons-nous sur les différents types de fonctions.

Les fonctions principales sont la raison d'être du système. Elles justifient sa conception. Par exemple, un stylo « permet à la main de laisser une trace sur un support » ; c'est sa fonction principale. En général, on associe à cette fonction des critères de performance quantifiables. Dans le cas du stylo par exemple, il s'agira d'une épaisseur de trait et d'une longueur d'écriture.

Les fonctions contraintes sont imposées par le contexte du système. Un stylo doit par exemple faciliter la prise en main. Ces fonctions disposent généralement d'un cadre normatif (ergonomie, sécurité, etc.)

Les fonctions complémentaires ne justifient pas la conception et ne répondent pas à des exigences normatives. Elles sont relatives, soit à des améliorations proposées par le concepteur, soit à des possibilités supplémentaires résultant de la conception. Toujours pour le stylo, faciliter l'identification de la couleur d'écriture est une fonction complémentaire.

Il est important de noter que l'**analyse fonctionnelle** est un document vivant. Elle peut évoluer durant le processus de conception, être enrichie, être corrigée ou elle peut prendre en compte de nouvelles exigences. Par ailleurs, l'analyse fonctionnelle conduite par le concepteur est réalisée au niveau du système et non pas au niveau de ses éléments constituants. En ce

sens, toutes les fonctions de l'analyse fonctionnelle sont des fonctions contextuelles (Niel and Craye 2002).

L'autre définition que réalise un concepteur est un **schéma synoptique** de son système. Ce schéma répondant aux codes de son métier permet au concepteur de cadrer son système dans un périmètre physique. La figure 8 est le synoptique d'un osmoseur série X de chez AQUA-BASE[1].

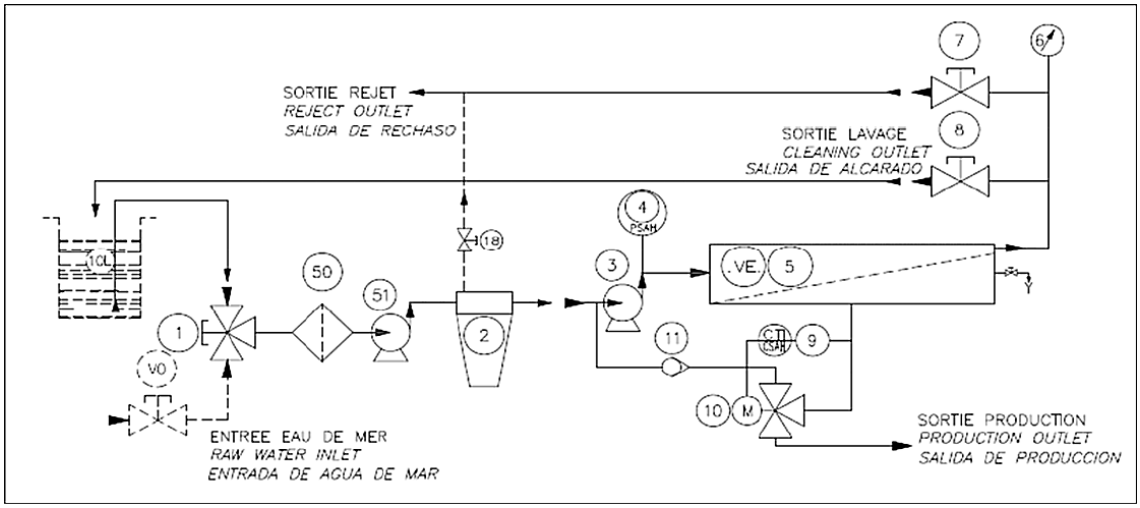


Fig. 8 Exemple de synoptique fonctionnel

Certains domaines ont déjà franchi depuis longtemps le cap de la normalisation. Par exemple, ce type de schéma est connu des ingénieurs en procédés et leur réalisation est normalisée par la norme ANSI/ISA-5.1-1984 (R1992) (ANSI 1992) ou son équivalent français, NF E 04-203. On les appelle alors PID (« Process and Instrumentation Diagram ») ou TI (Tuyauterie et Instrumentation). Il existe également ce type de norme pour réaliser des schémas électriques.

Une **nomenclature** (Fig. 9), généralement associée au schéma, décrit quant à elle les différentes instances des éléments du schéma. Elle permet de faire le lien (de manière implicite ou explicite) avec l'analyse fonctionnelle.

Ces deux définitions d'un système sont liées. En effet, le synoptique doit au minimum permettre de réaliser les fonctions principales et les fonctions contraintes extraites de l'analyse fonctionnelle. De même, en fonction de l'architecture du synoptique, on définit les fonctions complémentaires et on peut être amené à définir de nouvelles fonctions contraintes. Ces documents sont d'autant plus importants qu'en cas de problème, c'est toujours vers eux que va se tourner le concepteur pour trouver une solution.

Il est intéressant de noter que ce constat contredit la plupart des méthodes de conception qui conseillent de peaufiner une analyse fonctionnelle avant de travailler sur l'architecture d'un produit. En fait, quel que soit le projet, un concepteur ne part jamais complètement d'une feuille blanche. Soit, s'il est débutant, il s'inspire de documents produits sur des projets précédents ou sur des référentiels métiers qu'on lui a confiés (voir section I.1.3). Soit, s'il est plus expérimenté, il s'inspire de ses propres

[1] AQUA-BASE : http://www.slce.net/

connaissances pour former très tôt une représentation de son système, qu'il ajustera par la suite (Lahonde, Omhover et al. 2008). Dans ce sens et au moins sur les grands projets, Bruce Nussbaum avance dans la revue BusinessWeek (31/12/2008) : « Innovation is Dead. Herald the Birth of Transformation ».

REP.	DESIGNATION	FONCTION
EdM	*Passe coque*	*Toujours immergé, il permet d'alimenter de façon continue l'appareil en eau de mer.* ***Ne fait pas partie de la fourniture.***
V0	*Vanne de coque*	*Située à proximité du passe coque, permet de fermer l'alimentation en eau de mer.* ***Ne fait pas partie de la fourniture.***
A/B	Tuyauterie alimentation	Permet d'alimenter l'appareil à travers le filtre (2).
1	Vanne d'entrée	Vanne trois voies manuelle permettant d'alimenter l'appareil, soit avec de l'eau de mer en fonctionnement normal, soit avec l'eau ou la solution chimique contenue dans un récipient durant le rinçage ou le nettoyage des membranes.
2	Filtre 5µ	Contient un élément assurant la filtration de l'eau de mer à 5µ.
3	Pompe haute pression	Entraînée par un moteur électrique; élève la pression de l'eau de mer à la valeur souhaitée.
4	Pressostat HP	Arrête automatiquement l'appareil en cas de surpression dans le circuit.
5	Module d'osmose	Constitué de tubes résistant à la pression, contenant les membranes dans lesquelles s'effectue le dessalement de l'eau de mer.
6	Manomètre	Indique la pression dans les membranes d'osmose inverse.
7	Vanne de pression	Permet d'ajuster la pression conformément aux prescriptions du **Chap.C.**
8	Vanne de nettoyage	En ouvrant cette vanne, la vanne (1) étant elle-même basculée en position rinçage, on peut faire fonctionner l'appareil en circuit fermé sur un récipient contenant les solutions de nettoyage.**Voir Chap.C-3**
9	Sonde salinométrique	Mesure en continu la salinité de l'eau produite et commande la vanne (10) en fonction de cette mesure.
10	Vanne de production	Vanne trois voies électromagnétique commandée par le salinomètre. Elle dirige automatiquement l'eau produite vers le réservoir (tuyauterie F) si sa salinité est correcte, ou vers le rejet à la mer si elle ne l'est pas.
18	Vanne de purge d'air	Permet de purger l'air du circuit. Cette vanne, raccordée par un té au tuyau de rejet (C) reste toujours ouverte sauf pour les opérations de maintenance ou elle doit être fermée.
51	Pompe Basse Pression	Assure l'alimentation correcte de l'appareil en eau de mer.
C	Tuyauterie de rejet	Collecte la saumure concentrée produite par les membranes pour la rejeter à la mer.
R	*Passe coque*	*Situé au-dessus de la flottaison, permet d'assurer le rejet de la saumure à la mer.* ***Ne fait pas partie de la fourniture (éviter d'installer ce rejet devant l'entrée eau de mer).***
E	Tuyauterie de nettoyage	Permet d'alimenter l'appareil avec l'eau ou les solutions chimiques contenues dans un récipient auxiliaire, durant les opérations de rinçage et de nettoyage des membranes.
D	Tuyauterie de rinçage	Dirige le rejet de l'appareil vers le récipient auxiliaire, permettant ainsi de nettoyer les membranes en circuit fermé.

Fig. 9 Exemple de nomenclature

Dans ce contexte, la notion de « re-use » apparue dans les années 90 prend tout son sens. Rappelons que cette notion est héritée du concept de programmation orientée objet et prône la réutilisation d'un maximum de travail déjà effectué par la mise à disposition d'éléments de réponse à des problèmes ciblés. L'analyse des interviews menées démontre l'usage systématique de la **réutilisation** du travail, même si la pratique n'est pas formalisée en tant que telle. Ce point est d'ailleurs en accord avec l'étude menée par Nguyen en 1997 sur la pratique de la réutilisation dans les entreprises Québécoises (Nguyen, Abran et al. 1997).

I.1.2.2.3. Conception détaillée

La troisième étape du processus consiste à spécifier dans le détail les différents aspects du système. La définition générale issue de l'étape précédente est alors confrontée à la problématique de fabrication du système.

Le schéma synoptique, parce qu'il est cadré physiquement pose les bases de l'intégration du système.

Il permet de réaliser, en lien avec la nomenclature, le **schéma mécanique** détaillé. Comme le terme « cahier des charges », le terme

schéma mécanique est à prendre au sens large. Il comprend tous les documents techniques nécessaires à l'intégration physique du système. On peut citer à titre d'exemple les plans de routage des tuyauteries, les plans d'implantation des matériels, les plans de fixation, les plans d'élingage pour les matériels nécessitant d'être levés par une grue ou tout autre plan mécanique de production.

Le schéma synoptique et la nomenclature permettent également de réaliser le **schéma électrique** détaillé du système. Là aussi le terme Schéma Électrique est à prendre au sens large. Il comprend tous les documents techniques nécessaires à l'intégration physique du système, parmi lesquels on peut citer le plan de routage des câbles, le plan des armoires et coffrets électriques et les plans de bornage.

L'analyse fonctionnelle en lien avec le schéma synoptique permet la spécification du fonctionnement du système. La **notice de fonctionnement** qui en résulte, décrit les éléments utilisés pour la réalisation d'une fonction et la façon dont ils sont utilisés.

I.1.2.2.4. Réalisation

La dernière étape du processus est celle de la production du système. Elle voit la réunification de l'axe fonctionnel et de l'axe physique de conception réunis dans le **système** conçu.

Les plans mécaniques et électriques sont utilisés pour la fabrication et l'assemblage matériel du système. La notice de fonctionnement et les plans électriques (pour la partie bornage) servent à la production du programme de contrôle commande du système. L'ensemble, logiciel et matériel, constitue le système produit.

I.1.2.3 Compléments

Il est important de noter que chaque entreprise adapte ce processus à sa culture. Ainsi, certains vont inclure la nomenclature dans l'analyse fonctionnelle plutôt que l'associer au synoptique, ou même ne faire qu'un seul document regroupant toutes les informations. Le vocabulaire est également souvent différent pour chaque entreprise. Elles s'approprient ainsi la démarche.

De même, pour des montages industriels complexes, des spécifications d'interfaces particulières peuvent être mises en place. Par exemple dans le cas d'une intelligence répartie mais associée à une supervision centralisée, on peut trouver la nécessité de produire une spécification de fonctionnement local, une spécification de supervision et un document d'interface entre les deux.

Enfin certaines entreprises formalisent à l'extrême leurs processus alors que dans d'autres, la démarche est complètement informelle et appartient à la culture de l'entreprise. Il faut faire attention à ce que la formalisation soit une aide à la conception mais ne bride pas l'innovation. A l'inverse, l'absence de formalisation complique considérablement les échanges entre les intervenants d'un projet et l'intégration de nouveaux intervenants.

I.1.2.4 Conclusion sur le processus

Le processus simplifié issu de notre retour d'expérience industriel (Fig. 7) représente la démarche commune partagée par beaucoup d'entreprises (que la démarche soit formalisée ou non).

Un autre point commun de ces entreprises est que le concepteur est généralement un spécialiste du domaine d'application de son système mais n'est pas ou peu sensibilisé aux domaines connexes. Par exemple dans le cas d'un système de production et de distribution d'eau sur un navire, le concepteur est familier des notions d'hydraulique et de mécanique des fluides, il connaît également les normes relatives à l'eau de boisson mais n'a pas de notion d'automatisme ou de compatibilité électromagnétique.

A l'inverse, le programmeur n'est pas familier du comportement du système. Bien souvent, un intermédiaire est mis en place entre le concepteur et le programmeur. Celui-ci partage les notions d'automatisme et de comportement des installations, et produit en conséquence la spécification de fonctionnement.

Cependant, le fait est que la plupart des incohérences de fonctionnement des systèmes a pour origine les erreurs de compréhension entre les intervenants qui ne partagent pas la même culture. Des outils, dont nous allons voir un aperçu dans la partie suivante sont donc parfois mis en place pour faciliter les échanges entre les différents intervenants.

I.1.3 Les outils

De même qu'il existe des disparités dans la formalisation des processus, il existe de grandes différences quant aux outils mis à disposition des concepteurs. En fait, un outil étant l'instrument d'une formalisation, en règle générale, moins une entreprise formalise ses processus moins elle offre d'outils d'aide à la conception. Par ailleurs, plus les systèmes nécessitent des compétences transverses dans leur implémentation et plus on trouve d'outils d'aide à la conception. Par exemple le système vu dans une raffinerie n'est que très peu automatisé même s'il dispose d'une interface de commande à distance, et il se trouve que le processus de conception est très peu formalisé (la conception nécessite en fait peu de dialogue entre les différents concepteurs du système). Au contraire, dans l'industrie navale l'automatisation des fonctions de conduite (allant croissant avec la diminution des équipages) entraîne la nécessité de cadrer les échanges entre les différents intervenants d'un projet par un formalisme partagé.

Il faut aussi noter que plus on remonte dans le processus et moins il y a d'outils à la disposition du concepteur. Par exemple, les entreprises peuvent imposer un outil de schématique pour les schémas mécaniques et électriques, mais très peu offrent des outils pour l'analyse fonctionnelle ou les synoptiques, alors même qu'en cas de problème c'est toujours vers ces deux entités que l'on revient pour analyser les causes et chercher des réponses.

On peut distinguer deux catégories d'outils. Les outils méthodologiques d'une part et les outils d'aide au développement d'autre part.

I.1.3.1 Les outils méthodologiques

Les outils méthodologiques ont pour but de préserver l'intégrité conceptuelle d'un grand projet en donnant des moyens de standardisation aux différents concepteurs du projet.

I.1.3.1.1. Le Catalogue

Le premier des outils de standardisation largement répandus est le catalogue des matériels préférentiels. Ce catalogue est construit par des spécialistes de chaque domaine technologique (instrumentiste, automaticien, électricien, mécanicien, etc.) et regroupe des matériels répondant déjà à des contraintes transverses du projet. Par exemple, dans le cas d'une raffinerie, tous les éléments de ce catalogue répondront à une norme type ATEX[1].

L'application de cet outil permet de simplifier le choix des concepteurs et donc de recentrer leur effort sur le matériel spécifique à leur système. Il permet également d'assurer une homogénéité matérielle du projet et donc de limiter en nombre les procédures de montage, de raccordement, d'entretien, etc.

La réduction des coûts de fabrication par cet outil n'est cependant pas démontrée. En effet, on peut penser que la perspective de vendre beaucoup de matériel peut inciter les fournisseurs à baisser les tarifs. Cependant, l'assurance de vendre du matériel n'encourage pas la baisse de ces tarifs.

I.1.3.1.2. Les référentiels métiers

Ces référentiels existent pour la plupart sous forme de guide et offrent au concepteur des exemples de solutions techniques basées sur le retour d'expérience. Ils représentent le savoir-faire d'une entreprise dans le métier concerné. Ils peuvent contenir des exemples d'installations existantes et/ou le modèle d'une installation type. Ils fournissent également un cadre méthodologique de conception en proposant une démarche de conception.

Ces guides permettent donc de standardiser les règles de conception en formalisant la création des analyses fonctionnelles, des synoptiques (voir norme NF E 04-203 pour des schémas type PID) ou de tout autre document exigé par le processus. Nous pouvons reprendre ici l'exemple de l'analyse fonctionnelle qui peut être cadrée par une méthode issue des diagrammes FAST[2] ou SADT[3] ou bien encore s'inspirer du GEMMA[4]. Il en est de même des méthodes de conception que des processus. Chaque entreprise se les approprie pour les adapter à sa culture. Ainsi le référentiel métier d'une entreprise ne sera pas facile à appréhender pour quelqu'un.

Dans le domaine de l'automatisme, ces guides peuvent également standardiser la spécification de certains éléments faisant partie d'une bibliothèque de programmation. Prenons le cas d'une pompe. Le programmeur dispose dans sa bibliothèque d'un objet pompe standard qu'il doit utiliser pour toutes les pompes. Il doit cependant savoir comment le paramétrer et comment le raccorder, ce que seul le concepteur peut lui

[1] ATEX : ATmosphère EXplosible
[2] FAST: Function Analysis System Technique
[3] SADT: Structured Analysis Design Technique
[4] GEMMA: Guide d'Étude des Mode de Marche et d'Arrêt

donner. Pour sa spécification, le concepteur utilise un guide référençant les différents objets standardisés et la manière de les représenter comme celui proposé dans le projet UNICOS du CERN (Bornand 2004).

Ce formalisme, en se rapprochant des représentations usuelles en programmation (voir langages FDB, et ST de la norme CEI-61131 par exemple), cherche à rendre plus accessible la spécification au programmeur. En contrepartie, le concepteur est souvent appelé à s'appuyer sur un spécificateur intermédiaire car il n'est pas familier du tout avec ces notations.

Plus particulièrement, dans le domaine de la supervision, les normes ergonomiques en vigueur sont déclinées au travers d'un guide propre à l'entreprise ou au moins au projet. Il définit les codes visuels à respecter pour la construction des IHM[1], propose une liste de symboles à utiliser et organise l'interface afin d'assurer la cohérence des différentes images représentatives de tout ou partie du système. En effet, si les moyens de surveillance d'une installation sont du ressort de son concepteur, la cohérence d'ensemble du système ne peut être assurée par ce dernier.

Ce « guide de style » est un outil bien connu des développeurs informatiques (Ambone, Ernu et al.). On peut même citer l'exemple de la recommandation, du W3C[2] dans le développement des sites internet (Chissholm, Vanderheiden et al. 1999).

De la même manière, on peut trouver des guides aidant à la spécification des interfaces mécaniques ou de l'intégration, à la schématique, à la logistique, etc.

La mise en place de ce genre d'outils doit cependant être accompagnée. Si le concepteur les juge trop complexes, il risque de les laisser de côté. Ce risque n'est pas à négliger et le cas est fréquemment rencontré.

I.1.3.1.3. Le référentiel

Certaines entreprises offrent au concepteur un référentiel normatif et contractuel ajusté à sa conception. On peut assimiler ce référentiel à une matrice de conformité vide que le concepteur doit compléter. Le travail d'analyse du contrat ou des normes est ainsi dégrossi en amont de la conception et les exigences de chaque référence sont affectées aux systèmes concernés. Cela permet de faciliter la vérification de la prise en compte des exigences au cours de la conception. Par contre une erreur dans un référentiel peut ne pas être découverte avant les essais de qualification. Aussi un grand soin doit être apporté à la construction de ces documents.

I.1.3.2 Les outils d'aide au développement

Depuis la globalisation de l'utilisation des outils informatiques, le développement de la Conception Assistée par Ordinateur et des ERP[3] (ou PGI[4] en Français), on assiste à l'émergence du « Product Lifecycle Management » (« PLM »). Les entreprises que nous avons rencontrées sont toutes dans une démarche plus ou moins avancée d'utilisation de ces progiciels et les outils

[1] IHM : Interface Homme-Machine
[2] World Wide Web Consortium : www.w3.org
[3] ERP : Enterprise Resource Planning
[4] PGI : Progiciel de Gestion Intégré

que nous présentons ci-dessous sont généralement intégrés dans un système informatique global.

I.1.3.2.1. La schématique

La plupart des entreprises imposent l'utilisation d'un outil de schématique afin d'assurer la compatibilité des formats des fichiers. Ils sont généralement associés à des conventions de représentation qui garantissent l'homogénéité des schémas produits sur un projet.

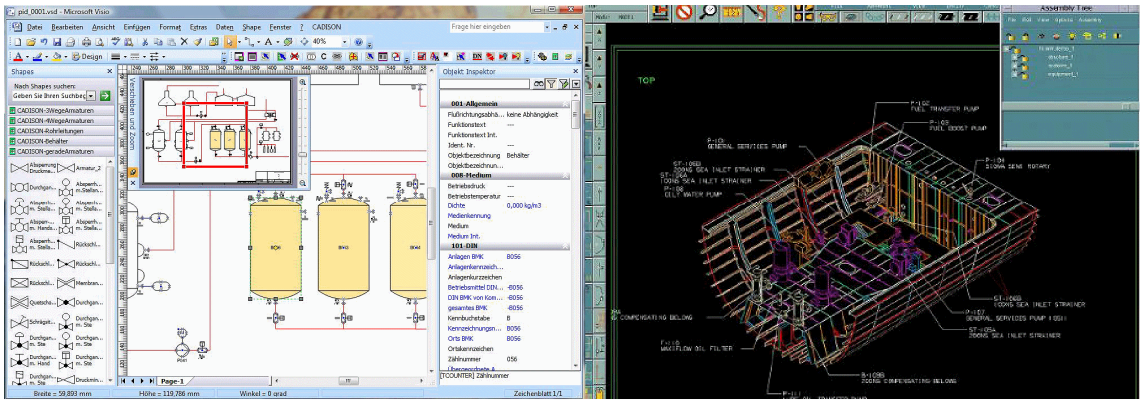


Fig. 10 Exemple de schémas sous Visio et Cadds

Par exemple, l'outil Microsoft® Visio (à gauche sur la Fig. 10) permet la création de modèles prédéfinis, partagés par tous les schémas produits par la suite. L'outil Cadds de PTC® (à droite sur la Fig. 10) permet l'assemblage de modèles complexes en parallèle.

L'outil de maquettage des IHM est un autre outil de schématique couramment utilisé en conception de systèmes interactifs. Il s'agit en général d'un simple outil informatique de présentation (Microsoft Power Point, Visio, etc.) dont l'utilisation est cadrée par le référentiel graphique de présentation des IHM.

I.1.3.2.2. La gestion documentaire

La gestion électronique de documents (« GED ») est aujourd'hui très répandue (toutes les entreprises rencontrées disposent de leur système de GED). « Elle recouvre l'ensemble des techniques qui permettent de gérer les flux de documents qui pénètrent, sortent ou circulent à l'intérieur de l'entreprise. Ces techniques ont pour fonction de capturer ou de dématérialiser des documents, afin de gérer, indexer, stocker, rechercher, consulter, traiter et transmettre des fichiers numériques de toutes origines. » (Mary).

Elle facilite le partage des informations entre les différents acteurs d'un projet (chacun a accès à l'ensemble des documents depuis son poste de travail), la recherche d'informations par l'acteur d'un projet (grâce au moteur de recherche) et la traçabilité des documents produits par le projet (circuits de visa numérique, verrouillage des documents).

La plupart des GED utilisées aujourd'hui sont des systèmes propriétaires développés spécialement pour une entreprise, ce qui en fait un outil onéreux.

Néanmoins, on voit apparaître aujourd'hui quelques outils libres de droit parmi lesquels on peut citer MAARCH[1], Freedom[2] ou Quotero[3].

I.1.3.2.3. Les essais

Les entreprises prennent conscience des risques encourus lors des premiers essais en réel d'un système. On voit ainsi, aujourd'hui, apparaître des outils d'émulation.

L'émulation n'est pas à confondre avec la simulation. Alors que la simulation est utilisée pour tester et développer différentes solutions basées sur un ensemble de paramètres prédéfinis, l'émulation est utilisée afin de tester le fonctionnement d'un système de contrôle sous différentes conditions d'exploitation (McGregor 2002). C'est un moyen de valider le fonctionnement d'un système et d'en former les opérateurs sans risque pour le matériel ou pour le personnel. On peut cependant retrouver le terme simulation employé indifféremment dans les deux cas.

Un émulateur peut être purement logiciel. On parle dans ce cas d'émulation logicielle du matériel d'automatisme sur une machine de test avec émulation logicielle de la partie opérative (exemple de gauche de la Fig. 11). Cette fonctionnalité est par exemple proposée par SIMSED (Lallican and Berruet 2006).

Un émulateur peut également reprendre une partie de l'architecture matérielle du système, il s'agit alors d'une plate-forme d'essai représentative de l'architecture de contrôle commande ; la partie opérative fait alors l'objet d'une émulation logicielle (exemple de droite de la Fig. 11).

Fig. 11 Exemple d'émulateurs

La première solution est très bien adaptée pour des essais réalisés par des développeurs. La deuxième solution, quant à elle, sera préférable pour les essais réalisés par les concepteurs. En effet, les conditions des essais seront alors plus proches d'un environnement réel ; en contrepartie, le coût d'une plate-forme est supérieur à celui d'une émulation logicielle complète.

I.1.3.3 Conclusion sur les outils

L'emploi de tels outils s'est généralisé ; la tendance actuelle est de créer des liens entre ces outils afin de globaliser la gestion des informations sur l'ensemble du cycle de vie du produit (PLM[4]). Ainsi, un élément introduit sur

[1] MAARCH : www.maarch.org/
[2] Freedom : www.frdom.org/doku.php
[3] Quotero : www.quotero.com/
[4] PLM : Product Lifecycle Management

un schéma du logiciel SchemPID de FTZ[1] (voir Fig. 12) peut être automatiquement référencé dans la Gestion Électronique de Document qui lui attribue certaines caractéristiques (nature de l'élément, dimension, mode d'approvisionnement, raccordement, etc.). Ces caractéristiques peuvent, elles-mêmes, être intégrées dans d'autres outils (un modèle 3D sous Cadds par exemple). Il reste au concepteur à les compléter au fur et à mesure du processus de conception. Ces liens permettent la définition d'objets et posent les bases d'une ingénierie documentaire.

Néanmoins, si le développement physique du produit est aujourd'hui bien intégré dans les solutions existantes comme Cadds (PTC[2]), Pro Engineer (PTC) ou Catia (Dassault Système[3]), la production des programmes, nécessaires à son fonctionnement, fait généralement l'objet d'un développement séparé qui peut entraîner des incohérences avec la réalité matérielle du produit. Nous voyons ici qu'il serait intéressant de lier la production physique d'un produit (hardware) et la production des logiciels nécessaires à son fonctionnement (software).

Cependant l'interopérabilité d'outils propriétaires est parfois compliquée du fait de formats de données spécifiques au logiciel concerné. Même si des formats génériques sont parfois proposés (format XML par exemple), la conversion d'un format à un autre peut provoquer des pertes d'information. Qui n'a jamais vu, par exemple, sa mise en forme disparaître en passant d'un logiciel de traitement de texte à un autre ?

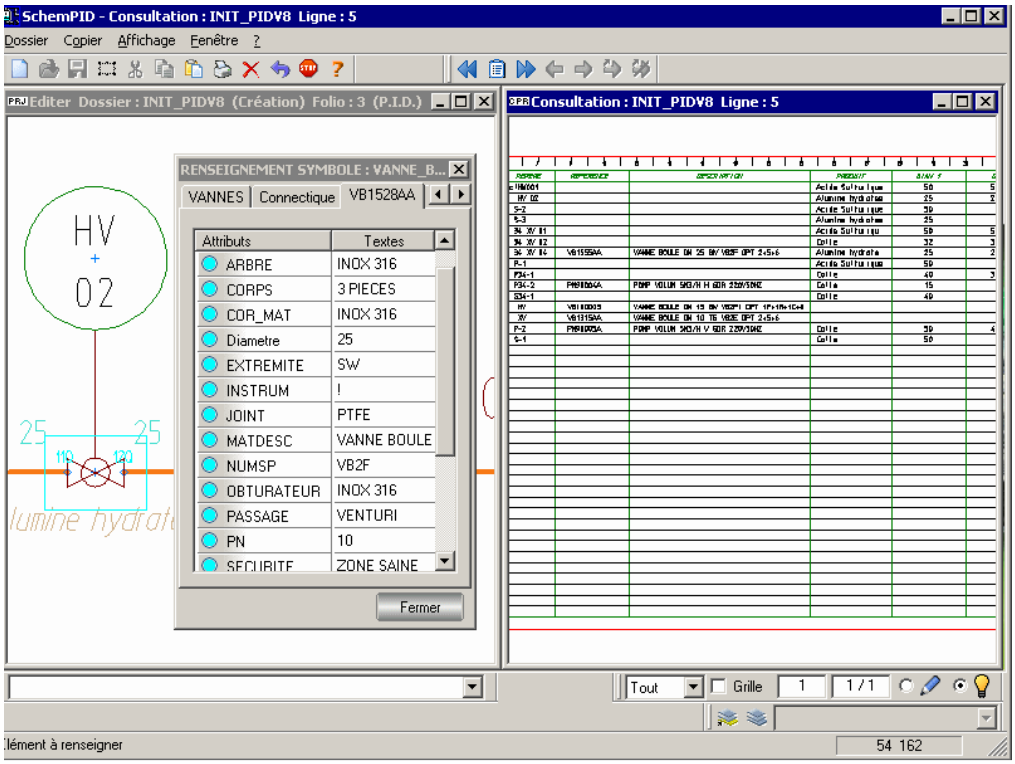


Fig. 12 Nomenclature sous SchemPID de FTZ

[1] FTZ : www.ftz.fr
[2] PTC : www.ptc.com
[3] Dassault Système : www.3ds.com

Il faut enfin noter que la mise en service des outils est très laborieuse et que les progrès technologiques des outils sont plus rapides que l'avancement des grands projets. Il en résulte que certaines entreprises mettent en place des outils en cours de projet. Sans même parler de leur fiabilité, si cette démarche n'est pas suffisamment accompagnée et si les outils ne sont pas adaptés (en termes d'interface) aux concepteurs, l'entreprise s'expose au risque de ne pas voir ses outils utilisés (ce qui signifie une perte financière en terme d'investissement), ou pire, de voir les concepteurs mener leur conception de manière habituelle puis d'adapter leurs données au format attendu par les outils (ce qui constitue une perte financière et une perte de temps).

I.1.4 Bilan sur le retour d'expérience industriel

De l'enquête menée auprès des industriels, nous retenons, outre le processus de conception simplifié présenté (Fig. 7), deux points essentiels qui conditionnent la suite de nos travaux.

D'une part, nous reconnaissons **la volonté unanime des industriels de développer la réutilisation** pour la conception de leurs produits. Cette réutilisation nécessite une plus grande standardisation, que cette dernière soit matérielle, logicielle ou méthodologique. En effet, les entreprises mettent en place des procédures standardisées pour conduire les études mais elles peuvent également proposer des catalogues de solutions standards aux concepteurs. Cette tendance, confirmée par l'étude menée par Nguyen au Québec (Nguyen, Abran et al. 1997) peut aller jusqu'à l'imposition de matériel préalablement retenu par l'entreprise, indépendamment de tout projet (catalogue de matériel standard). L'objectif de cette standardisation est de limiter les libres interprétations des concepteurs en posant un cadre méthodologique, tout en favorisant la réutilisation par l'imposition de matériels ou de blocs logiciels standards. Pour les activités de programmation, cette tendance évoque le passage de l'artisanat à la production industrielle (Sverdlov, Kopec et al. 2005) et suggère le déroulement d'une démarche de conception dite « ascendante », car on conçoit le système par agrégations successives de briques élémentaires.

D'autre part, nous avons constaté **l'utilisation systématique, et au plus tôt dans le projet, d'un schéma** qui suivra le système tout au long de son cycle de vie. Ce schéma est une synthèse descriptive du système conçu et permet d'en appréhender l'architecture et le fonctionnement. La démarche suggérée ici, est plutôt celle d'une conception « descendante » par raffinage successif d'un modèle source. Il est intéressant de noter que ce constat contredit les méthodes de conception séquentielles, qui conseillent de valider une analyse fonctionnelle d'un produit avant d'en étudier l'architecture. En fait, quel que soit le projet, un concepteur ne part jamais complètement d'une feuille blanche. Soit, s'il est débutant, il s'inspire de documents produits sur des projets précédents ou sur des référentiels métiers qu'on lui a confiés. Soit, s'il est plus expérimenté, il s'inspire de ses propres connaissances pour former très tôt une représentation de son système qu'il ajustera par la suite (Lahonde, Omhover et al. 2008).

David Harel évoquait déjà en 1994 la nécessité de rendre la conception informatique plus visuelle : « Il est de notre devoir d'aller de l'avant et de transformer la modélisation de système en un processus essentiellement visuel et graphique » (Harel 1992). Et plus récemment selon Dromey (Dromey 2006) « des représentations simplifiées sont la clé de nouveaux progrès substantiels dans le génie logiciel ». Par ailleurs, il a fallu des années au dessin technique pour devenir un médiateur de la conception (Lavoisy and Vinck 2000). Il apparaît alors pertinent d'exploiter cette caractéristique pour générer du code de contrôle commande.

I.2 Problématique

La problématique à l'origine du sujet de la thèse est directement tirée de mon expérience personnelle sur des projets d'industrie navale, en tant que responsable d'étude du système de conduite. Mon rôle était alors d'assister les concepteurs mécaniciens dans la définition du contrôle-commande de leurs systèmes, dans le respect des contraintes d'exploitation du navire et des capacités du système de supervision. Je réalisais ensuite les spécifications techniques de programmation destinées à la réalisation de la supervision centrale du navire.

Cette problématique se rapporte à un double constat. D'une part de nombreux problèmes apparaissant en fin de projet sont issus d'incompréhensions entre les différents intervenants tout au long du cycle de vie du projet. En effet, sur de tels projets, les concepteurs interviennent par centaines et sont parfois issus d'horizons technologiques très variés, ce qui entraîne des difficultés de communication.

D'autre part, dans un contexte d'amélioration permanente imposé par la concurrence, la diminution des délais et la recherche de qualité de conception ont entraîné l'obsolescence des processus séquentiels (comme le cycle en V Fig. 13), au profit de processus itératifs. L'émergence des processus de conception participative, pour les systèmes en interface avec l'humain, renforce également la nécessité de prendre en compte des exigences tardives. En d'autres termes, on n'attend plus aujourd'hui que la totalité d'une étape de conception soit validée avant d'entamer la suivante. Il faut donc simplifier la prise en compte tardive de contraintes qui seraient issues d'étapes précédentes et limiter leur impact sur la conception en facilitant leur propagation aux étapes suivantes.

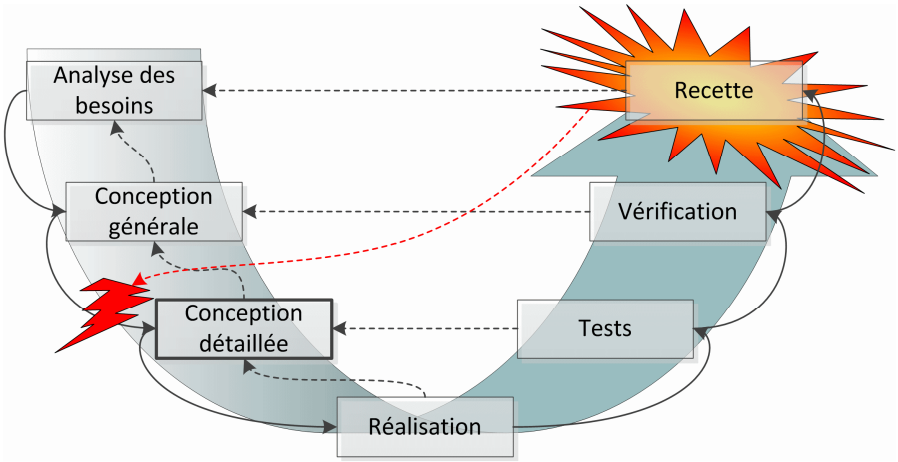


Fig. 13 Constat

Ces constats ne sont d'ailleurs pas spécifiques au domaine de l'industrie navale.

En 1975 dans « Le Mythe du Mois Homme » (Brooks 1996), Brooks revient sur son expérience du développement d'OS/360 chez IBM. Il estime

alors qu'une grande partie des bugs rencontrés au cours des projets sont issus de problèmes de communication entre les acteurs du projet. Il introduit donc la notion **d'intégrité conceptuelle** du logiciel dont l'architecte est le garant. Il insiste sur la formalisation des données partagées par les concepteurs, que ce soit d'un point de vue méthodologique ou documentaire.

Plus récemment, dans le domaine du BTP, on peut noter le projet CoCAO (Bignon, Halin et al. 2000) du Centre de Recherche en Architecture et Ingénierie né d'une volonté de faciliter la coopération au sein d'une « équipe de projet fortement hétérogène », dans le but de répondre à l'accroissement des exigences qualitatives (de la maîtrise d'ouvrage) et de la pression concurrentielle.

Les méthodes systémiques qui trouvent leur sens dans la décomposition élémentaire d'ensembles cohérents, ont émergé pour répondre à ces problématiques. Or, « le système de systèmes n'est pas rigide dans sa conception » (Luzeaux 2004). Le concepteur doit alors être capable de suivre les évolutions en traçant et en minimisant les impacts sur les systèmes, tout en se concentrant sur l'architecture d'ensemble. Il s'agit donc de maîtriser l'information associée à l'ingénierie.

Ces problématiques sont d'ailleurs reprises par l'Association Française d'Ingénierie Système (Fiorese 2005) qui identifie, entre autres, comme défauts :

- des besoins insuffisamment exprimés
- des attentes de certaines parties prenantes, mal perçues ou non formalisées
- des spécifications imprécises et incomplètes
- des responsabilités et rôles des acteurs mal définis
- une communication entre acteurs non maîtrisée

Dans le cadre de la conception des logiciels, des études ont démontré que ces problématiques se traduisent par des erreurs quantifiables. Ainsi, 79% des défaillances trouvent leur origine dans les prestations d'étude et de réalisation (Fig. 14), et 72% des défaillances ne sont détectées que lors des essais et de l'exploitation (Fig. 15).

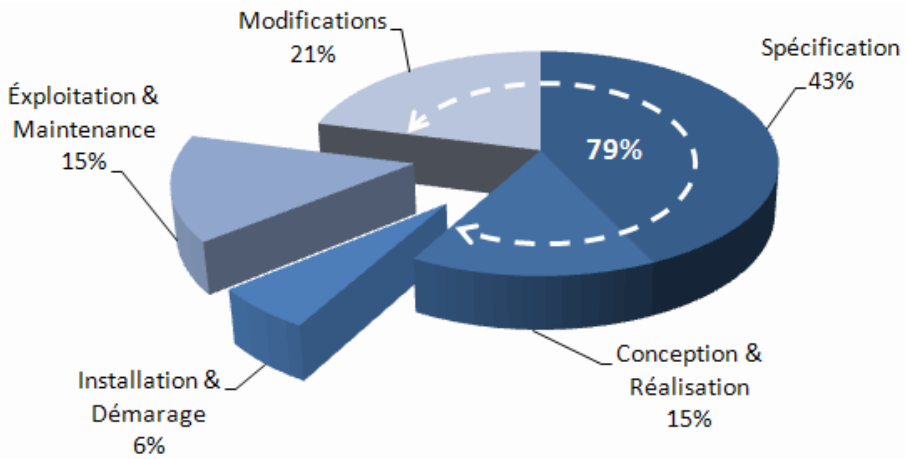


Fig. 14 Origine des défaillances dans le cycle de vie du logiciel (Sourisse and Boudillon 1997)

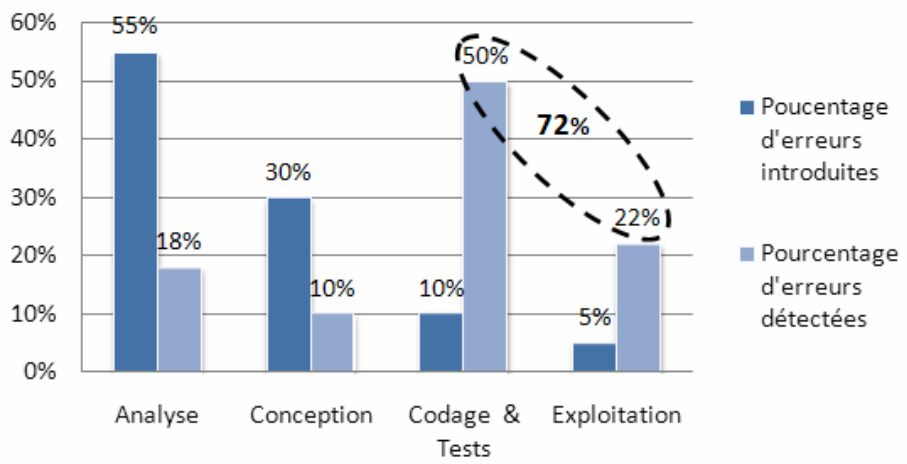


Fig. 15 Pourcentage d'erreurs introduites et détectées par phase du cycle de vie (Pham 2005)

Ces erreurs résultent du fait que l'intégrité conceptuelle du produit reste difficile à maintenir, notamment à cause de la taille importante des équipes de travail. La conception d'un navire par exemple, fait intervenir des centaines de personnes d'horizons techniques variés (mécaniciens, électriciens, informaticiens, etc.), sur une période de plusieurs années.

Par ailleurs, la correction de ces erreurs nécessite d'importants efforts de re-conception, qui sont à la fois générateurs de surcoûts et chronophages pour la production du système final.

Il est donc important de gérer la complexité de tels projets en dégageant les concepteurs des tâches à faible valeur ajoutée. En diminuant la durée des activités de programmation pour de telles tâches nous permettons ainsi aux experts de se consacrer d'avantage à la spécification du comportement global du système. Conserver un rôle important de l'expert dans les tâches à forte valeur ajoutée permet, par ailleurs, de ne pas brider l'innovation.

La génération automatique de programme décharge le concepteur de ces tâches et assure, de surcroît, la conformité du code produit à la spécification du concepteur.

Par ailleurs, la génération automatique conjointe d'un programme de commande et de son interface de supervision, permet d'assurer la cohérence entre les deux codes produits.

Enfin, la production d'un projet d'interface de supervision disponible dans les phases amont du projet permet une confrontation rapide de la conception du système à son usage par les utilisateurs. Or l'utilisateur final et son environnement restent au centre de la conception des applications ergonomiques (Caelen 2009).

I.3 État de l'art académique

Frederick P. Brooks utilisait en 1987 le loup-garou comme métaphore aux dépassements en délais et en coût des projets d'ingénierie logiciel. Il prédisait qu'il n'existerait jamais de balle en argent susceptible d'abattre ce monstre (Brooks 1987). Brooks divise les difficultés pour concevoir un système (en l'occurrence, un système logiciel) en deux parties. D'une part les difficultés essentielles qui viennent de la nature même du produit ; d'autre part, les difficultés substantielles (accidents) qui gênent la production mais qui ne sont pas inhérentes au produit.

Il est intéressant de noter que les propriétés intrinsèques des systèmes logiciels décrites par Brooks, sont, en fait applicables à des systèmes d'autre nature.

Ainsi la gestion de la **complexité** est reprise comme leitmotiv dans de nombreux travaux sur la conception parmi lesquels (Brandt, Hartmann et al. 1999; Fondement 2007; OMG 2008; Gholipour, Bignon et al. 2009).

La **conformité** à l'environnement et aux exigences n'est pas propre à la conception de logiciels. Il existe de très nombreuses normes touchant tous les domaines.

S'il est vrai que la **variabilité** d'un produit physique est limitée une fois le produit fabriqué, il est important de noter que durant la conception la gestion des évolutions est un vrai problème (Dartigues 2001; Rivière 2004; Eynard 2005).

Seule l'**invisibilité** est une propriété propre au logiciel dans le sens ou « la réalité du logiciel n'est pas inscrite dans l'espace » (Brooks 1987). Encore qu'aujourd'hui UML[1] (OMG 2009; OMG 2009) propose des outils pour représenter graphiquement un logiciel.

L'histoire semble avoir donné raison à Brooks (Brooks 1996; Fraser and Mancl 2008), puisque, au fur et à mesure que nous inventons des solutions miraculeuses permettant de résoudre des problèmes insolubles jusque là, un nouvel horizon de problème insoluble apparaît avec son lot de frustrations pour les concepteurs (Berry 2008). Cependant, s'il n'existe pas de solution miracle pour augmenter de façon significative la productivité, l'espoir se trouve dans l'utilisation conjointe de différentes solutions.

S'il est clair que l'implémentation d'une approche composant peut contribuer à répondre aux besoins en standardisation, exprimés par les entreprises, notre objectif de limiter les erreurs de conception tout en facilitant la réutilisation de solutions existantes, trouve des réponses partielles dans de nombreux travaux. Ainsi les approches orientées composants, qu'elles soient basées sur le paradigme d'objet, de patron ou de modèle se proposent toutes d'élever le niveau d'abstraction par une formalisation poussée, et d'améliorer la transmission du savoir au moyen de bibliothèques ou de collections. Dans ce chapitre nous passons en revue ces différents paradigmes.

[1] UML : Unified Modeling Language

I.3.1 Le paradigme d'objet

Le paradigme d'objet s'est largement répandu depuis son apparition à la fin des années 60. Le concept est de proposer des conteneurs abstraits, indépendants et réutilisables exécutant des fonctions transparentes pour l'utilisateur et qui sont mis en relation avec leur environnement par des interfaces prédéfinies.

L'OMG[1], fondé en 1989, a d'ailleurs pour objectif de promouvoir et de standardiser ce concept.

L'OMG est notamment à l'origine du standard UML (OMG 2009; OMG 2009) très utilisé dans le génie logiciel. UML est un langage graphique pour la conception orientée objet, née de l'unification des méthodes OMT (Rumbaugh 1996), OOD (Booch 1994) et OOSE (Jacobson 1992).

Mais il propose également, pour le domaine de l'ingénierie système, le standard SysML (OMG 2008) ainsi que toute une batterie de spécifications appliquées à des domaines particuliers. Ces standards sont associés à la recommandation « Model Driven Architecture » (OMG 2003) et portés par le méta-méta-modèle « Meta-Object Facility » (MOF) (OMG 2006).

I.3.1.1 Concepts

Dans l'approche orientée objet, un programme est vu comme une collection d'entités distinctes, identifiées et possédant des caractéristiques (Audibert 2009). Ces caractéristiques peuvent définir l'état de l'entité (un attribut) ou son comportement (une fonction). La fonctionnalité du logiciel émerge des interactions entre les différentes entités qui le composent. Suivant cette approche, les données et leurs traitements sont associés au sein d'un objet réutilisable.

Un objet est défini par :

1. Une **identité** qui permet de distinguer un objet d'un autre, indépendamment de son état. Cette identité est posée par un identifiant découlant du problème (code, n° de série).
2. Des **attributs** qui sont les données de caractérisation de l'objet. Il s'agit de variables représentatives de l'état de l'objet.
3. Des **méthodes** qui caractérisent le comportement de l'objet. Chaque méthode représente une action que l'objet peut mener en réponse à des sollicitations extérieures. Les méthodes sont étroitement liées aux attributs car les actions peuvent être déclenchées par (ou avoir un impact sur) les changements de valeur des attributs.

Le paradigme d'objet introduit également plusieurs notions dont la notion de **classe**. Une classe déclare les propriétés (attributs ou méthodes) communes à un ensemble d'objets. Il s'agit d'un modèle à partir duquel un objet peut être créé. Dans ce sens, un objet est une instance d'une classe.

Un objet peut être encapsulé. L'**encapsulation** consiste à masquer une partie des propriétés d'une classe qui ne devient utilisable que par le biais de ses interfaces. Les propriétés masquées de la classe constituent la partie

[1] OMG : Object Management Group

privée de la classe, par opposition aux interfaces qui constituent la partie publique. Seule la partie publique est accessible aux utilisateurs de l'objet. L'encapsulation permet de stabiliser l'utilisation des objets puisque les interfaces sont figées ; elle garantit également l'intégrité des propriétés privées de l'objet puisque celles-ci sont inaccessibles à l'environnement de l'objet.

La notion d'**héritage** quant à elle permet de limiter la duplication du code et d'en faciliter la réutilisation. L'héritage est le mécanisme qui permet la transmission des propriétés d'une classe à une sous classe. La création d'une nouvelle classe peut alors se faire par dérivation d'une classe existante qui répond partiellement au besoin. L'héritage peut également être multiple dans le cas d'une classe qui hériterait des propriétés de plusieurs autres classes. La notion d'héritage permet la spécialisation ou la généralisation des classes et leur hiérarchisation.

Le **polymorphisme** représente la capacité d'une méthode à pouvoir s'appliquer à des objets de classes différentes. Cette notion augmente la généricité du code puisque la même méthode peut être appelée par des objets de types différents.

I.3.1.2 Application

L'IDE Eclipse[1] est un exemple de plate-forme de développement basée sur ce concept. Chaque plugin de la plate-forme peut être considéré comme un composant logiciel dédié à des tâches particulières. L'ensemble des plugins du projet Topcased[2] propose ainsi une boîte à outil pour la conception de systèmes critiques.

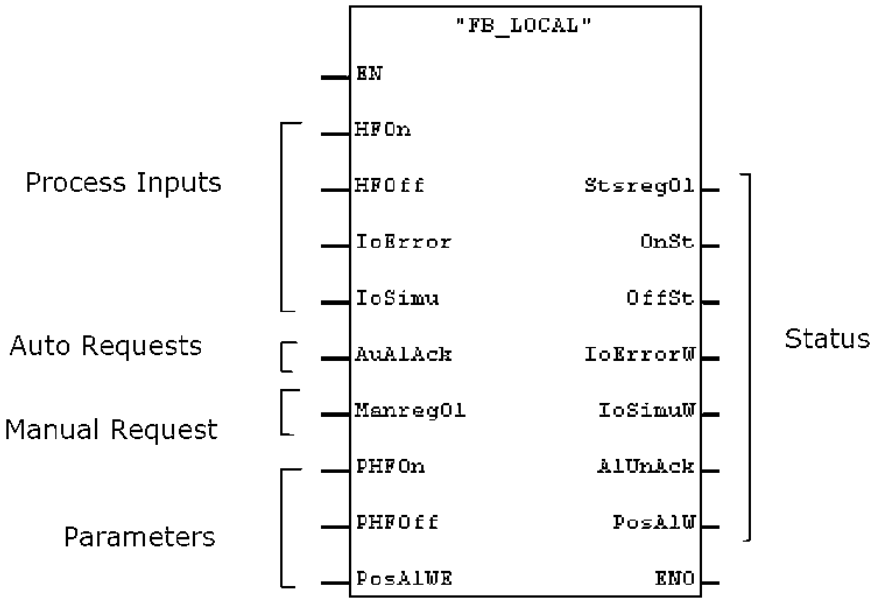


Fig. 16 Objet « Local » du projet UNICOS

[1] Eclipse : www.eclipse.org
[2] Topcased : www.topcased.org/

L'idée de proposer des composants « sur étagère » (COTS[1]) prêts à l'emploi a même dépassé le cadre du génie logiciel. Ainsi, il existe des profils UML pour différents secteurs d'activité. On peut citer par exemple, SYSML (OMG 2008) qui est une application d'UML au domaine de l'ingénierie Système, MARTE (OMG 2009) une application d'UML pour les systèmes temps-réel embarqués et QoS un profil UML pour modéliser la qualité de service et les caractéristiques de tolérance aux fautes.

Le paradigme d'objet est également présent en automatisme au travers du concept de « User Defined Function Block » (UDFB) (IEC 2003). On peut citer le projet UNICOS du CERN (Bornand 2004) comme une mise en œuvre concrète de ce paradigme (Fig. 16). Le protocole Bacnet de l'ASHRAE (ASHRAE 2010) est également fondé sur ce paradigme. Chaque matériel physique d'une installation HVAC[2] est vu comme un composant logiciel par le système de contrôle commande. L'ingénierie Système s'intéresse également à cette approche, en considérant des systèmes entiers comme composants d'un système de systèmes. Autran propose par exemple, une matrice de couplage qui permet d'identifier les interactions entre les systèmes d'un SoS[3] (Autran, Auzelle et al. 2008) ; dans ce cas, le système est défini par ses interfaces avec les autres systèmes et par sa contribution au système de systèmes.

I.3.2 Le paradigme de patron

Le paradigme de patron est issu du génie civil et des problématiques rencontrées en architecture. La définition proposée par le livre fondateur de Christopher Alexander (Alexander 1979) est la suivante : « chaque patron décrit un problème qui se produit encore et encore dans notre environnement, et ensuite décrit le cœur d'une solution à ce problème de manière à ce qu'on puisse utiliser cette solution plus d'un million de fois, sans jamais le faire deux fois exactement de la même manière. »

Le but d'un patron est de recenser le savoir-faire concernant la résolution d'un problème spécifique répandu et de diffuser ce savoir-faire au plus grand nombre. Les solutions représentées par un patron peuvent ainsi être réutilisées et adaptées à un problème similaire.

L'idée d'utiliser des patrons pour la conception de logiciel date de la fin des années 80 (Jézéquel 2006) et a été popularisée entre autre par le « Hillside Group », fondé par Cunningham, Johnson, Auer, Hildebrand, Booch, Beck et Coplien. Mais la contribution qui fait aujourd'hui référence, est celle du livre « Design Patterns : Elements of reusable Object Oriented Software » (Gamma, Helm et al. 1995). Les patrons de conception, en complément d'une approche objet, sont vus comme un bon moyen d'améliorer la réutilisation et donc de diminuer l'effort de conception (Arnaud 2008).

Un objet offre une solution instrumentée et réutilisable pour un problème simple, un patron propose un moyen de répondre à un problème de plus grande ampleur et plus générique. Par exemple, le patron « Composite »

[1] COTS : Commercial Of-The-Shelf
[2] HVAC : Heat, Ventilation and Air Conditioning
[3] SoS : System of Systems

(Gamma, Helm et al. 1995) propose une solution pour manipuler un groupe d'objets de la même façon que s'il s'agissait d'un seul objet. Les objets ainsi regroupés doivent posséder des opérations communes. Pour ce faire, il s'appuie sur les classes « composant », « feuille » et « composite ».

I.3.2.1 Concepts

Chaque patron offre une **solution** réutilisable à un **problème** récurrent donné, dans un **contexte** défini (Arnaud 2008). Ces solutions peuvent ainsi être mutualisées de projet en projet au travers des patrons.

Nom du patron

Problème : description des conditions d'application. On y décrit aussi le contexte d'utilisation.

Solution : description des éléments (objets, relations, responsabilités et collaboration) permettant de concevoir la solution du problème.

Conséquences : description des résultats de l'application du patron sur le système. Contient la description des effets induits par cette application

Fig. 17 Représentation simplifiée d'un patron (Saidane 2005)

En général un patron se compose de 4 éléments essentiels (Fig. 17) (Gamma, Helm et al. 1995) :

1. Le **nom** du patron est un moyen de décrire le problème de conception, ses solutions et ses conséquences en un mot ou deux. Nommer un patron augmente immédiatement le vocabulaire de la conception. Il permet de concevoir à un niveau d'abstraction plus élevé. Avoir un vocabulaire pour les patrons permet d'en parler avec ses collègues, dans sa documentation, et même à soi-même. Il rend plus facile la projection des idées et la communication.
2. Le **problème** décrit les cas d'application du patron. Il explique le sujet à traiter et son contexte. Il peut décrire des problèmes de conception spécifique comme la représentation d'algorithme sous forme d'objet. Il peut également décrire des structures d'objet symptomatique d'une conception immuable. Parfois le problème contient une liste de conditions nécessaires à l'application du patron.
3. La **solution** décrit les éléments qui répondent au problème ainsi que leurs relations, leurs rôles et leurs coopérations. La solution ne décrit pas une conception concrète particulière ou une implémentation puisqu'un patron peut être appliqué de différentes manières suivant les situations. Le patron propose plutôt une description résumée d'un problème de conception et une manière d'agencer certains éléments, permettant de résoudre ce problème.
4. Les **conséquences** sont le résultat de l'application du patron. Bien que les conséquences soient rarement évoquées lors de la description des choix de conception, elles sont déterminantes pour l'évaluation des alternatives de conception et pour l'appréciation des avantages et des inconvénients de l'application du patron. Les conséquences pour un logiciel sont souvent

exprimées en taille (de programme) et en temps (de traitement). L'analyse des conséquences permet de choisir un langage ou une implémentation adaptée. En termes de réutilisation, les conséquences d'un patron comprennent son impact sur la flexibilité, la portabilité ou l'ouverture d'un système. Le simple fait de décrire les conséquences contribue implicitement à les comprendre et à les évaluer.

Les patrons couvrent un spectre large des activités de production. Conte (Conte, Fredj et al. 2001) propose trois critères, issus d'une classification plus large des approches composants, pour catégoriser les patrons.

- Un critère portant sur le type de connaissance capitalisée par le patron. Il peut s'agir de patron de type **produit** comme ceux du GOF[1] (Gamma, Helm et al. 1995) ou un patron de type **processus** (Hug, Front et al. 2008).
- Un critère portant sur la couverture du patron. Ce dernier peut offrir des solutions **génériques** (Coad, North et al. 1997), des solutions applicables à un **domaine** particulier comme celui des IHM (Hariri, Tabary et al. 2009) ou à une **entreprise** particulière (Apple 2010) (Microsoft 2011).
- Un critère concernant la portée du patron. Celui-ci peut en effet intervenir au niveau de la phase d'**analyse** (Coad, North et al. 1997) du processus de développement, de la phase de **conception** (Gamma, Helm et al. 1995) ou de la phase d'**implantation** (Coplien and Schmidt 1995).

Les patrons sont parfois décrits de manière narrative comme ceux dans « the timeless way of building », (Alexander 1979) mais aujourd'hui des représentations plus formelles sont introduites pour faciliter la diffusion et la réutilisation des patrons. Certains de ces formalismes sont adaptés à la représentation de patron produit (Gamma, Helm et al. 1995) ou de processus (Tran 2007). Le formalisme P-Sigma (Conte, Fredj et al. 2001) se propose de modéliser tout type de patron.

I.3.2.2 Application

Nous l'avons vu, les patrons tels qu'ils se conçoivent aujourd'hui sont issus du domaine de la construction (Alexander 1979). Ils ont trouvé un écho favorable dans le domaine du génie logiciel (Coplien and Schmidt 1995; Gamma, Helm et al. 1995; Coad, North et al. 1997). UML propose ainsi une déclinaison des patrons sous la forme du concept de collaboration paramétrée (Jézéquel 2006).

Mais comme pour les objets, l'utilisation de ce concept n'est pas restreinte à ce domaine. En automatisme, Drath (Drath, Fay et al. 2006), propose une approche basée sur des patrons pour l'implémentation d'un contrôle d'exclusion mutuelle. Jéron (Jéron, Marchand et al. 2006) propose des patrons de supervision pour le diagnostic des systèmes à événement discret. Hariri (Hariri, Tabary et al. 2009) propose d'utiliser des patrons de

[1] GOF : Gang Of Four, acronyme utilisé pour désigner les auteurs du livre « Design Patterns : Elements of reusable Object Oriented Software »

contexte d'utilisation (plate-forme, usage et utilisateur sont pris en compte) pour réaliser des IHM adaptatives.

I.3.3 Le paradigme de modèle

Plus récemment, depuis le début du 21ème siècle, l'ingénierie du logiciel s'est tournée vers le paradigme du modèle. S'il existe diverses définitions de ce qu'est un modèle (Muller, Fondement et al. 2010), nous reprenons à notre compte celle proposée par Combemale : « Un modèle est une abstraction d'un système modélisé sous la forme d'un ensemble de faits construits dans une intention particulière. Un modèle doit pouvoir être utilisé pour répondre à des questions sur le système modélisé » (Combemale 2008).

Le paradigme d'objet est issu de la décomposition en brique élémentaire d'un système permettant ainsi de concevoir un système par composition d'objets. Le paradigme de modèle est, quant à lui, issu de la décomposition d'une activité de conception en fonction des différents points de vue portés par les concepteurs sur le système à concevoir (Bézivin 2004). Si ces deux approches sont fondamentalement différentes, elles peuvent néanmoins être utilisées de manière complémentaire.

I.3.3.1 Concepts

L'ingénierie dirigée par les modèles définit plusieurs concepts que nous illustrons sur la figure 18. La notion de modèle introduit le concept de **représentation**. La représentation est la relation notée « RepresentionOf » sur la figure 18, qui lie un sujet à son modèle.

Un modèle est effectivement une représentation partielle d'un sujet suivant un point de vue. Ce point de vue est caractérisé par la notion de metamodel. Un metamodel décrit un langage de modélisation et dans ce sens définit le point de vue du modèle sur son sujet. Une nouvelle relation apparaît alors. Il s'agit d'une relation de **conformité**, notée « ConformantTo » sur la figure 18. Pour exprimer parfaitement un point de vue, un modèle doit être conforme à son metamodel.

Le monde des modèles est **hiérarchisé** (Bézivin and Gerbé 2001), il se situe au dessus du monde réel (level M0) en terme d'abstraction et se compose de 3 niveaux.

Le niveau M1 correspond aux modèles représentant le monde réel et le niveau M2 est celui des métamodèles. Au sommet de cette hiérarchie se situe le niveau des métamétamodèles. Un métamétamodèle est un langage de métamodelisation qui permet de spécifier des métamodèles. Il a en outre la capacité de se décrire lui-même. Ainsi un métamétamodèle est conforme à lui même. L'OMG propose le MOF[1] (OMG MOF) comme métamétamodèle et nous nous appuyons sur ce dernier pour décrire nos métamodèles.

[1] MOF : Meta-Object Facility

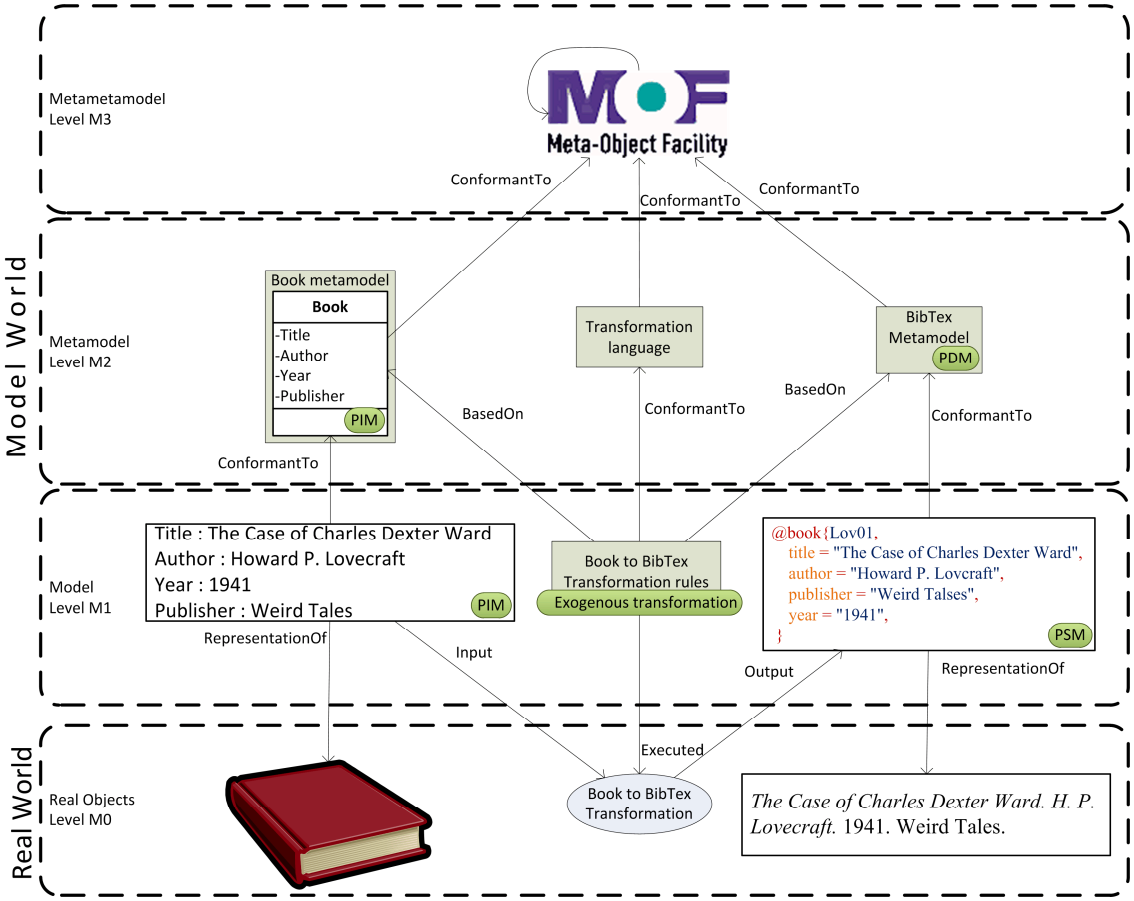


Fig. 18 Concepts de l'IDM

La spécification MDA[1] (OMG MDA) pose également le concept de **transformation** de modèle permettant, à partir d'un modèle indépendant de la plateforme (PIM[2]) et d'un modèle de plateforme (PDM[3]), de produire un modèle spécifique à une plateforme (PSM[4]) et donc exploitable par cette dernière.

Au delà de la définition initialement proposée par l'OMG le concept de transformation a été étendu.

La figure 19 présente une taxonomie des transformations de modèle. Une transformation d'un modèle Ma en un modèle Mb, conformes à un métamodèle MM est une transformation **endogène**. Une transformation d'un modèle Ma, conforme à un métamodèle MMa, en un modèle Mb, conforme à un métamodèle MMb, est une transformation **exogène**. Une transformation entre deux modèles de même niveau hiérarchique (de PIM à PIM par exemple) est une transformation **horizontale**. Une transformation entre deux modèles de niveaux hiérarchiques différents est une transformation **verticale**.

[1] MDA : Model Driven Architecture
[2] PIM : Platform Independent Model
[3] PDM : Platform Description Model
[4] PSM : Platform Specific Model

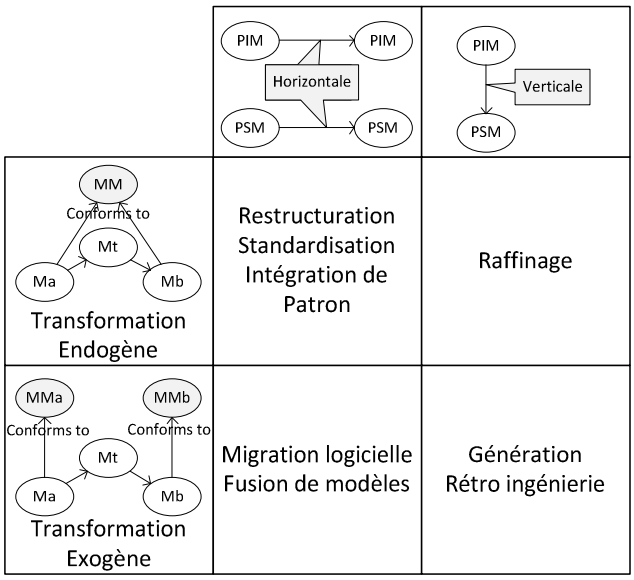


Fig. 19 Types de transformation et leurs usages principaux (Mens and Van Gorp 2006; Combemale 2008)

Les règles d'une transformation de modèle font références aux métamodèles. Cette relation est notée « BasedOn » sur la figure 18. Sur notre exemple, la transformation est horizontale et exogène. Elle est basée sur un métamodèle de livre et sur le métamodèle de BibTex et produit un modèle conforme à BibTex à partir d'un modèle de livre. Quel que soit le modèle d'entrée, s'il est conforme à son métamodèle, la transformation pourra s'exécuter et produira ainsi le modèle de sortie.

I.3.3.2 Application

L'OMG introduit enfin le concept de Computer Independent Model (CIM). Ce dernier porte les connaissances et les exigences propres à l'utilisateur final du système. Il est supposé que l'utilisateur principal du CIM ne maîtrise pas les modèles et artefacts utilisés pour réaliser le produit spécifié dans le CIM (OMG 2003). Ce modèle permet aux experts du domaine d'application du système de dialoguer sur une base commune avec les experts de la conception du système.

Les concepts de l'IDM offrent un background en parfaite adéquation avec notre démarche de conception. Par ailleurs, l'IDE Eclipse, notamment grâce à l'Eclipse Modeling Project[1], offre un panel complet d'outils pour l'implémentation des différentes opérations de notre démarche.

UML[2](OMG 2009; OMG 2009) par exemple est un métamodèle pour le développement orienté objet de logiciels. La chaîne de développement AUKOTON (Vepsalainen, Hastbacka et al. 2008) est une approche mixte, basée sur un langage de domaine spécifique (DSL[3]) issu d'un modèle métier, pour définir le système, et d'un profil UML pour définir les exigences. AUKOTON permet la génération partielle d'un programme basé sur les blocs fonctionnels.

[1] Eclipse Modeling Project (www.eclipse.org/modeling/)
[2] UML : Unified Modeling Language
[3] DSL : Domain Specific Language

Le projet MEDEIA (Ferrarini, Dedè et al. 2009) propose une approche basée sur des vues de domaine spécifique (DSV[1]) et un modèle d'implémentation de composant d'automatisme (ACIM[2]) pour produire des vues spécifiques à une plate-forme (PSV[3]). MEDEIA est un environnement de travail pour la conception d'automatismes industriels.

Dans le même registre, Mertens (Mertens and Epple 2010) propose une approche orientée service pour la gestion des propriétés. Le but est de rendre accessible des objets décrits par leurs propriétés à l'ensemble des domaines de conception.

Les ergonomes s'intéressent aussi à l'IDM[4] pour la conception d'IHM comme le montrent les articles de prospectives (Gabillon, Calvary et al. 2008) et (Sottet, Calvary et al. 2006).

I.3.4 Bilan des approches

Les trois paradigmes présentés ci-dessus ont ceci de commun qu'ils proposent des outils pour améliorer la réutilisation, dans les démarches de conception. Dans ces trois approches, la transmission du savoir se fait par l'utilisation de composants. Les critères de comparaison des composants réutilisables, introduits par Conte (Conte, Fredj et al. 2001), nous offrent un moyen de les comparer (Fig. 20).

En fait, ces différents types de composants ne sont strictement semblables que dans leur couverture. Celle-ci peut être générale, c'est par exemple le cas du patron observateur (Gamma, Helm et al. 1995). Elle peut également être rattachée à un domaine, c'est le cas des « widgets » qui sont des objets spécialement dédiés aux interfaces graphiques. Elles peuvent enfin être propres à une entreprise, c'est par exemple le cas des blocs fonctionnels du projet Unicos (Bornand 2004).

Pour les autres critères, les composants diffèrent. Parce qu'un objet est un extrait de code, il porte des connaissances rattachées à un produit spécifique. La solution qu'il apporte est purement logicielle et sa portée ne couvre que l'implantation. Il s'agit généralement d'une boîte noire ou en verre que l'on ne peut modifier directement et sa granularité est faible. On pourrait même la qualifier de très faible car elle est inférieure à celle des patrons. En effet, même s'il existe des objets composites, ceux-ci ne contiennent qu'un nombre limité de sous-objets.

Concernant les patrons, seuls les trois premiers critères permettent de les classer. Nous avons présenté des exemples pour chacun d'eux. Concernant les trois derniers critères, un patron est toujours de nature conceptuelle et d'une granularité faible. Tous les patrons sont des boîtes blanches car ils doivent pouvoir être adaptés à un contexte particulier.

Les modèles, enfin, peuvent porter des connaissances indifféremment sur un produit (Frizon De Lamotte 2006; Charfi, Gamatié et al. 2008; Ferry

[1] DSV : Domain Specific View
[2] ACIM : Automation Component Implementation Model
[3] PSV : Platform Specific View
[4] IDM : Ingénierie Dirigée par les Modèles

2008) ou sur un processus (Breton 2002; Rochet 2007; Garcia, Combemale et al. 2009). Ils peuvent couvrir aussi bien les phases d'analyse (Gerbé, Mineau et al. 2007), de conception (Lallican 2007) ou d'implantation (Frizon De Lamotte, Berruet et al. 2008). Les transformations de modèle permettent la génération automatique de programmes, cette particularité confère au modèle une nature mixte, à la fois conceptuelle et logicielle. L'intérêt de l'IDM est de pouvoir travailler directement sur les modèles. Ceux-ci doivent alors être considérés comme des boîtes blanches. Néanmoins, des modèles intermédiaires utilisés dans les étapes de génération ou des modèles de plate-forme (PDM) peuvent être des boîtes noires ou en verre.

Critère	Descriptif et valeurs possibles	Cas général des objets	Cas général des patrons	Cas général des modèles
Type de connaissance	Il peut s'agir de capitaliser des spécifications ou des implantations de **produit** – un produit correspondant au but à atteindre – ou de capitaliser des spécifications ou des implantations de **processus** – un processus correspondant au chemin à parcourir pour atteindre le résultat.	Produit	Produit ou processus	Produit ou processus
Couverture	Il peut s'agir de composants généraux (resp. **domaine**, **entreprise**) si le problème traité est fréquent dans de nombreux domaines d'application (resp. dans un domaine d'application, dans une entreprise particulière).	Généraux ou domaine ou entreprise	Généraux ou domaine ou entreprise	Généraux ou domaine ou entreprise
Portée	La portée d'un composant est évaluée en fonction de l'étape d'ingénierie (**analyse**, **conception**, **implantation**) à laquelle le composant s'adresse.	Implantation	Analyse ou conception ou implantation	Analyse ou conception ou implantation
Nature de la solution	Les composants sont de nature **conceptuelle** et/ou **logicielle**	Logicielle	Conceptuelle	Conceptuelle ou logicielle
Ouverture	L'ouverture caractérise le niveau de transparence du composant. Selon que son adaptation entraîne ou non une modification de sa structure interne, il est possible d'en distinguer plusieurs types qui vont de la **boîte noire** (la structure interne du composant n'est ni visible, ni modifiable, seule l'interface est accessible), à la **boîte blanche** (le composant est complètement transparent) en passant par la **boîte en verre** (la structure interne du composant est visible mais non modifiable).	Boîte noire ou boîte en verre	Boîte blanche	Boîte blanche, Boîte en verre ou Boîte noire
Granularité	Pour les composants orientés objets, la granularité des composants est souvent mesurée en nombre de classes. De manière plus générale elle peut être mesurée en nombre d'entités (des modules, des activités, etc.). Elle peut être **faible** (< 10), **moyenne** (< 100) ou **forte**.	Faible	Faible	Faible ou moyenne

Fig. 20 Comparaison des composants réutilisables

La mise en œuvre conjointe de ces paradigmes permet d'envisager une démarche unifiée couvrant l'ensemble d'un processus de conception, permettant de répondre au besoin de standardisation exprimé par les industriels sans remettre en cause des méthodes et outils aujourd'hui maîtrisés par les acteurs de la conception.

I.4 Proposition d'une démarche unifiée

La démarche de conception que nous proposons (Bignon, Berruet et al. 2010) se veut applicable dans un contexte industriel. Elle ne doit cependant pas tourner le dos aux travaux de recherche actuels. La littérature ne manque pas lorsqu'on s'intéresse aux problèmes de conception. Néanmoins, le bilan de notre retour d'expérience nous a permis d'identifier deux approches de conception, aujourd'hui concurrentes et que nous espérons unifier au travers de notre démarche.

La première est une approche ascendante orientée sur l'implantation de composants génériques. A ce titre, elle permet de prendre en compte les contraintes entre composants. Son principal avantage réside dans l'utilisation d'éléments en bibliothèque (notion de réutilisation). Mais la notion de système global n'est à ce jour pas prise en compte.

La seconde est une approche descendante qui apporte une description de haut niveau. Elle permet de profiter des avantages des transformations de modèles afin de décliner le modèle de description en différents modèles utilisés à des fins d'analyse et d'implantation. Néanmoins, le modèle de description ne permet pas de modéliser des contraintes entre ressources élémentaires. De plus, il est nécessaire d'ajouter en annexe au modèle, des informations relatives à l'implantation pour générer du code de commande.

Cette partie doit démontrer, d'une part, comment ces approches étudiées précédemment au Lab-STICC (Berruet 2007) peuvent être rattachées à notre démarche de conception générique présentée précédemment, et d'autre part ce que nous apporte l'intégration de ces approches.

I.4.1 Approches expertisées au Lab-STICC

Avant de présenter notre flot de conception, nous faisons donc état des travaux du Lab-STICC dans le domaine de la conception de système autour des approches ascendante et descendante.

I.4.1.1 Approche ascendante

Cette approche a été initiée au Lab-STICC par Sébastien Mouchard (Mouchard 2002) puis elle a été enrichie par Jean-Louis Lallican (Lallican 2007). Elle prend son origine dans le concept de programmation orientée objet, présenté dans l'état de l'art. Dans cette approche ascendante, la conception se fait par agrégation de composants prédéfinis, complétée par un développement traditionnel.

Le concept de programmation orientée objet part du constat qu'une grande partie du code pour réaliser un projet, peut être réutilisée pour d'autres projets. Une étude menée par Jones en 1984 (Jones 1984), estime qu'entre 30% et 85% du code est réutilisable en fonction de l'application.

Si nous étendons cette hypothèse à une conception partagée entre plusieurs intervenants, la réutilisation est également valable au sein d'un projet mais dans ce cas les objets doivent avoir été développés au préalable afin de les mettre à disposition des concepteurs. L'hypothèse est donc : concevoir des objets standards indépendamment de tous projets permet, d'une part, de répartir le coût de conception de chaque objet entre plusieurs projets et d'autre part, de gagner du temps sur la réalisation d'un projet en permettant au concepteur de se concentrer sur le code spécifique qui ne représente qu'une partie du code total.

Dans le cadre d'un système sociotechnique, une partie du code peut donc être constituée par des objets standards servant à plusieurs systèmes sur plusieurs projets. Le code restant étant constitué des objets spécifiques propres au système concerné et des artifices nécessaires à l'intégration des objets standards. En contrepartie, pour être facilement exploités, les objets doivent être d'une conception optimisée et être parfaitement documentés pour faciliter leur insertion dans le système. Cela nécessite pour chaque objet générique un travail de conception plus approfondi.

Cette notion d'objet générique portant sur la conception logicielle se rapproche également, d'une part, de la démarche de standardisation matérielle mise en place dans les entreprises (les bibliothèques d'objets sont alors à rapprocher des catalogues de matériels standards) et, d'autre part, de la démarche de formalisation documentaire.

Comme nous l'avons vu, des guides visant à standardiser la spécification de l'automatisme sont mis en place (Bornand 2004). De plus, les outils de Gestion Électronique de Document qui sont mis en place aujourd'hui disposent déjà d'une approche orientée composant. En effet, un symbole déposé sur un schéma crée une instance dans une base de données de matériel (nomenclature) qui contient déjà les attributs connus de la technologie standard que le concepteur n'a plus qu'à compléter. Il reste donc à voir comment intégrer dans ces outils des attributs de contrôle commande et de supervision (issus des guides et outils vus précédemment), premier pas vers une génération automatique de programme et d'interface au moins pour la partie représentée par des objets standards.

Cette démarche doit cependant être cadrée ; en effet, les expériences menées dans le cadre de (Lallican 2007) ont montré que cette approche donne des programmes de commande beaucoup plus volumineux que ceux développés par des méthodes empiriques. Ains,i dans le cadre d'une application industrielle, l'implantation de ce code sur un automate oblige parfois le concepteur à « monter en gamme » et en conséquence à augmenter le coût de son système. Il est donc important que les composants définis soient optimisés en termes de code.

Pour être facilement manipulables par des concepteurs non automaticiens, ces composants doivent aussi parfaitement correspondre aux matériels standard imposés dans le cadre d'un projet. Cela peut impliquer que plusieurs objets standards disposent rigoureusement des mêmes parties commande et surveillance/supervision et ne diffèrent que d'un point de vue technologique. Par exemple, une vanne à deux voies avec fin de course de DN15 en cupronickel aura exactement le même principe de contrôle commande qu'une vanne à deux voies avec fin de course de DN15 en inox.

Pour autant elles ne sont pas identiques et constituent donc des références différentes dans la bibliothèque.

Pour s'adapter aux particularités de chaque système, la bibliothèque doit également mettre à disposition des composants vides qui fournissent une trame mais dont aucun attribut n'est renseigné. Cela permet la définition de composants spécifiques n'existant qu'à peu d'exemplaires. Prenons le cas d'un système de traitement des eaux usées à bord d'un navire. La partie collecte du système peut être basée sur une technologie standard faite de vannes et de pompes. Par contre, l'unité de traitement sera vraisemblablement conçue et fournie par un expert du domaine. La gestion de cette unité passera donc par la création d'un composant spécifique. Par ailleurs, la bibliothèque doit pouvoir être enrichie de nouveaux composants standards si le besoin s'en fait sentir.

I.4.1.2 Approche descendante

Si l'approche ascendante (orientée composant) permet une modélisation fine du système à concevoir, elle peut conduire à passer à côté de problèmes globaux du système. Un peu à la manière d'une loupe qui permettrait de vérifier la précision du tracé sur un dessin mais pas le respect de la perspective globale du dessin.

Une approche descendante permet justement de limiter ce risque puisque la conception se fait par raffinage successif d'un modèle de base représentant le système dans son ensemble. Cette approche est directement issue du modèle de conception en cascade qui a émergé dans les années 70 (Royce 1970). Ce modèle part du principe que la conception est un enchaînement linéaire de tâches à réaliser dans un ordre précis. Dans le cas du BTP par exemple, cela se matérialise par le fait que la toiture ne peut être réalisée avant les fondations. L'ensemble des processus de conception séquentiels ont pour origine le modèle cascade, y compris le cycle en V qui est largement admis par la communauté industrielle.

L'approche descendante proposée au Lab-STICC dans (Frizon De Lamotte 2006) présente donc des similitudes avec le processus simplifié issu du retour d'expérience. La conception par raffinages successifs de différents modèles correspond aux différentes étapes du processus. Par ailleurs, la séparation de l'architecture et des configurations peut être assimilée à la décomposition schéma synoptique d'une part et analyse fonctionnelle d'autre part.

Comme nous l'avons montré dans notre retour d'expérience (cf. section I.1.2.2.2 p. 10) la représentation schématique du système est utilisée dès le début de la conception. Dans ce cadre une approche descendante peut répondre au besoin de formalisation méthodologique exprimé par les entreprises.

La difficulté avec ce type de démarche vient de l'implémentation des solutions spécifiées par le modèle d'entrée. En cas d'erreur dans le modèle, la détection peut intervenir tardivement dans le projet obligeant ainsi les concepteurs à reprendre le processus depuis la tâche où l'erreur a été introduite. Par ailleurs, le choix du modèle d'entrée est crucial pour l'acceptabilité de la démarche par les concepteurs (Chevallereau, Bernard et al. 2009).

Pour répondre à ce problème, l'OMG propose sa recommandation Model Driven Architecture (OMG 2003) dont l'objectif est de promouvoir l'utilisation de modèles dans le développement automatique des logiciels, elle pose notamment les concepts de modèle et de transformation qui ne sont pas sans rappeler les raffinages successifs évoqués précédemment. L'OMG fait d'ailleurs un parallèle avec le BTP : « Imaginez qu'un travailleur du bâtiment puisse prendre son plan, le mette dans une machine et d'un tour de manivelle, voir simplement apparaître les fondations du bâtiment (OMG 2003). » Les transformations de modèles, lorsqu'elles sont réalisées automatiquement, garantissent la cohérence des modèles entre eux. Par ailleurs, l'automatisation des transformations permet des gains de temps important lors de la génération des modèles.

I.4.2 Approche intégrée, notre proposition

Notre proposition prend la forme d'un flot de conception que nous illustrons sur la figure 21.

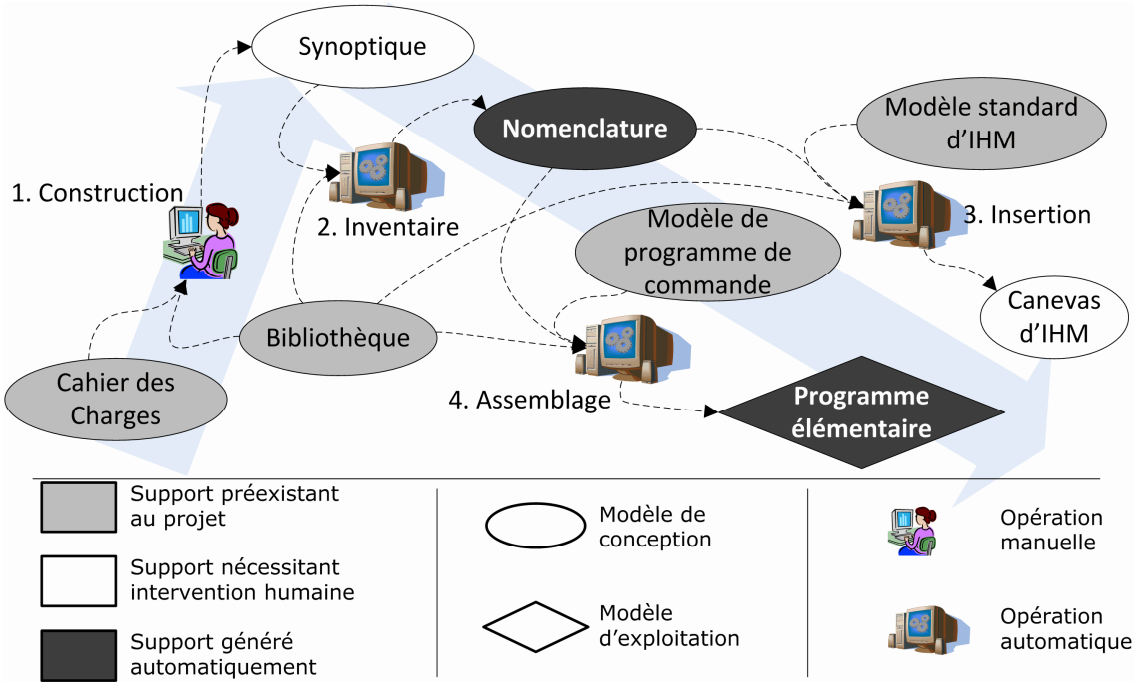


Fig. 21 Flot de conception intégré

Nous souhaitons exploiter au mieux l'expertise des concepteurs. En ce sens, nous proposons que l'expert soit l'initiateur de notre approche. Il doit cependant pouvoir s'appuyer sur des éléments de standardisation pour construire le modèle de son système lors d'une **phase ascendante**.

Néanmoins nous souhaitons mieux séparer les activités de conception et de programmation en automatisant davantage le passage de l'une à l'autre.

D'une part, la séparation de ces activités permettra aux experts de différents domaines de maintenir leurs efforts de conception sur leurs corps de métier respectifs plutôt que sur l'expression de leurs besoins aux autres corps de métier (Chevallereau, Bernard et al. 2009).

D'autre part, l'automatisation du flot de conception, suivant une **phase descendante**, rendra la production de code plus rapide et permettra donc une vérification au plus tôt dans le projet des principes d'automatisation et d'interface de supervision du système.

Notre proposition de flot de conception fait apparaître des modèles et des opérations que nous introduisons ci-après.

I.4.2.1 Les modèles

En s'appuyant sur les constats de notre retour d'expérience, nous proposons de baser notre démarche (Fig. 21) sur un schéma de principe réalisé par l'expert à partir d'une bibliothèque d'éléments standards. Notre démarche doit permettre, à partir d'un modèle métier (le synoptique sur la Fig. 21), de générer un programme de commande simple (programme élémentaire sur la Fig. 21) et un projet d'interface de supervision (canevas d'IHM sur la Fig. 21).

Le **synoptique** est un schéma fonctionnel constitué d'un ensemble de formes géométriques et de connexions relatives à l'architecture du procédé et au fonctionnement d'un système. Sa réalisation est régie par des règles très précises dépendant fortement du domaine d'application.

Les éléments de la **bibliothèque** sont les unités constitutives d'un système. Un élément est l'unité constitutive du procédé du système. L'élément peut être relatif à du matériel (une vanne, une pompe, etc.) ou à des fonctionnalités du système (alarmes, informations, etc.). La bibliothèque met en relation les différents points de vue d'un élément standard ; elle regroupe pour chaque élément les aspects relatifs à la commande, à la supervision ainsi que sa représentation sur le synoptique.

Le **canevas d'IHM** généré est une ébauche de l'application de supervision du système à concevoir. Il doit permettre la surveillance des éléments du système identifiés sur le synoptique en proposant une interface graphique dont l'agencement reprend la représentation sur synoptique. Le canevas d'IHM doit servir de base, d'une part à une démarche de conception participative basée sur des échanges avec l'utilisateur final du système et d'autre part au développement de l'application de supervision globale du système.

Le **programme élémentaire** est destiné à être implanté dans l'automate de contrôle du système. Il doit permettre le contrôle individuel des éléments du système identifiés sur le synoptique. Il peut être directement exploitable et peut également servir de base pour le développement d'un programme de contrôle global du système.

I.4.2.2 Les opérations

Nous présentons dans cette partie les quatre opérations de notre flot de conception. Des précisions sont apportées quant aux modèles intermédiaires.

A La construction

La première étape de notre démarche (**1. Construction** sur la Fig. 21) relève d'une approche ascendante. Elle vise à l'obtention du synoptique. L'expert en charge de la conception du système maîtrise les connaissances

nécessaires à la construction de ce modèle à partir des éléments stockés dans la bibliothèque et conformément au cahier des charges.

Les étapes suivantes de la démarche relèvent d'une approche descendante. Le synoptique et la bibliothèque vont subir une suite d'opérations de raffinage permettant l'obtention d'un projet d'interface de supervision et d'un programme de commande.

B L'inventaire

La seconde étape (**2. Inventaire** sur la Fig. 21) est une opération automatique qui regroupe dans un modèle unique (la nomenclature), les informations issues du synoptique et de la bibliothèque, nécessaires à la création d'une interface de supervision et d'un programme de commande.

Le synoptique lie les différents éléments d'un système et en donne une représentation globale et schématique. La nomenclature permet de lier les différents attributs d'un élément à son instance.

La nomenclature est donc le modèle central de notre démarche. C'est un ensemble d'items ; chaque item représente, soit une instance d'un élément présent sur le synoptique, soit une connexion établie entre deux de ces éléments sur le synoptique. L'ensemble des attributs des éléments indépendants des plateformes et issus de la bibliothèque sont associés à leur instance dans la nomenclature.

C L'insertion

La troisième étape est une opération automatique de génération du projet d'interface de supervision (**3. Insertion** sur la Fig. 21). Cette opération réalise l'insertion, dans un modèle standard d'IHM, des informations relatives à la supervision des éléments stockés dans la bibliothèque, en fonction des instances répertoriées dans la nomenclature.

La conception d'une IHM se fait dans le respect, à la fois des règles d'ergonomie et du code visuel du concepteur de l'IHM, généralement présenté sous la forme d'un guide de style. Microsoft® (Microsoft 2011) ou Apple® (Apple 2010) proposent par exemple chacun un guide style pour les concepteurs de logiciel destinés à leurs OS[1]. Nous proposons d'automatiser la prise en compte de ces guides en imposant aux équipes du projet, un fond visuel préalablement défini.

Afin d'intégrer au mieux les exigences ergonomiques liées à l'exploitation du système complet, le modèle standard d'IHM doit répondre à une analyse de son domaine de travail (Burns and Hajdukiewicz 2004). Cette analyse doit permettre notamment de définir l'arborescence du modèle standard. La génération automatique vient compléter la branche de cette arborescence associée au système concerné.

Ainsi, le modèle standard d'IHM est à la fois un guide de style, puisqu'il définit l'identité visuelle du produit fini, et un patron de conception puisqu'il impose des règles d'agencement des zones de l'IHM et son arborescence.

[1] OS : Operating System

C'est également un "Framework" puisque le modèle standard d'IHM est un projet de supervision sous un logiciel de conception d'application SCADA[1].

Le projet d'IHM est le modèle standard d'IHM, complété des symboles de supervision référencés dans la nomenclature.

D L'assemblage

La quatrième et dernière étape de notre démarche est une opération automatique de génération du programme de commande (**4. Assemblage** sur la Fig. 21). Cette opération réalise l'assemblage, dans un projet de programme de commande, des informations relatives à la commande des éléments stockés en bibliothèque en fonction des instances répertoriées dans la nomenclature.

Le projet de commande modélise la structure d'un programme de commande suivant l'outil d'implémentation retenu pour le projet. Dans cette structure sont assemblés les objets de commandes référencés dans la nomenclature. Ce programme permet la commande, élément par élément, du système. Un programmeur peut alors se servir de ce programme comme d'une base pour produire le programme complet permettant le contrôle global du système.

E Complément

Notre démarche se base sur une série d'opérations automatiques permettant le passage d'un modèle à un autre. Elle correspond exactement à la définition d'une approche IDM donnée par l'OMG (OMG 2003). Chaque opération de notre démarche peut ainsi être implémentée au moyen d'une ou plusieurs transformations de modèles (Mens and Van Gorp 2006). Les transformations sont définies au niveau des metamodels (Bézivin and Gerbé 2001), cela élève le niveau d'abstraction et garantit l'application de la démarche à tout modèle conforme aux métamodèles que nous avons définis.

[1] SCADA : Supervisory Control and Data Acquisition

I.5 Bilan de l'analyse du besoin

Nous avons présenté dans cette partie notre retour d'expérience industriel, issu de la campagne d'entretiens, formalisée et réalisée auprès d'agents opérationnels travaillant dans différents secteurs de l'industrie.

L'analyse du processus de conception simplifié, inspiré par ce retour d'expérience, ainsi que l'état de l'art académique présenté, nous ont permis d'identifier deux démarches académiques, à priori concurrentes, susceptibles d'améliorer sensiblement le déroulement de la conception.

Nous avons alors proposé une démarche de conception tirant partie des deux approches académiques retenues. Basée sur l'utilisation d'un modèle métier, construit d'après une démarche ascendante, elle vise à générer conjointement, par une démarche descendante, un canevas d'IHM ainsi qu'un programme élémentaire. Si l'on se replace dans le contexte de la démarche de conception simplifiée (Fig. 7) issue de notre retour d'expérience industriel, nous avons déroulé la branche issue de la définition physique du système.

A contrepied des démarches généralement présentées proposant de rendre les outils de programmation abordables par les experts métiers, nous nous proposons ici d'exploiter au maximum les outils de l'expert métier pour débuter la programmation.

La description succincte de notre flot de conception intégré doit maintenant faire place à une définition détaillée des concepts nécessaires à son implémentation ; c'est l'objet de la partie II.

Partie. II Implémentation de la proposition

Cette partie détaille l'implémentation de notre proposition pour la génération conjointe de programme de commande et d'interface de supervision. Rappelons que cette proposition prend la forme d'un flot de conception comportant quatre étapes.

La première étape de ce flot est une opération de **construction**. Elle vise la création d'un modèle métier représentatif du système : le synoptique. A partir d'une bibliothèque d'éléments standards et du cahier des charges, l'expert en charge de la conception établit le synoptique du système. Nous détaillons les concepts relatifs à cette opération dans le premier chapitre de cette partie.

La seconde étape de notre flot de conception est une opération d'**inventaire**. Elle vise à rassembler automatiquement, dans une nomenclature, les informations nécessaires à la génération de la commande et de la supervision. Ce modèle central à notre démarche est généré à partir de la bibliothèque et du synoptique. Nous présentons dans le deuxième chapitre de cette partie, l'ensemble des concepts et des transformations relatifs à cette opération.

La troisième étape est une opération d'**insertion**. Les informations relatives à la supervision, contenues dans la bibliothèque et référencées dans la nomenclature sont insérées dans un modèle standard d'IHM afin de produire un canevas d'IHM, ébauche de l'interface de supervision du système. Les concepts et les transformations de cette opération sont détaillés dans le troisième chapitre de cette partie.

La quatrième et dernière étape de notre flot est une opération d'**assemblage**. Les informations relatives à la commande du système, stockées dans la bibliothèque et référencées dans la nomenclature, sont utilisées pour la génération d'un programme de commande élémentaire. Nous présentons les concepts et les transformations de l'opération d'assemblage dans le quatrième chapitre de cette partie.

Le flot de conception proposé est **outillé** et **appliqué** sur un exemple concret issu de notre support d'étude, présenté en annexe. La description des outils, leur implémentation ainsi que les résultats de l'application de la démarche sont présentés dans le cinquième chapitre de cette partie.

II.1 Opération de construction

L'opération de **construction** (Fig. 22) est la première de notre flot de conception intégrée. Elle vise à l'obtention, par une **démarche ascendante**, d'un modèle de synoptique à partir d'un modèle de bibliothèque et du cahier des charges. Définissons tout d'abord ces concepts avant de présenter en détail l'implémentation de cette opération.

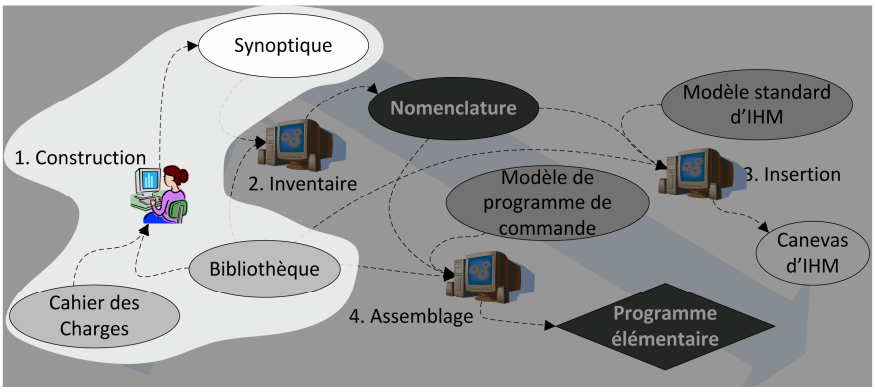


Fig. 22 Opération de Construction

II.1.1 Le cahier des charges

Le modèle du cahier n'est pas formalisé, il s'agit d'un modèle de type CIM[1] (OMG 2003). Dans notre cas, le cahier des charges regroupe l'ensemble des exigences auxquelles l'expert doit se soumettre pour concevoir son système (cf. section I.1.2.2.1 p. 10). Il n'est pas contenu dans un document unique et peut évoluer au fur et à mesure de la conception, en fonction de l'évolution des normes ou des besoins de l'utilisateur final.

II.1.2 La bibliothèque

La mise en œuvre d'une politique de standardisation au sein d'une entreprise passe nécessairement par la mise à disposition des éléments standardisés aux concepteurs au travers d'un catalogue, ou dans notre cas d'une bibliothèque.

Nous avons plusieurs fois abordé la notion d'objet, de composant ou d'élément. Ce concept mérite d'être approfondi. Un élément est « une partie identifiable à l'intérieur d'un ensemble qu'elle contribue à former. (…) Les éléments d'un circuit électrique. (…) Par extension. Ce qui vient s'intégrer à un ensemble, ce qui peut compléter un ensemble » (extrait de la 9ème édition du dictionnaire de l'Académie française).

Dans notre cas, nous retenons la définition suivante.

[1] CIM : Computationnal Independant Model

Un élément est l'unité constitutive du procédé du système. L'élément peut être relatif à du matériel (une vanne, une pompe, etc.) ou à des fonctionnalités du système.

Déf. 1 Élément

Notre bibliothèque a pour vocation de rassembler les informations relatives à ces éléments et de les classer par items selon le point de vue des différents concepteurs.

II.1.2.1 Le concept de vue

Nous avons déjà évoqué les origines multiples des concepteurs sur les projets qui nous intéressent. Chacun de ces intervenants, suivant son corps de métier (mécanicien, automaticien, électricien, intégrateur, etc.), a sa propre vision d'un élément.

Considérons l'exemple d'une vanne. Le mécanicien y verra un ustensile de robinetterie permettant le sectionnement d'un circuit. L'automaticien y verra un traitement permettant de gérer la commande et la surveillance de cette vanne. L'intégrateur, lui, verra les dimensions de la vanne et son dispositif de raccordement (bride, fileté, etc.). Enfin, l'électricien considérera la vanne au travers de ses interfaces de bornage et de la puissance qu'elle consomme.

Chaque intervenant dispose également d'outils qui lui sont propres et dans lesquels il doit pouvoir retrouver ces éléments (schématique de procédé, schématique électrique, maquette d'intégration 3D, logiciel de programmation, etc.).

Cette notion de **subjectivité** des concepteurs nous permet d'introduire le **concept de vue** ou de point de vue d'un élément.

Reprenons la définition posée par Mouchard (Mouchard 2002) et reprise par Lallican (Lallican 2007); « une vue correspond à un modèle d'un aspect du composant avec un niveau de granularité et un point de vue propre ». Cette définition correspond à la notion introduite précédemment. Pour autant, un élément n'est pas un composant au sens de Mouchard puisque celui-ci est uniquement dédié à la production du code de commande, de supervision et de simulation des éléments, alors que l'élément désigne également le matériel derrière ce code, et toutes les autres vues possibles. Ainsi, on peut dire qu'un élément contient un composant et d'autres vues.

De même, une vue d'un élément ne peut se résumer à la notion de modèle puisque l'élément est aussi bien le matériel qui sera implanté que « l'objet » (au sens des bases de données) enregistré dans le système d'information.

Le concept d'élément est donc très vaste. En qualité de concepteurs du système de contrôle commande, nous nous devons de restreindre notre étude aux vues qui nous intéressent. Les exemples qui suivent montrent les différentes vues que nous avons créées pour une vanne motorisée à 2 voies.

II.1.2.1.1. La vue synoptique

La vue synoptique permet la représentation de l'élément sur le schéma synoptique. Elle se caractérise par un symbole nommé, identifié, positionné

et disposant éventuellement d'informations complémentaires (comme les seuils de niveau associés à une détection).

Dans notre cas, la vue synoptique est un fichier décrivant la forme du symbole et les informations qui lui sont associées au format XML, interprétable par Microsoft® Visio. Une fois insérée sur le schéma, la vue synoptique en devient un élément à part entière. La position du symbole dans le schéma est ajoutée aux attributs.

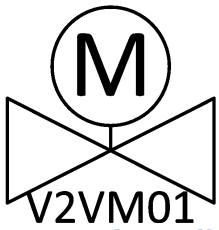


Fig. 23 Vue synoptique d'un élément

La figure 23 représente le symbole associé à une vanne motorisée à 2 voies pour les schémas de tuyauterie et instrumentation (ANSI 1992). Ce type de schéma peut être considéré comme un synoptique.

II.1.2.1.2. La vue supervision

La seconde vue qui nous intéresse est celle qui représente l'élément sur l'IHM. On l'appellera la vue de supervision (Fig. 24). Cette vue se caractérise également par un ou plusieurs symboles.

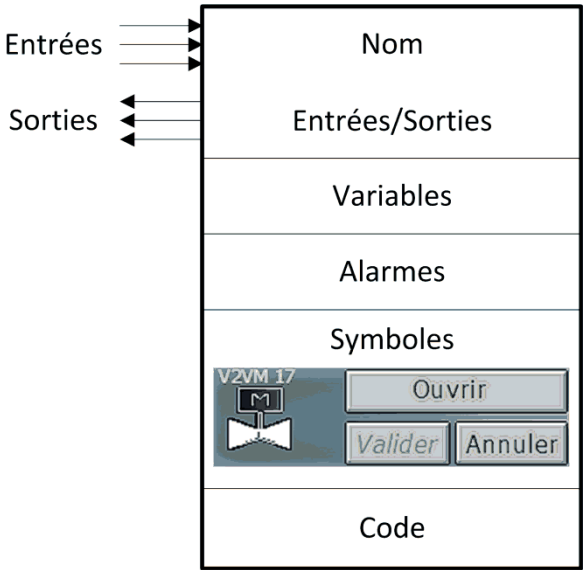


Fig. 24 Structure d'un objet de supervision

Mais cette fois-ci on y associe un objet logiciel qui gère l'interface visuelle de ces symboles (y compris la liste d'alarme associée à l'élément). Les animations ou les alarmes sont conditionnées par des variables internes ou issues de la base de données du superviseur (*Entrées* sur la Fig. 24). Les actions de l'opérateur sur le symbole entraînent une mise à jour des variables de commande destinées à la partie opérative (*Sorties* sur la Fig. 24).

La figure 25 présente la vue de supervision d'une vanne motorisée à 2 voies implémentée sur Panorama E2.

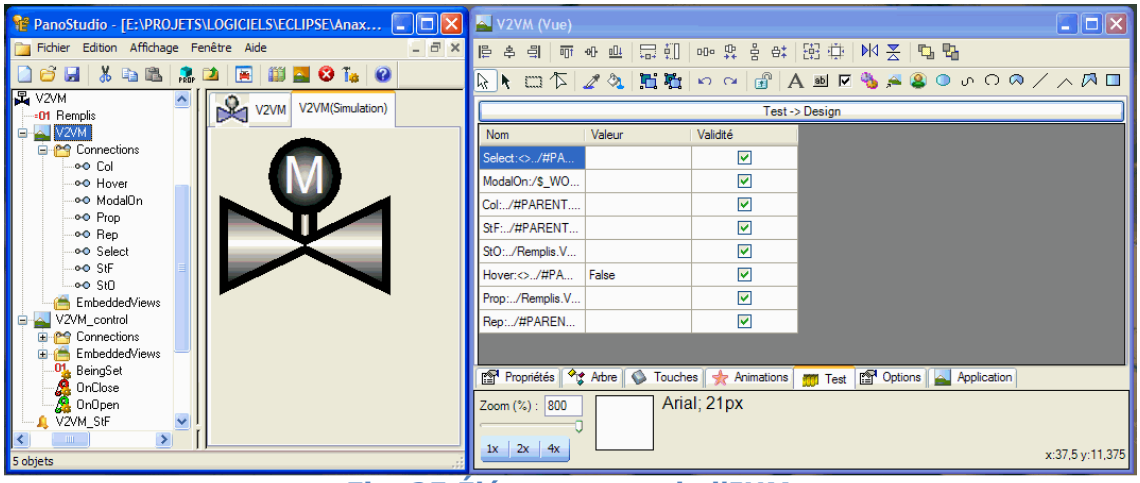


Fig. 25 Éléments vus de l'IHM

On y retrouve le symbole animé dans la rubrique V2VM, les boutons de commande dans la rubrique V2VM_control, une alarme associée à l'objet de supervision (V2VM_StF sur la Fig. 25), les variables d'entrée et de sortie internes de cet objet ainsi que le code nécessaire à l'animation du symbole (*BeingSet*, *OnClose* et *OnOpen* sur la Fig. 25).

II.1.2.1.3. La vue commande

Une troisième vue est nécessaire pour appliquer notre démarche. Il s'agit de la vue commande. Elle se caractérise par un traitement élémentaire que nous définissons de la manière suivante.

Un traitement élémentaire est une partie d'un programme qui met en relation les variables d'entrée et de sortie « au niveau » d'un élément.

Déf. 2 Traitement élémentaire

Dans notre cas, nous considérons un traitement élémentaire comme un POU[1] (Fig. 26) régi par la norme IEC-61131 (IEC 2003). Il se compose d'interfaces qui le mettent en contact avec son environnement logiciel, de variables internes et d'un code source qui s'exécute en fonction de l'état des interfaces et des variables.

On y retrouve :

- Les Entrées / Sorties qui peuvent être des variables de commande à destination des sorties physiques, des variables de surveillance à destination de la supervision, des variables de commande issues de la supervision ou des variables issues des entrées physiques de l'automate.

[1] POU : Programmable Organization Unit

- Les variables internes du bloc fonctionnel qui, si elles sont publiques, peuvent permettre de paramétrer l'élément (comme la valeur d'une temporisation associée à une vanne).
- Le code qui régit la mise à jour des sorties du bloc.

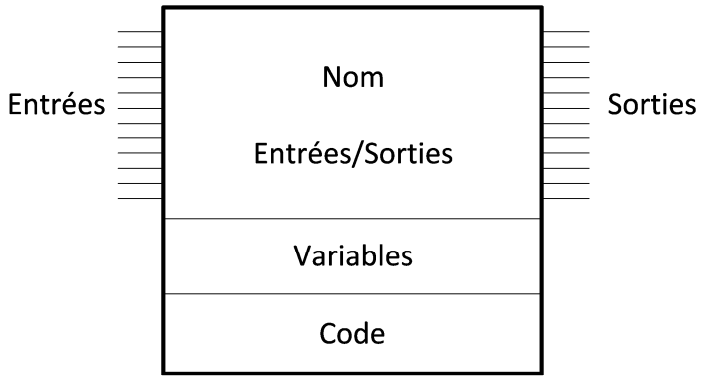


Fig. 26 Structure d'un Traitement élémentaire

Le traitement élémentaire peut aussi intégrer une gestion de priorité de commande (si plusieurs lieux ou plusieurs modes de commande sont prévus) ou toute autre opération prévue par la standardisation, pour peu qu'elle se rapporte à l'élément considéré. On peut par exemple prévoir sur le bloc plusieurs entrées de commande pour l'ouverture d'une vanne, pouvant provenir de différents lieux de commande, la priorité entre ces commandes étant gérée au sein du bloc.

Par contre, à un élément matériel on peut vouloir associer plusieurs éléments fonctionnels.

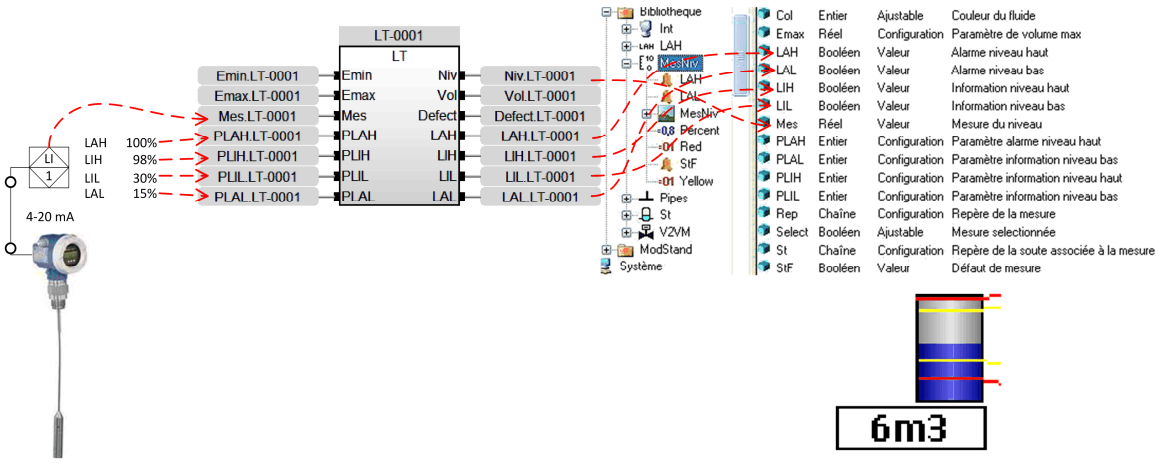


Fig. 27 Exemple d'utilisation de traitements élémentaires

Ainsi, au matériel de mesure de niveau, on pourra associer des paramètres de mise à l'échelle ainsi que des seuils de détection d'alarme (Fig. 27).

II.1.2.2 Les interfaces

Il est important de noter que, par nature, l'élément n'est pas une entité abstraite projetable sur n'importe quel matériel. L'élément est directement lié

à un matériel précis et les vues qui lui sont associées ne sont pas toutes indépendantes des plateformes sur lesquelles elles seront implantées.

Dans notre cas, les vues de commandes et de supervision sont des objets logiciels codés dans des langages déjà prévus pour des cibles précises. S'il est vrai que les langages d'automatisme utilisés pour la partie commande (IEC 2003) sont relativement ouverts et transposables d'un automate à un autre, avec quelques réserves toutefois, la vue de supervision est codée sous Panorama E2 qui est un logiciel propriétaire comme la plupart des superviseurs.

Afin de rendre notre démarche la plus indépendante de ces supports, nous devons disposer d'un moyen de description approprié qui permettra la propagation des informations relatives à la commande et à la surveillance tout au long de notre démarche.

Pour ce faire, nous avons défini un modèle d'interface centré sur les informations possiblement échangées entre les différentes vues d'un même objet. A chaque élément de la bibliothèque nous associons donc une liste d'interface (Fig. 28), caractérisée par l'information portée par cette interface et par la vue de l'élément à laquelle cette interface est rattachée. Cette interface est typée suivant la norme IEC 61131-3 (IEC 2003) et nous lui affectons un genre (Entrée, Sortie, Constante, Variable).

Reference	Information	Designation	Type	Kind	View
V2VM	CtrlO	Commande bistable issue de la supervision (1=ouverture; 0=fermeture)	BOOL	Input	VCmd
V2VM	Tempo	Temporisation de manœuvre de la vanne	TIME	Var	VCmd
V2VM	FdcO	Fin de course d'ouverture	BOOL	InPhy	VCmd
V2VM	FdcC	Fin de course de fermeture	BOOL	InPhy	VCmd
V2VM	StO	Etat "vanne ouverte"	BOOL	Output	VCmd
V2VM	StF	Etat "vanne fermée"	BOOL	Output	VCmd
V2VM	Defect	Etat "défaut vanne"	BOOL	Output	VCmd
V2VM	CmdO	Commande d'ouverture vers la vanne	BOOL	OutPhy	VCmd
V2VM	CmdC	Commande de fermeture vers la vanne	BOOL	OutPhy	VCmd
V2VM	CtrlO	Commande bistable issue de la supervision (1=ouverture; 0=fermeture)	BOOL	Output	VSup
V2VM	StO	Etat "vanne ouverte"	BOOL	Input	VSup
V2VM	StC	Etat "vanne fermée"	BOOL	Input	VSup
V2VM	StF	Etat "défaut vanne"	BOOL	Input	VSup
V2VM	Col	Couleur du fluide	INT	Var	VSup
V2VM	Col1	Couleur du fluide1	INT	Var	VSup
V2VM	Prop1	Animation de la propagation1	BOOL	Var	VSup
V2VM	Prop2	Animation de la propagation2	BOOL	Var	VSup
V2VM	Rep	Repère de la vanne	STRING	Const	VSup
V2VM	Select	Vanne selectionnée	BOOL	Var	VSup

Fig. 28 Interfaces de la vanne à 2 voies, motorisée

Cette liste peut très simplement être codée en langage XML ; la figure 29 présente le code correspondant à l'interface « État Vanne Ouverte » de l'élément vanne.

Ces interfaces seront enrichies lors de la création du modèle de nomenclature puis serviront à l'instanciation des vues auxquelles elles sont associées.

```xml
<Interface>
        <Reference>V2VM</Reference>
        <Information>StO</Information>
        <Designation>État Vanne Ouverte</Designation>
        <Type>BOOL</Type>
        <Genre>Input</Genre>
        <Vue>VSup</Vue>
</Interface>
```

Fig. 29 Exemple d'interface en XML

Dans le cadre de notre démarche nous ne définissons que les interfaces des blocs de commande et de supervision, mais il est intéressant de noter que les interfaces pourraient également servir à définir le bornage de l'automate. Il suffirait pour cela de rajouter une vue bornage pour définir les entrées et sorties physiques à associer à un élément.

II.1.2.3 Un métamodèle pour la bibliothèque

Pour appliquer notre démarche, nous mettons en œuvre les techniques de l'ingénierie dirigée par les modèles. L'implémentation de transformations de modèle nécessite la définition préalable des métamodèles sur lesquels les règles des transformations sont basées.

La bibliothèque regroupe donc l'ensemble des données relatives aux unités constitutives d'un système, appelées éléments. Chaque facette subjective de ces entités est appelée vue. Certaines de ces vues sont assimilables à des objets logiciels (traitement élémentaire ou composant de supervision), à des objets de base de données (documentation, référence, etc.) ou à des modèles (vue synoptique, vue 3D, etc.). Chaque intervenant sur le projet n'est concerné que par un nombre limité de vues mais elles sont liées entre elles par la bibliothèque.

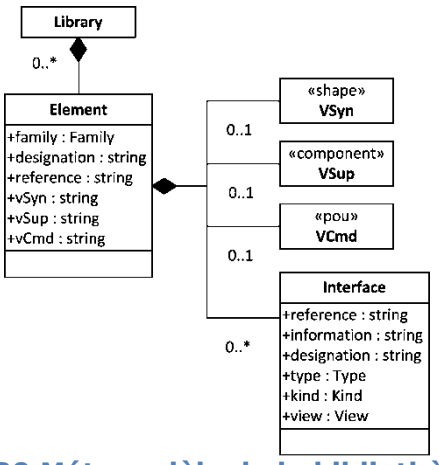


Fig. 30 Métamodèle de la bibliothèque

La figure 30 est le méta modèle de notre bibliothèque (*Library* sur la Fig. 30). Nous voyons qu'elle contient les éléments (*Element* sur la Fig. 30) qui sont caractérisés (Fig. 31) par les attributs *family* et *reference*, et rattachés à leurs vues par les attributs *VSyn*, *VSup* et *VCmd* qui sont des URI[1].

[1] URI : Uniform Resource Identifier, il s'agit dans notre cas d'une chaîne de texte représentant le chemin d'accès à une vue.

Family	Designation	Reference	VSyn	VSup	VCmd
Process	Vanne 2 voies motorisée	V2VM	V2VM\Vsyn\	V2VM\Vsup\	V2VM\VCmd\

Fig. 31 Caractérisation de la vanne 2 voies, motorisée

Chaque élément contient une vue synoptique *VSyn* stéréotypée suivant les fichiers *Shape* de Microsoft® Visio. Il peut également contenir une vue commande *VCmd* stéréotypée suivant les « Program Organisation Unit » (POU) de PLCOpen (PLCOpen 2009), et il peut contenir une vue supervision *VSup* stéréotypée suivant les composants Panorama E2. Enfin il peut contenir des interfaces (*Interface* sur la Fig. 30).

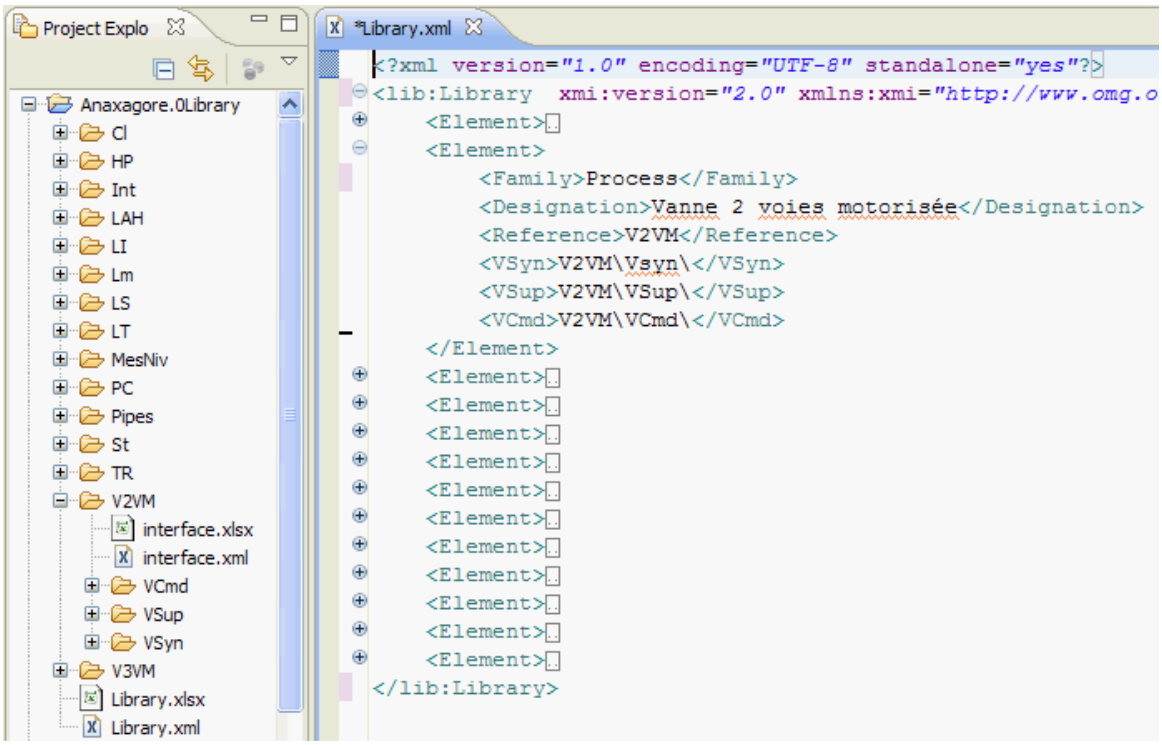


Fig. 32 Structure de la bibliothèque

La figure 32 présente la structure de notre bibliothèque. Les données de caractérisation des éléments sont regroupées dans un fichier *Library.xml* se trouvant sur la racine de la bibliothèque. Chaque élément de la bibliothèque dispose d'un sous-répertoire contenant les interfaces sous la forme d'un fichier *interface.xml* ainsi qu'un répertoire par vue de l'élément (*Vcmd*, *VSup* et *VSyn* sur la Fig. 32).

II.1.3 Le synoptique

Le point d'entrée de notre démarche est le synoptique. Mais qu'est-il exactement ? En fait, un synoptique désigne une présentation, en général graphique, qui permet de saisir d'un seul coup d'œil un ensemble d'informations liées ou un système complexe. D'ailleurs, l'adjectif synoptique s'applique à quelque chose « qui permet d'embrasser, de saisir d'un même coup d'œil les diverses parties d'un ensemble, qui en offre une vue

générale » (extrait de la 8ème édition du Dictionnaire de l'Académie française). Nous retenons la définition suivante.

Le synoptique est un schéma structuro-fonctionnel constitué d'un ensemble de symboles et de connexions relatifs à l'architecture du procédé et au fonctionnement d'un système. Sa réalisation est régie par des règles très précises dépendant fortement du domaine d'application.

Déf. 3 Synoptique

Pour tester notre démarche nous considèrerons le synoptique de la figure 33, extrait du synoptique de l'installation de production de distribution d'eau douce sanitaire présenté en annexe.

Ce synoptique représente une soute (St3 sur la Fig. 33) contenant au maximum 10m^3 d'eau déminéralisée, isolée du réseau de remplissage par une vanne motorisée (V2VM09 sur la Fig. 33). La soute est instrumentée par un indicateur de niveau mécanique (LI0003 sur la Fig. 33), un transmetteur de niveau (LT0003 sur la Fig. 33) et un détecteur de niveau (LS0003 sur la Fig. 33).

Rappelons que « la description d'un procédé, d'un équipement ou d'une installation à l'aide de schémas utilise un langage graphique symbolique, qui doit être commun et universel pour être compris de tous » (Auroy 1999).

Il s'agit donc bien d'un langage qui, dans notre cas, est cadré par la norme ANSI/ISA-5.1 (ANSI 1992), dont l'équivalent français est la NF E 04-202 pour les « Process and Instrumentation Diagram » (PID). Notre langage est également spécifique à un domaine (celui du génie des procédés).

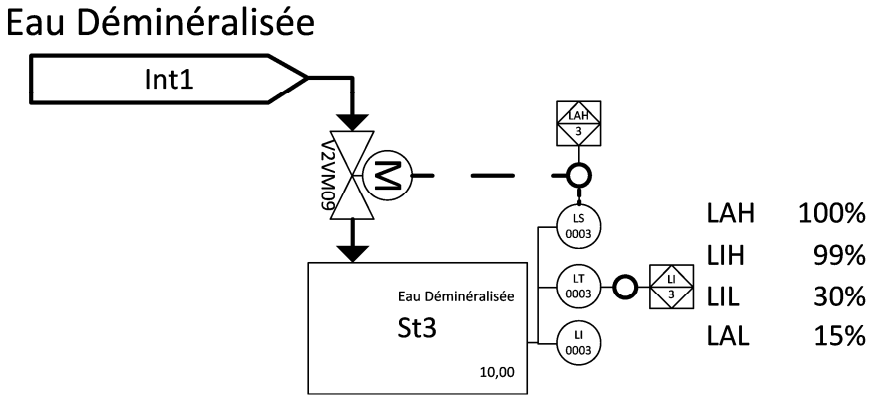


Fig. 33 Exemple de synoptique

La création d'un langage de conception peut s'aborder de deux manières suivant les concepts de l'OMG (Turki 2008).

Ainsi la manière la plus simple pour mettre en œuvre un tel langage est de créer un profil UML pour particulariser le langage générique. Cette solution offre l'avantage de compatibilité de tous les standards et outils compatibles avec UML pour réaliser notre spécification. À l'inverse, cette solution nécessite au préalable de maîtriser UML, la syntaxe restant identique.

Comme différents problèmes requièrent différentes abstractions (Thomas 2003), nous préférons la solution qui consiste à créer un DSL[1]. Si cette solution nous impose de recréer des outils spécifiques, elle nous permet tout de même de définir exactement notre langage et les modèles associés. Nous nous attacherons tout de même à respecter la spécification « Meta Object Facility » (OMG 2004) pour favoriser l'emploi des outils déjà développés par la communauté dans le domaine de l'ingénierie dirigée par les modèles.

Un DSL est caractérisé par un métamodèle et une syntaxe (Frizon De Lamotte 2006) auxquels on peut adjoindre des règles supplémentaires sous forme de contraintes OCL[2] par exemple (OMG 2010).

La syntaxe de notre langage est fournie par la norme ANSI/ISA 5.1 (ANSI 1992) et nous proposons le métamodèle suivant pour le synoptique (cf. Fig. 34).

Nous voyons ici que le synoptique (*Synoptic* sur la Fig. 34) est référencé par rapport à un domaine (attribut *domain* sur la Fig. 34), à un système (attribut *system* sur la Fig. 34) et à un sous-système (attribut *subSystem* sur la Fig. 34). Ainsi, conformément à la hiérarchie d'abstraction présentée en annexe, notre cas d'application appartient au domaine PF (pour « contrôler la plate-forme propulsée »), au système Aux (pour « localisation, apparence et état des dispositifs auxiliaires ») et au sous-système EdS (pour « eau douce sanitaire »).

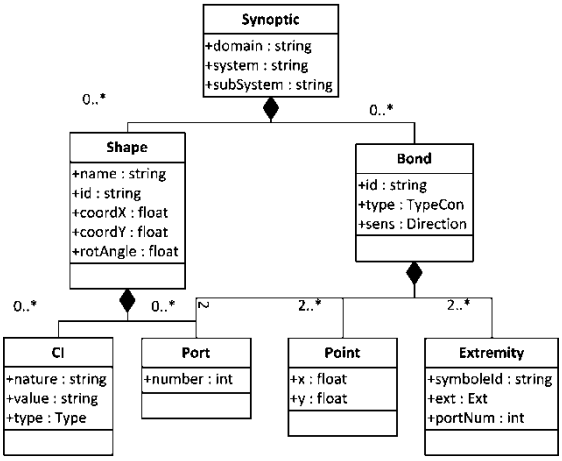


Fig. 34 Métamodèle d'un synoptique

Un synoptique se compose de symboles (Shape sur la Fig. 34) et de connexions (Bond sur la Fig. 34). Ces termes méritent précisions et nous les apportons dans les paragraphes suivants. En pratique, le synoptique est un fichier informatique de schématique. Nous retenons Microsoft® Visio pour la réalisation de ces schémas. Ce logiciel de la suite Microsoft® Office est largement répandu dans les usages professionnels. Il permet par ailleurs une grande personnalisation de la palette et de l'environnement de dessin.

[1] DSL: Domain Specific Language
[2] OCL: Object Constraint Language

II.1.3.1 Le symbole

Le symbole est une instance de la vue synoptique de la bibliothèque dans le synoptique. La figure 35 présente l'instanciation de la vue synoptique d'une information de seuil remontée à la supervision. On y voit le métamodèle d'un symbole, la représentation graphique d'un symbole sur le synoptique et sa représentation au format XML.

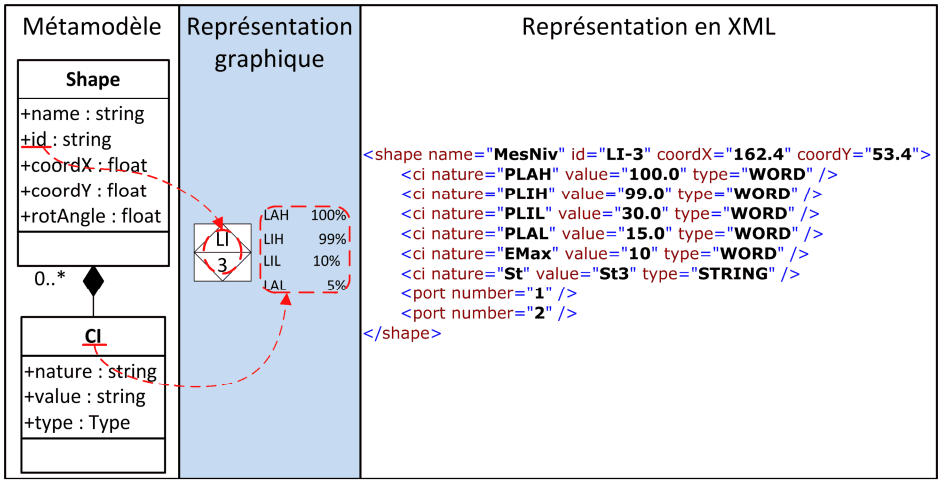


Fig. 35 Symbole d'un synoptique

Le nom du symbole (*Shape* sur la Fig. 35) est MesNiv pour « mesure de niveau ». Son identifiant est LI-3, pour « Level Indicator 3 » en accord avec la norme ANSI/ISA 5.1 (ANSI 1992). Ce symbole correspond à une information de niveau affichée en supervision. Elle dispose de quatre informations complémentaires (*CI* sur la Fig. 35) correspondant aux seuils de la détection sur la plage de mesure du transmetteur de niveau associé à l'information. L'information « niveau haut », notée *LIH*, sera vraie pour une mesure du transmetteur de niveau correspondant à 99% de sa plage de mesure.

Le symbole dispose également de connecteurs (*Port* sur la Fig. 34) qui permettent, lors de la construction du synoptique, d'établir des connexions entre symboles.

II.1.3.2 La connexion

La définition de la vue synoptique d'un élément a fait apparaître le concept de connexion que nous allons détailler.

On lui trouve deux définitions. Une première au sens de relation ou rapport étroit entre deux entités. En technique, il s'agit d'une liaison entre des conducteurs, des organes ou des machines (extrait de la 9ème édition du dictionnaire de l'Académie française). Nous retenons la définition suivante.

Une connexion est un lien entre plusieurs éléments.

Déf. 4 Connexion

La figure 36 présente un exemple de connexion établie sur un synoptique.

Une connexion dispose d'un identifiant (attribut *id* sur la Fig. 36). Pour ne pas surcharger le synoptique, par défaut, cet attribut n'est pas affiché sur le schéma.

Chaque connexion peut recouvrir une réalité matérielle, comme un tuyau ou un câble, ou représenter un lien immatériel comme un lien logiciel. La connexion est donc typée (attribut *type* sur la Fig. 36) suivant la norme ANSI/ISA 5.1 (ANSI 1992).

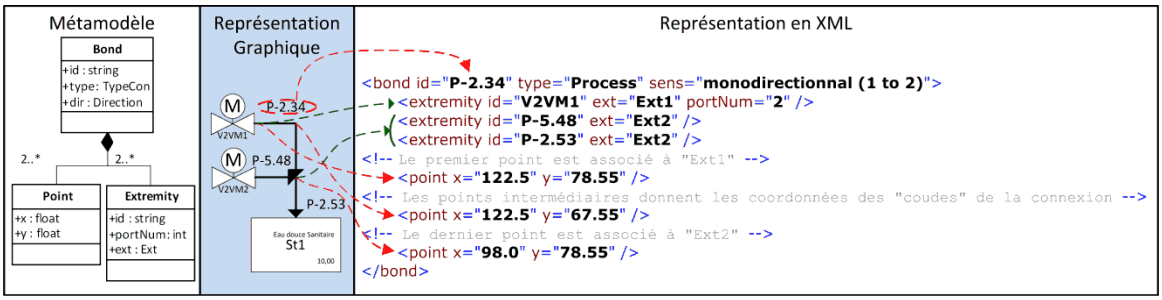


Fig. 36 Connexion d'un synoptique

Les différents types de connexion de cette norme sont présentés figure 37.

Une connexion représentant une ligne de tuyauterie peut être orientée. Dans ce cas l'attribut *sens* (Fig. 36) prend la valeur *monodirectionnal* et la connexion est orientée d'Ext1 à Ext2. Dans tous les autres cas, l'attribut *sens* prend la valeur *bidirectionnal*.

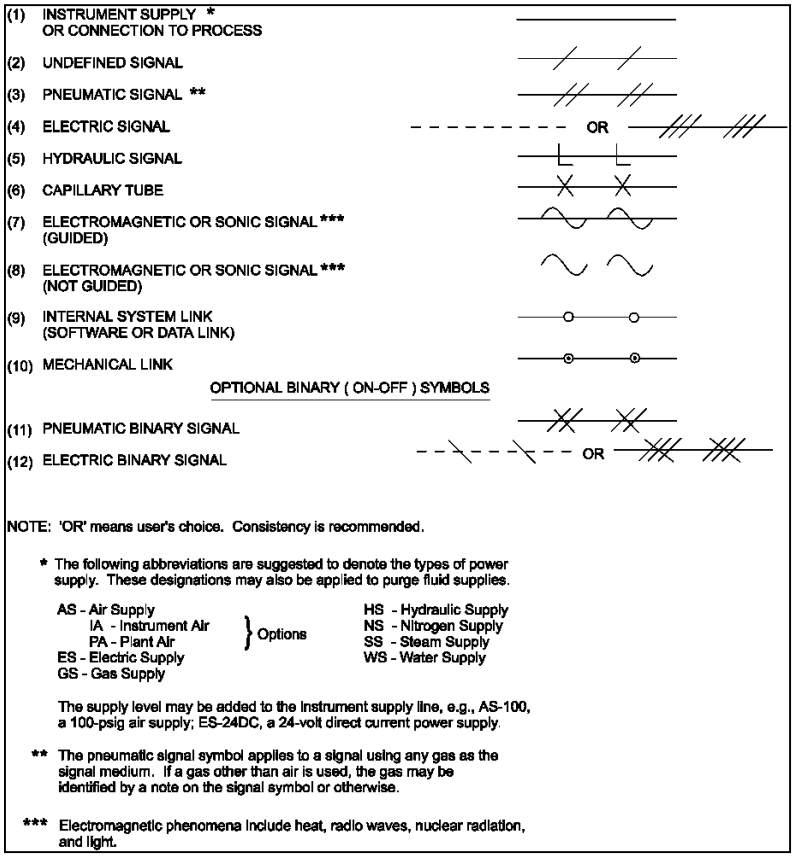


Fig. 37 Type de connexion suivant la norme ANSI/ISA 5.1

Une connexion s'établit entre les deux points Ext1 et Ext2. Chaque extrémité de la connexion est affectée à l'un de ces deux points (attribut *ext* sur la Fig. 36). Les extrémités (*Extremity* sur la Fig. 36) référencent les objets liés à ces points. Il peut s'agir d'autres connexions (P-5.48 et P-2.53 sur la Fig. 36) ou d'un symbole (V2VM1 sur la Fig. 36). Dans le cas d'un symbole, le numéro du port (attribut *portNum* sur la Fig. 36) auquel la connexion est reliée est renseigné.

Les classes *Point* contiennent des coordonnées géométriques. La première instance de cette classe donne les coordonnées du point Ext1, la dernière instance donne les coordonnées du point Ext2. Les instances intermédiaires donnent les coordonnées des « coudes » de la connexion.

II.1.4 Transformation d'épuration

Dans les précédents paragraphes, nous avons présenté les concepts relatifs à l'opération de construction (cahier des charges, bibliothèque et synoptique). Nous présentons maintenant le déroulement de cette opération.

L'essentiel de l'opération de construction est réalisé manuellement par l'expert en charge de la conception du système. Le schéma est saisi sous le logiciel Microsoft® Visio à l'aide des vues synoptique de la bibliothèque. Néanmoins, afin de garantir la généricité de notre démarche, nous avons implémenté une transformation d'**épuration** (Fig. 38) dont l'objectif est de produire, à partir d'un schéma Visio, un synoptique conforme à notre metamodel.

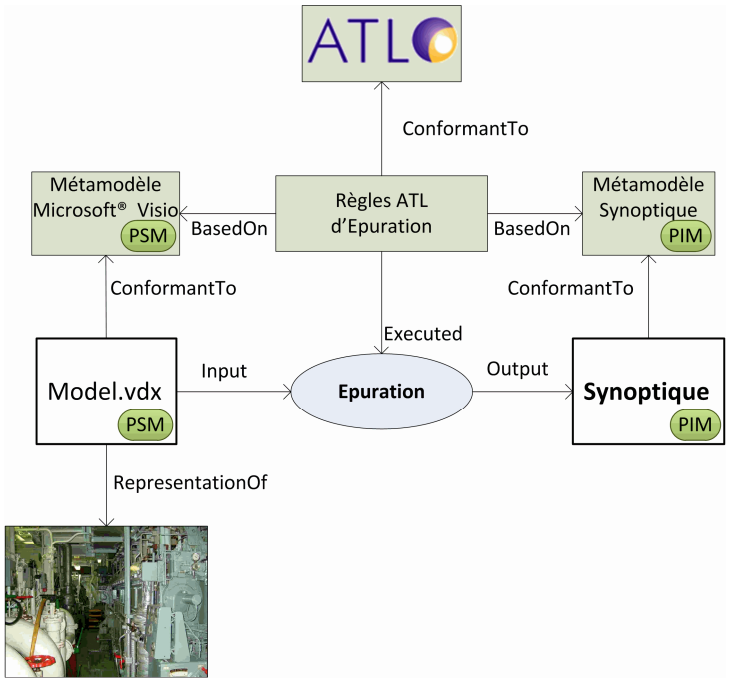


Fig. 38 Transformation d'épuration

Cette transformation exogène verticale (cf. section I.3.3 p. 31) nous permet d'obtenir un modèle de synoptique indépendant de toute plate-forme

(PIM) à partir d'un fichier *model.vdx* dépendant de la plate-forme Visio (PSM), en ne conservant que les informations nécessaires à notre démarche.

Nous avons extrait le métamodèle de Microsoft Visio à partir de son schéma XML. Ce dernier fait partie du SDK téléchargeable sur centre de téléchargement de Microsoft® (Microsoft 2011).

Le métamodèle de Microsoft® Visio comporte de nombreuses classes ; seule une partie d'entre elles nous sont utiles. La figure 39 présente un extrait du métamodèle de Microsoft® Visio regroupant les classes qui nous intéressent.

Les informations permettant de référencer le synoptique se trouvent dans les sous-classes *PropType* associées aux classes *DocumentSheetType*.

Dans Microsoft® Visio, une connexion est représentée par une forme (*ShapeType*) à une seule dimension géométrique contrairement aux symboles qui sont des formes (*ShapeType*) à deux dimensions. Les liens entre les symboles et les connexions sont établis dans les sous-classes *ConnectionType*.

Les classes *MasterType* contiennent les vues synoptiques de la bibliothèque. Elles permettent d'afficher les catalogues de symboles disponibles sur la palette du logiciel et utilisables pour dessiner le synoptique. Les sous-classes *ShapeType* associées aux classes *MasterType* permettent notamment d'obtenir l'attribut *name* de nos symboles.

Les sous-classes *ShapeType* associées aux classes *PageType* contiennent les informations relatives aux symboles et aux connexions instanciées sur le schéma. Elles permettent d'obtenir l'attribut *id*, les coordonnées du symbole et les valeurs des informations complémentaires associées.

La transformation d'épuration est composée de trois règles :

1. La règle **Visio2Synoptic** traite les informations relatives au document (domaine, système et sous-système). Elle permet également la création des instances des symboles et des connexions qui sont complétées par les autres règles. Elle est exécutée une fois lors de la transformation.
2. La règle **VisioShape2SynopticShape** traite les informations relatives à un symbole. Elle permet de renseigner les attributs de la classe *Shape* et de créer les classes *Port* et *CI* associées. Elle est exécutée pour chaque symbole référencé lors de la première règle.
3. La règle **VisioShape2Bond** traite les informations relatives aux connexions. Cette règle permet de renseigner les attributs de la classe *Bond* et de créer les classes *Extremity* et *Point* associées. Elle est exécutée pour chaque connexion référencée lors de la première règle.

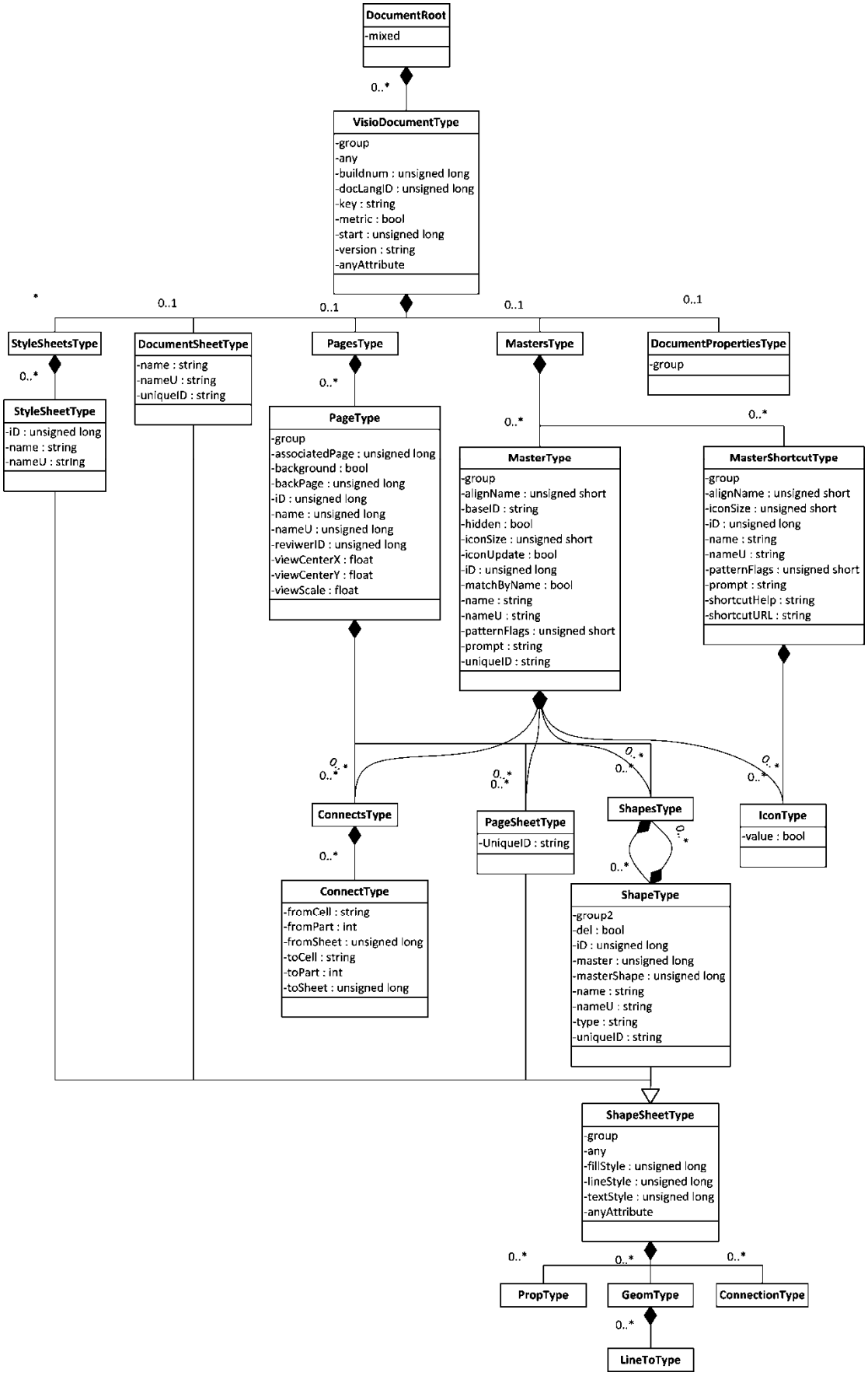


Fig. 39 Extrait du métamodèle de Microsoft Visio

A l'issue de cette première opération, nous disposons d'un modèle de synoptique indépendant de toute plateforme. Ce synoptique est une synthèse des connaissances de l'expert concernant l'architecture du système qu'il conçoit. L'expert a utilisé les vues synoptiques de la bibliothèque pour établir son schéma en suivant une **démarche ascendante**. Nous pouvons maintenant croiser ces informations avec celles des autres vues de la bibliothèque pour générer automatiquement un programme de commande et une interface de supervision.

II.2 Opération d'inventaire

Une fois le synoptique dessiné, pour préparer la création d'une interface de supervision et d'un programme de commande, nous devons regrouper dans un modèle les informations nécessaires à l'instanciation des différents objets logiciels. L'opération d'**inventaire** (Fig. 40) vise à l'obtention de ce modèle central : la nomenclature. Il s'agit de la deuxième opération de notre flot de conception intégré ; elle marque le début de la phase **descendante** de notre démarche. Elle reçoit en entrée un modèle de synoptique (précédemment défini) et un modèle de bibliothèque (précédemment présenté) et génère la nomenclature. Avant de présenter les différentes transformations de l'inventaire, nous définissons le concept de nomenclature.

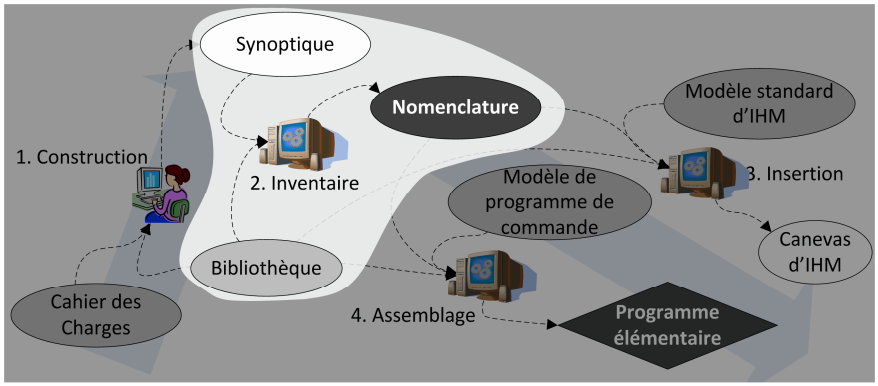


Fig. 40 Opération d'Inventaire

II.2.1 La nomenclature

Une nomenclature est définie comme « un relevé systématique des éléments qui constituent un ensemble » (extrait de la 9ème édition du dictionnaire de l'Académie française). Nous retenons la définition suivante.

La nomenclature est un ensemble d'items ; chaque item représente une instance d'un élément du système à laquelle sont associés des informations.

Déf. 5 Nomenclature

Généralement une nomenclature est associée à un schéma et contient les repères du schéma qu'elle lie à des références matérielles.

Nous proposons de compléter cette nomenclature par les éléments non matériels et d'y associer également des attributs spécifiques (Fig. 41).

Alors que le synoptique lie les différents éléments d'un système et en donne une représentation globale et schématique, la nomenclature permet de lier les différents attributs d'un élément de la bibliothèque à son instance ainsi que les connexions établies entre les éléments.

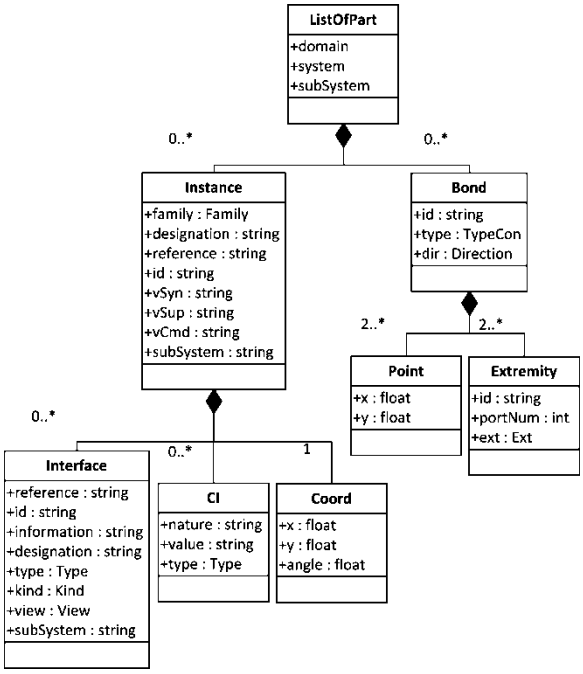


Fig. 41 Métamodèle de la nomenclature

La nomenclature est référencée de la même manière que le synoptique. La figure 42 présente l'en-tête du fichier de la nomenclature de notre cas d'application (Fig. 33). On retrouve les attributs *domain*, *system* et *subSystem* qui prennent respectivement les valeurs PF, Aux et EdS.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<nomcl:ListOfPart xmi:version="2.0" [...] domain="PF" system="Aux" subSystem="EdS">
...
</nomcl:ListOfPart>
```

Fig. 42 En-tête du fichier de nomenclature

II.2.1.1 Les instances

Pour en revenir à notre cas d'application (Fig. 33), nous voyons figure 43 l'exemple d'une instance d'une vanne motorisée à 2 voies. Il s'agit de la V2VM09 utilisée pour l'isolation de la soute à eau déminéralisée du cas d'application présenté figure 33. Cet exemple fait partie du système plus large de production et de traitement de l'eau douce sanitaire que nous avons noté « EdS » (cf. annexe).

Instance	
family	Process
designation	Vanne 2 voies, motorisée
reference	V2VM
id	V2VM09
vSyn	V2VM\VSyn\
vSup	V2VM\VSup\
vCmd	V2VM\VCmd\
subSystem	EdS

Fig. 43 Instance de la vanne 2 voies, motorisée numéro 09

Les informations complémentaires sont listées dans des sous-balises de la balise instance. La vanne V2VM09 ne dispose pas d'information complémentaire. Par contre, le symbole de l'indicateur de niveau (Fig. 35) dispose d'informations complémentaires associées aux seuils de détection des niveaux très bas, bas, haut et très haut.

Les coordonnées du symbole sur le synoptique sont conservées de la même manière dans la nomenclature pour pré-positionner les symboles de supervision sur l'IHM (cf. Fig. 44).

Coord	
x	32.52
y	21.4
Angle	-90.0

Fig. 44 Coordonnées de l'instance V2VM09

Les interfaces issues de la bibliothèque sont enfin enrichies de l'attribut repère (id sur la Fig. 45) et de l'attribut sous-système (subSystem sur la Fig. 45) et deviennent ainsi des interfaces de l'instance V2VM09. Ces interfaces seront déclarées comme variables dans le programme de commande (pour celles rattachées à *VCmd*) ou dans le programme de supervision (pour celles rattachées à *VSup*).

Interfaces							
reference	id	information	designation	type	kind	view	subSystem
V2VM	V2VM09	CtrlO	Commande bistable issue de la supervision (1=ouverture; 0=fermeture)	BOOL	Input	VCmd	EdS
V2VM	V2VM09	Tempo	Temporisation de manœuvre de la vanne	TIME	Var	VCmd	EdS
V2VM	V2VM09	FdcO	Fin de course d'ouverture	BOOL	InPhy	VCmd	EdS
V2VM	V2VM09	FdcC	Fin de course de fermeture	BOOL	InPhy	VCmd	EdS
V2VM	V2VM09	StO	Etat "vanne ouverte"	BOOL	Output	VCmd	EdS
V2VM	V2VM09	StC	Etat "vanne fermée"	BOOL	Output	VCmd	EdS
V2VM	V2VM09	Defect	Etat "défaut vanne"	BOOL	Output	VCmd	EdS
V2VM	V2VM09	CmdO	Commande d'ouverture vers la vanne	BOOL	OutPhy	VCmd	EdS
V2VM	V2VM09	CmdC	Commande de fermeture vers la vanne	BOOL	OutPhy	VCmd	EdS
V2VM	V2VM09	CtrlO	Commande bistable issue de la supervision (1=ouverture; 0=fermeture)	BOOL	Output	VSup	EdS
V2VM	V2VM09	StO	Etat "vanne ouverte"	BOOL	Input	VSup	EdS
V2VM	V2VM09	StC	Etat "vanne fermée"	BOOL	Input	VSup	EdS
V2VM	V2VM09	StF	Etat "défaut vanne"	BOOL	Input	VSup	EdS
V2VM	V2VM09	Col	Couleur du fluide	INT	Var	VSup	EdS
V2VM	V2VM09	Prop1	Animation de la propagation 1	BOOL	Var	VSup	EdS
V2VM	V2VM09	Prop2	Animation de la propagation 2	BOOL	Var	VSup	EdS
V2VM	V2VM09	Rep	Repère de la vanne	STRING	Const	VSup	EdS
V2VM	V2VM09	Select	Vanne selectionnée	BOOL	Var	VSup	EdS

Fig. 45 Interfaces de l'instance V2VM09

Les instances et leurs sous-classes sont répertoriées dans le fichier de la nomenclature sous forme de balises XML. Du fait de la longueur des balises, nous avons représenté les informations des instances sous forme de tableau.

II.2.1.2 Les connexions

Les connexions reliées aux symboles sur le synoptique sont également listées. La figure 46 présente les connexions associées à la vanne V2VM09 au format XML.

Trois connexions sont reliées au symbole de la vanne sur notre cas d'application. La première (P-1.29 sur la Fig. 46) est monodirectionnelle et de type *Process*. Elle relie le port 2 de la vanne V2VM09 au port 5 de la soute St3.

La seconde est de type *Electric signal*. Elle relie le port 1 de la vanne V2VM09 au port 4 du détecteur LS-0003. Cette dernière connexion nous informe que la détection du niveau associé à LS-0003 entraîne une action sur la vanne par le biais d'une logique câblée.

La dernière connexion (P-2.39 sur la Fig. 46) est bidirectionnelle et de type *Process*. Elle relie le port 2 du lien d'entrée du synoptique (Int1 sur la Fig. 46) au port 3 de la vanne V2VM09. Outre les points donnant les coordonnées des extrémités (le premier point et le dernier), le deuxième point nous informe que la connexion dispose d'un coude aux coordonnées x=31.04, y=28.02365.

```
<bond id="P-1.29" type="Process" dir="monodirectionnal (1 to 2)">
    <extremity id="V2VM09" portNum="2" ext="Ext1" />
    <extremity id="St3" portNum="5" ext="Ext2" />
    <point x="31.07" y="18.4" />
    <point x="31.07" y="15.44" />
</bond>
<bond type="Electric signal">
    <extremity id="V2VM09" portNum="1" ext="Ext1" />
    <extremity id="LS-0003" portNum="4" ext="Ext2" />
    <point x="35.14" y="21.4" />
    <point x="46.56921" y="21.4" />
    <point x="46.56921" y="19.413094" />
</bond>
<bond id="P-2.39" type="Process" dir="bidirectionnal">
    <extremity id="Int1" portNum="2" ext="Ext1" />
    <extremity id="V2VM09" portNum="3" ext="Ext2" />
    <point x="27.986364" y="28.02365" />
    <point x="31.04" y="28.02365" />
    <point x="31.04" y="24.4" />
</bond>
```

Fig. 46 Connexions associées à V2VM09

II.2.2 Transformations de l'opération d'inventaire

La nomenclature est le modèle central de notre démarche. C'est à partir d'elle et des vues contenues dans la bibliothèque que nous pourrons créer un programme de commande et une interface de supervision. L'opération d'inventaire qui permet de générer la nomenclature est détaillée sur la figure 47. Cette opération se compose de deux étapes exécutées successivement.

La première étape de notre opération (**Collecte des interfaces (1/2)** sur la Fig. 47) vise à référencer les interfaces des éléments de la bibliothèque représentés sur le synoptique. Cette étape peut être assimilée à une transformation exogène horizontale (cf. section I.3.3 p. 31). Elle reçoit le modèle synoptique et la bibliothèque en entrée et produit un PIM d'interface.

L'étape de collecte des interfaces est implémentée au moyen d'un applet Java. Trois méthodes sont définies.

1. La méthode **ListeReference** parcourt le fichier du synoptique pour lister les références des symboles représentés.
2. La méthode **CreerInterface** génère le fichier temporaire destiné à recevoir la liste des interfaces, puis elle parcourt la bibliothèque pour lister les interfaces des symboles référencés par la méthode

ListeReference. Elle complète enfin le fichier temporaire de collecte avec la liste des interfaces répertoriées.

3. La méthode **Enregistre** sauvegarde le fichier temporaire dans le répertoire de la nomenclature.

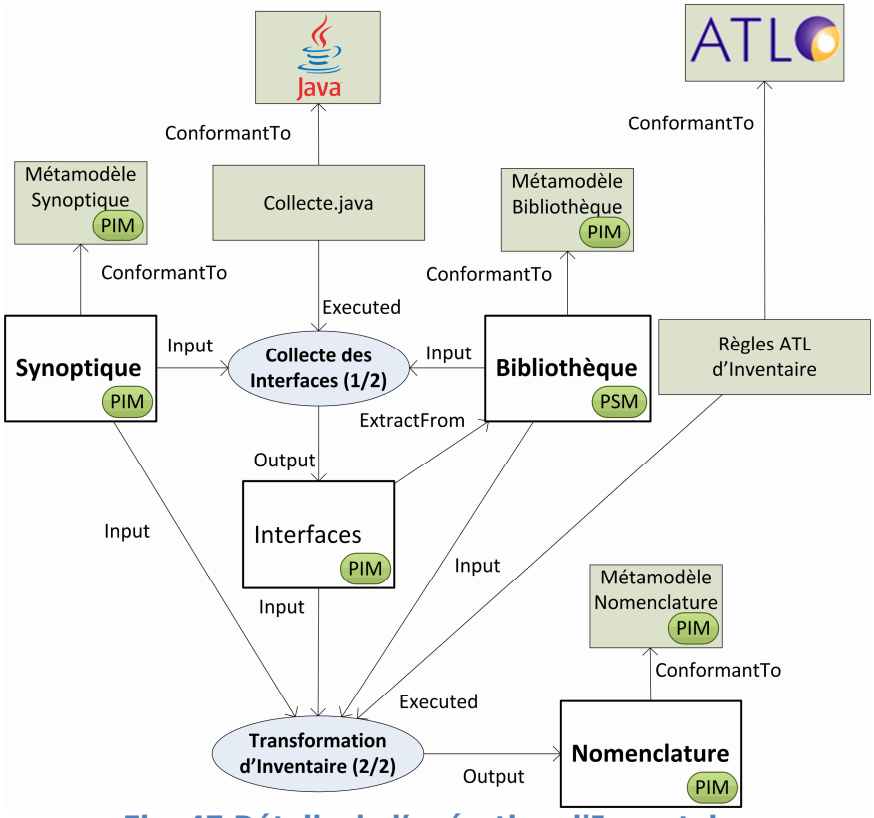


Fig. 47 Détails de l'opération d'Inventaire

La deuxième étape (**Transformation d'Inventaire (2/2)** sur la Fig. 47) génère le modèle de nomenclature. Cette transformation ATL, exogène et horizontale (cf. section I.3.3 p. 31) crée dans la nomenclature une instance pour chaque élément de la bibliothèque présent dans le synoptique. Elle y associe les interfaces collectées précédemment qui sont alors instanciées, les informations associées à l'élément de la bibliothèque, les connexions réalisées entre les différentes formes sur le synoptique ainsi que les informations complémentaires des formes renseignées sur le synoptique.

La transformation d'inventaire se compose de trois règles.

1. La règle **Synoptique2Nomenclature** traite les informations relatives au fichier (domaine, système et sous-système). Elle permet également la création des instances des éléments (de la bibliothèque) et des connexions qui sont complétées par les autres règles. Elle est exécutée une fois lors de la transformation.
2. La règle **Symbole2Instance** traite les informations relatives à une instance de la nomenclature. Elle permet de renseigner les attributs de la classe *Instance* et de créer les classes *CI* et *Coord*. Elle reprend également les interfaces générées par la transformation de collecte et les instancie dans des classes

Interface. Elle est exécutée pour chaque instance référencée lors de la première règle.

3. La règle **ConnexionS2ConnexionN** recopie simplement les connexions du synoptique dans le fichier de nomenclature. Chaque attribut de la classe *Bond* du synoptique est ainsi recopié dans les attributs correspondant de la classe *Bond* de la nomenclature. Les sous-classes *Point* et *Extremity* sont traitées de la même manière.

A l'issue de cette opération, nous avons obtenu par une **démarche descendante**, un modèle qui contient toutes les références d'accès aux différentes vues de la bibliothèque, nécessaires à la génération d'un programme de commande et d'une interface de supervision conforme au synoptique proposé par l'expert. Ce modèle de nomenclature va maintenant nous servir à instancier les vues de supervision pour créer l'interface et les vues de commande pour créer le programme.

II.3 Opération d'insertion

Le modèle de nomenclature issu de l'opération d'inventaire regroupe l'ensemble des références des vues de supervision nécessaires à la génération d'une interface graphique. Cette troisième opération est réalisée par l'insertion des vues de supervision dans un modèle standard d'IHM préexistant. L'opération d'**insertion** (Fig. 48) reçoit en entrée les modèles de bibliothèque et de nomenclature précédemment définis ainsi que le modèle standard d'IHM. Elle produit en sortie un canevas d'IHM. Avant de présenter l'implémentation de cette opération, nous détaillons les concepts relatifs à la génération d'une IHM dans notre contexte.

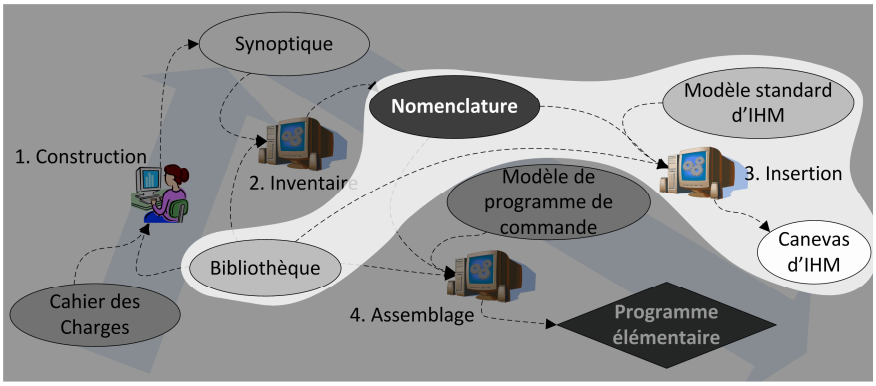


Fig. 48 Opération d'Insertion

II.3.1 Les modèles d'IHM

Le titre de cette thèse fait apparaître le terme « interface de supervision ». Nous entendons par interface de supervision d'un système, l'image affichée sur l'écran de l'utilisateur. Nous ne nous intéresserons pas aux aspects anthropomorphiques de l'interface. De même nous ne remettons pas en cause la présence d'écrans (qu'ils soient tactiles ou associés à un dispositif de commande de type clavier et souris) comme interface privilégiée pour le contrôle commande. En effet, les panneaux de commande « en dur » tendent à disparaître ou sont réduits à quelques interfaces spécifiques (arrêt d'urgence, mise sous tension, etc.) et si d'autres dispositifs existent ou sont à l'étude (réalité augmentée, environnements virtuels, etc.), ils ne sont, pour l'instant, pas d'actualité dans l'industrie.

C'est pourquoi ce développement des Interfaces Homme-Machine par une équipe de conception hétérogène se doit d'être cadré. Il existe des normes (AFNOR 1991; ISO 2006) définissant les grands principes. Au sein d'une entreprise ou pour un projet, on doit trouver un guide de style pour les développeurs (Apple 2010; Microsoft 2011). Ce guide est conçu par des ergonomes et des informaticiens sur les pré-requis suivants :

- **Tenir compte de l'existant.** Une bonne interface suit tout d'abord les codes généralement établis (interface de type WIMP[1] par exemple). Une révolution en IHM ne peut pas marcher ; si elle s'éloigne trop des codes existants elle sera rejetée par les utilisateurs, à moins d'être extrêmement intuitive. Des études démontrent par exemple que le clavier Dvorak (breveté en 1935) améliore la productivité et le confort des utilisateurs. Il n'a cependant jamais supplanté le Qwerty et ses dérivés (breveté depuis 1868), du fait probablement de son trop grand écart avec les standards qui nécessiterait donc un réapprentissage. Le paradoxe est que les mêmes études démontrent qu'il est plus rapide d'apprendre à taper sur un clavier Dvorak que sur un Qwerty (Liebowitz and Margolis 1990).
- **Connaître l'utilisateur.** Une bonne connaissance *a priori* des futurs utilisateurs limitera les écueils (inévitables) de conception et les refontes importantes lors des confrontations avec le client. Un processus de conception d'IHM place en effet l'utilisateur au centre de la conception par le biais d'une démarche incrémentale et participative (ISO 2010). Des outils existent pour analyser l'environnement de travail des utilisateurs et le contexte d'exploitation de l'interface. Nous utilisons la hiérarchie d'abstraction (Rasmussen 1985) pour analyser le domaine de travail (cf. annexe).

Ce guide fédérateur doit regrouper toutes les règles graphiques assurant l'homogénéité de l'identité visuelle du produit final. Ainsi, il définira les points suivants :

- **L'organisation des différentes zones** de l'image (bandeau de commande, de menu, de lien, possibilité de multifenêtrage ou non, etc.) en tenant compte de la tendance naturelle de l'homme à se focaliser sur le tiers haut d'une image.
- **Le code des couleurs** applicable (pour le fond, les animations, etc.).
- **Le principe des commandes** et les animations associées (pointage, surbrillance, effet 3D, etc.).
- **Les symboles** prédéfinis pour l'exploitation de l'IHM (boutons, listes de choix, liens, etc.) mais aussi pour l'exploitation de l'installation (symbole d'une vanne, d'une pompe, mesure de niveau, de pression, etc.).

La prise en compte de l'utilisateur final nécessite de réaliser au plus tôt des maquettes fidèles des IHM, ce qui va à contre-pied des habitudes puisqu'en général l'IHM est le dernier point abordé lors d'un processus conception séquentiel.

Notre but n'est pas de définir exactement ce que doit être une bonne IHM mais notre démarche permet un prototypage rapide de l'image, afin de l'exploiter dans le cadre d'une conception participative. Plus le prototype sera semblable à l'IHM finale, et plus les problèmes (ou écarts) seront découverts en amont.

[1] WIMP: Windows, Icons, Menus and Pointing device

Nous nous proposons d'automatiser la prise en compte de ces guides en imposant un fond visuel matérialisé par le modèle standard d'IHM. Ce dernier sera complété des vues de supervisions issues de la bibliothèque pour produire un canevas d'IHM. Nous présentons ici ces deux concepts.

II.3.1.1 Le modèle standard d'IHM

Le modèle standard d'IHM regroupe les règles de présentation de l'interface à produire, il pose ainsi l'identité visuelle du produit fini. En ce sens, c'est un **guide de style** au même titre que ceux proposés par Microsoft® (Microsoft 2011) et Apple® (Apple 2010). C'est également un **patron de conception** puisqu'il impose des règles d'agencement des zones de l'IHM. La figure 49 présente notre modèle standard d'IHM sous le formalisme de patrons P-Sigma (Conte, Fredj et al. 2001).

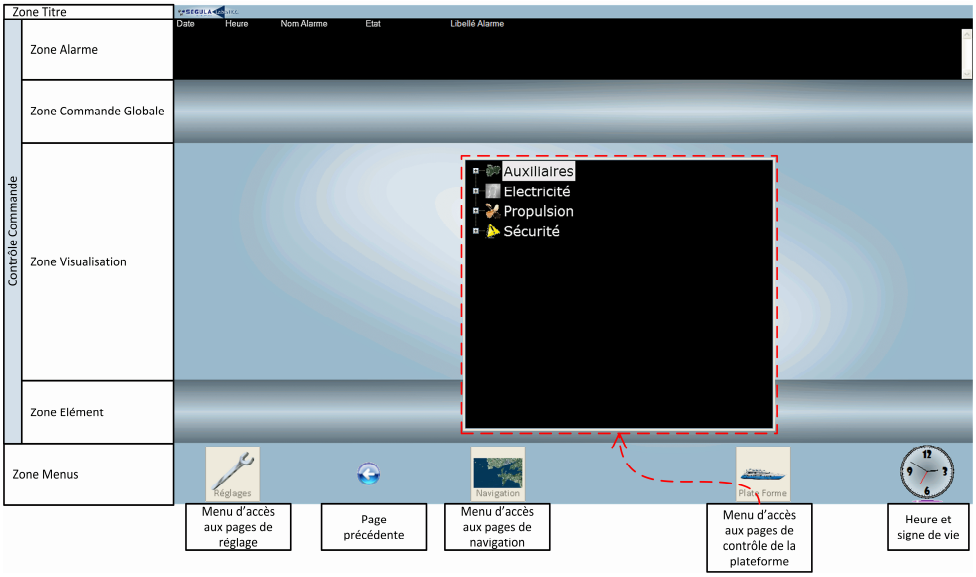
Notre modèle standard est enfin un « **framework** » puisque, dans notre cas, le modèle standard d'IHM est un projet de supervision réalisé sous le logiciel SCADA[1] Panorama E2 de la société CODRA, notamment utilisé pour la supervision du LMJ[2] (Nicoloso, dupas et al. 2009).

Les logiciels de supervision comme Panorama E2 peuvent contraindre légèrement la conception des IHM. Aussi, le modèle standard d'IHM ne peut pas être complètement indépendant du logiciel sur lequel il sera appliqué. Exactement comme un programme prévu pour Microsoft® Windows n'a pas tout à fait la même interface que ce même programme prévu pour Mac OS d'Apple®.

Rubrique	Champ
Interface	
Identifiant	Modèle standard d'IHM Navire
Classification	Offre un modèle de conception d'interface de supervision pour le contrôle commande centralisé d'un navire.
Contexte	Nécessite l'existence d'une bibliothèque d'objet de supervision préétabli et compatible avec le modèle standard d'IHM.
Problème	Permet de construire une IHM pour un sous-système du navire à partir des objets de la bibliothèque.
Force	Permet de garantir l'homogénéité des interfaces de supervision du navire par l'application d'un cadre d'application strict.

Rubrique	Champ
Réalisation	
Solution/ Démarche	1. Créer un fond d'écran commun à l'ensemble des interfaces. Pose l'identité visuelle du produit. 2. Définir les zones de l'interface graphique. 3. Définir l'arborescence de l'interface à partir de l'analyse du domaine de travail (Burns and Hajdukiewicz 2004). 4. Compléter les zones identiques pour toutes les interfaces (menus, horodatage).
Solution/ Modèle	Le modèle obtenu après application du patron contient les zones d'affichages suivantes : ▪ Une zone de menu qui permet la navigation entre les différentes pages ainsi que l'affichage d'informations indépendantes de tout sous-système (heure et signe de vie). La zone contient un menu de réglage de l'interface (paramètre graphique, de sécurité, etc.) un menu ou plusieurs menus d'accès aux pages de contrôle commande du système. ▪ Une zone de titre qui affiche la page active. ▪ Une zone d'affichage « au fil de l'eau » des alarmes ▪ Une zone de commande globale qui regroupe les dispositifs de commande du sous-système affichés sur la page. ▪ Une zone de visualisation du sous-système sous la forme d'un schéma animé. ▪ Une zone de commande élémentaire qui regroupe les dispositifs de commande d'un élément sélectionné sur la zone de visualisation.

[1] SCADA : Supervisory Control and Data Acquisition
[2] LMJ : Laser MegaJoule

Cas d'application	Une solution pour la supervision d'un navire est la suivante :  La zone de menu occupe la partie basse de l'interface. Outre le bouton d'accès au menu de réglage, l'heure et le signe de vie, elle propose un bouton pour le menu d'accès aux pages concernant la navigation, un bouton pour le menu d'accès aux pages de contrôle commande de la plate-forme propulsée du navire ainsi qu'un bouton permettant de revenir à la page précédemment affichée. La zone de titre occupe la partie haute de l'interface. La zone d'alarme se situe sous la zone de titre. La zone de visualisation du sous-système occupe la partie centrale de l'interface. Les zones de commandes sont réparties de part et d'autre de la zone de visualisation. La zone haute située entre le bandeau d'alarme et la zone de visualisation donne accès aux commandes globales du sous-système affiché et la zone basse située au dessus des menus donne accès aux commandes de l'élément sélectionné dans la zone de visualisation du sous-système.
Conséquence d'application	Ce patron offre un modèle d'interface graphique généralisable à un contexte (celui du contrôle-commande d'un navire) et réutilisable pour tout projet similaire. Il nécessite cependant l'utilisation d'un outil de conception prédéfini qui sera par la suite imposé pour compléter ces interfaces (dans notre cas, le logiciel Panorama E2).

Rubrique	Champ
Relation	
Utilise	/
Raffine	/
Requiert	L'utilisation de ce patron pour la génération automatique d'une interface complète, nécessite une bibliothèque d'éléments graphiques complémentaire et une nomenclature des éléments présents sur l'interface.
Alternative	/

Fig. 49 P-Sigma du modèle standard d'IHM

II.3.1.2 Le canevas d'IHM

Le canevas d'IHM est une ébauche de l'application de supervision du système à concevoir. Il doit permettre la surveillance des éléments du système, identifiés sur le synoptique en proposant une interface graphique dont l'agencement reprend la représentation du synoptique. Le canevas d'IHM doit servir de base, d'une part à une démarche de conception participative basée sur des échanges avec l'utilisateur final du système et d'autre part au développement de l'application de supervision globale du système.

La figure 50 donne une représentation simplifiée du canevas d'IHM. Il s'agit du modèle standard d'IHM complété des symboles de supervision référencés dans la nomenclature. Les informations de la nomenclature (Fig. 41) servent à l'instanciation des vues de supervision stockées dans la bibliothèque (*VSup* sur la Fig. 30). Une fois instanciées, ces vues sont

insérées dans le canevas, positionnées à l'aide des coordonnées (Coord sur la Fig. 41) et reliées grâce aux connexions (Bond sur la Fig. 41).

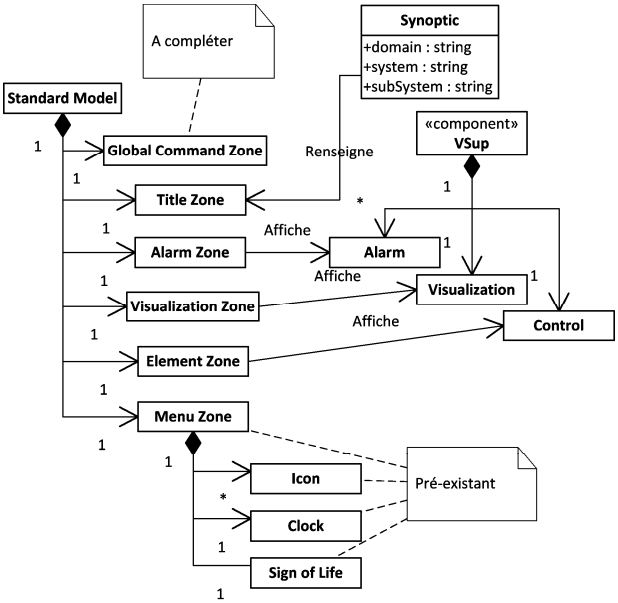


Fig. 50 Représentation simplifiée du canevas d'IHM

La zone de titre (Title Zone sur la Fig. 50) pourra être renseignée par les attributs *domain*, *system* et *subSystem* du synoptique. La zone de commande globale (Global Command Zone sur la Fig. 50) reste à compléter par les experts.

II.3.2 Transformations de l'opération d'insertion

L'opération de génération d'un projet d'IHM se déroule en 8 étapes regroupées en 3 phases (Fig. 51). La première phase prépare le modèle standard à recevoir le canevas (2 étapes), la seconde permet de créer le premier fichier de configuration du canevas (3 étapes) et la dernière crée le second fichier de configuration du canevas (3 étapes).

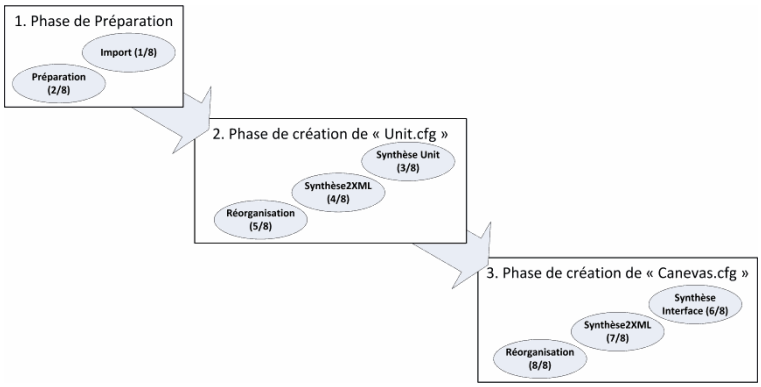


Fig. 51 Organisation de l'opération d'insertion

La génération automatique du canevas de supervision est réalisée par une suite de transformations de modèles. Pour être correctement exécutées

ces transformations doivent pointer des métamodèles de sortie auxquels les modèles produits seront conformes. Ces métamodèles sont des PSM qui permettent d'assurer la compatibilité avec le logiciel Panorama E2 grâce auquel nous avons réalisé le modèle standard d'IHM.

Nous présentons tout d'abord la structure de Panorama E2 avant de détailler les transformations nécessaires à la génération du canevas.

II.3.2.1 Structure de Panorama E2

Un projet d'interface graphique réalisé sous Panorama E2 se compose, pour simplifier, d'une bibliothèque de composants de supervision prédéfinis, d'un fichier de configuration *Unit.cfg* par interface qui définit l'architecture de l'interface, ainsi que d'un fichier de configuration du projet *canevas.cfg* pour chaque interface qui définit la structure de l'affichage de l'interface.

La figure 52 présente un extrait de la structure de notre modèle standard réalisé sous Panorama E2. Nous retrouvons ici la hiérarchie d'abstraction issue de l'analyse du domaine de travail (cf. annexe). Le modèle standard est divisé en domaines (Plateforme, Navigation, Réglages). Le domaine *Plateforme* est lui-même subdivisé en systèmes (*Auxiliaires*, *Propulsion*, *Sécurité*, *Électricité*). Notre cas d'application est un sous-système du système auxiliaire, il est défini par un fichier *Unit.cfg* et par un fichier *Canevas.cfg*.

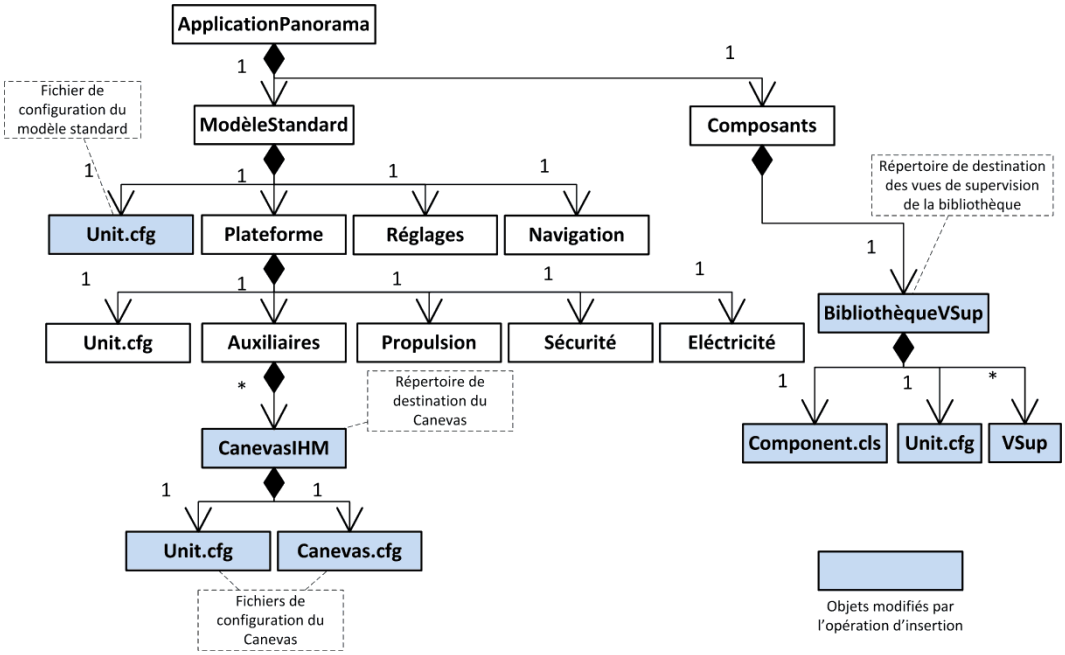


Fig. 52 Structure du modèle standard sous Panorama E2

Le fichier *Unit.cfg* du modèle standard contient l'arborescence de l'application. Notre canevas doit être déclaré dans ce fichier pour être accessible.

Les vues de supervision de la bibliothèque nécessaires à la génération de l'interface sont stockées dans le répertoire *Composants*. Nous les regrouperons dans un sous-répertoire *BibliothèqueVSup* qui contiendra le

fichier de configuration *Unit.cfg* de la bibliothèque ainsi qu'un fichier de description des classes (*Component.cls*).

Nous rappelons que l'opération d'insertion se déroule en trois phases. La première se charge de préparer la bibliothèque de vue de supervision ainsi que le fichier *Unit.cfg* du modèle standard. La deuxième génère le fichier *Unit.cfg* du canevas et la troisième le fichier *Canevas.cfg*. Elles sont détaillées ci-après.

II.3.2.2 Phase de préparation

La première phase de l'opération d'insertion sert à préparer le modèle standard existant à recevoir le canevas d'IHM. Elle se déroule en deux étapes présentées sur la figure 53.

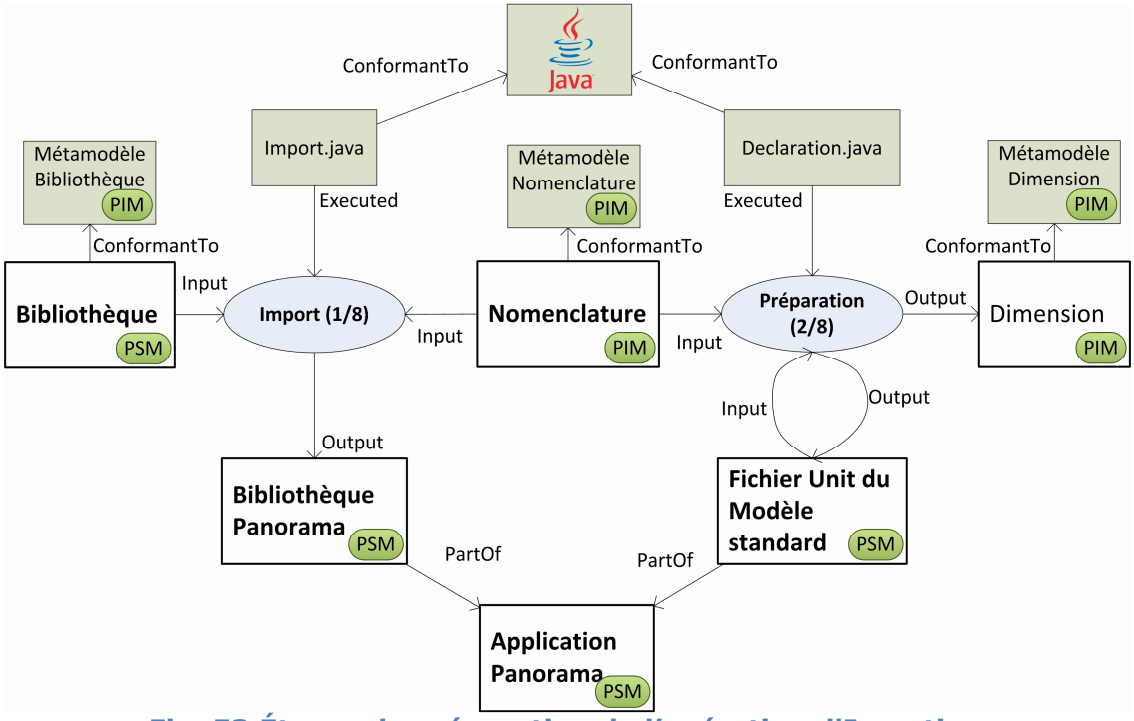


Fig. 53 Étapes de préparation de l'opération d'Insertion

La première étape de cette phase de préparation (**Import (1/8)** sur la Fig. 53) crée la bibliothèque de supervision associée au projet d'IHM. Elle est implémentée au moyen d'un applet Java ; elle peut être assimilée à une transformation exogène verticale (cf. section I.3.3 p. 31). Six méthodes sont définies pour réaliser trois sous étapes :

1. La méthode **CheminVSup** parcourt la nomenclature pour lister les chemins d'accès aux vues de supervisions.
2. Les méthodes **CreerComponent** et **CreerUnit** créent respectivement le fichier *Component.cls* et le fichier *Unit.cfg* de la bibliothèque. Ces fichiers sont sauvegardés à leur place dans l'arborescence de l'application par la méthode **Enregistre** ; le répertoire de destination est créé à cette étape.
3. Les méthodes **CopyFile** et **CopyDirectory** comme leurs noms l'indiquent, copient le Modèle standard d'IHM dans le répertoire de destination du canevas. Elles sont également utilisées pour

copier le contenu des vues synoptiques (répertoire et fichiers) dans leur répertoire de destination.

La seconde étape de cette phase (**Préparation (2/8)** sur la Fig. 53) prépare l'application de Panorama E2 pour la génération de l'interface. Elle est également implémentée au moyen d'un applet Java et on peut l'assimiler à une transformation exogène verticale (cf. section I.3.3 p. 31). Trois méthodes sont définies pour réaliser deux sous étapes :

1. La méthode **UnitDeclaration** ajoute une balise au fichier *Unit.cfg* du modèle standard pour déclarer le canevas dans l'arborescence de l'application.
2. Les méthodes **ListeElements** et **CreerDimensio** servent à créer un fichier temporaire contenant les dimensions des symboles de l'interface. Ce fichier est utilisé par la suite pour positionner les symboles sur l'image de supervision.

Les étapes d'import et de préparation ont préparé l'application de supervision à recevoir notre canevas d'IHM. Nous avons généré la bibliothèque de Panorama E2 et nous avons modifié le fichier *Unit.cfg* du modèle standard. Ces deux modèles sont des PSM de Panorama E2. Nous avons enfin généré un PIM *Dimension* sous la forme d'un fichier temporaire.

Nous pouvons désormais nous attaquer à la génération du canevas en lui-même. Deux phases sont nécessaires à cette génération. La première sert à créer le fichier de configuration *Unit.cfg* du canevas, la dernière génère le fichier de configuration du projet (*Canevas.cfg*) à partir du fichier *Unit.cfg*.

II.3.2.3 Phase de Création du fichier *Unit.cfg*

Cette phase vise la génération du fichier *Unit.cfg* du canevas d'IHM. Le fichier *Unit.cfg* associé aux interfaces dans Panorama E2 contient déjà la description complète des images de contrôle-commande et des instances des vues de supervision de la bibliothèque. La figure 54 présente le métamodèle simplifié d'un fichier *Unit.cfg*.

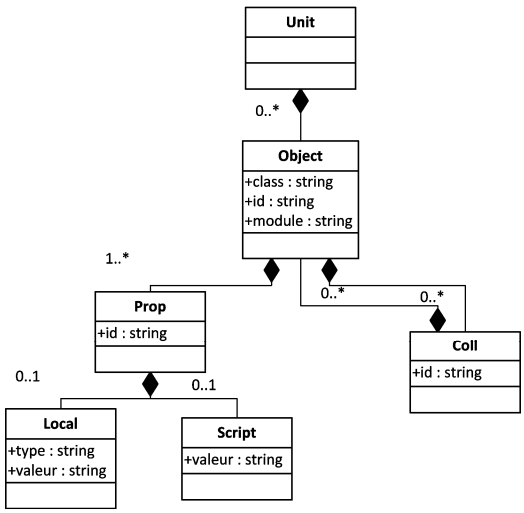


Fig. 54 Métamodèle du fichier Unit.cfg

Le fichier *Unit.cfg* se compose d'objets (classe *Object* sur la Fig. 54). Les objets peuvent contenir des propriétés (*Prop* sur la Fig. 54). Ces propriétés peuvent contenir soit une variable locale (*Local* sur la Fig. 54) soit du code interprétable (*Script* sur la Fig. 54). Les objets peuvent être agrégés dans des collections (*Coll* sur la Fig. 54) pour former des objets plus complexes.

Chaque vue de supervision est référencée deux fois dans le fichier *Unit.cfg*. Une fois en tant qu'objet, instance d'une classe de la bibliothèque et une fois en tant que vue embarquée dans l'objet *SynoptHMI* qui représente l'image de supervision du système avec la position des différents symboles sur l'écran. La figure 55 présente un extrait du fichier *Unit.cfg*, on y retrouve la vanne V2VM09 de notre cas d'application en tant qu'objet et en tant que vue embarquée (*EmbeddedViewHMI* sur la Fig. 55).

```
<UNIT PRELOAD="FALSE">
...
  <OBJECT CLASS="SynoptHMI" ID="EdS" MODULE="CODRA_Synopt">
  ...
    <COLL ID="embeddedviews">
    ...
      <OBJECT CLASS="EmbeddedViewHMI" ID="EmbeddedView_V2VM9" MODULE="CODRA_Synopt">
        <PROP ID="referenceviewname">
          <LOCAL TYPE="BSTR">../V2VM9/V2VM</LOCAL>
        ...
      </OBJECT>
    </COLL>
  ...
  </OBJECT>
  <OBJECT CLASS="V2VM" ID="V2VM9" MODULE="Bibliotheque">
    <COLL ID="objects" />
    <PROP ID="rep">
      <LOCAL TYPE="BSTR">V2VM9</LOCAL>
    </PROP>
  ...
  </OBJECT>
</UNIT>
```

Fig. 55 Extrait du fichier Unit.cfg

La création du fichier *Unit.cfg* du canevas se déroule en trois étapes (Fig. 56). La première (**Synthèse Unit (3/8)** sur la Fig. 56) crée un fichier temporaire conforme à notre métamodèle de fichier Unit (Fig. 54) à partir du modèle de nomenclature et du modèle temporaire *Dimension*.

Cette étape est implémentée au moyen d'une transformation ATL exogène verticale (cf. section I.3.3 p. 31). La transformation **Synthèse Unit (3/8)** comporte une règle principale (**ListOfPart2Unit**) qui appelle 13 règles secondaires permettant la génération du fichier *Unit temporaire*. On peut distinguer trois sous-étapes lors de cette génération :

1. La première sous-étape est celle de l'instanciation des vues de supervision qui seront affichées. Elles sont déclarées et paramétrées (seuils de détection, couleur d'affichage des fluides, etc.) dans le fichier *Unit temporaire*.
2. La seconde sous-étape est celle de la construction de l'interface avec la déclaration et le positionnement des vues embarquées sur l'image de conduite. C'est dans cette phase que le fichier temporaire *Dimension* est utilisé.
3. La troisième sous-étape vient compléter le fichier *Unit temporaire* avec des informations définies par défaut et dépendantes du projet de canevas (titre de l'interface) ou de notre modèle standard (taille de l'interface, style d'affichage, image de fond).

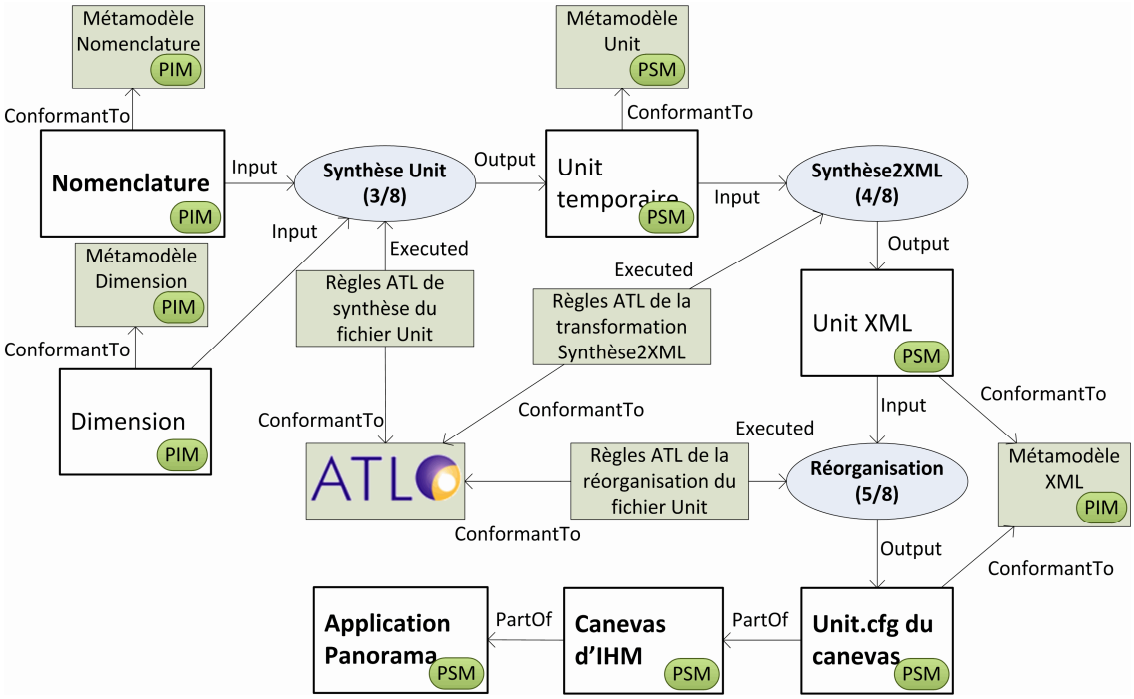


Fig. 56 Création du fichier Unit.cfg du canevas

Bien qu'au format XML, le fichier *Unit.cfg* interprétable par Panorama E2 a une structure particulière puisque certains des attributs ont leur valeur directement renseignée dans la balise (cas de l'attribut *class* de la balise *Object* sur la Fig. 55), alors que d'autres attributs ont leur valeur renseignée entre une balise d'ouverture et une balise de fermeture (cas de l'attribut *valeur* de la balise *Local*, représenté par la chaine de texte « ../V2VM9/V2VM »).

Les deux dernières étapes de la création du fichier *Unit.cfg* ont donc pour but de remettre en forme le fichier *Unit temporaire* généré pour qu'il soit interprétable par Panorama E2.

L'étape **Synthèse2XML (4/8)** (Fig. 56) transforme le fichier temporaire conforme à notre métamodèle Unit en un fichier conforme au métamodèle XML (voir Fig. 57).

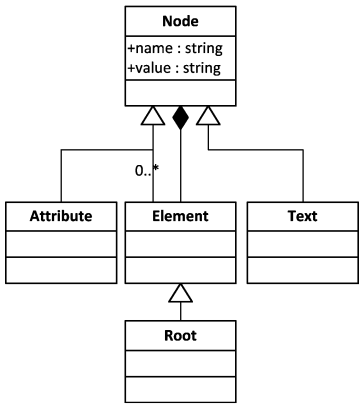


Fig. 57 Métamodèle XML

Chaque classe du fichier *Unit temporaire* est transformée en un nœud de type *Element*, *Attribute*, *Text* ou *Root* en fonction de la forme interprétable par Panorama E2. Cette étape est implémentée au moyen d'une transformation ATL exogène horizontale (cf. section I.3.3 p. 31) qui comporte sept règles.

L'étape **Réorganisation** termine la mise en forme du fichier. Elle est implémentée par une transformation ATL endogène et horizontale (cf. section I.3.3 p. 31) qui ne comporte pas de règle mais uniquement une *requête ATL* (*query*). « Une requête ATL peut être considérée comme une opération qui calcule une valeur primitive à partir d'un ensemble de modèles de sources. L'utilisation la plus courante de requêtes ATL est la nouvelle génération d'une sortie textuelle (codée dans une valeur de chaîne) à partir d'un ensemble de modèles de sources » (Fortin 2010).

Le résultat de ces trois étapes est la création du fichier de configuration *Unit.cfg* de notre canevas d'IHM (Fig. 55) ; ce fichier est un PSM propre à Panorama E2. Il nous reste à créer le deuxième fichier de configuration nommé *Canevas.cfg*.

II.3.2.4 Phase de création du fichier *Canevas.cfg*

Le fichier *Canevas.cfg* peut être construit à partir du fichier *Unit.cfg*. Il ne comporte pas d'information supplémentaire mais les informations qu'il contient y sont présentées différemment. La figure 58 présente le métamodèle simplifié du fichier de configuration d'une interface (*Canevas.cfg*).

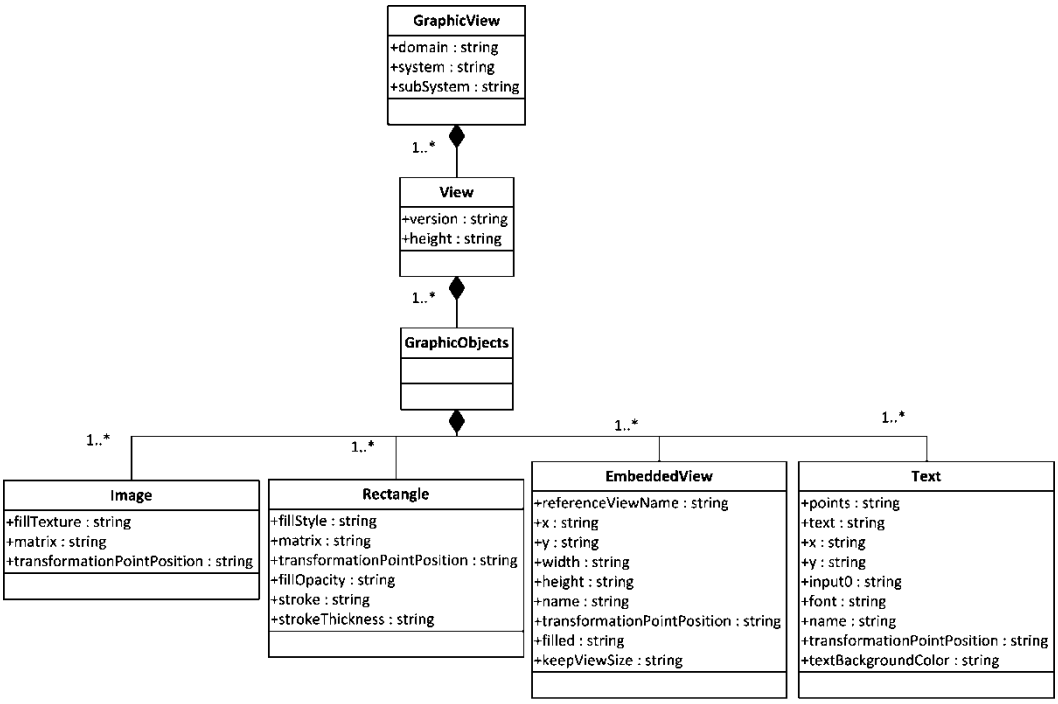


Fig. 58 Métamodèle du fichier de configuration d'une interface

Ce fichier contient des informations relatives à la structure graphique de l'interface (dimensions, couleurs, vues incrustées, etc.). Ainsi la vue graphique (classe *GraphicView* sur la Fig. 58) définie par le fichier

Canevas.cfg contient des vues (classe *View* sur la Fig. 58) qui sont composées d'objets graphiques (classe *GraphicObjects* sur la Fig. 58).

Ces objets graphiques peuvent être des images (classe *Image* sur la Fig. 58), comme le fond de notre interface. Ils peuvent également être des formes, comme la forme rectangle (classe *Rectangle* sur la Fig. 58) que nous utilisons pour distinguer les différentes zones de l'interface.

Les vues embarquées (classe *EmbeddedView* sur la Fig. 58) sont également des objets graphiques. La figure 59 présente un extrait du fichier *Canevas.cfg* à produire. On y retrouve la vue embarquée de la vanne V2VM09.

```
<GraphicView domain="PF" systeme="Aux" subSystem="EdS">
  <View Version="1.3.0.0" Height="464">
...
    <GraphicObjects>
...
      <EmbeddedView ReferenceViewName="../V2VM09/V2VM" X="84" Y="310" Width="23" Height="22"
      KeepViewSize="False" Name="EmbeddedView_V2VM09" TransformationPointPosition="Center" Filled="False"/>
...
    </GraphicObjects>
  </View>
</GraphicView>
```

Fig. 59 Extrait du fichier Canevas.cfg

Enfin, les objets graphiques peuvent être des zones de texte (classe *Text* sur la Fig. 58) ; c'est par exemple le cas de la zone qui affiche le titre de l'interface.

Le déroulement de la création du fichier de configuration *Canevas.cfg* (Fig. 60) est similaire à celui de la création du fichier *Unit.cfg* précédemment présenté (Fig. 56). Trois étapes sont nécessaires ; la première (**Synthèse Interface (6/8)** sur la Fig. 60) crée un fichier temporaire conforme au métamodèle du fichier de configuration d'interface (Fig. 58) à partir du modèle temporaire du fichier Unit (issu de l'étape **Synthèse Unit (3/8)** sur la Fig. 56).

L'étape **Synthèse Interface (6/8)** est implémentée au moyen d'une transformation ATL exogène horizontale (cf. section I.3.3 p. 31), constituée de deux règles :

1. La règle **Unit2Interface** est la règle principale de cette transformation. Elle crée le fichier temporaire, renseigne les attributs généraux et appelle la deuxième règle.
2. La règle **GraphicObjects** crée les différents objets graphiques du fichier de configuration et renseigne leurs attributs. Les vues embarquées, créées dans le fichier *Unit temporaire*, sont ici reprises pour déclaration dans le fichier temporaire de configuration de l'interface.

Comme pour le fichier Unit, afin de rendre le fichier de configuration de l'interface interprétable par Panorama E2, il est nécessaire de le remettre en forme. C'est le but des deux dernières transformations de cette phase.

La transformation **Synthèse2XML (7/8)** reçoit en entrée le fichier d'interface temporaire et produit un fichier d'interface conforme au métamodèle XML (Fig. 57). Il s'agit d'une transformation exogène horizontale (cf. section I.3.3 p. 31) implémentée sur ATL. Elle est tout à fait similaire à la

transformation **Synthèse2XML (4/8)** présentée précédemment (cf. section II.3.2.3 p. 74).

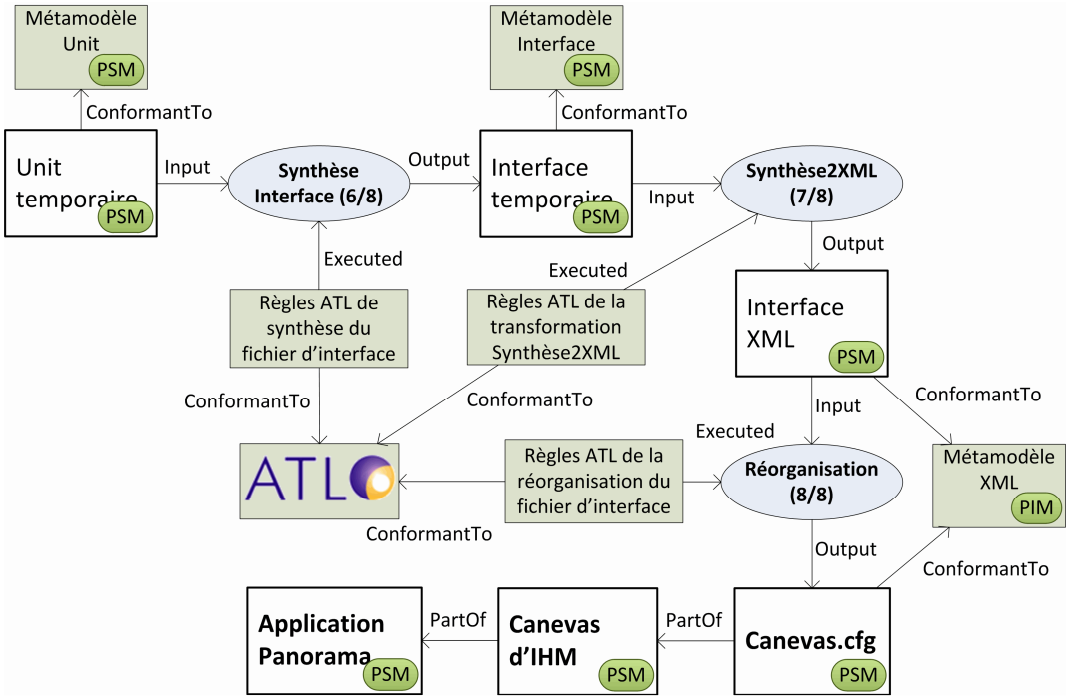


Fig. 60 Création du fichier de configuration Canevas.cfg

La transformation **Réorganisation (8/8)** termine la mise en forme du fichier. Elle est implémentée par une transformation ATL endogène et horizontale (cf. section I.3.3 p. 31) qui ne comporte pas de règle mais uniquement une *requête ATL*. Elle est tout à fait similaire à la transformation **Réorganisation (5/8)** présentée précédemment (cf. section II.3.2.3 p. 74).

L'opération d'insertion poursuit la **démarche descendante** entamée avec l'opération d'inventaire (cf. chapitre II.2 p. 61).

Après avoir préparé le modèle standard à recevoir notre interface (cf. section II.3.2.2 p. 73), nous avons créé le premier fichier de configuration *Unit.cfg* (cf. section II.3.2.3 p. 74). Cette dernière étape que nous venons de présenter crée le second fichier de configuration « Canevas.cfg ».

A l'issue de ces trois étapes, le canevas d'IHM est interprétable sous Panorama E2. Il peut, ensuite, faire l'objet de travaux de finition, d'analyse et de validation par le biais d'un cycle de conception participative (Caelen 2009).

II.4 Opération d'assemblage

Le modèle de nomenclature nous permet de générer un canevas d'IHM. Néanmoins, il contient également l'ensemble des références des vues de commande nécessaire à la génération d'un programme de commande. Cette quatrième opération est réalisée par l'assemblage des vues de commande dans un modèle de programme de commande. L'opération d'assemblage (Fig. 61) reçoit en entrée les modèles de bibliothèque et de nomenclature précédemment définis ainsi que le modèle standard d'IHM. Elle produit en sortie un programme élémentaire. Avant de présenter l'implémentation de cette opération, nous détaillons les concepts relatifs à la génération d'un programme de commande dans notre contexte.

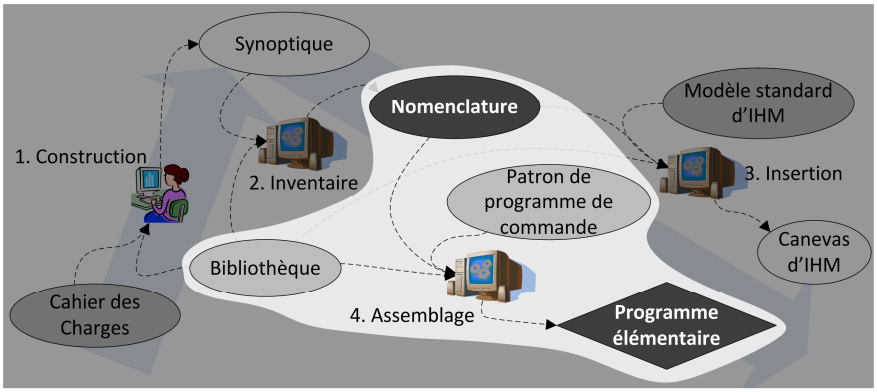


Fig. 61 Opération d'Assemblage

II.4.1 Les modèles de commande

La commande est le moyen d'intervenir sur le système à contrôler. C'est le sens donné à ce terme dans le domaine de l'automatisme. Plus précisément, dans notre cas, nous pouvons distinguer deux types de commandes. Les commandes dites « élémentaires », qui gèrent les sorties du contrôleur et les commandes dites « globales », qui gèrent des ensembles de commandes élémentaires.

Les commandes élémentaires peuvent en général être associées à un matériel précis (vanne, pompe) au travers des sorties du contrôleur qu'elles gèrent (bornage). En conséquence, pour du matériel standardisé cette commande peut être prédéfinie, indépendamment du contexte global d'utilisation du matériel.

Par opposition, les commandes globales sont propres à chaque système puisqu'elles sont dépendantes de l'architecture du système (commande de transfert entre deux cuves faisant intervenir une pompe et plusieurs vannes par exemple). Ce type de commande ne peut être spécifié que par le concepteur qui maîtrise le fonctionnement de son système.

A ce stade, le programme de commande que nous proposons de générer regroupe l'ensemble des commandes associées aux éléments du système. La génération de la commande globale est laissée à la charge des experts.

II.4.1.1 Le modèle de programme de commande

Pour être exploitable, notre programme de commande doit être adapté à la plateforme destinée à le recevoir. Le modèle de programme de commande reprend la structure d'une d'application vide créée sous le logiciel de programmation retenu. C'est une coquille vide, compatible avec l'environnement de developpement, destinée à recevoir les vues de commande des éléments répertoriés dans la nomenclature.

Nous avons retenu l'environnement de programmation Straton[1] de la société Copalp®. Il a été utilisé avec succès pour des projets similaires parmi lesquels nous pouvons citer SimSED (Bévan, Allègre et al. 2011), ou le développement d'un IMCS[2] (Byunghee, Sungho et al. 2010). La figure 61 présente notre modèle de programme de commande sous Straton et un extrait de son export au format XML.

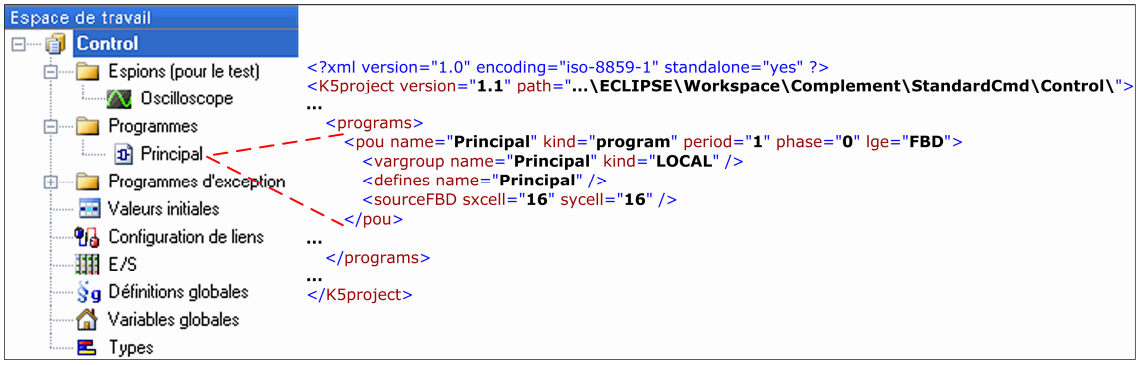


Fig. 62 Le modèle de programme sous Straton et son équivalent XML

L'**IDE Straton** permet de programmer les automates Straton conformément aux langages de la norme IEC 61131-3 (IEC 2003) et il permet également d'importer les programmes sous forme de fichier XML conformes au standard PLCOpen (PLCOpen 2009).

Par ailleurs, les programmes de commande sont exécutés sur le **Straton Runtime**. Il s'agit d'une machine virtuelle portable sur tout type de plateforme incluant ou non un OS. Le code généré peut donc être testé sur cette machine virtuelle installée sur un PC conventionnel puis être implantée sur un automate sans transcription supplémentaire.

II.4.1.2 Le programme de commande élémentaire

Le programme de commande élémentaire reprend la structure du modèle de programme de commande ; il contient en outre les vues de commande référencées dans la nomenclature ainsi que les variables issues des interfaces de la nomenclature.

[1] Straton IDE : www.copalp.com
[2] IMCS : Integrated Management Control System

Les vues de commande sont instanciées en tant que POU dans le programme principal du projet. Les interfaces de la nomenclature servent quant à elle à déclarer les variables.

Nous avons retenu le langage FBD (Fig. 63) de la norme IEC 61131 (IEC 2003) pour l'écriture de notre programme de commande. Ce langage est une évolution du « Ladder » basé sur la représentation de schéma électrique de contacts. Le FBD est donc un langage graphique ; il est constitué de blocs fonctionnels dont les entrées sont visuellement représentées sur la gauche du bloc et les sorties sur la droite.

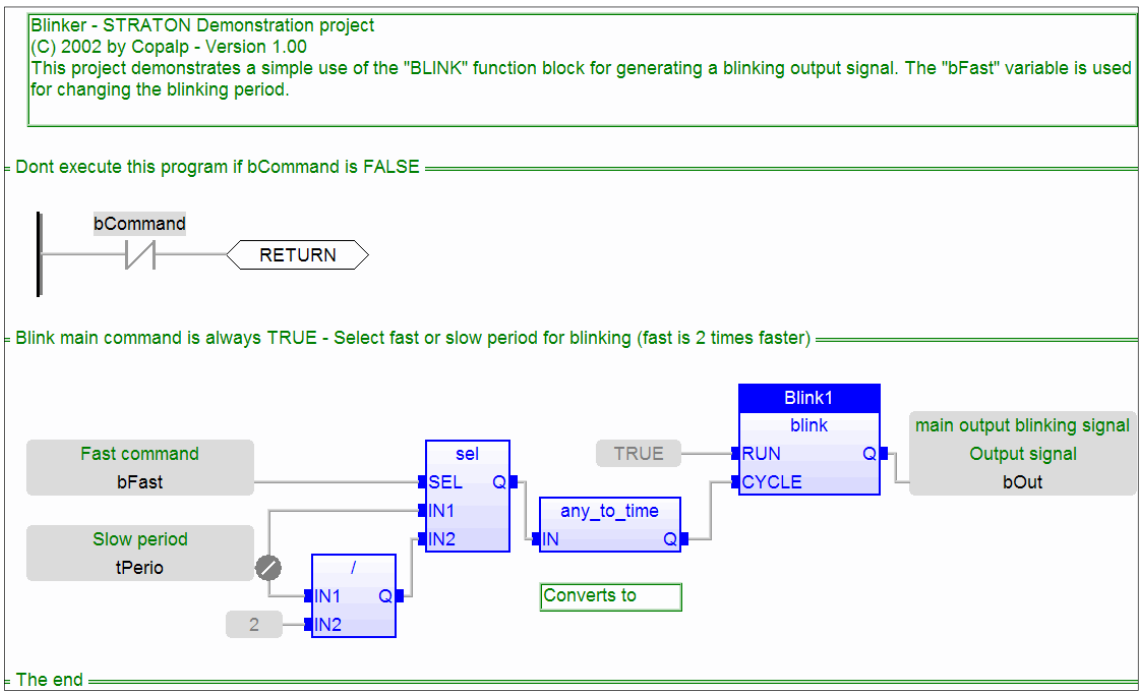


Fig. 63 Exemple de programme en langage FDB (Copalp 2002)

L'état des entrées d'un bloc en conditionne l'état des sorties, mais la logique n'est pas purement combinatoire dans la mesure où les blocs fonctionnels peuvent gérer des temporisations ou des expressions de calcul complexes.

Un des intérêts du langage FBD réside dans la réduction du nombre d'erreurs par l'utilisation de blocs validés. Le réemploi de sous-ensembles se trouve également facilité (Bertrand 2010). Dans notre cas, sa représentation graphique nous permet surtout de bien identifier les instances des vues de commande dans le programme généré.

II.4.2 Transformations d'assemblage

La génération automatique du programme élémentaire est réalisée suivant deux phases. La première génère un programme interprétable par Straton. La seconde importe ce programme ainsi que les vues de commande de la bibliothèque dans le modèle de programme de commande. Avant de détailler la réalisation de ces phases il est important de présenter la structure d'un projet réalisé sous Straton.

II.4.2.1 Structure d'un projet sous Straton

Un projet de commande sous Straton (Fig. 64) se compose d'un répertoire contenant les différents programmes, d'un fichier de configuration *projet.W5L* contenant le chemin d'accès au répertoire et d'un fichier *projet.xml* contenant la description des programmes. Le modèle de programme de commande contient un fichier *ModProg.xml* ainsi qu'un répertoire *ModProg*.

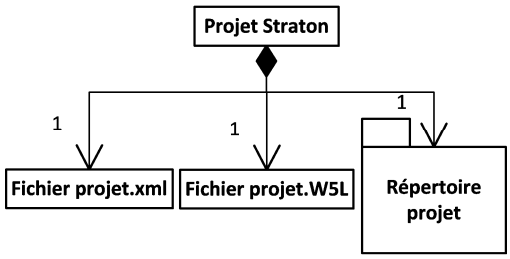


Fig. 64 Structure d'un projet de commande sous Straton

Le fichier *projet.xml* n'est pas exécutable directement mais peut être importé grâce à l'IDE Straton. Cette manipulation vient modifier le répertoire du projet. Le code devient alors compilable et exécutable.

II.4.2.2 Phase d'instanciation

La première phase de l'opération d'assemblage (**Génération Instances (1/2)** sur la Fig. 65) vise à créer un fichier regroupant les déclarations des variables issues des interfaces de la nomenclature ainsi que les instances des vues de commande. Elle génère un PSM temporaire *Instances.xml* à partir du PIM de nomenclature.

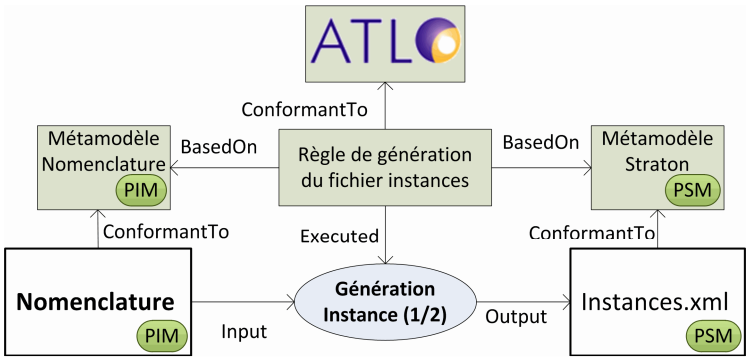


Fig. 65 Création du fichier temporaire Instances.xml

Nous avons déjà présenté le métamodèle de la nomenclature (Fig. 41), nous allons maintenant présenter le métamodèle de Straton auquel notre fichier *Interfaces.xml* est conforme.

Un projet Straton (*K5Project* sur la Fig. 66) se compose de programmes, de variables et de types. Chaque type déclaré peut se voir associer des variables (classe *TypeVar* sur la Fig. 66). Les variables (classe *Var* sur la Fig. 66) peuvent, elles, être rassemblées dans un groupe (classe *VarGroup* sur la Fig. 66). Les types et les variables sont utilisés dans les programmes. Ces derniers sont constitués de POUs qui regroupent des groupes de variables,

une définition (classe *Defines* sur la Fig. 66) ainsi que du code en langage FDB (classe *SourceFBD* sur la Fig. 66). Ce code est lui-même composé de blocs fonctionnels (classe *FBD* sur la Fig. 66), de blocs de variables (classe *FBDVarBox* sur la Fig. 66) et de liens (classe *FBDLine* sur la Fig. 66).

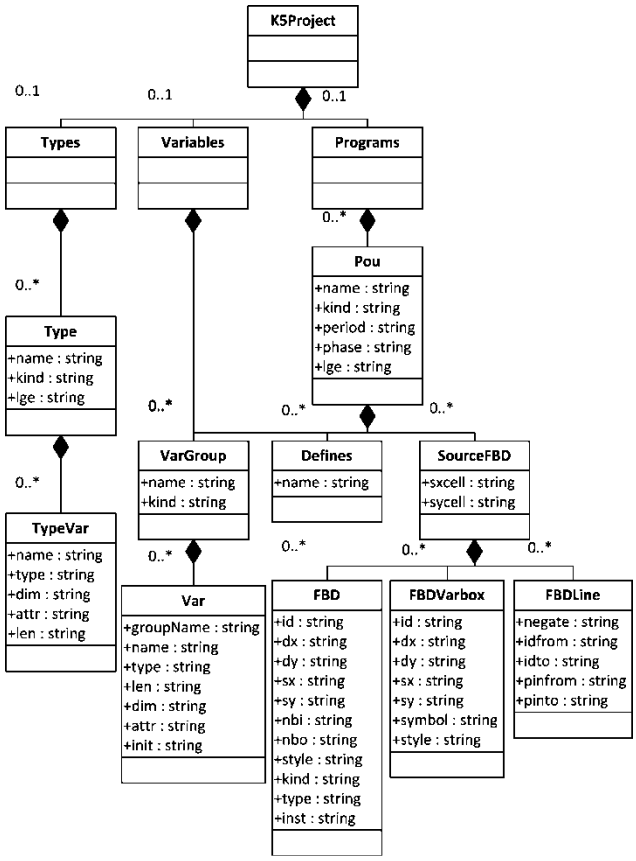


Fig. 66 Métamodèle d'un projet sous Straton

La création du fichier temporaire *Instances.xml* (**Génération Instance (1/2)** sur la Fig. 65) est implémentée au moyen d'une transformation ATL exogène verticale (cf. section I.3.3 p. 31). Elle comporte une règle principale (**ListOfPart2K5Project**) qui appelle 7 règles secondaires. On peut distinguer 3 étapes lors de l'exécution de cette transformation :

1. La première étape est celle de la déclaration des types et des classes associées (*Type* et *TypeVar* sur la Fig. 66). Chaque vue de commande référencée dans la nomenclature est ainsi déclarée comme type et les interfaces associées à cette vue de commande sont déclarées comme *TypeVar*.
2. La seconde étape est celle de la déclaration des variables et des classes associées (*VarGroup* et *Var* sur la Fig. 66). Chaque interface, associée aux vues de commandes référencées dans la nomenclature, est déclarée dans le groupe *Global*.
3. La dernière étape génère le code FBD du programme. Le programme comporte un seul POU nommé *Principal*. Les vues de commandes sont référencées comme variables locales de ce POU puis sont instanciées en tant que blocs fonctionnels (*FBD* sur la Fig. 66). Enfin les variables issues de la déclaration des interfaces

sont appelées sous forme de *FBDVarBox* (cf. Fig. 66) et elles sont reliées aux entrées et sorties des *FBD* par des *FBDLine* (cf. Fig. 66).

Le résultat de cette première étape est un fichier *Instances.xml* qui contient la description du programme de commande dans un format interprétable par Straton. Il reste néanmoins à inclure ce résultat dans le modèle de programme de commande ; c'est l'objet de la seconde étape de l'opération d'assemblage.

II.4.2.3 Phase d'importation

La deuxième phase de l'opération d'assemblage (**Import (2/2)** sur la Fig. 67) vise à finaliser le PSM de programme élémentaire à partir du PSM *Instances.xml* précédemment généré, des vues de commande de la bibliothèque (PSM), du PIM de nomenclature et du PSM de programme de commande.

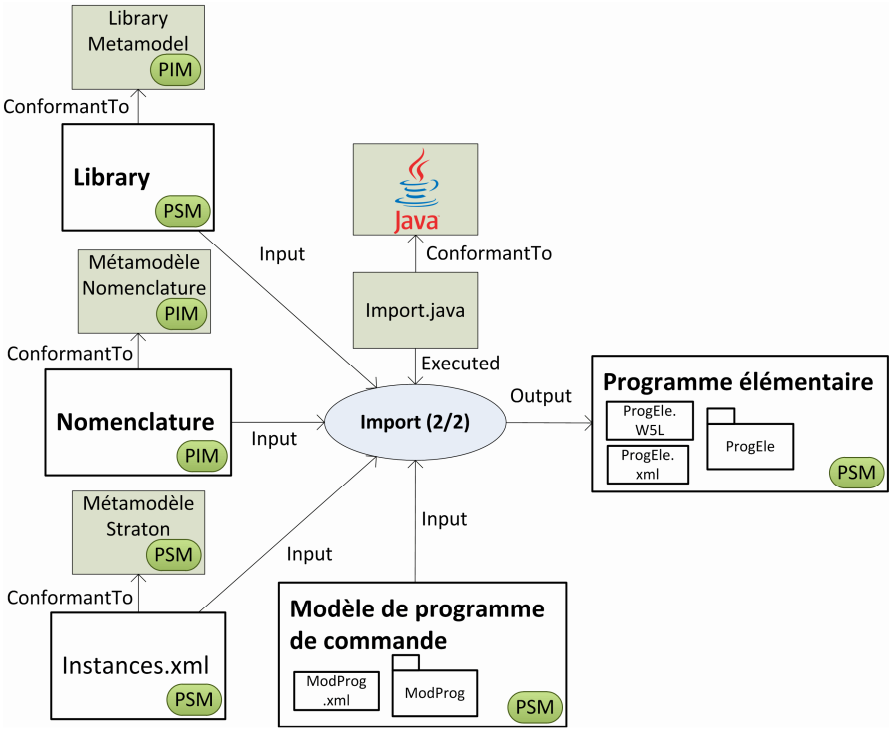


Fig. 67 Création du programme élémentaire

Cette phase est implémentée au moyen d'un applet java (**Import.java** sur la Fig. 67) que l'on peut assimiler à une transformation exogène verticale (cf. section I.3.3 p. 31). Six méthodes sont définies pour réaliser quatre étapes :

1. La méthode **CheminVCmd** parcourt la nomenclature pour lister les chemins d'accès aux vues de commande.
2. Les méthodes **CopyFile** et **CopyDirectory** sont utilisées pour copier le modèle de programme de commande dans le répertoire de destination du programme élémentaire. Elles sont également utilisées pour copier le contenu des vues synoptiques (répertoire et fichiers) dans leur répertoire de destination.

3. La méthode **CreerW5L** permet de générer le fichier de configuration *projet.W5L* (cf. Fig. 64). Ce fichier contient le chemin d'accès au répertoire du projet.
4. La méthode **ImportInstances** copie le contenu du fichier Instances.xml issu de la première phase de l'opération d'assemblage, dans le fichier *ProgEle.xml* du modèle de commande.

L'opération d'assemblage poursuit la **démarche descendante** entamée par l'opération d'inventaire (cf. chapitre II.2 p. 61).

Après avoir généré un fichier de commande au format XML interprétable par Straton (cf. section II.4.2.1 p. 83), nous avons intégré le contenu de ce fichier dans un modèle de programme de commande. Puis nous avons complété ce dernier par les vues de commande issues de la bibliothèque et référencées dans la nomenclature.

Il est alors possible de lire le programme élémentaire résultant avec l'IDE Straton. A la première utilisation il est nécessaire d'importer le fichier *ProgEle.xml* ; le contenu du répertoire *ProgEle* est alors modifié en conséquence par l'IDE. Le programme de commande élémentaire est alors compilable et exécutable.

II.5 Outillage, illustration et validation

Nous avons présenté les opérations de **construction** (cf. chapitre II.1 p. 45), d'**inventaire** (cf. chapitre II.2 p. 61), d'**insertion** (cf. chapitre II.3 p. 67) et d'**assemblage** (cf. chapitre II.4 p. 80) qui constituent notre flot de conception intégré (Fig. 21, p. 39).

Ces opérations ont été implémentées dans l'outil Anaxagore développé à des fins de validation de notre démarche. Nous présentons ci-après cet outil ainsi que l'application de la démarche sur le cas d'application de la soute à eau déminéralisée présentée, au début de cette partie (Fig. 33 p. 53).

II.5.1 Anaxagore

L'ensemble des transformations présentées ont été implémentées dans un outil sous forme de plugins à la plateforme Eclipse. L'ensemble de ces plugins compose l'outil Anaxagore développé pour expérimenter notre démarche.

Fig. 68 Écosystème de l'outil Anaxagore

La figure 68 schématise les interactions d'Anaxagore avec les logiciels sources et cibles de notre environnement de travail. Ainsi, il reçoit en entrée des modèles issus de Microsoft® Visio sous la forme de dessins XML et produit en sortie un répertoire contenant le programme de commande élémentaire pour Straton et un répertoire contenant le canevas d'IHM pour Panorama E2.

Anaxagore se compose de six ensembles distincts. La **bibliothèque**, tout d'abord (Anaxagore.0Library sur la Fig. 69) ; elle contient pour notre application les éléments vanne, soute, détecteur de niveau, transmetteur de niveau et mesure de niveau.

Ensuite nous retrouvons un ensemble relatif à l'opération de **construction**, constitué du modèle Visio utilisé pour la création du synoptique (*Anaxagore.1Building.0StandardSyn* sur la Fig. 69), ainsi que de

la transformation ATL d'épuration (*Anaxagore.1Building.0StandardSyn* sur la Fig. 69).

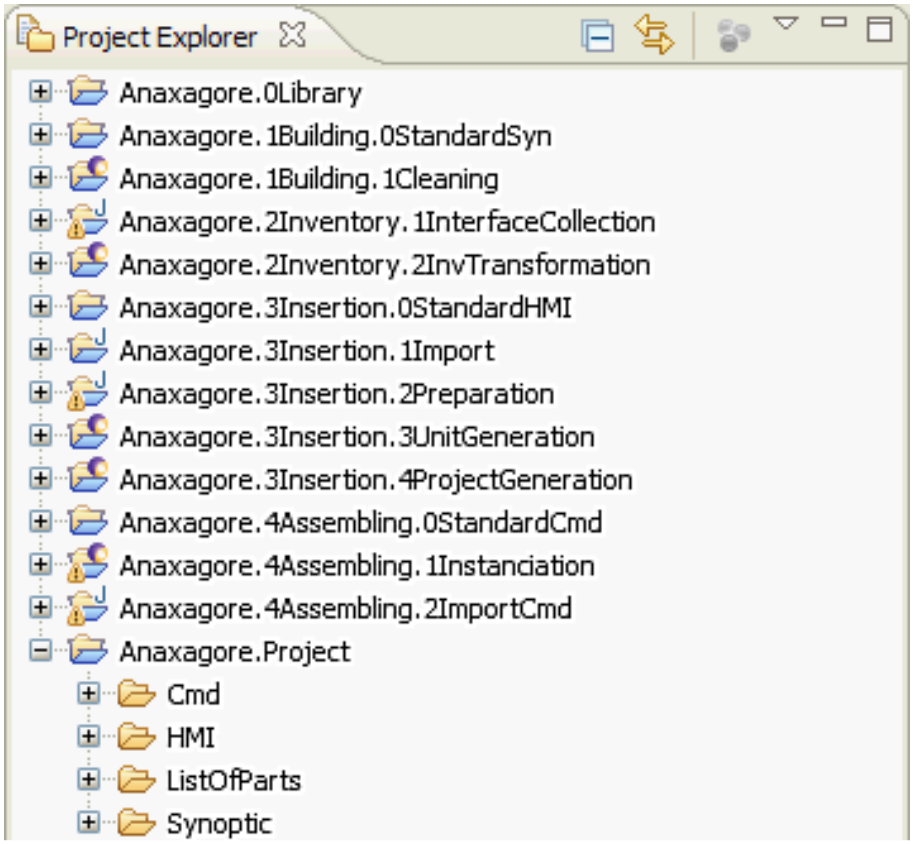


Fig. 69 Organisation Anaxagore

Nous avons également un ensemble relatif à l'opération d'**inventaire** qui contient l'applet Java (*Anaxagore.2Inventory.1InterfaceCollaction* sur la Fig. 69) de collecte des interfaces, ainsi que la transformation ATL d'inventaire (*Anaxagore.2Inventory.2InvTransformation* sur la Fig. 69).

Le troisième ensemble est relatif à l'opération d'**insertion**. Il est constitué du modèle standard d'IHM (*Anaxagore.3Insertion.0StandardHMI* sur la Fig. 69), de l'applet Java d'import des vues de supervision (*Anaxagore.3.Insertion.1Import* sur la Fig. 69), de l'applet Java de préparation du canevas (*Anaxagore.3Insertion.2Preparation* sur la Fig. 69), des transformations ATL de génération du fichier *Unit.cfg* (*Anaxagore.3Insertion.3UnitGeneration* sur la Fig. 69) et des transformations ATL de génération du fichier de configuration du canevas (*Anaxagore.3Insertion.4ProjectGeneration* sur la Fig. 69).

Le quatrième ensemble regroupe l'implémentation de l'opération d'**assemblage**. Il se compose du modèle de programme de commande (*Anaxagore.4Assembling.0StandardCmd* sur la Fig. 69), de la transformation ATL d'instanciation des vues de commande de la nomenclature (*Anaxagore.4Assembling.1Instanciation* sur la Fig. 69), ainsi que de l'applet Java d'importation des instances et des vues de commande dans le programme élémentaire (*Anaxagore.4Assembling.2ImportCmd* sur la Fig. 69).

Le dernier ensemble regroupe les modèles du **projet** (*Anaxagore.Project* sur la Fig. 69) incluant une composante synoptique (*Synoptic*), une composante nomenclature (*ListOfParts*), une composante interface de supervision (*HMI*) et une composante commande (*Cmd*).

Cette conception modulaire permet de diminuer l'impact d'un éventuel changement de logiciel source ou cible sur l'outil Anaxagore.

II.5.2 Illustration

Afin d'illustrer notre démarche et de valider le fonctionnement des outils développés, nous présentons dans cette section l'application de la démarche sur un cas d'exemple.

II.5.2.1 Construction

Nous nous basons sur le synoptique présenté au paragraphe II.1.3 (Fig. 33). Nous rappelons qu'il s'agit d'un extrait du synoptique de l'installation de production, de stockage et de distribution de l'eau douce sanitaire (EdS) détaillée en annexe.

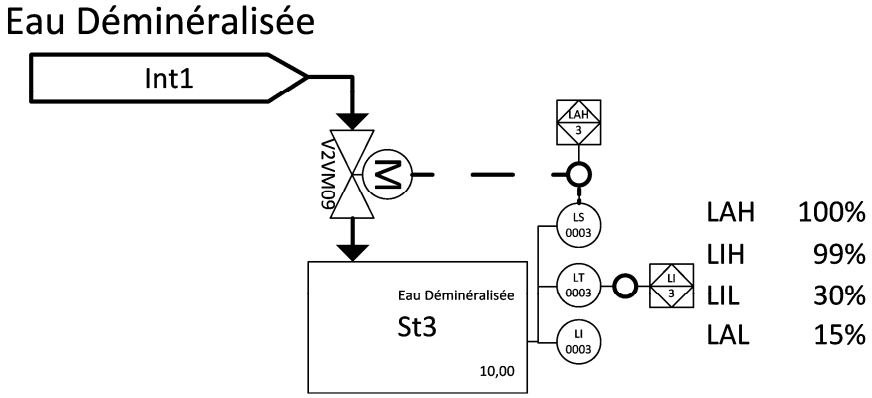


Fig. 70 Cas d'illustration - Soute à eau déminéralisée

Ce synoptique (Fig. 70) représente une soute de stockage d'eau déminéralisée (St3 sur la Fig. 70), isolée de sa ligne de remplissage (Int1 sur la Fig. 70) par une vanne motorisée à deux voies (V2VM9 sur la Fig. 70). La soute est équipée d'un indicateur de niveau mécanique (LI0003 sur la Fig. 70), d'un transmetteur de niveau (LT0003 sur la Fig. 70) et d'un détecteur de niveau (LS0003 sur la Fig. 70). Une indication du niveau de la soute est remontée à la supervision (LI3 sur la Fig. 70) depuis le transmetteur de niveau ; des seuils de détection d'information et d'alarme de niveau haut et bas sont associés à cette indication (LAH, LIH, LIL et LAL sur la Fig. 70). Une alarme de niveau haut (LAH3 sur la Fig. 70) est également remontée depuis le détecteur de niveau. Enfin, ce dernier envoie un signal électrique à la vanne en cas de détection afin de la fermer, indépendant du système de contrôle/commande.

La palette de l'outil de saisie présente les vues synoptiques des éléments de la bibliothèque. Chaque forme de ce synoptique correspond à l'une d'entre elles ; elles sont instanciées sur le schéma par glissé/déposé à

partir de la palette de l'outil. Le dessinateur peut alors compléter les informations complémentaires, comme la nature du fluide stocké ou le volume de la soute.

Le synoptique est saisi sous Microsoft® Visio 2010, le fichier qui en résulte est ensuite soumis à la transformation, d'épuration (Fig. 38).

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<syno:Synoptic [...] domain="PF" system="Aux" subSystem="EdS">
  <shape name="St" id="St3" coordX="35.52" coordY="10.938194">
    <ci nature="Fluide" value="Eau Déminéralisée" type="STRING" />
    <ci nature="Capacité" value="10" type="WORD" />
    <port number="1" />
    <port number="2" />
    <port number="3" />
    <port number="4" />
    <port number="5" />
    <port number="6" />
  </shape>
  <shape name="V2VM" id="V2VM09" coordX="32.52" coordY="21.398195" rotAngle="-90.0">
    <port number="1" />
    <port number="2" />
    <port number="3" />
    <port number="4" />
    <port number="1" />
  </shape>
...
  <bond id="P-1.29" type="Process" sens="monodirectionnal (1 to 2)">
    <extremity id="V2VM09" ext="Ext1" portNum="2" />
    <extremity id="St3" ext="Ext2" portNum="5" />
    <point x="31.07" y="18.4" />
    <point x="31.07" y="15.44" />
  </bond>
...
  <bond id="P-3.40" type="Process" sens="bidirectionnal">
    <extremity id="Int1" ext="Ext1" portNum="2" />
    <extremity id="V2VM09" ext="Ext2" portNum="3" />
    <point x="27.986364" y="28.081528" />
    <point x="31.04" y="28.081528" />
    <point x="31.04" y="24.398195" />
  </bond>
</syno:Synoptic>
```

Fig. 71 Extrait du fichier Synoptic.xml

La figure 71 présente un extrait du fichier issu de la transformation d'épuration. Ce synoptique appartient au domaine « plate-forme » (PF), au système auxiliaire (Aux) et représente le sous-système « eau douce sanitaire » (EdS).

On y retrouve la soute d'eau déminéralisée St3, la vanne motorisée à deux voies V2VM09 ainsi que la connexion de type « procédé » établie sur le schéma entre les deux (P-1.29 sur la Fig. 70). La connexion P-3.40 est celle établie entre l'interface du réseau (INT1 sur la Fig. 70) et la vanne.

Le fichier *Synoptic.xml* est ensuite repris comme point d'entrée pour l'opération d'inventaire.

II.5.2.2 Inventaire

L'exécution des transformations **collecte des interfaces** et **inventaire** (Fig. 47) nous renvoie un modèle de nomenclature conforme à notre metamodel (Fig. 41), sous la forme d'un fichier XML (Fig. 72).

La figure 72 présente un extrait de ce fichier XML et notamment les balises représentant la soute (St3 sur la Fig. 70), la vanne (V2VM09 sur la Fig. 70), la connexion de type « procédé » établie entre les deux (P-1.29 sur la Fig. 70), ainsi que la connexion établie entre l'interface INT1 et la vanne V2VM09 (P-3.40 sur la Fig. 70).

La balise *instance* de la soute comporte les attributs conformément au métamodèle (*family*, *designation*, *reference*, *id*, *vSyn*, *vSup* and *subSystem* sur la Fig. 72). Contrairement à la vanne, l'élément soute ne dispose pas de vue commande (*vCmd*). Les sous-balises *interface* sont donc toutes associées à la vue de supervision (attribut *view* sur la Fig. 72) de la soute St3. Les coordonnées de la soute sur le synoptique apparaissent dans la balise *coord*. On retrouve également les informations complémentaires (balises *ci*) *Capacity* et *Fluid*.

La première balise *bond* contient les informations relatives à la connexion monodirectionnelle de type procédé entre la vanne V2VM09 et la soute St3 (balises *extremity* sur la Fig. 72). Cette connexion est automatiquement référencée (P-1.29 sur la Fig. 72) lors du dessin du synoptique. Les coordonnées géométriques de la connexion apparaissent dans les sous-balises *point*.


```
<?xml version="1.0" encoding="ISO-8859-1"?>
<nomcl:ListOfPart [...] domain="PF" system="Aux" subSystem="EdS">
  <instance family="Process" designation="Soute" reference="St" id="St3" vSyn="St\Vsyn\" vSup="St\VSup\" subSystem="EdS">
    <interface reference="St" id="St3" information="Rep" designation="Repère de la soute" type="STRING" kind="Const" view="VSup" subSystem="EdS" />
    <interface reference="St" id="St3" information="Col" designation="Couleur du fluide" type="INT" kind="Var" view="VSup" subSystem="EdS" />
    <interface reference="St" id="St3" information="Fluide" designation="Nature du fluide stocké" type="STRING" kind="Const" view="VSup" subSystem="EdS" />
    <interface reference="St" id="St3" information="Capacite" designation="Capacité de la soute (m3)" type="REAL" kind="Const" view="VSup" subSystem="EdS" />
    <interface reference="St" id="St3" information="Select" designation="Soute selectionnée" type="BOOL" kind="Var" view="VSup" subSystem="EdS" />
    <coord x="35.52" y="10.938194" />
    <ci nature="Fluide" value="Eau Déminéralisée" type="STRING" />
    <ci nature="Capacité" value="10" type="WORD" />
  </instance>
  <instance family="Process" designation="Vanne 2 voies motorisée" reference="V2VM" id="V2VM09" vSyn="V2VM\Vsyn\" vSup="V2VM\VSup\" vCmd="V2VM\VCmd\" subSystem="EdS">
    <interface reference="V2VM" id="V2VM09" information="CtrlO" designation="Commande bistable issue de la supervision (1=ouverture; 0=fermeture)" type="BOOL" kind="Input" view="VCmd" subSystem="EdS" />
    <interface reference="V2VM" id="V2VM09" information="Tempo" designation="Temporisation de manoeuvre de la vanne" type="TIME" kind="Var" view="VCmd" subSystem="EdS" />
    <interface reference="V2VM" id="V2VM09" information="FdcO" designation="Fin de course d'ouverture" type="BOOL" kind="InPhy" view="VCmd" subSystem="EdS" />
    <interface reference="V2VM" id="V2VM09" information="FdcC" designation="Fin de course de fermeture" type="BOOL" kind="InPhy" view="VCmd" subSystem="EdS" />
    <interface reference="V2VM" id="V2VM09" information="StO" designation="Etat vanne ouverte" type="BOOL" kind="Output" view="VCmd" subSystem="EdS" />
    <interface reference="V2VM" id="V2VM09" information="StC" designation="Etat vanne fermée" type="BOOL" kind="Output" view="VCmd" subSystem="EdS" />
    <interface reference="V2VM" id="V2VM09" information="StF" designation="Etat défaut vanne" type="BOOL" kind="Output" view="VCmd" subSystem="EdS" />
    <interface reference="V2VM" id="V2VM09" information="CmdO" designation="Commande d'ouverture vers la vanne" type="BOOL" kind="OutPhy" view="VCmd" subSystem="EdS" />
    <interface reference="V2VM" id="V2VM09" information="CmdC" designation="Commande de fermeture vers la vanne" type="BOOL" kind="OutPhy" view="VCmd" subSystem="EdS" />
    <interface reference="V2VM" id="V2VM09" information="CtrlO" designation="Commande bistable issue de la supervision (1=ouverture; 0=fermeture)" type="BOOL" kind="Output" view="VSup" subSystem="EdS" />
    <interface reference="V2VM" id="V2VM09" information="StO" designation="Etat vanne ouverte" type="BOOL" kind="Input" view="VSup" subSystem="EdS" />
    <interface reference="V2VM" id="V2VM09" information="StC" designation="Etat vanne fermée" type="BOOL" kind="Input" view="VSup" subSystem="EdS" />
    <interface reference="V2VM" id="V2VM09" information="StF" designation="Etat défaut vanne" type="BOOL" kind="Input" view="VSup" subSystem="EdS" />
    <interface reference="V2VM" id="V2VM09" information="Col" designation="Couleur du fluide" type="INT" kind="Var" view="VSup" subSystem="EdS" />
    <interface reference="V2VM" id="V2VM09" information="Prop1" designation="Animation de la propagation1" type="BOOL" kind="Var" view="VSup" subSystem="EdS" />
    <interface reference="V2VM" id="V2VM09" information="Prop2" designation="Animation de la propagation2" type="BOOL" kind="Var" view="VSup" subSystem="EdS" />
    <interface reference="V2VM" id="V2VM09" information="Rep" designation="Repère de la vanne" type="STRING" kind="Const" view="VSup" subSystem="EdS" />
    <interface reference="V2VM" id="V2VM09" information="Select" designation="Vanne selectionnée" type="BOOL" kind="Var" view="VSup" subSystem="EdS" />
    <coord x="32.52" y="21.398195" angle="-90.0" />
  </instance>
...
  <bond id="P-1.29" type="Process" dir="monodirectionnal (1 to 2)">
    <extremity id="V2VM09" portNum="2" ext="Ext1" />
    <extremity id="St3" portNum="5" ext="Ext2" />
    <point x="31.07" y="18.4" />
    <point x="31.07" y="15.44" />
  </bond>
...
  <bond id="P-3.40" type="Process" dir="bidirectionnal">
    <extremity id="Int1" portNum="2" ext="Ext1" />
    <extremity id="V2VM09" portNum="3" ext="Ext2" />
    <point x="27.986364" y="28.081528" />
    <point x="31.04" y="28.081528" />
    <point x="31.04" y="24.398195" />
  </bond>
</nomcl:ListOfPart>
```


Fig. 72 Extrait du fichier ListOfParts.xml

La seconde balise *bond* (P-3.40 sur la Fig. 72) contient les informations relatives à la connexion bidirectionnelle de type procédé entre la vanne V2VM09 et l'interface du réseau (Int1 sur la Fig. 72). A noter que le deuxième point correspond aux coordonnées du coude que fait cette connexion sur le schéma.

Le modèle de nomenclature est ensuite utilisé pour générer le projet d'interface de supervision.

L'opération d'insertion (cf. chapitre II.3) produit une application exécutable par le logiciel Panorama E2 et basée sur notre modèle standard. Graphiquement, notre modèle standard est une application vide du contenu propre à la supervision des systèmes.

II.5.2.3 Insertion

Le résultat de l'opération d'insertion est illustré par la figure 73. Le canevas d'IHM généré contient les symboles issus des vues de supervision associées aux instances de la nomenclature.

On retrouve la soute St3 dont le volume de 10m^3 est conforme aux informations complémentaires extraites du synoptique (attribut *capacity* sur la Fig. 72). La couleur du barre-graphe de la soute dépend de la nature du fluide stocké ; le bleu correspond à de l'eau douce conformément à la norme ISO 14726-1 (ISO 1999). La hauteur du barre-graphe ainsi que la valeur affichée sous celui-ci correspondent à l'indication de mesure de niveau associée à la soute (LI3 sur la Fig. 70 et sur la Fig. 73), forcée pour l'exemple

à 6m^3. La position des seuils sur le barre-graphe correspond à leur valeur sur le synoptique (LAH, LIH, LIL, LAL sur la Fig. 70).

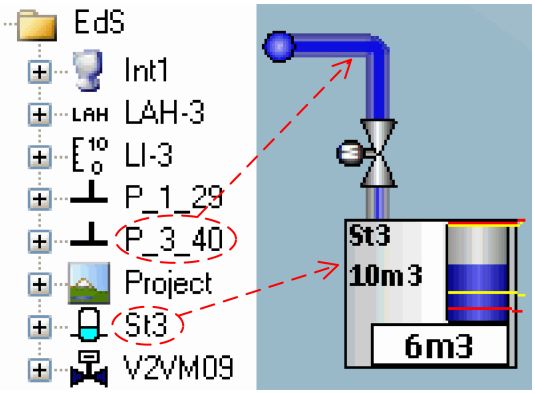


Fig. 73 Résultat de l'opération d'insertion

On retrouve également la connexion entre la vanne V2VM09 et l'interface du réseau Int1 (P_3_40 sur la Fig. 73). Le premier point (attribut point sur la Fig. 72) associé à cette connexion dans la nomenclature nous permet de positionner le premier segment de tuyauterie (le segment horizontal de l'interface au coude). Le deuxième point nous permet de positionner le coude de tuyauterie. Le troisième point nous permet de positionner le deuxième segment (le segment vertical entre le coude et la vanne).

II.5.2.4 Assemblage

Le modèle de nomenclature est également utilisé pour générer le programme de commande du sous-système. Le langage de programmation retenu pour l'exemple est le langage FDB (IEC, 2003). Il s'agit d'un langage graphique commode à visualiser.

La figure 74 est extraite du logiciel Straton IDE. Il s'agit du programme principal du projet de commande, généré par l'opération d'assemblage. Il regroupe les blocs fonctionnels issus des vues de commande associées aux instances de la nomenclature.

On retrouve le bloc de commande de la vanne V2VM09 qui reçoit en entrée les informations des capteurs de fin de course ouvert et fermé (*FdcO* et *FdcC* sur la Fig. 74), la commande issue de la supervision (*Ctrl_O* sur la Fig. 74), ainsi qu'un paramètre de temporisation (*Tempo* sur la Fig. 74) permettant la détection d'un problème de manœuvre de la vanne (*Ctrl_F* sur la Fig. 74).

En sortie, le bloc V2VM09 renvoie à la supervision les variables d'état de la vanne (*St_O*, *St_C* et *St_F* sur la Fig. 74) ainsi que les commandes d'ouverture et de fermeture destinées au matériel (*Cmd_O* et *Cmd_F* sur la Fig. 74).

Le bloc LS-0003 correspond au détecteur de niveau. Il reçoit le signal du capteur (*Mes* sur la Fig. 74) et renvoie l'information de détection (*Detect* sur la Fig. 74) et un défaut du capteur (*Defect* sur la Fig. 74).

Le dernier bloc de commande correspond au transmetteur de niveau (LT-0003 sur la Fig. 74). Il reçoit en entrée les paramètres de mise à l'échelle (*Emin* et *Emax* sur la Fig. 74), le signal correspondant à la mesure du capteur (*Mes* sur la Fig. 74) et les paramètres de seuil de détection (*PLAH*, *PLIH*, *PLIL* et *PLAL* sur la Fig. 74). Il renvoie en sortie le niveau mesuré et mis à l'échelle (*Niv* sur la Fig. 74), la capacité de la soute (*Vol* sur la Fig. 74), un défaut de mesure (*Defect* sur la Fig. 74) et des informations de détection de niveau liées aux seuils en entrées (*LAH*, *LIH*, *LIL* et *LAL* sur la Fig. 74).

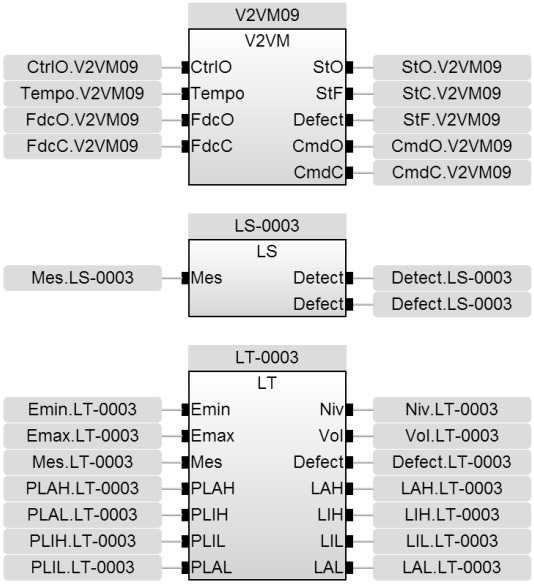


Fig. 74 Résultat de l'opération d'assemblage

Les variables connectées à ces blocs sont générées à partir des interfaces de la nomenclature. Les mnémoniques sont créés par concaténation des attributs *information* et *id* de l'interface (Fig. 41). Seules les interfaces associées aux vues de commande (attribut *view* sur la Fig. 41) sont ici générées. Les attributs *designation*, *type* et *kind* (Fig. 41) servent à la déclaration des variables sous l'IDE Straton.

Le projet d'interface et le programme de commande générés sont cohérents avec le modèle de synoptique utilisé pour la génération.

II.5.2.5 Alternative

Si nous modifions le modèle d'entrée, nous pouvons, très simplement et en quelques minutes, générer un nouveau projet d'IHM et un nouveau programme de commande. Ceci nous permet par exemple de proposer à moindre coût des alternatives de conception ou de prendre rapidement en compte des demandes d'évolution du système.

La figure 75 illustre la prise en compte d'une modification de notre synoptique d'entrée. Nous avons ajouté une interface (Int 2 sur la Fig. 75) de remplissage ainsi qu'une vanne motorisée (V2VM10 sur la Fig. 75) d'isolement de la soute.

Anaxagore génère un canevas d'IHM et un programme élémentaire dans lesquels figurent les modifications.

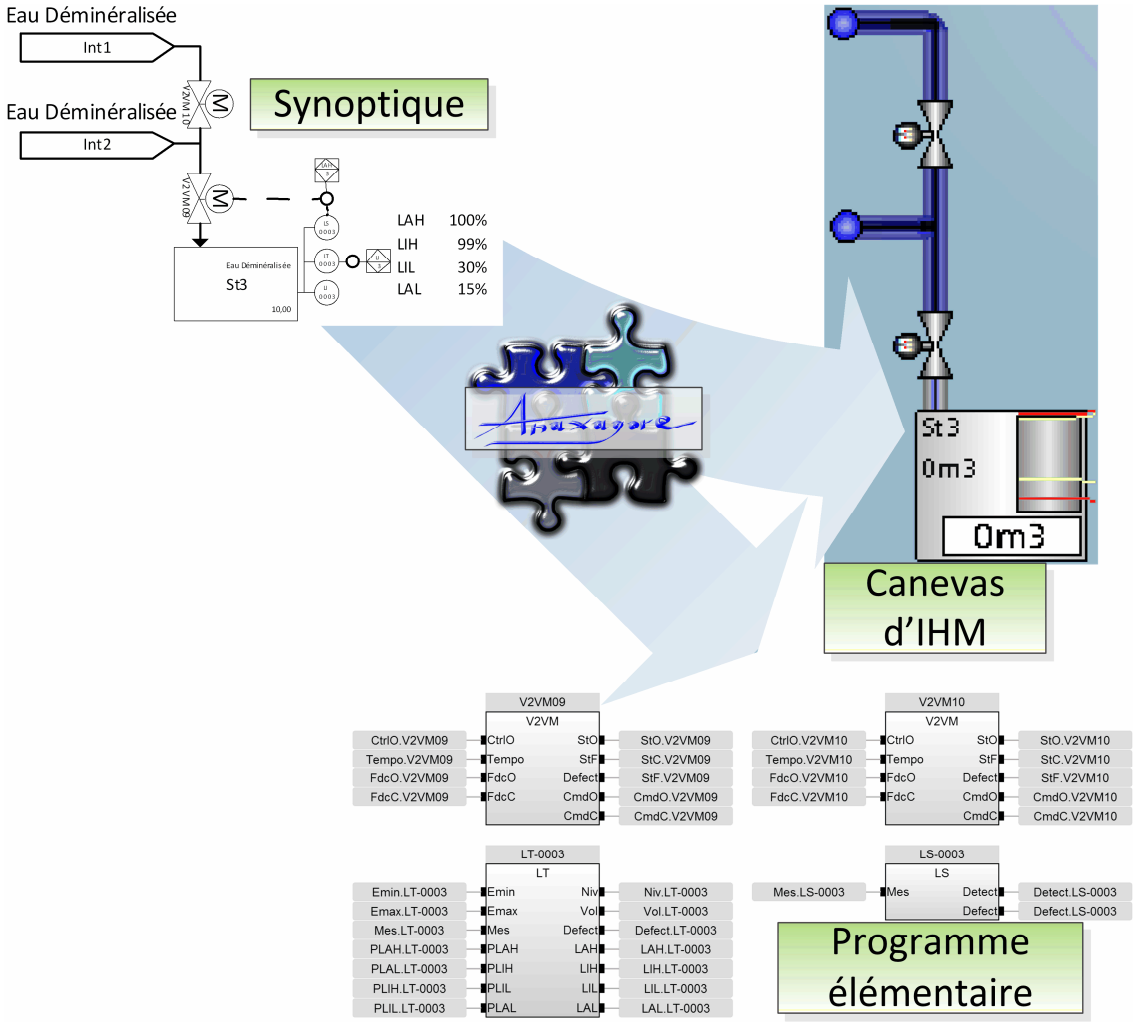


Fig. 75 Illustration d'une alternative

II.6 Bilan de l'implémentation du flot

Notre implémentation démontre que l'application croisée de différents paradigmes (objet, patron et modèle) et d'approches de conception vues comme concurrentes (ascendante et descendante) dans une démarche unifiée donne des résultats encourageants.

L'utilisation d'une bibliothèque d'éléments standards basée sur le concept d'objet, ainsi que sur des modèles de conception de programme et d'interface de supervision (ce dernier étant basé sur le concept de patron) répondent au besoin de **standardisation** exprimé par les industriels.

La mise en œuvre des techniques de transformation de modèle pour la génération automatique du programme de commande et d'interface de supervision garantit quant à elle la **cohérence**, d'une part entre le modèle d'entrée et les modèles générés, et d'autre part entre le projet d'interface et le programme de commande généré.

Par ailleurs, cette automatisation partielle du flot de conception concentre les échanges entre les intervenants du projet sur les sujets à **forte valeur ajoutée** (fonctionnement global, exploitation), facilite la correction d'erreur ou la prise en compte de nouvelles exigences (la propagation des évolutions est automatisée de la même manière) et réduit le temps de programmation.

L'utilisation d'un modèle métier existant **simplifie** la mise en œuvre de ce genre de démarche (si le langage fait déjà consensus et ne nécessite pas d'apprentissage supplémentaire), **limite les erreurs** de spécification puisque le langage est déjà maîtrisé et facilite la réutilisation de modèles existants qu'il suffit de saisir dans l'outil.

Cette démarche a fait l'objet d'une implémentation sous la forme du programme Anaxagore développé sur la plate-forme Eclipse. Nous avons vérifié que ce démonstrateur répondait à notre attente sur différents cas d'application. Il permet, en quelques minutes la génération d'un programme de commande et d'une interface de supervision conformes au synoptique utilisé en donnée d'entrée et cohérents entre eux.

Cependant, les programmes générés ne prennent pas en compte le fonctionnement global du système qui est pour l'instant laissé à la charge des concepteurs. Nous proposons dans la suite une extension du flot de conception qui permet justement de prendre en compte le système dans son ensemble et d'exploiter son caractère reconfigurable.

Enfin, si notre démarche offre un moyen rapide de générer des modèles de programme et d'interface, la cohérence du modèle d'entrée avec les exigences ne peut être garantie que par l'expert. De la même manière, toute intervention humaine sur les modèles générés devra se faire dans le respect de ces exigences, sous la responsabilité de l'intervenant.

Partie. III Extension fonctionnelle de la proposition

Cette partie présente une extension de notre flot de conception dont l'objectif est d'aborder l'aspect fonctionnel de la conception. Cette extension doit nous permettre de considérer le système conçu dans son ensemble et non plus seulement élément par élément.

Nous introduisons tout d'abord, les notions de fonction, configuration et commande dans le contexte de l'exploitation globale du système.

Nous poursuivons par la définition d'un modèle d'analyse des configurations, nécessaire à la validation des configurations et qui de plus permet la prise en compte du caractère reconfigurable du système.

Nous présentons l'implémentation de ce modèle d'analyse et son impact sur les modèles et transformations déjà définis.

Nous terminons par une proposition de démarche pour la génération de la commande et de la supervision complète du système.

III.1 Concepts et définitions

Nous avons proposé une démarche de conception qui permet, à partir d'un modèle très fortement orienté métier (le synoptique), de produire conjointement un canevas d'IHM et un programme élémentaire. Si l'on se replace dans le contexte de la démarche de conception simplifiée issue de notre retour d'expérience industriel (Fig. 7, p. 9), nous avons déroulé la branche issue de la définition physique du système.

La prise en compte du système conçu dans son ensemble et de son caractère reconfigurable nécessite le déroulement de la branche issue de la définition fonctionnelle. L'analyse fonctionnelle du système permet d'exprimer la manière dont les différents éléments du système doivent être utilisés pour satisfaire les besoins exprimés dans le cahier des charges (cf. section I.1.2.2.2 p. 10).

Certains de ces besoins nécessitent le développement de commandes globales et de dispositifs d'IHM associés facilitant la tâche des opérateurs en charge de l'exploitation du système. Nous définissons ci-après ce qu'est une commande globale et son implémentation suivant les points de vue « commande » et « supervision ».

La prise en compte du système conçu dans son ensemble et de son caractère reconfigurable, passe également par la définition d'un modèle dédié à l'analyse des possibilités de configuration et de reconfiguration qui s'appuie sur notre définition d'une configuration et d'une commande globale dans notre contexte. Ce modèle d'analyse se compose d'un graphe et de contraintes. Nous présentons également ces concepts dans la suite.

III.1.1 Fonction, configuration et commande

La démarche que nous avons décrite jusqu'à présent, met en avant les éléments constituants du système, en les considérant comme autant de briques utilisables pour le processus de conception. Si ce concept permet de constituer un système complexe par agrégation de ces briques élémentaires, l'exploitation globale de ce système ne peut se faire qu'en le considérant dans son ensemble. Pour cela, nous définissons les concepts de fonction, de configuration et de commande au niveau du système.

III.1.1.1 Fonction

Rappelons que l'on peut distinguer deux types de fonction (Niel and Craye 2002), les fonctions génériques et les fonctions contextuelles.

Les fonctions génériques sont associées aux éléments indépendamment de leur environnement. Dans le cas d'une vanne à deux voies par exemple, une fonction générique est sa fermeture.

Les fonctions contextuelles, comme leur nom l'indique, tiennent compte du contexte. Elles s'expriment dans le cadre de l'environnement d'un ou de plusieurs éléments. Si nous reprenons l'exemple de la vanne à deux voies,

une fonction contextuelle est le « sectionnement d'un segment de tuyauterie par fermeture de la vanne ». La fermeture de la vanne entraîne une modification du contexte d'exploitation.

Dans le contexte de l'exploitation globale du système, l'ensemble des fonctions est contextuel. Pour l'opérateur, l'exploitation du système consiste à surveiller son bon fonctionnement et à mettre en œuvre les fonctions contextuelles, qu'elles soient principales, contraintes ou complémentaires (au sens de la norme AFNOR X50 – 151 (AFNOR 2007)).

Nous retenons la définition suivante pour la notion de fonction :

Une fonction, du point de vue du système, est caractérisée par un couple configuration/objectif.

Déf. 6 Fonction

Dans ce cadre, l'objectif matérialise la raison d'être de la fonction et donc du système, tandis que la configuration matérialise le contexte du système.

Le cas d'application que nous avons défini précédemment n'est pas suffisamment riche pour illustrer ce concept ; aussi, nous prendrons désormais comme exemple une autre partie du système EdS défini en annexe. Il s'agit du stockage et de la distribution de l'eau douce, composé de deux soutes de stockage d'eau douce et de trois groupes hydrophores de distribution de l'eau. Ce système est bouclé de manière à permettre le transfert entre les deux soutes ainsi que le brassage d'une soute.

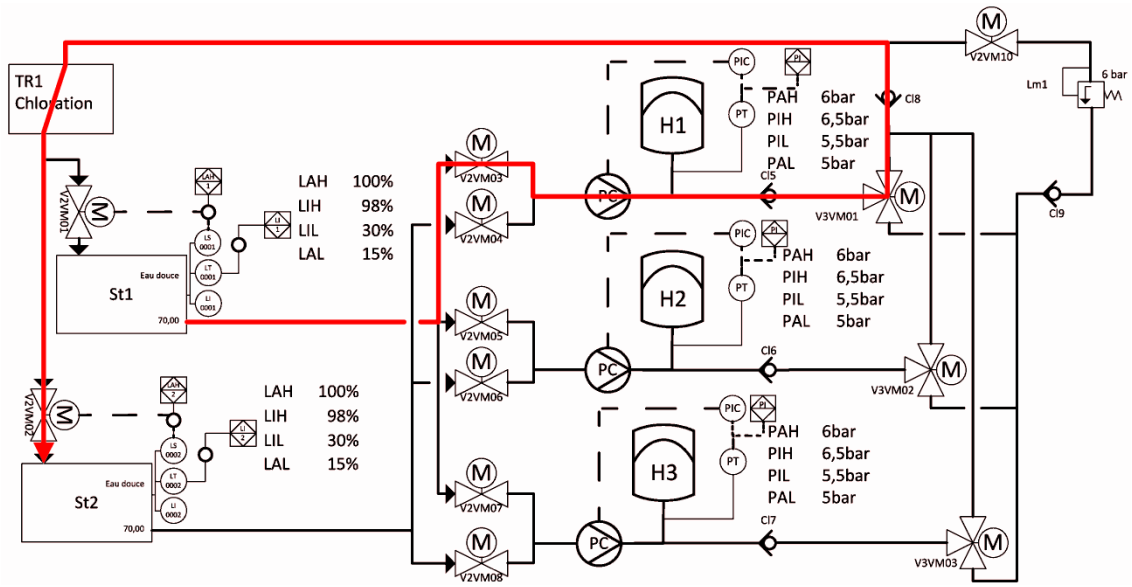


Fig. 76 Illustration d'une fonction

La figure 76 est l'illustration d'une fonction de transfert de la soute St1 à la soute St2 via l'hydrophore H1. L'objectif est ici de transférer une certaine quantité de fluide d'un point du système à un autre et nous présentons ci-après ce qu'est la configuration.

La fonction de transfert est une fonction complémentaire au sens de l'Afnor X50-151 (AFNOR 2007) et contextuelle suivant (Niel and Craye 2002). La description fonctionnelle du système en annexe fait apparaître la nécessité de développer une commande globale pour cette fonction.

III.1.1.2 Configuration

Conformément à la définition que nous avons donnée d'une fonction, nous retenons qu'une fonction donne le rôle d'une configuration. Une configuration est « une figure, ou une forme, résultant de la dépendance mutuelle ou de la disposition cohérente des parties d'un ensemble » (extrait de la 9ème édition du dictionnaire de l'Académie française). Plus précisément, dans notre cas, nous proposons la définition suivante :

Une configuration définit l'état des éléments du système nécessaires à la réalisation d'une fonction.

Déf. 7 Configuration

Cette définition rejoint celle proposée par (Frizon De Lamotte 2006) puis reprise par (Kanso 2010).

La spécification des configurations nominales associées aux fonctions est à la charge de l'expert qui conçoit le système. Il maîtrise l'architecture du système et a les connaissances nécessaires pour en expliquer le fonctionnement. Pour être exploitable par les outils de l'ingénierie dirigée par les modèles, une configuration doit être formalisée. A cette fin, nous avons défini le métamodèle suivant (Fig. 77).

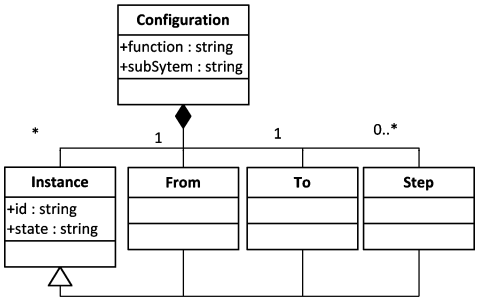


Fig. 77 Métamodèle d'une configuration

Une configuration est rattachée à une fonction (attribut *function* sur la Fig. 77) et à un sous-système (attribut *subSystem* sur la Fig. 77). Une configuration se compose d'instances ; chaque instance est identifiée (attribut *id* sur la Fig. 77). Un attribut permet d'exprimer l'état de l'instance souhaité pour la configuration (attribut *state* sur la Fig. 77). Cet état est exprimé sous la forme d'une chaîne de texte reprenant le nom d'une variable associé à l'instance et éventuellement une expression booléenne. Une instance particulière caractérise le point de départ de la configuration (classe *From* sur la Fig. 77) et une autre caractérise le point d'arrivée (classe *To* sur la Fig. 77). D'autres instances particulières caractérisent des points de passage obligés pour réaliser la fonction (classe *Step* sur la Fig. 77).

Si nous considérons la fonction de transfert de fluide de la soute St1 à la soute St2 via le groupe hydrophore H1 (Fig. 76), la configuration correspond à l'état du réseau de transport de fluide associé à cette fonction. Le tableau ci-dessous (Fig. 78) propose une configuration permettant de réaliser la fonction ; néanmoins, il peut exister d'autres configurations permettant de réaliser la même fonction.

	From	To	Step	Instances														
id	St1	St2	H1	V2VM01	V2VM02	V2VM03	V2VM04	V2VM05	V2VM06	V2VM07	V2VM08	V2VM10	V3VM01	V3VM02	V3VM03	H2	H3	Tr1
State	/	/	St_On	St_C	St_O	St_O	St_C	St_C	nc	St_C	nc	St_C	St_O2	not St_O2	not St_O2	nc	nc	nc

Fig. 78 Illustration d'une configuration

Dans un système reconfigurable, une même fonction peut être réalisée par plusieurs configurations différentes. Par ailleurs, une configuration ne considère que les éléments nécessaires à la réalisation d'une fonction, donc, sur l'ensemble du système il peut être possible de réaliser plusieurs fonctions simultanément.

Il existe néanmoins deux cas particuliers de configuration qui ne sont pas forcément associés à des fonctions, ou tout du moins dont le but n'est pas borné. Il s'agit des configurations de repos et de repli. La configuration de repos correspond à l'état que le système doit atteindre à la fin d'une fonction, la configuration de repli correspond à un état que le système doit atteindre pour garantir la sécurité.

Parfois ces deux configurations correspondent au même état du système, parfois elles sont différentes.

Concernant notre cas d'application présenté en annexe, la configuration de repos prépare le système à effectuer la distribution d'eau à bord. Les vannes entre les soutes et les groupes hydrophores sont ouvertes, les vannes à 3 voies sont positionnées vers la distribution et les groupes hydrophores sont en mode automatique.

La configuration de repli doit permettre d'assurer la sécurité des biens et des personnes en cas d'avarie. Elle revient à arrêter tous les matériels et à fermer toutes les vannes motorisées.

Cependant, les configurations de repos et de repli dépendent du système. Nous pouvons citer l'exemple d'un système de refroidissement d'un réacteur nucléaire pour lequel la position de repli privilégiée serait d'ouvrir les vannes afin de limiter le risque de fusion du cœur.

La mise en œuvre d'une configuration est réalisée par une commande globale.

III.1.1.3 Commande globale

III.1.1.3.1. Définition d'une commande globale

Une fonction est exprimée dans le système par le biais d'une commande globale. Au sens littéral, une commande est un ordre et l'adjectif global signifie que cet ordre s'applique à un ensemble de choses matérielles (extrait de la 9ème édition du dictionnaire de l'Académie française).

Une commande globale est un ensemble de commandes élémentaires exécutées dans un ordre précis pour répondre au besoin exprimé par la fonction.

Déf. 8 Commande globale

On retrouve ici la notion de but qui déterminera, d'une part, la consigne nécessaire au lancement de la commande et, d'autre part, la condition mettant fin à la commande et entraînant le retour à une configuration de repos.

Si nous reprenons l'exemple de la figure 76, la commande globale de cette fonction définit les commandes élémentaires à émettre pour atteindre la configuration (Fig. 78), ainsi que l'ordre d'exécution de ces commandes. La prise en compte du besoin est une condition qui permet la fin de la fonction, comme un volume à transférer qui serait atteint.

Une commande globale comporte un volet « commande » comparable à la vue commande des éléments de la bibliothèque et un volet « supervision » comparable à la vue supervision des éléments.

Ces vues ne peuvent être complètement définies par avance puisqu'elles dépendent de l'architecture du système définie par le synoptique. Néanmoins, les invariants de ces vues peuvent faire l'objet de patrons de conception que nous présentons ci-après.

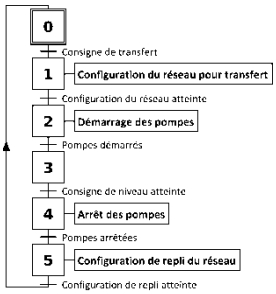
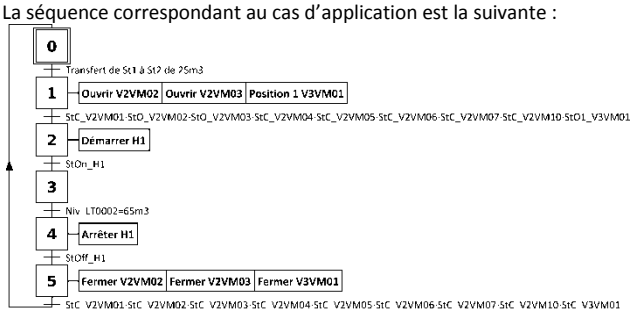
III.1.1.3.2. Le patron de séquence

Si nous reprenons la définition 8, nous comprenons que la commande globale se compose de commandes élémentaires. Ces dernières ne peuvent être définies par avance puisqu'elles dépendent des éléments du système. Par contre, ces commandes élémentaires sont réalisées dans un ordre précis qui, lui, peut-être prédéterminé.

Cet ordre est matérialisé par une séquence qui vise à répondre au besoin exprimé par la fonction. Il existe donc une séquence par fonction.

Le patron de séquence (Fig. 79) représente le volet « commande » de la commande globale. Nous le représentons suivant le formalisme P-Sigma (Conte, Fredj et al. 2001).

Rubrique	Champ
Interface	
Identifiant	Séquence de transfert
Classification	Offre un modèle de séquence d'un transfert entre deux points sur un système de gestion des fluides embarqués.
Contexte	Nécessite l'existence d'une bibliothèque d'objets de commande préétablis et compatibles avec le modèle de séquence.
Problème	Permet de construire une commande globale de transfert pour un système de gestion des fluides.
Force	Permet d'automatiser la prise en compte de l'ensemble du système dans la création d'une commande de transfert.

Rubrique	Champ
	Réalisation
Solution/ Démarche	1. Définir la consigne de la séquence (objectif de la fonction). 2. Définir le déroulement de la séquence. 3. Définir les contraintes de la séquence.
Solution/ Modèle	Un modèle obtenu après application de la solution est le suivant : La consigne de séquence prend la forme : « **Transfert du point A au point B d'un volume de X** » Le déroulement de la séquence doit être conforme au grafcet ci-contre. Les contraintes sont les suivantes : 1. Le volume stocké au point A doit être supérieur à son volume minimal. 2. Le volume stocké au point B doit être inférieur à son volume maximal. 3. Le volume X doit être inférieur ou égal au volume stocké en A et au volume disponible en B La consigne de transfert est issue de la supervision exclusivement sur demande d'un opérateur. 
Cas d'application	Nous appliquons le patron pour une fonction de transfert et pour la configuration illustrée par la figure 78. Les conditions initiales sont les suivantes : ▪ St1 contient 50m3 d'eau et St2 40m3. ▪ Le réseau est en configuration de repli (toutes les vannes sont fermées et toutes les pompes sont arrêtées). Les contraintes à respecter sont les suivantes : 1. Le volume stocké en St1 est supérieur à $0m^3$. 2. Le volume stocké en St2 est inférieur à $70m^3$. 3. Le volume à transférer doit être inférieur à $30m^3$ (volume disponible en St2). La consigne correspondant au cas d'application est la suivante : **Transférer $25m^3$ de St1 à St2.** La séquence correspondant au cas d'application est la suivante : 
Conséquence d'application	Ce patron offre une séquence pour la réalisation d'un transfert entre deux points sur un système de gestion des fluides embarqués. Cette séquence suppose de connaître par avance la configuration à atteindre. Il peut être enrichi pour prendre en compte des aléas dans le déroulement de la séquence (interruption en cas de défaut de commande par exemple).

Rubrique	Champ
	Relation
Utilise	Le modèle standard d'IHM navire.
Raffine	/
Requiert	L'existence d'un dispositif de commande susceptible de recevoir la consigne (patron de séquence de transfert).
Alternative	/

Fig. 79 Patron de séquence de la commande globale de transfert

Le patron de séquence est exprimé sous la forme d'un pseudo-code en SFC[1] (IEC 2003) supporté par l'atelier de développement Straton pour lequel nous avons défini un métamodèle (Fig. 66, p. 84) présenté dans la partie II. Par ailleurs, le langage SFC peut être retranscrit sous la forme d'un fichier XML (PLCOpen 2009) facilement manipulable par les outils de l'IDM.

III.1.1.3.3. Le patron de supervision

Pour être exploitable par l'opérateur, la commande globale doit être accessible depuis une IHM. Le patron de supervision propose une ébauche

[1] SFC : Sequential Function Chart

d'interface de contrôle pour une commande globale. La figure 80 présente le patron de supervision de la commande globale de transfert sous le formalisme P-Sigma (Conte, Fredj et al. 2001).

Rubrique	Champ
Interface	
Identifiant	Interface de transfert
Classification	Offre un modèle d'interface pour la saisie d'une consigne de transfert entre deux points sur un système de gestion des fluides embarqués.
Contexte	Nécessite l'existence d'une bibliothèque d'objets de supervision préétablis et compatible avec le modèle d'interface ainsi que d'un modèle standard d'IHM.
Problème	Permet de construire une interface commande globale de transfert pour un système de gestion des fluides.
Force	Permet d'automatiser la prise en compte de l'ensemble du système dans la création d'une interface de commande de transfert.

Rubrique	Champ
Réalisation	
Solution/ Démarche	1. Définir la consigne de la séquence (objectif de la fonction). 2. Définir le principe de la saisie.
Solution/ Modèle	Le modèle obtenu après application de la solution doit être conforme au modèle suivant : La consigne de séquence prend la forme : « **Transfert du point A au point B d'un volume de X** » L'interface de saisie doit permettre de renseigner cette consigne, quatre dispositifs sont nécessaires : 1. Saisie du point A 2. Saisie du point B 3. Saisie du volume X 4. Validation de la consigne Les contraintes sont les suivantes : 1. Pour pouvoir être sélectionné en point A, un point doit contenir un stock supérieur à son volume minimal. 2. Pour pouvoir être sélectionné en point B, un point doit contenir un stock inférieur à son volume maximal. 3. Le volume X doit être inférieur ou égal au volume stocké en A et au volume disponible en B.
Cas d'application	Une solution pour la supervision de la séquence est la suivante : Il s'agit d'une fenêtre ouvrable à partir d'un bouton accessible dans la zone « commande globale » du canevas d'IHM du système. La fenêtre comporte : ▪ une liste de choix pour le point de départ (point A), ▪ une liste de choix pour le point d'arrivée (point B), ▪ une zone de saisie pour le volume à transférer avec des boutons d'ajustement de la valeur (volume X), ▪ une zone de validation/annulation de la consigne. Nous ajoutons en tête de la fenêtre un dispositif permettant de régler la priorité de cette consigne par rapport aux consignes d'autres commandes globales. Cette priorité peut prendre 3 valeurs (haute, moyenne ou basse). Cette notion de priorité permet d'assurer la continuité de service pour les commandes les plus importantes par rapport aux commandes jugées moins importantes.
Conséquence d'application	Ce patron offre une interface de saisie d'une consigne pour la réalisation d'une commande globale de transfert.

Rubrique	Champ
Relation	
Utilise	Le modèle standard d'IHM navire.
Raffine	/
Requiert	L'existence d'un dispositif de commande susceptible de recevoir la consigne (patron de séquence de transfert).
Alternative	/

Fig. 80 Patron de supervision de la commande globale de transfert

Le patron de supervision est un projet d'interface réalisé sous Panorama E2 (cf. section II.3.2.1 p. 72). Il doit faire l'objet d'une spécialisation pour être inséré dans le canevas d'IHM du système conçu. Cette spécialisation peut être implémentée par des transformations de modèle.

Le patron de supervision est nécessairement lié à un patron de séquence (Fig. 79) et au modèle standard d'IHM (Fig. 49, p. 70).

III.1.1.4 Conclusion sur les concepts

Nous avons défini la notion de **commande globale** dont le but est de mettre en œuvre une **fonction** du système en s'appuyant sur une **configuration** du système et sur une **séquence** d'exécution. Afin de formaliser ces notions, nous avons proposé un métamodèle de configuration. Puis nous avons défini un **patron de séquence** pour une commande globale de transfert et, associé à cette séquence, nous avons également proposé un **patron de supervision** permettant à l'opérateur de saisir la consigne de la commande globale. Cette consigne matérialise l'**objectif** de la fonction.

Ces concepts permettent d'envisager la génération automatique de commandes globales en complément du programme élémentaire présenté précédemment. Le canevas d'IHM peut également être modifié automatiquement pour y ajouter l'interface de la commande globale.

Au delà de la génération automatique de la commande, il nous apparaît important de disposer d'un modèle d'analyse des configurations que nous présentons ci-après.

III.1.2 Modèle d'analyse des configurations

La spécification des configurations nominales reste à la charge de l'expert qui conçoit le système. Néanmoins les configurations proposées par l'expert peuvent faire l'objet d'une vérification automatique. Cela constitue la première étape vers la recherche automatique de configurations alternatives et donc la prise en compte du caractère reconfigurable du système.

III.1.2.1 Le graphe de potentiel et les contraintes

Nous avons vu précédemment que construire une commande globale s'apparente à rechercher un chemin dans un graphe. Le synoptique s'apparente en effet à un graphe complexe puisqu'il est constitué de symboles reliés entre eux par des connexions. Pour exploiter facilement des techniques de recherche de chemin, nous devons cependant construire une représentation plus adaptée de notre système.

Une solution nous a été inspirée par les travaux de Butler (Butler, Sarma et al. 2001). Elle consiste à considérer le système comme un ensemble de potentiels représenté par des sommets entre lesquels il peut exister des liaisons orientées représentées par des arcs. Nous appelons ce modèle le graphe de potentiel.

Associé à ce graphe et afin de prendre en compte les spécificités d'un matériel ou les exigences d'une configuration, nous définissons un modèle de contraintes. L'ensemble graphe de potentiel et contraintes forme le modèle d'analyse des configurations (MAC). Chaque élément de type procédé de la bibliothèque se verra attribuer une vue MAC pour modéliser son comportement. Nous avons défini le métamodèle suivant pour le MAC (Fig. 81).

Le MAC (*AnalysisModel* sur la Fig. 81) est référencé par rapport à un système (attribut *subSystem* sur la Fig. 81). Il se compose d'un graphe de

potentiel (*Potential Graph* sur la Fig. 81) et d'un ensemble de contraintes (*Constraints* sur la Fig. 81).

Le graphe de potentiel est composé de sommets (*Vertex* sur la Fig. 81) et d'arcs (*Edge* sur la Fig. 81).

Les sommets peuvent être des nœuds internes à une vue MAC (*InternalNode* sur la Fig. 81) ; dans ce cas ils contiennent un port virtuel (*GhostPort* sur la Fig. 81) qui permet d'une part, d'associer le nœud à son instance (attribut *id* de la classe *GhostPort* sur la Fig. 81) et d'autre part, de référencer le nœud dans la vue MAC (attribut *number* de la classe *GhostPort* sur la Fig. 81). Les sommets peuvent également être des nœuds d'ancrages (*AnchorageNode* sur la Fig. 81) ; dans ce cas ils contiennent un ou plusieurs ports (*Port* sur la Fig. 81) qui matérialisent les interfaces physiques d'une instance (comme le raccord d'une vanne par exemple).

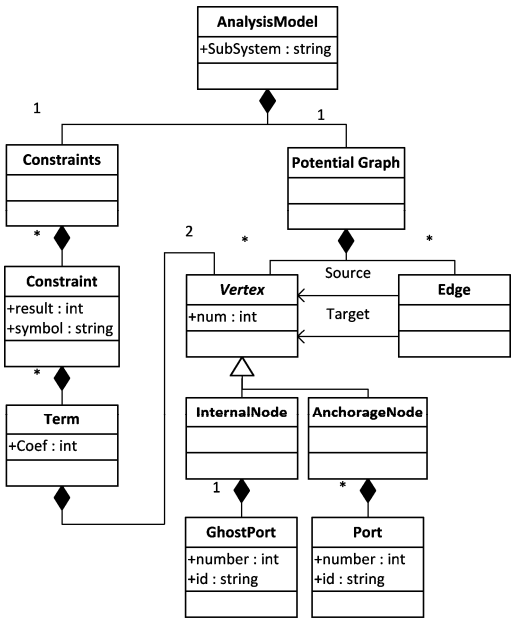


Fig. 81 Métamodèle du MAC

Les arcs sont associés aux sommets par des relations d'origine (*source* sur la Fig. 81) et de cible (*target* sur la Fig. 81).

Une contrainte est exprimée par une équation ou une inéquation portant sur des arcs du graphe de potentiel. Elle contient un résultat (attribut *result* sur la Fig. 81), un symbole mathématique (attribut *symbol* sur la Fig. 81) et des termes (*Term* sur la Fig. 81). Chaque terme de la contrainte associe un coefficient (attribut *coef* sur la Fig. 81) et un arc du graphe. Ce dernier est matérialisé par la relation de composition qui lie 2 sommets (*Vertex* sur la Fig. 81) à un terme.

Le modèle d'analyse des configurations d'un système peut être construit par assemblage des vues MAC associées aux éléments de la bibliothèque.

Le modèle d'analyse des configurations permet la mise en œuvre d'algorithmes de recherche de flot maximum dans un graphe orienté pour déterminer les configurations.

Certaines configurations nécessitent d'emprunter des chemins disjoints pour maximiser le flot. C'est par exemple le cas d'un transfert utilisant les trois groupes hydrophores simultanément.

Pour cela il est nécessaire d'affecter des capacités aux arcs. En première approximation, nous considérons que le réseau de tuyauterie est suffisamment dimensionné pour supporter la charge de l'ensemble de ses pompes tournant simultanément. Cette hypothèse n'est pas très éloignée de la réalité pour les systèmes embarqués sur les navires. Dans ce cas, la capacité maximale des arcs correspond à la somme des capacités affectées aux pompes.

Une fois cette hypothèse prise en considération, une configuration peut emprunter des chemins disjoints dans la limite du flot maximal admissible, matérialisé par la capacité maximale de l'ensemble des pompes.

Notre modèle d'analyse des configurations fait apparaître la notion de potentiel. Nous illustrons cette notion dans la suite en prenant pour exemple la vue MAC d'un élément.

III.1.2.2 La notion de potentiel et la vue graphe

Un potentiel est une ressource dont un objet peut disposer (inspiré de la 9ème édition du dictionnaire de l'Académie française).

Dans le cas d'un système de transport et de traitement de fluide :

Le potentiel d'un point du système correspond à la pression du fluide en ce point en régime permanent.

Déf. 9 Potentiel

La disposition des arcs d'un potentiel à un autre modélise le comportement du système.

Pour illustrer notre définition, nous nous basons sur la vue MAC d'une vanne à 2 voies. Cette vue MAC est vierge de toute contrainte, elle se compose uniquement du graphe de potentiel suivant (Fig. 82).

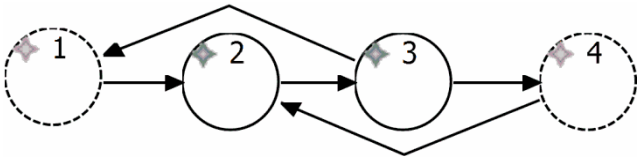


Fig. 82 Vue graphe d'une vanne à deux voies

Les sommets en pointillés (1 et 4 sur la Fig. 82) sont des nœuds d'ancrages qui servent à la construction du graphe complet. Un nœud d'ancrage hérite des caractéristiques d'un sommet classique mais il contient également une liste de connecteurs (Port sur la Fig. 81) nous permettant d'identifier les connexions qui lui seront associées.

Le comportement modélisé pour la vanne à 2 voies est le suivant : l'arc {2,3} peut laisser passer un flot si la vanne est ouverte, et le fluide peut alors circuler des sommets 1 à 4 par les arcs {1,2}, {2,3}, et {3,4} ou des

sommets 4 à 1 par les arcs {4,2}, {2,3} et {3,1}. En aucun cas le fluide ne peut circuler dans les deux sens en même temps dans la vanne.

L'état « fermé » de la vanne peut être exprimé par une contrainte associée à ce graphe. Cette contrainte prend la forme de l'équation suivante :

$$x_{2,3} = 0$$

La variable $x_{2,3}$ renseigne l'état de l'arc {2,3}. Cette contrainte empêche le flot de traverser la vanne, les potentiels des sommets 1 et 4 peuvent alors être différents. Les contraintes associées au graphe de potentiel du système permettent d'exprimer des exigences particulières d'une configuration (vanne fermée, pompe en marche, etc.).

Il est également possible d'associer des contraintes, propres à un élément, dans sa vue MAC. Prenons l'exemple d'une vanne à trois voies dont la vue graphe est présentée sur la figure 83.

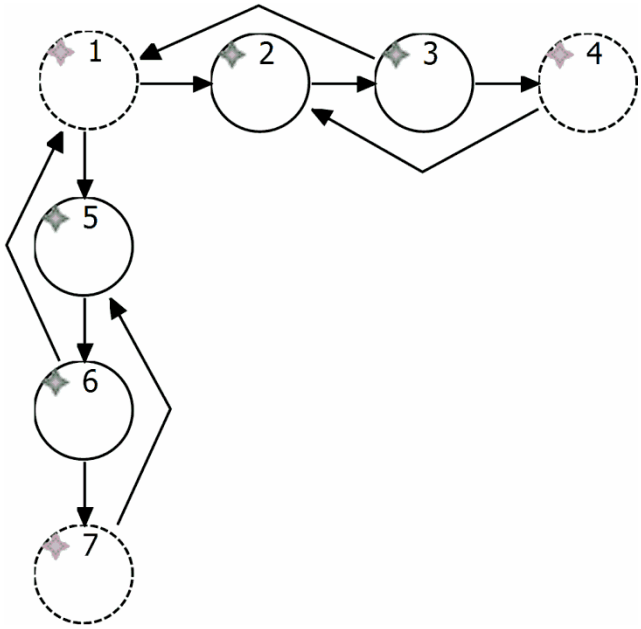


Fig. 83 Vue graphe d'une vanne à trois voies

Le comportement modélisé permet de faire passer un flot du sommet 1 au sommet 4 et inversement, ainsi que du sommet 1 au sommet 7 et inversement. Mécaniquement, la vanne à trois voies que nous avons retenue, ne permet pas de mettre en communication les sommets 4 et 7 avec le sommet 1 en même temps (voir Fig. 84).

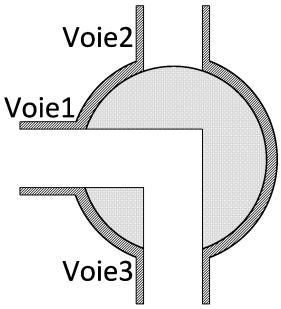


Fig. 84 Coupe d'une vanne à trois voies

Il est donc nécessaire d'ajouter une contrainte. Elle s'exprime par l'équation suivante :

$$x_{2,3} + x_{5,6} \leq 1$$

Cette contrainte impose que le flot ne puisse passer que par un des arcs {2,3} ou {5,6} à la fois. Dans ce cas, la contrainte n'est pas visible sur le graphe de potentiel mais est écrite dans le fichier XML du MAC (Fig. 85).

Les graphes de potentiel, bien que saisis à l'aide d'un outil développé sous GMF, peuvent être considérés comme des PIM dans la mesure où les fichiers résultants sont des fichiers textes au format XML. En conséquence, comme les contraintes sont exprimées sous la forme de texte XML, notre modèle d'analyse des configurations est indépendant de toute plateforme.

```
<Constraints>
  <Constraint symbol="=Inf" result="1">                    ≤1
    <Term Coef="1">
      <Vertex xsi:type="con:InternalNode" num="2">
        <GhostPort id="V3VM1" number="1" />
      </Vertex>
      <Vertex xsi:type="con:InternalNode " num="3">        1×x2,3
        <GhostPort id="V3VM" number="2" />
      </Vertex>
    </Term>                                                +
    <Term Coef="1">
      <Vertex xsi:type="con:InternalNode " num="5">
        <GhostPort id="V3VM" number="5" />
      </Vertex>
      <Vertex xsi:type="con:InternalNode " num="6">        1×x5,6
        <GhostPort id="V3VM" number="3" />
      </Vertex>
    </Term>
  </Constraint>
</Constraints>
```

Fig. 85 Contrainte d'une vanne à troies voies

La construction du modèle d'analyse complet, par assemblage des vues MAC des éléments qui composent le système, nécessite d'identifier les nœuds d'ancrages communs à plusieurs graphes de potentiel des vues MAC. Ce mécanisme passe par la recherche d'équipotentiels dans le système. Nous définissons ci-après cette notion.

III.1.2.3 La notion d'équipotentiel

La première étape pour la construction d'un graphe de potentiel est d'identifier les zones d'équipotentiels du système sur le synoptique. Un équipotentiel dans les cas d'un système de gestion des fluides répond à la définition suivante :

Un équipotentiel désigne une zone du système dont tous les points sont au même potentiel.

Déf. 10 Équipotentiel

De fait, toutes les connexions de type procédé de la nomenclature qui ont une extrémité commune ont le même potentiel, elles font donc partie du même équipotentiel.

La figure 86 illustre le concept d'équipotentiel sur les portions de tuyauterie situées respectivement entre St1, V2VM03, V2VM05 et V2VM07 ainsi qu'entre St2, V2VM04, V2VM06 et V2VM08. Un équipotentiel associe les connexions de la nomenclature à un sommet fictif du graphe de potentiel. Ce

dernier servira de « point d'ancrage » aux nœuds d'ancrages des vues graphes.

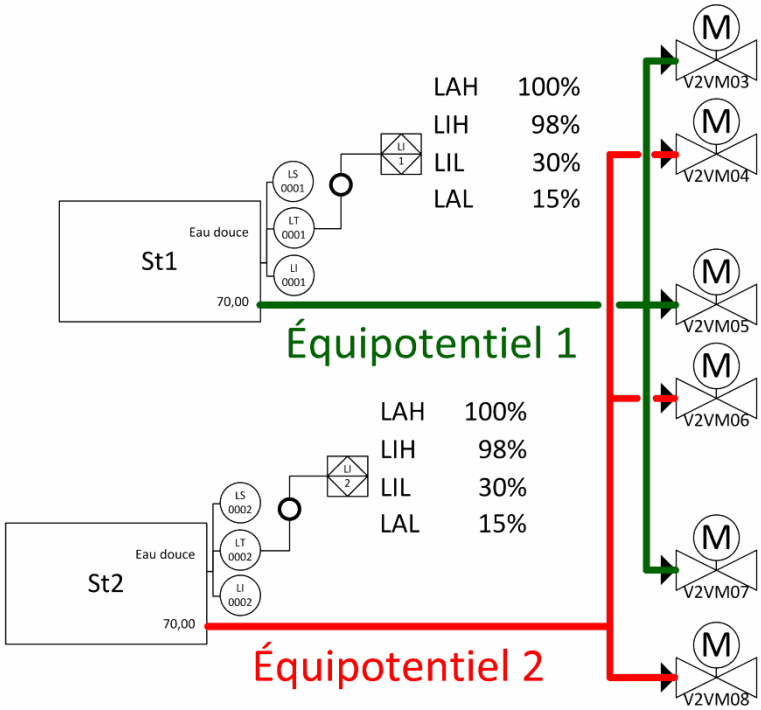


Fig. 86 Illustration des équipotentiels

Les équipotentiels sont regroupés dans une liste pour laquelle nous définissons le métamodèle de la figure 87.

La liste d'équipotentiels (*Equipotentiels* sur la Fig. 87) contient plusieurs équipotentiels (*Equipotentiel* sur la Fig. 87). Chaque équipotentiel contient une ou plusieurs connexions (*Bond* sur la Fig. 87). Les équipotentiels regroupent des zones du système dont tous les points ont le même potentiel. Concrètement, il s'agit de portions de tuyauterie, donc seules les connexions de type procédés peuvent être référencées dans un équipotentiel (l'attribut *type* prend la valeur *Process* sur la Fig. 87).

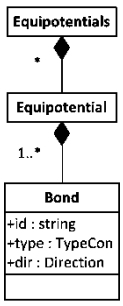


Fig. 87 Métamodèle de la liste d'équipotentiel

Au delà de la génération du graphe de potentiel, les équipotentiels servent également à la prise en compte du système dans son ensemble pour l'animation de l'interface de supervision. La liste des équipotentiels du système est utilisée pour générer des équations d'animation de la

propagation de fluide dans le réseau de tuyauterie, représenté sur l'écran de contrôle.

III.1.2.4 Conclusion sur le MAC

Dans le but de proposer un modèle d'analyse des configurations du système, nous avons défini un modèle de graphe de potentiel basé sur les notions de potentiel et d'équipotentiel.

La validation d'une configuration consiste à vérifier l'existence d'un chemin dans le graphe de potentiel. Ce chemin doit respecter les contraintes imposées par la configuration (une vanne fermée se traduisant par un arc du graphe interdit).

La recherche d'une nouvelle configuration consiste en la recherche d'un chemin alternatif dans le graphe ; cette recherche impliquant des contraintes à minima pour la fonction (seulement le point de départ et le point d'arrivée par exemple).

Nous avons proposé un modèle de vue MAC associé aux éléments de la bibliothèque et un métamodèle du MAC permettant sa génération grâce aux outils de l'IDM.

Ces nouveaux modèles vont également nous permettre de vérifier l'ouverture de notre démarche de conception à l'introduction de compléments que nous présentons ci-après.

III.2 Implémentation du MAC

Nous avons déjà évoqué le fait que le MAC d'un système est construit par assemblage des vues MAC des éléments composant le système.

Nous présentons ici une extension de notre flot de conception permettant cette génération.

Les impacts de cette extension sur les modèles et transformations présentés dans la partie II sont détaillés, ainsi que les nouvelles transformations implémentées pour la génération automatique du modèle d'analyse des configurations.

III.2.1 Extension du flot de conception

Les vues MAC des éléments sont ajoutées à notre bibliothèque et sont référencées comme les autres vues dans la nomenclature. Une opération de modélisation est ajoutée à notre flot de conception (Fig. 88).

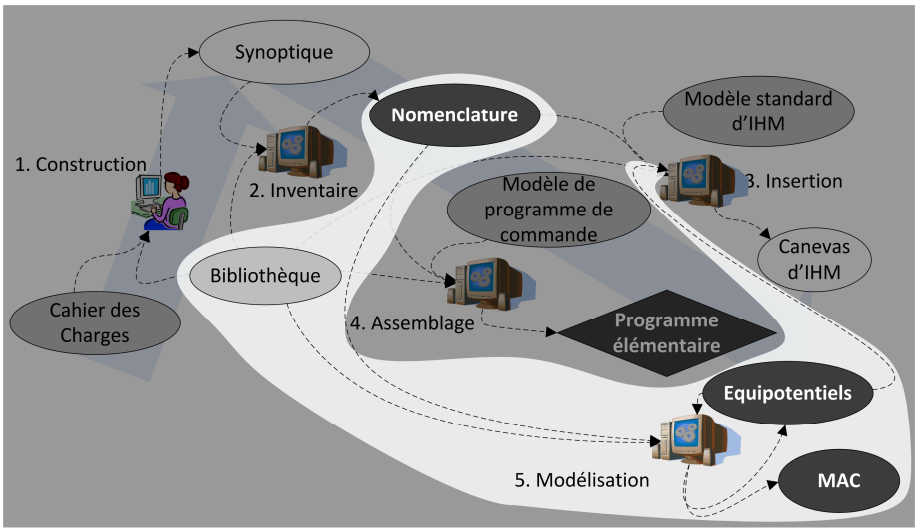


Fig. 88 Extension du flot pour la génération du graphe de potentiel

Cette opération permet de générer une liste d'équipotentiels et un modèle d'analyse des configurations à partir de la bibliothèque et de la nomenclature. La liste d'équipotentiels est injectée dans l'opération d'insertion afin de générer des formules d'animation pour les symboles de tuyauterie.

Maintenant que nous avons présenté l'opération de modélisation dans le contexte du flot de conception, nous pouvons présenter les modifications que nous avons apportées aux métamodèles de bibliothèque et de nomenclature pour prendre en compte cette évolution.

III.2.1.1 Extension des métamodèles

Nous reprenons, pour la création du MAC, la même logique que pour la génération du programme élémentaire ou du canevas d'IHM. Nous ajoutons donc au métamodèle de la bibliothèque (Fig. 89) une vue MAC matérialisée par la classe *VMAC* stéréotypée graph. Cette classe contient les fichiers XML du graphe de potentiel. Nous rajoutons également un attribut *vMAC* dans la classe *Element* qui contient le chemin d'accès à la vue MAC.

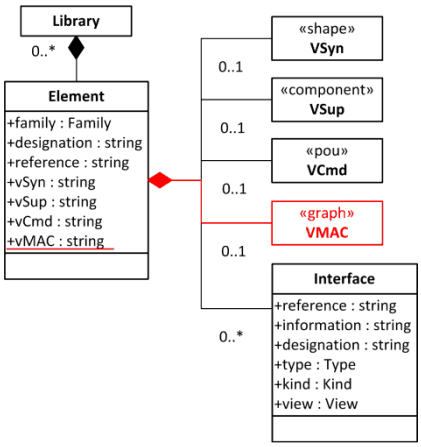


Fig. 89 Extension de la bibliothèque

De la même manière, la classe instance de la nomenclature est modifiée. On y ajoute l'attribut *vMAC* qui contient le chemin d'accès à la vue MAC de l'instance.

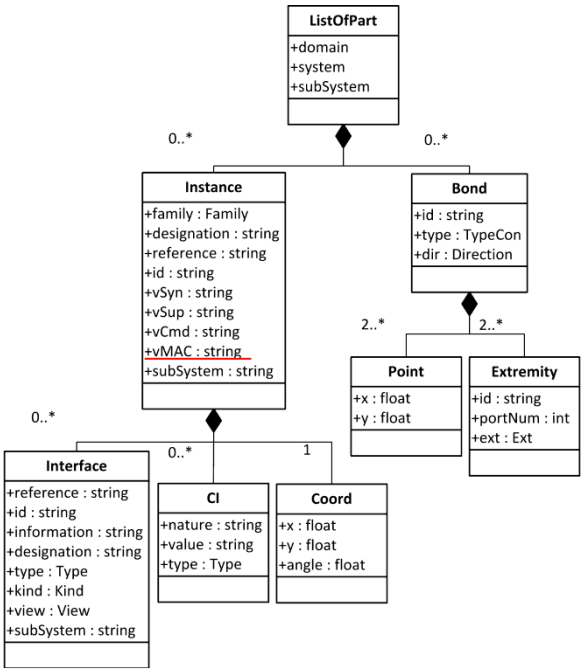


Fig. 90 Extension de la nomenclature

La règle **Symbol2Instance** de la transformation d'inventaire (cf. Fig. 47, p. 65) est modifiée pour prendre en compte l'attribut *vMAC*. Ainsi la

nomenclature générée contient désormais l'ensemble des références des vues graphes, nécessaires à la création du graphe de potentiel complet.

III.2.1.2 Opération de modélisation

L'opération de modélisation du système sous la forme d'un MAC est détaillée sur la figure 91. Elle se déroule en trois étapes exécutées successivement.

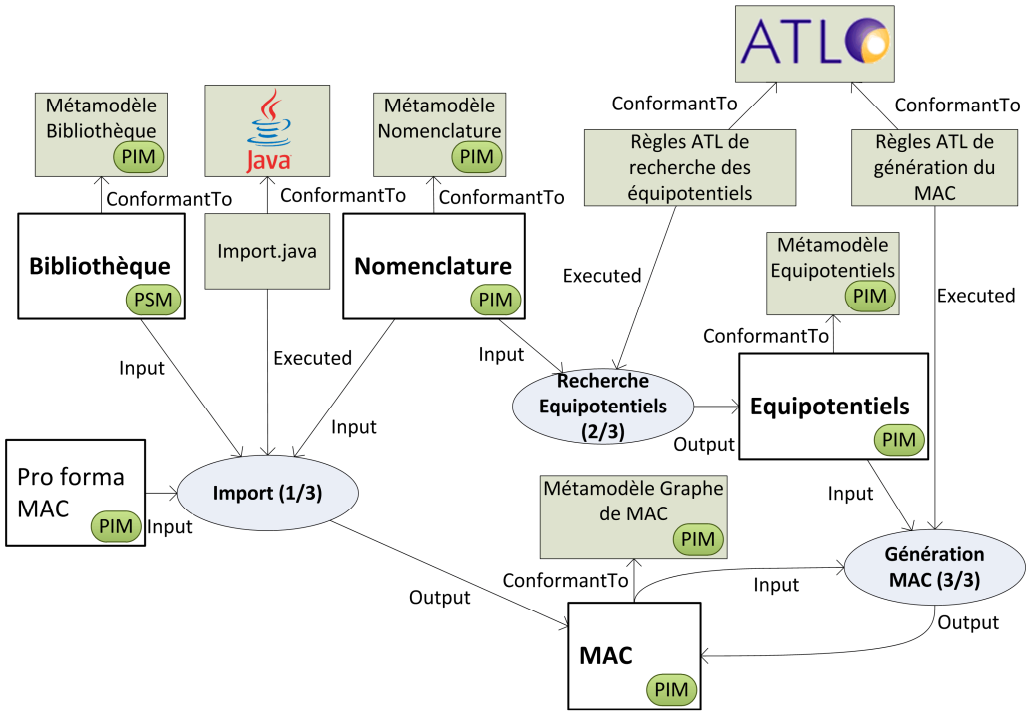


Fig. 91 Détails de l'opération de modélisation

La première étape de notre opération (**Import (1/3)** sur la Fig. 91) vise à insérer dans un pro forma de modèle d'analyse des configurations, l'ensemble des vues MAC de la bibliothèque référencées dans la nomenclature. Cette étape, implémentée au moyen d'un applet Java, peut être assimilée à une transformation exogène verticale. Elle reçoit le modèle de bibliothèque, le modèle de nomenclature et le pro forma de MAC en entrée et elle produit en sortie un premier modèle contenant l'ensemble des vues MAC référencées. Les graphes de potentiel de ces vues ne sont pas encore assemblés et ne forment donc pas le graphe de potentiel du système.

Trois méthodes sont définies pour l'implémentation de cette étape.

1. La méthode **ListeChemin** parcourt la nomenclature pour lister les chemins d'accès aux vues MAC des instances dont elle recense également les références et les identifiants.
2. La méthode **CréerMAC** génère le fichier du modèle d'analyse des configurations à partir du pro forma, de la bibliothèque et de la liste des chemins établie par la méthode **ListeChemin**.
3. La méthode **Enregistre** sauvegarde le MAC généré dans le répertoire du projet.

La deuxième étape de l'opération de modélisation (**Recherche Equipotentiels (2/3)** sur la Fig. 91) crée une liste d'équipotentiels (Fig. 87) à partir des connexions de la nomenclature. Cette étape est implémentée au moyen d'une transformation ATL exogène horizontale.

Deux règles ATL sont définies pour implémenter cette étape :

1. La règle **Bond2Equipotentiel** transforme les connexions de type procédé de la nomenclature en équipotentiels.
2. La règle **Nomenclature2Equipotentiel** regroupe les équipotentiels listés par la règle précédente et qui ont une extrémité commune dans un équipotentiel unique.

La troisième étape de l'opération de modélisation (**Génération MAC (3/3)** sur la Fig. 91) permet de relier les graphes de potentiel des vues MAC, importées dans la première étape aux équipotentiels issus de la deuxième étape. Le MAC résultant de cette étape contient donc le graphe de potentiel complet du système.

Cette étape est implémentée au moyen d'une transformation ATL exogène horizontale. La transformation de génération du MAC comporte une règle principale (**MACGeneration**) qui appelle cinq règles secondaires. On peut distinguer quatre sous-étapes lors de cette génération :

1. Les équipotentiels listés lors de l'étape précédente sont associés aux nœuds d'ancrages du fichier issu de la première étape. L'ensemble des nœuds associés à un équipotentiel sont fusionnés pour former un nœud unique.
2. L'ensemble des nœuds (internes et d'ancrages) sont renumérotés.
3. L'ensemble des arcs est redéfini en fonction de la nouvelle numérotation.
4. L'ensemble des contraintes du MAC est redéfini en fonction de la nouvelle numérotation.

La transformation de génération du MAC termine l'opération de modélisation du système. Cette opération est réalisée suivant la démarche descendante de notre flot de conception, entamée par l'opération d'inventaire (voir Partie II). Par ailleurs, cette opération offre un modèle exploitable pour la branche fonctionnelle du processus de conception.

Après avoir présenté cette nouvelle opération, nous pouvons exposer les modifications que nous avons apportées à l'opération d'insertion pour prendre en compte l'architecture du système dans l'animation des symboles de tuyauterie.

III.2.1.3 Modification de l'opération d'insertion

Nous rappelons que l'opération d'insertion (voir chapitre II.3 p. 67) est l'opération de notre flot de conception qui permet la génération automatique d'un canevas d'IHM. La prise en compte de la liste des équipotentiels dans cette opération permet de générer automatiquement des animations de propagation du fluide pour la tuyauterie représentée sur l'IHM de l'opérateur.

Il est important de noter que cette animation dépasse le cadre d'un simple assemblage de composants. En effet l'animation de propagation doit tenir compte de l'état de plusieurs éléments du système.

Cette animation a son importance dans notre contexte. En effet, dans un procédé continu, le fluide parcourt les tuyauteries en permanence et ce sont des valeurs remontées par l'instrumentation qui permettent à l'opérateur de maintenir la continuité du procédé.

Dans le cas d'un système embarqué sur un navire, l'action de l'opérateur porte davantage sur une gestion du stock et sur une surveillance des fonctions automatisées. Ici le procédé est discontinu. L'animation des tuyauteries permet à l'opérateur d'évaluer la configuration du système par le suivi de la ligne de tuyauterie animée.

La prise en compte de la liste des équipotentiels a un impact limité sur l'opération d'insertion (Fig. 92). Si nous reprenons la figure présentant la structure de cette opération, nous voyons qu'elle se déroule en trois phases. Seule la phase de création du fichier « Unit.cfg » est concernée, et dans cette phase, seule la transformation **Synthèse Unit (3/8)** doit être modifiée.

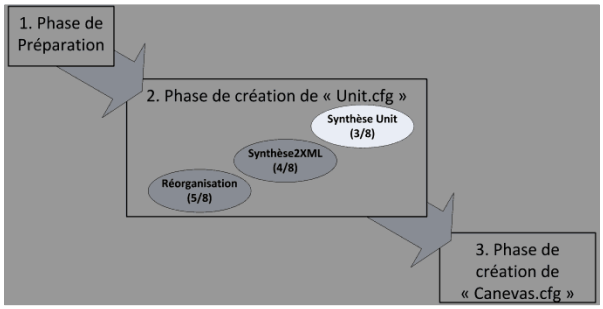


Fig. 92 Impact sur l'opération d'insertion

La transformation **Synthèse Unit (3/8)** est une transformation ATL exogène verticale (Fig. 56, p. 76). Elle comporte une règle principale qui appelle plusieurs règles secondaires en suivant trois grandes étapes (instanciation, construction et complément).

Nous ajoutons la règle **Formula** à cette transformation. C'est une règle secondaire appelée pendant la sous-étape d'instanciation.

Les formules sous Panorama E2 sont des objets au même titre que les symboles de supervision ; en conséquence le métamodèle du fichier Unit n'a pas besoin d'être modifié.

Les formules sont des équations booléennes, fonctions des instances raccordées à un équipotentiel. Elles renvoient un résultat utilisé pour l'animation des symboles de tuyauterie associés à un équipotentiel. Il y a une équation générée pour chaque équipotentiel de la liste.

Après avoir présenté l'impact de la prise en compte des modèles de graphe de potentiel et d'équipotentiel sur notre démarche, nous exposons l'application de la démarche sur un cas concret.

III.2.2 Outillage, Illustration et validation

La validation de nos concepts passe une fois de plus par leur implémentation. Nous présentons ici les modifications apportées à l'outil Anaxagore ainsi que l'application de notre extension sur un cas d'illustration.

III.2.2.1 Extension d'Anaxagore

L'opération de modélisation est implémentée sous la forme d'un plugin à la plateforme Eclipse. Ce plugin est ajouté à l'outil Anaxagore développé pour expérimenter notre démarche.

Le plugin de modélisation constitue un septième ensemble à l'outil après les ensembles relatifs à la bibliothèque, au projet, à l'opération de construction, à l'opération d'inventaire, à l'opération d'insertion et à l'opération d'assemblage.

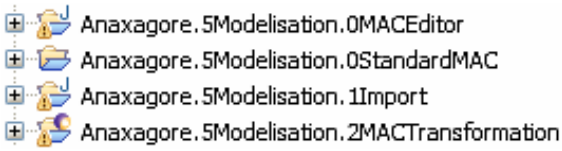


Fig. 93 Sous ensemble d'Anaxagore relatif à l'opération de modélisation

Le plugin de **modélisation** se compose d'un éditeur de MAC (*Anaxagore.5modelisation.0MACEditor* sur la Fig. 93), d'un répertoire contenant le pro forma de modèle d'analyse des configurations (*Anaxagore.5Modelisation.0StandardMAC* sur la Fig. 93), de l'applet Java d'import des vues MAC (*Anaxagore.5Modelisation.1Import* sur la Fig. 93), ainsi que d'un projet ATL regroupant les transformations de recherche d'équipotentiel et de génération du modèle d'analyse des configurations (*Anaxagore.5Modelisation.2MACTransformation* sur la Fig. 93).

III.2.2.2 Saisie et visualisation des graphes

Afin de compléter la bibliothèque, nous avons développé un outil de saisie et de visualisation des graphes de potentiel. Cet outil développé sous GMF, permet de créer une vue graphe par glissé-déposé des éléments de la palette sur la zone de dessin. Lors de la saisie d'une vue graphe, les sommets sont numérotés à la main. Les ports associés aux nœuds d'ancrages et les ports « fantômes » associés aux nœuds internes sont également renseignés manuellement.

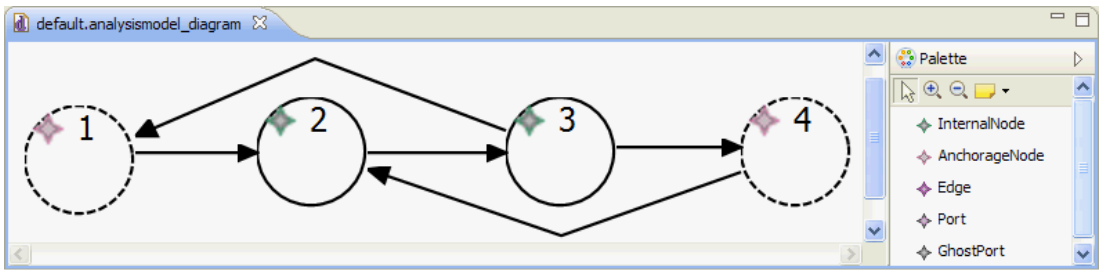


Fig. 94 Plug in de saisie et de visualisation des graphes de potentiels

Les éventuelles contraintes associées à un matériel sont ajoutées à la main sous forme de balise XML dans le fichier issu de l'outil de saisie. Le fichier résultant constitue la vue MAC d'un élément de la bibliothèque (Fig. 95).

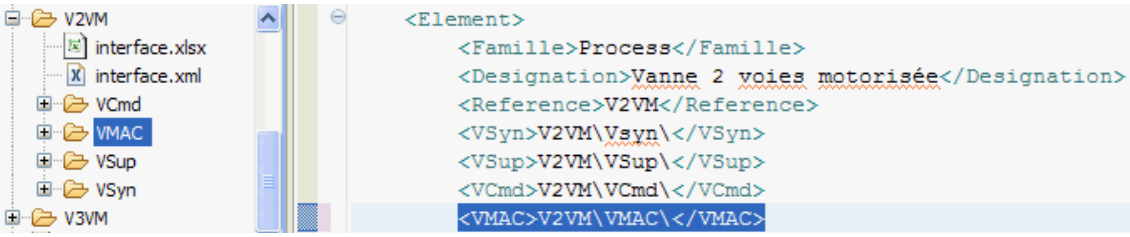


Fig. 95 Modification de la structure de la bibliothèque

La vue MAC est intégrée à la bibliothèque.

III.2.2.3 Cas d'illustration

Nous nous basons sur le synoptique présenté au paragraphe III.1.1.1 (Fig. 76, p. 98). Nous rappelons qu'il s'agit d'un extrait du synoptique de l'installation de production, de stockage et de distribution de l'eau douce sanitaire (EdS) présentée en annexe. Nous rappelons que ce type de synoptique est produit par l'expert en charge de la conception lors de l'opération de construction (cf. chapitre II.1 p. 45).

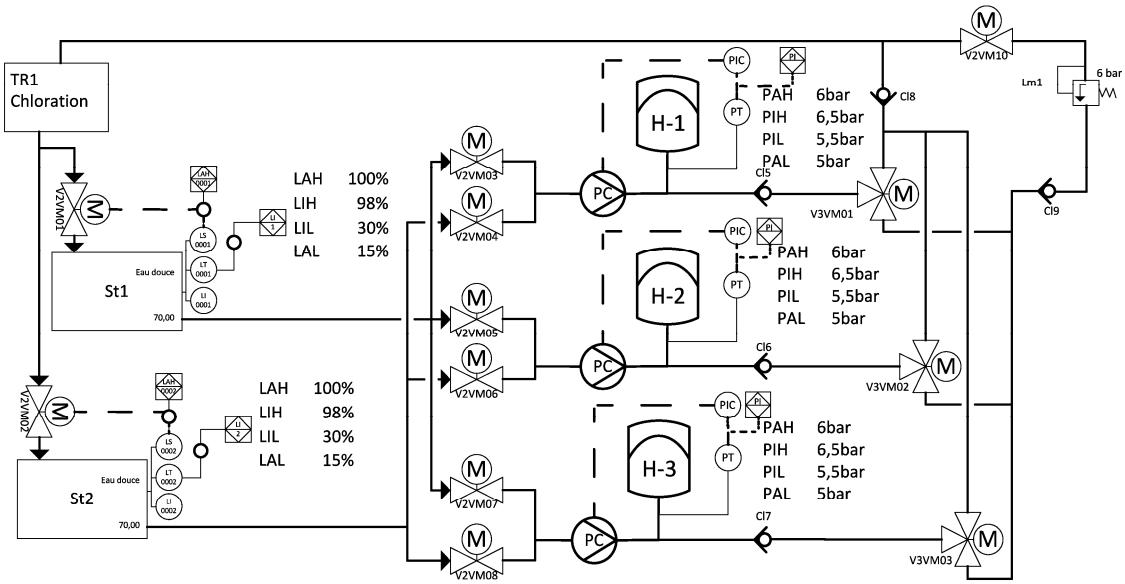


Fig. 96 Cas d'illustration

Ce synoptique représente deux soutes de stockage (St1 et St2 sur la Fig. 96). Chacune de ces soutes est équipée d'une jauge de niveau, d'un transmetteur de niveau et d'un détecteur de niveau. Ces deux derniers instruments renvoient leurs informations à la supervision.

En amont de chaque soute, il y a une vanne permettant de l'isoler de la ligne de remplissage (V2VM01 et V2VM02 sur la Fig. 96) sur laquelle est monté le dispositif de traitement de l'eau par chloration (Tr1 sur la Fig. 96).

En aval des soutes, un ensemble de 6 vannes (V2VM03, V2VM04, V2VM05, V2VM06, V2VM07 et V2VM08 sur la Fig. 96) permet l'aiguillage du fluide pompé par les trois groupes hydrophores (H1, H2 et H3 sur la Fig. 96).

Chaque groupe hydrophore est équipé en sortie d'un clapet anti-retour empêchant l'eau de revenir par ce tuyau (Cl5, Cl6 et Cl7 sur la Fig. 96). Un ensemble de 3 vannes à trois voies (V3VM01, V3VM02 et V3VM03 sur la Fig. 96) permet de choisir, pour chaque hydrophore, si l'eau pompée est envoyée à la distribution (vers Cl9 sur la Fig. 96) ou rebouclée sur la ligne de remplissage des soutes (vers Cl8 sur la Fig. 96).

La vanne V2VM10 est utilisée lors de l'alimentation directe du réseau de distribution par le quai. Dans toute autre configuration cette vanne est fermée.

III.2.2.4 Opération d'inventaire

Nous présentons ici un extrait du résultat de l'opération d'inventaire réalisé sur le synoptique de notre cas d'application (Fig. 96). Notre extrait porte sur l'instance de la vanne à trois voies, motorisée V3VM01 (Fig. 97). Outre les vues synoptique (*vSyn*), de supervision (*vSup*) et de commande (*vCmd*), nous retrouvons dans la balise instance, l'attribut *vMAC* concernant la vue MAC de la vanne. Cet attribut prend la valeur V3VM\VMAC\.


```
<instance family="Process" designation="Vanne 3 voies motorisée" reference="V3VM" id="V3VM01" vSyn="V3VM\Vsyn\" vSup="V3VM\VSup\" vCmd="V3VM\VCmd\" vMAC="V3VM\VMAC\" subSystem="EdS">
  <interface reference="V3VM" id="V3VM01" information="Col1" designation="couleur du fluide 1" type="INT" kind="Var" view="VSup" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="Col2" designation="couleur du fluide 2" type="INT" kind="Var" view="VSup" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="CtrlC" designation="Commande de fermeture issue de la supervision" type="BOOL" kind="Output" view="VSup" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="CtrlO1" designation=" Commande d'ouverture voie 1 issue de la supervision" type="BOOL" kind="Input" view="VSup" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="CtrlO2" designation=" Commande d'ouverture voie 2 issue de la supervision" type="BOOL" kind="Input" view="VSup" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="Over" designation="Animation survol" type="BOOL" kind="Input" view="VSup" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="PropIn" designation="Animation de la propagation entrée" type="INT" kind="Var" view="VSup" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="PropOut1" designation="Animation de la propagation voie 1" type="INT" kind="Var" view="VSup" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="PropOut2" designation="Animation de la propagation voie 2" type="INT" kind="Var" view="VSup" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="Repere" designation="Repère" type="STRING" kind="Const" view="VSup" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="Select" designation="Animation selection" type="BOOL" kind="Var" view="VSup" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="StFCmd" designation="Défaut commande" type="BOOL" kind="Output" view="VSup" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="StFMat" designation="Défaut matériel" type="BOOL" kind="Output" view="VSup" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="CtrlC" designation="Commande de fermeture" type="BOOL" kind="Input" view="VCmd" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="CtrlO1" designation="Commande d'ouverture voie 1" type="BOOL" kind="Input" view="VCmd" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="CtrlO2" designation="Commande d'ouverture voie 2" type="BOOL" kind="Input" view="VCmd" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="Tempo" designation="Temporisation de manœuvre de la vanne" type="TIME" kind="Input" view="VCmd" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="FdcO1" designation="Fin de course d'ouverture voie 1" type="BOOL" kind="InPhy" view="VCmd" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="FdcO2" designation="Fin de course d'ouverture voie 2" type="BOOL" kind="InPhy" view="VCmd" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="FdcC" designation="Fin de course de fermeture" type="BOOL" kind="InPhy" view="VCmd" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="StO1" designation="Etat "vane en position 1"" type="BOOL" kind="Output" view="VCmd" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="StO2" designation="Etat "Vanne en position 2"" type="BOOL" kind="Output" view="VCmd" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="StC" designation="Etat "vanne fermée"" type="BOOL" kind="Output" view="VCmd" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="StFCmd" designation="Défaut commande" type="BOOL" kind="Output" view="VCmd" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="CmdO1" designation="Commande d'ouverture voie 1 vers la vanne" type="BOOL" kind="OutPhy" view="VCmd" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="CmdO2" designation="Commande d'ouverture voie 2 vers la vanne" type="BOOL" kind="OutPhy" view="VCmd" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="CmdC" designation="Commande de fermeture vers la vanne" type="BOOL" kind="OutPhy" view="VCmd" subSystem="EdS" />
  <interface reference="V3VM" id="V3VM01" information="StFMat" designation="Défaut materiel" type="BOOL" kind="Output" view="VCmd" subSystem="EdS" />
  <coord x="249.68333" y="72.075" />
</instance>
```


Fig. 97 Extrait du fichier ListOfParts.xml

Nous retrouvons également dans cet extrait les différentes interfaces de la vanne à trois voies ainsi que ses coordonnées sur le synoptique (balise *coord* sur la Fig. 97).

L'attribut *vMAC* est utilisé lors de l'opération de modélisation dont nous présentons les résultats ci-après.

III.2.2.5 Opération de modélisation

L'opération de modélisation vise à la création d'une liste d'équipotentiels et d'un modèle d'analyse des configurations à partir du modèle de bibliothèque et de la nomenclature. Nous présentons ces deux résultats successivement.

III.2.2.5.1. Liste des équipotentiels

La liste des équipotentiels générée est conforme à notre métamodèle (Fig. 87, p. 109). Pour notre cas d'illustration, la liste comporte 19 équipotentiels. La figure 98 présente un extrait de la liste générée.

Nous détaillons l'équipotentiel situé entre la soute St1 et les vannes V2VM03, V2VM05 et V2VM07 (équipotentiel 1 sur la Fig. 86).

Nous retrouvons dans cet équipotentiel la connexion (*bond* sur la Fig. 98) P-6.108 qui se trouve entre la soute St1 (extrémité 1) et les connexions P-6.114, P-42.115 et P-41.112 (extrémité 2).

La connexion P-6.114 est reliée aux connexions P-6.108, P-41.112 et P-42.115 par son extrémité 1 et à la vanne V2VM05 par son extrémité 2.

La connexion P-41.112 est reliée aux connexions P-6.108, P-6.114 et P-42.115 par son extrémité 1 et à la vanne V2VM03 par son extrémité 2.

Enfin, la connexion P-42.115 est reliée aux connexions P-6.108, P-6.114 et P-42.115 par son extrémité 1 et à la vanne V2VM07.

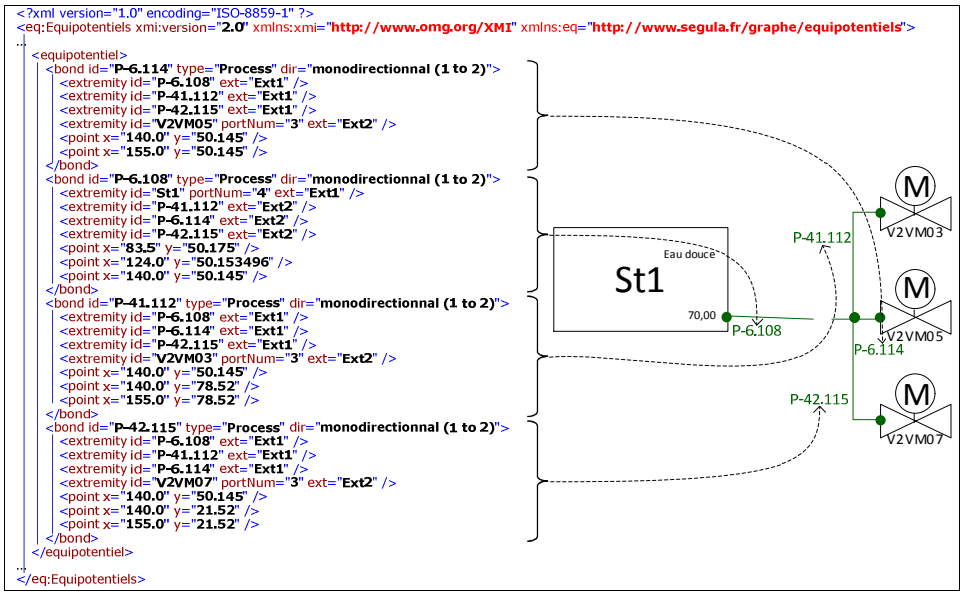


Fig. 98 Liste des équipotentiels

Outre l'aide à la construction du graphe de potentiel du MAC du système, la liste des équipotentiels permet de définir des équations booléennes pour l'animation des tuyaux. L'équation pour l'équipotentiel de la figure 98 est fonction du niveau d'eau dans la soute (s'il est supérieur à zéro) de l'état des vannes (si elles sont ouvertes) et de la présence d'eau dans les vannes. Chaque vue de supervision des éléments de la bibliothèque est pourvue d'une ou de plusieurs variables d'animation de présence de fluide.

Le deuxième résultat de l'opération de modélisation est la création d'un modèle d'analyse complet du système à partir des vues MAC de la bibliothèque et de la liste des équipotentiels.

III.2.2.5.2. Le modèle d'analyse des configurations

La figure 99 représente le graphe de potentiel issu de l'opération de modélisation, appliquée à notre cas d'illustration (Fig. 96). Nous avons placé les repères des instances pour visualiser plus facilement les motifs qui leur sont associés. Le graphe complet comprend toutes les liaisons possibles physiquement entre les places. Les nœuds d'ancrages correspondant aux extrémités des instances et appartenant à un même équipotentiel, sont fusionnés.

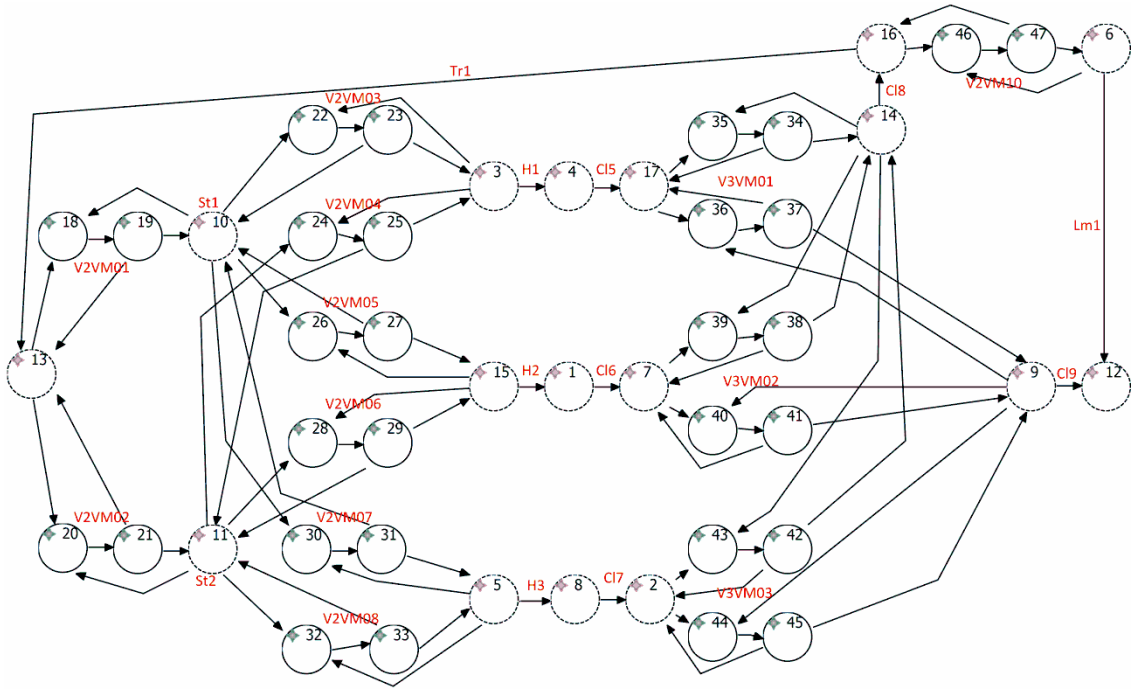


Fig. 99 Graphe de potentiel

La figure 100 illustre la fusion des nœuds d'ancrages pour l'équipotentiel de la figure 98. Les nœuds 1, 2, 6 et 10 des vues graphes présentées à gauche sont associés respectivement au port 4 de la soute St1, au port 3 de V2VM03, au port 3 de V2VM05 et au port 3 de V2VM07 qui appartiennent au même équipotentiel. Les nœuds sont donc fusionnés en un seul (nœud 1 du graphe de droite). L'ensemble des nœuds restant est ensuite renuméroté.

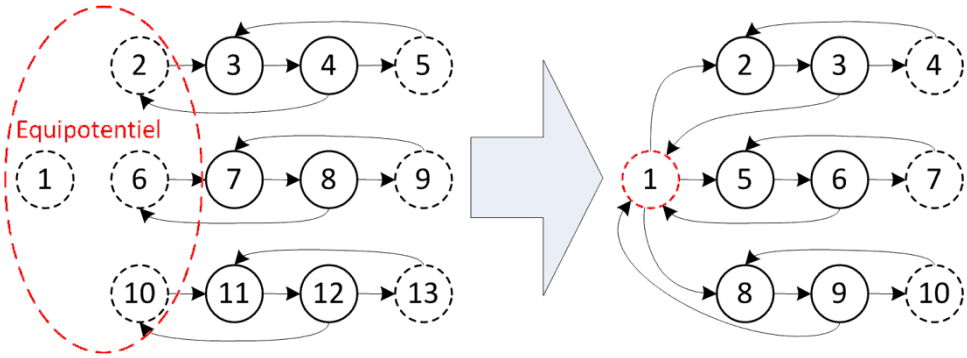


Fig. 100 Illustration de la fusion des nœuds d'ancrages

Les contraintes des vues MAC sont tout simplement copiées dans le modèle généré. Les numéros des nœuds, sur lesquels portent les contraintes, sont modifiés en fonction de la nouvelle numérotation.

```
<Constraints>
  <Constraint symbol="=Inf" result="1">
    <Term Coef="1">
      <Vertex xsi:type="con:InternalNode" num="35">
        < GhostPort id="V3VM01" number="1" />
      </Vertex>
      <Vertex xsi:type="con: InternalNode " num="34">
        < GhostPort id="V3VM01" number="2" />
      </Vertex>
    </Term>
    <Term Coef="1">
      <Vertex xsi:type="con: InternalNode " num="36">
        < GhostPort id="V3VM01" number="5" />
      </Vertex>
      <Vertex xsi:type="con: InternalNode " num="37">
        <GhostPort id="V3VM01" number="3" />
      </Vertex>
    </Term>
  </Constraint>
...
</Constraints>
```

≤ 1

$1 \times x_{35,36}$

$+$

$1 \times x_{36,37}$

Fig. 101 Contrainte de la vanne V3VM01

La figure 101 présente un extrait du fichier XML de notre modèle d'analyse, contenant la contrainte portant sur la vanne V3VM01. On retrouve l'équation suivante :

$$x_{35,36} + x_{36,37} \leq 1$$

Rappelons que cette équation contraint le flot traversant la vanne à trois voies à ne circuler que par l'une ou par l'autre des sorties de la vanne.

Nous présentons maintenant le résultat de l'opération d'insertion prenant en compte les équipotentiels.

III.2.2.6 Opération d'insertion

La figure 102 présente le résultat de l'opération d'insertion pour notre cas d'application (Fig. 96). Le canevas d'IHM généré reprend l'agencement du synoptique. On retrouve ainsi les deux soutes de stockage, les trois groupes hydrophores, le dispositif de chloration et le réseau de tuyauterie avec ses vannes.

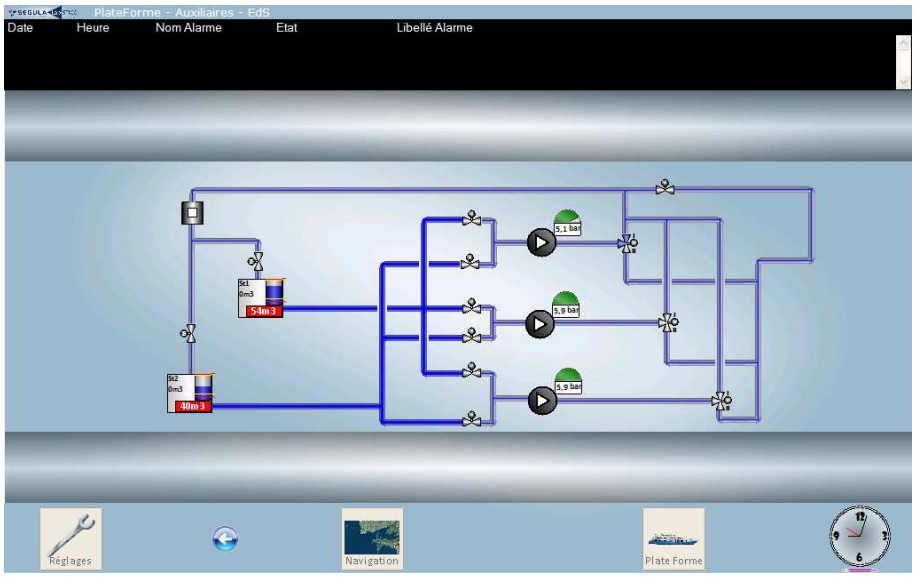


Fig. 102 Canevas d'IHM

Chaque symbole de tuyauterie du canevas est associé à une formule établie à partir de la liste des équipotentiels. La figure 103 présente la formule correspondant à l'équipotentiel 14 de notre liste. Cet équipotentiel se trouve entre la vanne V2VM03, la vanne V2VM04 et le groupe hydrophore H1 (en rouge sur la Fig. 103). La formule se base sur l'état des vannes pour animer les symboles de tuyauterie associés (en vert sur la Fig. 103). En effet le fluide ne peut pas remonter du groupe hydrophore ; il peut uniquement venir des vannes.

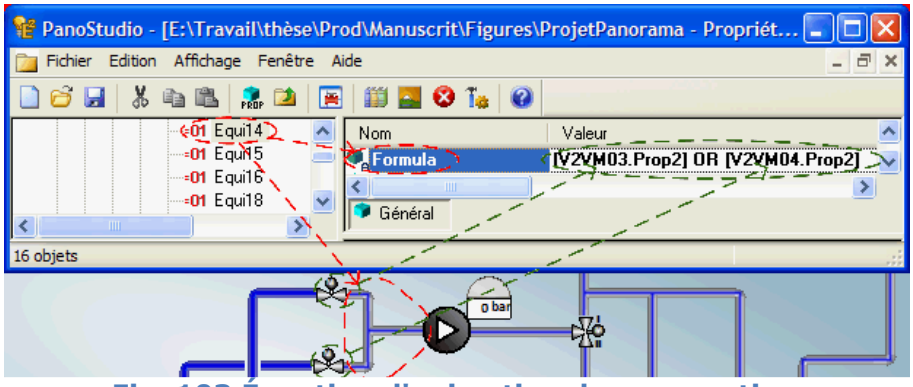


Fig. 103 Équation d'animation de propagation

La formule prend la forme suivante :

$$[V2VM03.Prop2]\ OR\ [V2VM04.Prop2]$$

Les variables *Prop2* des vannes sont « vraies » si la vanne est ouverte et qu'il y a du fluide en amont de la vanne. Dans notre cas, si l'une des deux vannes est ouverte, la formule devient « vraie » et les symboles de tuyauterie associés à l'équipotentiel s'animent en propagation.

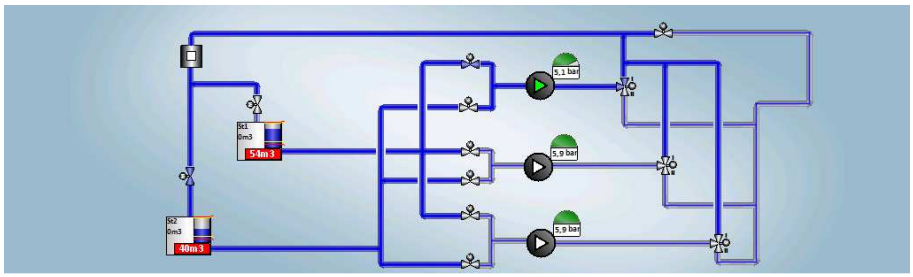


Fig. 104 Illustration de l'animation de propagation

La figure 104 illustre le principe de l'animation de la propagation. Le système est configuré pour réaliser un transfert de la soute St1 à la soute St2. Le groupe hydrophore H1 est en marche.

III.2.3 Bilan de l'implémentation du MAC

L'implémentation du modèle d'analyse des configurations concrétise une partie des concepts portant sur la prise en compte du système dans son ensemble. Le MAC ouvre notamment la voie à la vérification automatique des

configurations spécifiées par l'expert ainsi qu'à la recherche automatique de configurations alternatives.

Pensée comme une extension de notre flot de conception, cette concrétisation a démontré l'ouverture de notre démarche.

Par ailleurs, comme la prise en compte du système dans son ensemble ne se résume pas qu'aux commandes globales, nous avons modifié une transformation de l'opération de génération du canevas d'IHM afin de prendre en compte le concept d'équipotentiel. Ce dernier permet, au travers de la liste établie, de générer des équations d'animation de la propagation de fluide dans les symboles de tuyauterie. Les symboles du canevas d'IHM ne sont donc plus indépendants les uns des autres. Cet affichage permet à l'opérateur de visualiser rapidement la configuration du système.

Le développement d'Anaxagore en est rendu à ce point. Il nous semble néanmoins important de présenter à la suite la manière dont nous envisageons de concrétiser la génération de la commande globale et la prise en compte du caractère reconfigurable du système.

III.3 Proposition pour la commande globale

Ce chapitre présente notre proposition d'extension au flot de conception pour la génération des commandes globales et de leurs interfaces de supervision, ainsi que pour l'exploitation du MAC. Cela constitue nos perspectives à court terme pour le développement de l'outil Anaxagore.

Dans la partie II de ce manuscrit, nous avons détaillé un premier flot de conception basé sur l'utilisation d'un modèle métier pour la génération automatique de programme de commande et d'interface de supervision. Nous avons déjà évoqué les limites de ce flot qui portent notamment sur la prise en compte du système dans son ensemble.

Dans cette partie, nous présentons une proposition qui s'inscrit dans la continuité de notre flot de conception. L'idée est toujours de mieux séparer les activités de conception et de programmation, tout en automatisant davantage le passage de l'une à l'autre. Par ailleurs l'extension du flot que nous présentons a pour vocation d'être déroulée à la suite des opérations précédemment définies.

III.3.1 Les modèles

L'extension que nous proposons (Fig. 105) reprend plusieurs modèles présentés précédemment, notamment le cahier des charges, le synoptique, le canevas d'IHM, le programme élémentaire et le modèle d'analyse des configurations.

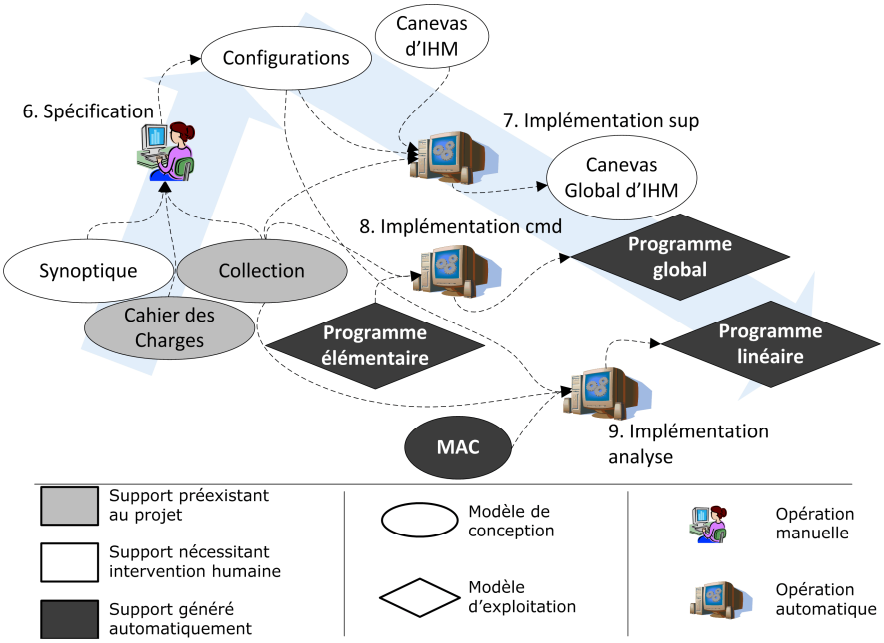


Fig. 105 Extension du flot de conception

En s'appuyant sur le processus de conception simplifié issu de notre retour d'expérience (Fig. 7, p. 9), et après en avoir déroulé l'axe de conception centré sur l'architecture du système, nous proposons maintenant d'en dérouler l'axe fonctionnel.

L'axe architectural se matérialise dans la construction du synoptique par l'expert en charge de la conception du système. L'axe fonctionnel se matérialise, quant à lui, dans l'établissement d'une liste de **configurations** nominales du système.

L'ensemble des patrons sont regroupés dans une **collection** (Fig. 106). Elle est organisée en « fonctions ». Chaque fonction regroupe plusieurs patrons conformément au concept de vue. La fonction transfert regroupe ainsi le patron de séquence de transfert et le patron de supervision de transfert. On peut y ajouter d'autres vues, comme par exemple une vue « objectif » qui matérialiserait un objectif à atteindre pour la fonction. Ce pourrait être « maximiser le flot » lors de la fonction de transfert.

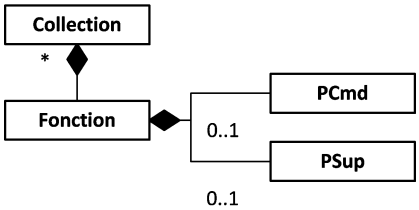


Fig. 106 Organisation de la collection

Le **canevas d'IHM global** contient le canevas d'IHM déjà généré mais on y trouve en plus les dispositifs d'interfaces permettant la saisie des consignes des commandes globales (issues des patrons de supervision). Il permet donc toujours une surveillance des éléments du système identifiés sur le synoptique, mais il permet en plus de les contrôler de manière globale en déclenchant des séquences regroupant un ensemble de commandes élémentaires.

De la même manière, le **programme global** contient le programme élémentaire. Il contient également le code des commandes globales (issu des patrons de séquence) qu'il met à disposition de la supervision au travers de variables de commande.

Enfin, nous introduisons le modèle de **programme linéaire**. Ce dernier a pour objectif, associé à un solveur, de rechercher une solution de configuration pour une ou plusieurs fonctions. Le solveur est basé sur un algorithme de recherche de chemin dans un graphe et résout le programme linéaire exprimé en fonction de l'état du système et des consignes de commande exprimées par l'opérateur. Ce modèle est directement exploitable. Il s'agit d'un programme pouvant être lancé automatiquement par la supervision, en cas de conflits entre plusieurs commandes globales ou en cas de problème lors du déroulement d'une commande.

III.3.2 Les opérations

Notre extension comprend quatre opérations déroulées à la suite de celles précédemment définies.

III.3.2.1 Opération de spécification

La première étape de notre extension (**6. Spécification** sur la Fig. 105) est une étape de description fonctionnelle du système par l'expert en charge de sa conception. A partir du synoptique qu'il a lui-même conçu, du cahier des charges et de la collection de fonctions, il doit être capable d'établir une configuration nominale pour chaque fonction de son système.

Cette liste de configurations prend la forme d'un ensemble de tableaux contenant les informations, conformément à notre métamodèle (Fig. 77, p. 99) et telle que nous l'avons illustrée (Fig. 78, p. 100).

Elle servira par la suite pour la génération de commandes globales, pour le paramétrage des patrons de supervision et enfin pour la définition de contraintes pour le solveur.

L'opération de spécification est à l'axe fonctionnel ce que l'opération de construction est à l'axe architectural. C'est la définition que l'expert donne de son système.

III.3.2.2 Implémentation de la supervision

La deuxième opération de notre extension (**7. Implémentation sup** sur la Fig. 105) réalise le paramétrage des patrons d'interface de la collection en fonction des configurations établies par l'expert. Les patrons paramétrés sont ensuite insérés dans le canevas d'IHM issu de la première partie du flot.

En quoi les patrons d'interface ont-ils besoin d'être paramétrés ? Reprenons l'exemple de la fonction de transfert que nous avons déjà présentée. Notre patron d'interface propose de choisir la soute de départ et la soute d'arrivée (voir Fig. 80, p. 103). Un paramétrage du patron de supervision consiste à établir les listes de choix possibles. Il est possible d'établir ces listes en recensant les configurations nominales de transfert (de St1 à St2 et de St2 à St1).

Rappelons enfin que pour l'insertion des patrons dans le canevas, une zone de l'écran est réservée aux boutons d'accès aux dispositifs d'interface des commandes globales. Cette zone se situe dans la partie haute de l'écran, entre le bandeau d'alarme et la zone d'affichage principale (voir Fig. 49, p. 70).

III.3.2.3 Implémentation de la commande

La troisième étape de notre extension (**8. Implémentation cmd** sur la Fig. 105) vise à l'intégration des patrons de séquence dans le programme élémentaire. Là encore, un paramétrage réalisé en fonction des configurations établies par l'expert, permettra de rendre cette intégration fonctionnelle.

Les conditions des transitions ainsi que les actions des étapes des séquences de commande sont paramétrées en fonction des configurations (voir Fig. 79, p. 102).

Ces patrons paramétrés sont ajoutés au programme élémentaire sous la forme de blocs fonctionnels supplémentaires (codés en SFC) ; l'ensemble forme le programme global.

III.3.2.4 Implémentation du modèle d'analyse

La dernière opération de notre extension (**9 Implémentation analyse** sur la Fig. 105) vise à transformer notre modèle d'analyse, issu de l'opération de modélisation (Fig. 88, p. 111), en solveur exécutable susceptible de proposer des solutions de reconfiguration en cas de consigne concurrente de la part de l'opérateur ou en cas d'avarie d'un élément du système.

Dans le MAC, une configuration du système est un chemin sur le graphe. Ce chemin doit satisfaire les objectifs de la fonction associée à la configuration.

Une reconfiguration est la possibilité de trouver un autre chemin entre ces deux places. Nous devons pouvoir poser des contraintes sur les chemins, comme la nécessité de passer par certains arcs. Par exemple, la configuration que nous avons présentée sur la figure 78 (p. 100) nécessite de faire passer le flot par le groupe hydrophore H1 modélisé sur le graphe par les sommets 3 et 4 et par l'arc entre les deux. La contrainte imposant au flot de traverser H1 s'exprime sous la forme de l'équation suivante :

$$x_{3,4} = 1$$

Afin de ne pas limiter la recherche de chemin au seul passage possible par H1, nous pouvons exprimer une contrainte qui impose le passage par l'un des groupes hydrophore (H1, H2, ou H3). Une telle contrainte s'exprime sous la forme de l'équation suivante :

$$x_{3,4} + x_{15,1} + x_{5,8} \geq 1$$

Associé à cette contrainte, nous devons également spécifier la capacité maximale des arcs. Elle correspond, en première approximation, à la somme des capacités des arcs associés aux pompes. Dans notre cas, nous avons trois pompes (les groupes hydrophores). Si nous considérons que chaque groupe a une capacité de 1, la capacité maximale des arcs sera de 3.

Les contraintes appliquées au solveur sont issues pour une part de la liste des configurations (point de départ et d'arrivée, balises) et pour une autre part, de l'état des éléments du système (défaillance d'une vanne, élément utilisé par une fonction plus prioritaire).

Les objectifs du solveur sont issus de la collection. Chaque fonction ayant un objectif prédéfini, par exemple, maximiser le débit pour la fonction de transfert.

Nous avons testé un solveur implémenté en C avec GLPK. Le solveur prend en entrée un fichier texte qui décrit l'objectif, le graphe de potentiel et les contraintes. Ce fichier est généré automatiquement par une transformation ATL à partir du MAC ; des contraintes supplémentaires

peuvent être rajoutées à la main. Le solveur renvoie une solution satisfaisant l'objectif.

Appliqué à notre cas d'illustration et sans contrainte autre que celles associées au matériel, le solveur nous renvoie une configuration composée de trois chemins, chacun empruntant un arc associé à un groupe hydrophore.

Si nous ajoutons une contrainte imposant la fermeture de la vanne en amont d'un des groupes hydrophores, le solveur nous renvoie une configuration composée de deux chemins, chacun empruntant un chemin associé à un groupe hydrophore accessible.

Enfin, si nous ajoutons une deuxième contrainte imposant la fermeture d'une vanne en amont d'un deuxième groupe hydrophore, le solveur nous renvoie une configuration composée d'un unique chemin empruntant le seul groupe hydrophore accessible.

Pour compléter la démarche il nous faut trouver un moyen d'utiliser le solveur lors de l'exploitation du système. Ce qui implique des interactions avec les logiciels Panorama E2 et Straton.

III.4 Bilan sur l'extension de la proposition

L'extension à la démarche que nous avons proposée dans cette partie permet de compléter la conception du système par la prise en compte de l'aspect **fonctionnel**.

Comme pour l'aspect architectural, nous nous sommes attachés à la définition que l'expert fait de son système. Contrairement au synoptique, la liste des configurations n'est pas un modèle normalisé. Néanmoins, nous avons croisé beaucoup de tableaux de configurations lors de nos entretiens. Associé au synoptique, un tableau de configuration est un moyen privilégié pour expliquer le fonctionnement d'un système.

La collection et ses patrons de séquence et de supervision poursuivent la réponse au besoin de **standardisation** exprimé par les industriels.

L'association des patrons de séquence avec des patrons de supervision, au sein de la même structure, permet d'affermir la **cohérence** entre le programme de commande et l'IHM de l'opérateur.

L'utilisation des techniques de l'IDM pour générer les différents modèles permet alors aux experts de se concentrer sur leurs cœurs de métier.

Par ailleurs, avec notre modèle d'analyse des configurations, nous ouvrons la voie à l'introduction d'outils et d'étapes de **vérification** dans notre flot. De plus, ce même modèle utilisé en exploitation permettrait de prendre en compte l'aspect **reconfigurable** du système.

A travers la prise en compte des équipotentiels, nous ouvrons également la voie à la génération d'IHM de haut niveau avec des fonctionnalités d'affichage avancées (animation de la propagation).

Les concepts de patron de supervision et de patron de séquence ne sont pas encore implémentés ; il s'agit néanmoins de perspectives à très court terme. Quant au solveur, son intégration dans l'outil proposé n'est pas finalisée mais il a été testé avec succès sur la fonction de transfert.

Conclusion et perspectives

La conception de grands systèmes sociotechniques, qu'ils soient grands par la taille, par le nombre de sous-ensembles identifiables ou par la multiplicité de leurs interactions internes et externes, reste une activité complexe mêlant les sciences exactes aux sciences humaines.

La nécessité de faire appel à des champs de compétence variés et surtout très éloignés les uns des autres (de la mécanique aux sciences humaines), entraîne des difficultés de communication au sein des équipes de conception. Ces problèmes de communication ont des répercussions coûteuses voire dangereuses sur le produit final et sa conception, obligeant à de nombreux contrôles et rebouclages pour tenter de détecter puis de résoudre ces problèmes.

Or, les entreprises qui sont aujourd'hui soumises à un contexte de concurrence globale, sont sans cesse poussées à améliorer leur productivité. Il nous est donc apparu pertinent d'orienter nos travaux sur les méthodes et outils de conception permettant d'améliorer cette productivité.

Rappel de la contribution

Nos travaux traitent de la génération de programmes de commande et d'interfaces de supervision ; nous nous sommes attachés à établir un état de l'art des pratiques des industriels et un état de l'art académique des méthodes et des outils développés dans ce contexte.

De l'**enquête** menée auprès des industriels, nous avons retenu tout d'abord leur volonté unanime de développer la standardisation à des fins de réutilisation. Nous avons également relevé l'habitude des experts d'appuyer leur conception sur des schémas de principe établis très tôt dans le processus.

Cette enquête nous a également permis d'établir un processus de conception générique sur lequel nous nous sommes appuyés pour élaborer notre contribution.

De l'état de l'art académique, nous retenons que les trois paradigmes présentés (objet, patron et modèle), pris individuellement, ne permettent pas d'établir une solution satisfaisante. Il nous a semblé opportun de tirer partie de ces trois paradigmes dans un **processus unifié,** qui conjugue une approche ascendante pour l'établissement de la spécification de l'expert, et une approche descendante pour la génération du programme de commande et de l'interface de supervision à partir de cette spécification.

Lors de la phase ascendante de notre processus, et pour répondre au besoin de standardisation exprimé par les industriels, l'expert s'appuie sur une **bibliothèque** d'éléments standards pour établir le modèle de son système. Conformément à notre retour d'expérience, ce modèle prend la forme d'un schéma, le **synoptique**. Le synoptique est un modèle métier que nous considérons comme un langage spécifique au domaine du système conçu. Dans notre cas d'application, le synoptique est normalisé : il s'agit d'un schéma PID.

Après le paradigme d'objet qui nous a inspiré la bibliothèque, nous nous inspirons du paradigme de modèle. Notre synoptique est le point de départ d'une série d'opérations implémentées sous la forme de **transformations** de modèle. Un métamodèle a ainsi été proposé pour formaliser chaque modèle intervenant dans le flot de conception.

Une première opération permet de rassembler l'ensemble des références aux informations, nécessaires à la génération de la commande et de la supervision dans un modèle central, la **nomenclature**.

Une seconde opération introduit le paradigme de patron puisque les informations relatives à la supervision du système et référencées dans la nomenclature sont insérées dans un patron de conception : le **modèle standard d'IHM**.

Une troisième opération réalise l'assemblage des informations relatives à la commande du système et référencées dans la nomenclature pour former un **programme élémentaire**.

Implémentation de la contribution

L'outil **Anaxagore** a été développé pour appliquer notre démarche. Nous avons pour cela défini un sujet d'étude (voir annexe). Réalisé sous Eclipse, Anaxagore permet d'établir le lien entre, d'une part, le logiciel Microsoft Visio utilisé pour la saisie du synoptique et, d'autre part, les logiciels Panorama E2 pour la supervision et Straton pour la commande.

L'application de la démarche donne des résultats satisfaisants dans le sens où elle permet de générer en quelques clics un projet d'interface graphique (le canevas d'IHM) et un programme élémentaire.

Les experts en charge de la conception voient le temps passé en spécifications intermédiaires diminuer au profit du temps passé sur les spécifications de leur corps de métier.

Au delà du temps que permet de gagner notre approche, son automatisation, dans la génération des modèles cibles, garantie également la cohérence entre le synoptique, le programme élémentaire, et le canevas d'IHM.

Néanmoins cette première approche est incomplète. Si elle prend bien en compte l'**architecture** du système, elle néglige l'aspect **fonctionnel** de la conception et ne permet pas de considérer le système dans son ensemble, mais uniquement à travers ses éléments constitutifs, pris indépendamment les uns des autres.

Extension de la contribution

Pour pallier cette lacune, nous avons proposé une **extension** de notre flot de conception dont l'objectif est de permettre la génération de commandes globales du système, tout en prenant en compte son caractère reconfigurable.

Après avoir défini les concepts liés à la prise en compte du système dans son ensemble, nous avons proposé des **patrons de séquence et de supervision** qui serviront de base à la génération automatique.

Nous avons également défini un **modèle d'analyse des configurations,** combinant un graphe de potentiel et des contraintes. A partir de ce modèle, des **programmes linéaires** peuvent être construits et résolus par un solveur.

La résolution de ces programmes peut servir à la **validation** d'une configuration spécifiée par l'expert, mais elle peut aussi servir à la recherche d'alternatives de commande en exploitant le caractère **reconfigurable** du système.

Le modèle d'analyse des configurations a fait l'objet d'une implémentation dans l'outil Anaxagore. La bibliothèque et la nomenclature ont été étendues pour permettre sa génération automatique.

Par ailleurs, l'opération de génération de l'interface de supervision a été modifiée pour prendre en compte la notion d'**équipotentiel**, issue du modèle d'analyse. Cette notion permet de proposer une animation des symboles de tuyauterie représentés à l'écran en fonction de l'état des éléments du système. Le but de cette animation est de permettre à l'opération de visualiser rapidement la configuration de son système.

Nous avons une nouvelle fois testé l'outil Anaxagore sur un modèle toujours issu de notre sujet d'étude présenté en annexe. Les résultats de nos tests s'avèrent concluants. Sans prendre davantage de temps que lors des précédents essais, nous générons maintenant une interface graphique dont l'animation reproduit fidèlement le comportement du système, et nous générons en plus un modèle d'analyse des configurations, exploitable par un solveur.

Perspectives

Nous avons déjà évoqué les perspectives à très court terme qui visent à implémenter la génération de la commande globale et de son interface de supervision. Un solveur exécutable lors de l'exploitation du système devra également être généré. Son utilisation permettra d'envisager une reconfiguration dynamique du système en cas d'aléas ou en cas de commandes globales conflictuelles. Les développements sont en cours sur ces deux points.

Ajout de vérifications au flot

A moyen terme, l'ajout de **vérifications** au flot de conception permettrait de valider plus rapidement les modèles de conception. Deux pistes sont envisagées pour explorer ce thème.

La première consiste en la génération automatique d'un dispositif de vérification par **simulation** du système.

L'utilisation de simulateurs dans la conception de systèmes de grande échelle est largement **répandue dans l'industrie** (SIMSED, Delmia/Quest, ARENA). Cette solution fonctionnelle nous paraît donc pertinente. L'objectif principal consiste à introduire dans les étapes amont de notre démarche, un ensemble de techniques de vérification par simulation.

La principale difficulté dans la mise en œuvre de ces techniques est **l'obtention du modèle de simulation**, qui nécessite une grande expertise à la fois dans le système produit et dans la simulation. Ce verrou est d'autant

plus important que nous visons l'utilisation de la simulation, non pas en fin de conception, mais tout au long du cycle ; le nombre de modèles de simulation étant plus important.

La seconde piste que nous souhaitons explorer nous est inspirée par notre modèle d'analyse des configurations. Rappelons qu'il peut être utilisé pour valider formellement une configuration spécifiée par l'expert.

L'utilisation de techniques de vérification par **preuve formelle** a démontré son intérêt, notamment dans la validation de composants électroniques. Cette solution fonctionnelle prend tout son sens dans un processus de génération automatique au sein duquel la vérification serait automatisée. Cette piste nous paraît donc également pertinente. L'objectif principal consiste à introduire dans les étapes amont de notre démarche, un ensemble de techniques de vérification par preuve formelle, que ce soit au niveau des **exigences** ou des bonnes **propriétés** des modèles obtenus.

La principale difficulté dans la mise en œuvre de ces techniques outre l'obtention du modèle de vérification, concerne **l'explosion combinatoire** dans l'exécution du modèle. Ces verrous sont d'autant plus importants que nous visons l'utilisation de la vérification, non pas en fin de conception, mais tout au long du cycle ; le nombre de modèles de vérification étant de ce fait plus important.

Ces techniques de vérification par simulation ou par preuve formelle permettront de proposer plus rapidement et à moindre coût, des systèmes plus sûrs et répondant aux exigences de l'utilisateur.

Ajout de critère ergonomiques

Dans un tout autre domaine, et comme nous traitons de systèmes **sociotechniques**, il nous paraît tout à fait intéressant d'étudier l'introduction de critères ergonomiques pour la génération de l'interface de supervision.

L'utilisation de **critères ergonomiques** dans la conception d'interfaces graphiques a démontré son intérêt dans de nombreux secteurs, des services (interfaces web par exemple) à l'industrie (interface de supervision). La prise en compte de tels critères dans un processus de génération automatique d'interface prend alors tout son sens.

Il s'agit dans le processus de conception de l'IHM d'évaluer automatiquement l'ergonomie de l'IHM et de la reconcevoir en conséquence. Les limites d'une automatisation de la démarche ergonomique ont pu être montrées. Il n'empêche que les efforts de **modélisation des connaissances** en ergonomie des IHM, permettent de plus en plus d'envisager la possibilité d'une automatisation de certains aspects de l'ergonomie des IHM.

L'approche écologique que nous avons déjà évoquée dans ce manuscrit, propose un cadre théorique qui constitue, en ergonomie, une approche proposant des **outils méthodologiques formalisant** certaines recommandations ergonomiques pour la conception d'IHM.

Par ailleurs, le cadre théorique des interfaces écologiques présente l'avantage d'avoir principalement été appliqué à des systèmes de travail impliquant la supervision de fluide, comme dans notre sujet d'étude.

Les travaux menés sur ce point permettraient, d'une part, de tester la possibilité de définir un **domaine de travail générique** de type « supervision de fluide », avec des spécifications en fonction de métiers particuliers (sous-domaines), et d'autre part, de tester la possibilité d'intégrer la démarche de conception d'interfaces écologiques dans un système de gestion des interfaces utilisateur.

D'autres extensions possibles

D'autres perspectives portant essentiellement sur des extensions du flot actuel sont également envisageables.

Ainsi, concernant **l'architecture du système de contrôle/commande** qui n'est aujourd'hui pas prise en compte, de futurs travaux pourraient s'attacher à développer cet aspect en intégrant, dès les phases amont, les caractéristiques de cette architecture (distribution des calculateurs et des E/S, bus, etc.). L'objectif serait de générer un **code partitionnable** ainsi que des modèles destinés aux opérateurs de montage (plans de bornage). Il serait également intéressant dans ce cadre, de rechercher l'**optimisation** du partitionnement. Ces travaux pourraient s'appuyer sur la norme IEC 61499 (IEC 2005).

Le flot actuel se concentre sur l'aspect programmation du système à concevoir et ne prend pas en compte l'aspect fabrication physique de ce même système. Une extension possible permettrait de générer un **modèle d'emménagement 3D** issu du synoptique ou de modèles complémentaires (plan de bornage pour les nappes de câbles par exemple). L'aboutissement de cette démarche étant la génération automatique de plans de montage pour les opérateurs. La cohérence avec l'architecture du système étant assurée par le flot proposé. Un aspect **validation** d'emménagement et de respect des **exigences** pouvant être associé à une analyse du modèle 3D généré. Ces travaux établiraient un portage de l'IDM dans la **CAO**.

Les techniques de génération automatique sont peu pérennes, en particulier par manque de document accompagnant les modèles exécutables générés. Une façon de **capitaliser le savoir faire** de l'entreprise dans les outils, consisterait à générer automatiquement la documentation associée à ces différents modèles, ainsi que les documents livrables. Une extension du flot initial pourrait être proposée en s'appuyant sur les concepts de la **gestion électronique de documents**. Elle se baserait sur des modèles de document électronique (métamodèle d'information, canevas de GED) et pourrait être outillée par des transformations.

Références bibliographiques

AFNOR (1991). Définition des critères ergonomiques de conception et évaluation des interfaces utilisateurs. **Z67-133.1**.

AFNOR (1991). Le management de projet – Concepts. **NF X50–105**.

AFNOR (2007). Management par la valeur - Expression Fonctionnelle du Besoin et cahier des charges fonctionnel - Exigences pour l'expression et la validation du besoin à satisfaire dans le processus d'acquisition ou d'obtention d'un produit. **NF X50–151**.

Alexander, C. (1979). The Timeless Way of Building, Oxford University Press.

Ambone, G., J. Ernu, et al. "Interaction." from http://interaction2.free.fr/.

ANFH. "Fiche outil - Entretien." from http://www.anfh.asso.fr/fonctioncadre/cadre.htm.

ANSI (1992). Instrumentation Symbols and Identification. A. N. STANDARD, ISA. **ANSI/ISA-5.1-1984 (R1992)**.

Apple (2010). Cocoa Fundamentals Guide. Cocoa Design Pattern, Apple Inc.

Arnaud, N. (2008). Fiabiliser la réutilisation des patrons par une approche orientée complétude, variabilité et généricité des spécifications. LIG. Grenoble, Université Joseph-Fourier - Grenoble I. **PhD thesis**.

ASHRAE (2010). BACnet A Data Communication Protocol for Building Automation and Control Networks (ANSI Approved) Standard 135-2010. R. a. A.-C. E. American Society of Heating, ASGRAE. **ANSI/ASHRAE 135-2010**.

Audibert, L. (2009). "UML 2 - cours." from http://laurent-audibert.developpez.com/Cours-UML/.

Auroy, M. (1999). "Processus d'industrialisation : Schémas en industrie de process." Techniques de l'ingénieur **vol. 1**(noAG3200): AG3200.3201-AG3200.3219.

Autran, F., J.-P. Auzelle, et al. (2008). Coupling component systems towards systems of systems. 18th Annual International Symposium of INCOSE, 6th Biennal European System Engineering Conference, INCOSE 2008. Utrecht, Netherlands.

Berruet, P. (2007). Ingénierie de la Commande et Analyse des Systèmes Reconfigurables. LESTER. Lorient, Université de Bretagne-Sud. **HDR**.

Berry, D. M. (2008). "The Software Engineering Silver Bullet Conundrum." Software, IEEE **25**(2): 18-19.

Berry, G. (2008). Pourquoi et comment le monde devient numérique. Leçon inaugurale. C. d. France.

Bertrand, M. (2010) "Automates Programmables Industriels." Techniques de l'ingénieur.

Bévan, R., W. Allègre, et al. (2011). The SimSED framework for modelling and simulation of transitic systems under uncertain environment. ISC2011. Milan, Italia, EUROSIS.

Bézivin, J. (2004). Sur les principes de base de l'ingénierie des modèles. Paris, FRANCE, Lavoisier.

Bézivin, J. and O. Gerbé (2001). Towards a precise definition of the OMG/MDA framework. 16th Annual International Conference on Automated Software Engineering, 2001. (ASE 2001). San Diego, USA**:** 273-280.

Bignon, A. (2010). Campagne d'entretiens menés auprès d'indsutriels - Synthèse. Lorient, Lab-STICC / SEGULA.

Bignon, A., P. Berruet, et al. (2010). Joint generation of controls and interfaces for sociotechnical and reconfigurable systems. IEEE International Conference on Systems Man and Cybernetics (SMC). Ieee. Istanbul, Turkey**:** 749-755.

Bignon, J.-C., G. Halin, et al. (2000). CoCAO : Collecticiel à l'usage des métiers du bâtiment. IBPSA'2002.

Booch, G. (1994). Object-oriented analysis and design with applications (2nd ed.), Benjamin-Cummings Publishing Co., Inc.

Bornand, M. (2004). "The UNICOS Project." from http://ab-project-unicos.web.cern.ch/ab-project-unicos/.

Brandt, D., E. Hartmann, et al. (1999). "Designing and simulating sociotechnical systems: Concepts and strategies." Human Factors and Ergonomics in Manufacturing & Service Industries **9**(3): 245-252.

Breton, E. (2002). Contribution à la représentation de processus par des techniques de méta-modélisation. CRGNA. Nantes, Université de Nantes. **PhD thesis**.

Brooks, F. P. J. (1987). "No Silver Bullet Essence and Accidents of Software Engineering." Computer **20**(4): 10-19.

Brooks, F. P. J. (1996). Le mythe du mois-homme - Essai sur le génie logiciel.

Burns, C. M. and J. R. Hajdukiewicz (2004). Ecological Interface Design, CRC PRESS.

Butler, K. L., N. D. R. Sarma, et al. (2001). "Network reconfiguration for service restoration in shipboard power distribution systems." Power Systems, IEEE Transactions on **16**(4): 653-661.

Byunghee, C., K. Sungho, et al. (2010). Development of excitation control process in IMCS. International Conference on Control Automation and Systems (ICCAS), 2010 515-518.

Caelen, J. (2009). Conception participative par "moments": une gestion collaborative. Paris, FRANCE, Presses Universitaires de France.

Charfi, A., A. Gamatié, et al. (2008). Validation de modèles dans un cadre d'IDM dédié à la conception de systèmes sur puce. IDM 2008, Mulhouse, France.

Chevallereau, B., A. Bernard, et al. (2009). Améliorer les performances de l'industrie logicielle par une meilleure compréhension des besoins. CIGI.

Chevallereau, B., A. Bernard, et al. (2009). Ingénierie dirigée par les modèles pendant la phase de spécification du besoin. Ingénierie Dirigée par les Modèles.

Chissholm, W., G. Vanderheiden, et al. (1999). Web Content Accessibility Guidelines. W3C. **1.0**.

Coad, P., D. North, et al. (1997). Object models: strategies, patterns, and applications, Yourdon Press.

Combemale, B. (2008). Approche de métamodélisation pour la simulation et la vérification de modèle -- Application à l'ingénierie des procédés. IRIT. Toulouse, Institut National Polytechnique de Toulouse - INPT. **PhD thesis**.

Conte, A., M. Fredj, et al. (2001). P-Sigma : un formalisme pour une représentation unifiée de patrons. XIXème Congrès INFORSID. essai, Genève, 69000**:** 67-86.

Copalp (2002). STRATON Demonstration project.

Coplien, J. O. and D. C. Schmidt (1995). Pattern languages of program design, Addison-Wesley.

Dartigues, C. (2001). État de l'art sur l'Ingénierie Concourante.

de Singly, F. (2008). L'enquête et ses méthodes : Le questionnaire, Armand Colin.

Drath, R., A. Fay, et al. (2006). Computer-aided design and implementation of interlock control code. CACSD-CCA-ISIC.2006. Munich, Germany**:** 2653-2658.

Dromey, R. G. (2006). "Climbing over the "No Silver Bullet" Brick Wall." IEEE Softw. **23**(2): 120-119.

Eynard, B. (2005). Gestion du cycle de vie des produits et dynamique des connaissances industrielles en conception intégrée. LSMIS. Troyes, UTC. **HDR**.

Ferrarini, L., A. Dedè, et al. (2009). Domain specific views in model-driven embedded systems design in industrial automation. 7th IEEE International Conference on Industrial Informatics, 2009. INDIN 2009. Cardif, UK**:** 702-707.

Ferry, N. (2008). Formalisation des modèles de la méthode Macao et réalisation d'un outil de génie logiciel pour la création d'interfaces homme machine. IRIT. Toulouse, Université Paul Sabatier. **PhD thesis**.

Fiorese, S. (2005). Pourquoi l'Ingénierie système? IS, pourquoi? Comment?, Association Française d'Ingénierie Système.

Fondement, F. (2007). Concrete syntax definition for modeling languages. ISIM-LGL. Lausanne, IC Faculté informatique et communications. **PhD thesis**.

Forsberg, K. and H. Mooz (2006). Architecture and Entity Vees Intersecting. Architecture_and_Entity_Vees_Intersecting.gif. Truckee, California, USA.

Fortin, T. (2010). "ATL/User Guide - Overview of the Atlas Transformation Language." from http://wiki.eclipse.org/ATL/User_Guide_-_Overview_of_the_Atlas_Transformation_Language.

Fraser, S. and D. Mancl (2008). "No Silver Bullet: Software Engineering Reloaded." IEEE Software **25**(1): 91-94.

Frizon De Lamotte, F. (2006). Proposition d'une approche haut niveau pour la conception, l'analyse et l'implantation des systèmes reconfigurables. LESTER. Lorient, Université de Bretagne-Sud. **PhD thesis**.

Frizon De Lamotte, F., P. Berruet, et al. (2008). Control/Command code generation using Model Engineering applied on an electric train. 17th IFAC World Congress (IFAC'08). Seoul, Korea**:** 8327-8332.

Gabillon, Y., G. Calvary, et al. (2008). L'IDM passerelle entre IHM et planification poour la composition dynamique de système interactifs. 4èmes Journées sur l'Ingénierie Dirigée par les Modèles. Mulhouse.

Gamma, E., R. Helm, et al. (1995). Design Patterns: Elements of Reusable Object-Oriented Software, Addisson-Wesley.

Garcia, A., B. Combemale, et al. (2009). "topPROCESS : vers une ingénierie des procédés dirigée par les modèles." Revue de l'Electricité et de l'Electronique (REE) **2/2009**: Société de l'Electricité, de l'Electronique et des Technologies de l'Information et de la Communicat.

Gerbé, O., G. W. Mineau, et al. (2007). Un métamodèle des graphes conceptuels. Cachan, FRANCE, Lavoisier.

Gholipour, V., J.-C. Bignon, et al. (2009). Eco-Modèles : Une méthode d'aide à l'éco-conception de bâtiments durables. CONFERE 09.

Harel, D. (1992). "Biting the Silver Bullet: Toward a Brighter Future for System Development." Computer **25**(1): 8-20.

Hariri, A., D. Tabary, et al. (2009). "Principes et étude de cas d'adaptation d'IHM dans les SI en fonction du contexte d'interaction de l'utilisateur." Ingénierie des Systèmes d'Information **14**(3): 141-162.

Hug, C., A. Front, et al. (2008). Ingénierie des processus : Une approche à base de patrons. Paris, FRANCE, Lavoisier.

IEC (2003). Programmable controllers. Part 3: Programming languages **IEC 61131-3**.

IEC (2005). Functions blocks. Part 1 - Part 4. **IEC 61499**.

ISO (1999). Ships and marine technology — Identification colours for the content of piping systems. Main colours and media. **ISO 14726-1:1999(E)**.

ISO (2006). Ergonomie de l'interaction homme-système. Principes de dialogue. **ISO 9241-110:2006**.

ISO (2010). Ergonomie de l'interaction homme-système. Conception centrée sur l'opérateur humain pour les systèmes interactifs. **ISO 9241-210:2010**.

Jacobson, I. (1992). Object-oriented software engineering, ACM.

Jéron, T., H. Marchand, et al. (2006). Supervision Patterns in Discrete Event Systems Diagnosis. Workshop on Discrete Event Systems, WODES'06. Ann Arbor, USA, IEEE Computer society**:** 262-268.

Jézéquel, J.-M. (2006). Patrons de conception. Encyclopédie Vuibert de l'informatique. J. Akoka and I. Comyn-Wattiau, Vuibert.

Jones, T. C. (1984). Reusability in programming: a survey of the state of the art. New York, NY, ETATS-UNIS, Institute of Electrical and Electronics Engineers.

Kanso, M. (2010). Contribution à la construction des configurations des systèmes manufacturiers : une approche basée sur la décision multi-critère. Lab-STICC. Lorient, Université de Bretagne-Sud. **PhD thesis**.

Kolski, C. and H. Ezzedine (2003). "Conception et évaluation des IHM de supervision : éléments méthodologiques." Génie Logiciel **65**: 2-11.

Lahonde, N., J.-F. Omhover, et al. (2008). Vers un outil intelligent de génération de modèles de processus de conception. CONFERE. Angers.

Lallican, J.-L. (2007). Proposition d'une approche composant pour la conception de la commande des systèmes transitiques. LESTER. Lorient, Université de Bretagne-Sud. **PhD thesis**.

Lallican, J.-L. and P. Berruet (2006). SIMSED : un environnement pour modéliser et simuler des systèmes transitiques. 6e Conférence Francophone de MOdélisation et SIMulation MOSIM'06. Rabat, Maroc.

Lavoisy, O. and D. Vinck (2000). Le dessin comme objet intermédiaire de l'entreprise. Communications organisationnelles. Objets, pratiques et dispositifs. P. Delchambre, Presses Universitaires de Rennes**:** 47-63.

Liebowitz, S. J. and S. E. Margolis (1990). "The Fable of the Keys." Journal of Law and Economics **33**(1): 26.

Luzeaux, D. (2004). La place de l'homme dans les systèmes de systèmes. Ergo'ia. Biarritz.

MacLeod, I. and D. Smeall (1999). A proposed integrated platform management system design for the RN future surface combatant. International Conference on Human Interfaces in Control Rooms, Cockpits and Command Centres, 1999.**:** 125-130.

Mary, S. (18/01/2011). "La GED (Gestion Électronique des Documents)." 2009, from http://www.cerpeg.ac-versailles.fr/tice/ged.htm.

McGregor, I. (2002). The relationship between simulation and emulation. Simulation Conference, 2002. Proceedings of the Winter. **2:** 1683-1688 vol.1682.

Mens, T. and P. Van Gorp (2006). "A Taxonomy of Model Transformation." Electronic Notes in Theoretical Computer Science **152**: 125-142.

Mertens, M. and U. Epple (2010). Metamodel-driven property management in process industries. 36th Annual Conference on IEEE Industrial Electronics Society. IECON 2010. Phoenix, USA**:** 1347-1352.

Microsoft. (2011). "MSDN Library." from http://msdn.microsoft.com/en-us/library/ms123401.aspx.

Mouchard, J.-S. (2002). Proposition d'une approche méthodique pour la conception des systèmes automatisés de production : application aux systèmes transitiques. LESTER. Lorient, Université de Bretagne-Sud. **PhD thesis**.

Muller, P.-A., F. Fondement, et al. (2010). "Modeling modeling modeling." Software and Systems Modeling: 1-13.

Nguyen, T.-H., A. Abran, et al. (1997). "Les concepts de la réutilisation du logiciel et la pratique institutionnelle dans les entreprises québécoises." Génie Logiciel.

Nicoloso, J., J. J. dupas, et al. (2009). Configuration and Sequencing Tools for the LMJ Control System. ICALEPCS2009. Kobe, Japan.

Niel, E. and E. Craye (2002). Maîtrise des risques et sûreté de fonctionnement des systèmes de production, Hermes Science.

OMG (2003). MDA Guide version 1.0.1. OMG.

OMG (2004). MOF 2.0 IDL Specification. OMG.

OMG (2006). Meta Object Facility (MOF) Core Specification. OMG.

OMG (2008). System Modeling Language. OMG.

OMG (2009). Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems. OMG.

OMG (2009). Unified Modeling Language Infrastructure. OMG.

OMG (2009). Unified Modeling Language Superstructure. OMG.

OMG (2010). Object Constraint Language. OMG.

OMS (2002). Hygiène et salubrité à bord des navires.

Pham, H. (2005). System Software Reliability (Springer Series in Reliability Engineering), Springer-Verlag New York, Inc.

PLCOpen (2009). XML Formats for IEC 61131-3, PLCOpen. **TC6 XML**.

Pomian, J.-L., T. Pradère, et al. (1997). Ingénierie et Ergonomie.

Rasmussen, J. (1985). "The role of hierarchical knowledge representation in decision making and system management." {IEEE} Transactions on Systems, Man, and Cybernetics **{SMC-15}**(2): 234--243.

Rivière, A. (2004). Gestion de configuration et des modifications lors du développement de grands produits complexes en ingénierie concourante. GILCO. Grenoble, INPG. **PhD thesis**.

Rochet, S. (2007). Formalisation des Processus de l'Ingénierie Système : Proposition d'une méthode d'adaptation des processus génériques à différents contextes d'application. LATTIS. Toulouse, Université Paul Sabatier - Toulouse III. **PhD thesis**.

Royce, W. W. (1970). Managing the development of large software systems: concepts and techniques. Western Electronic Show and Convention, WESCON/70. Los Angeles, USA, IEEE Computer Society Press**:** 1-9.

Rumbaugh, J. (1996). OMT insights: perspectives on modeling from the Journal of Object-Oriented Programming, SIGS Books.

Saidane, M. (2005). Formalisation de Familles d'Architectures Logicielles Coopératives : Démarches, Modèles et Outils. LSR-IMAG. Grenoble, Université Joseph-Fourier - Grenoble I. **PhD thesis**.

Selic, B. (2008). "Personal reflections on automation, programming culture, and model-based software engineering." Automated Software Engg. **15**(3-4): 379-391.

Siegel, C. (2008). The Politics of Simple Living - A new direction for liberalism. Berkeley, California, Preservation Institute.

Sottet, J.-S., G. Calvary, et al. (2006). IHM & IDM : Un tandem prometteur. Ergo'IA. Biarritz, CLIPS-IMAG.

Sourisse, C. and L. Boudillon (1997). La sécurité des machines automatisées: Techniques et moyens de prévention opératifs, systèmes de commande, utilisation des machines, Institut Schneider Formation.

Sperandio, J.-C. (2008). L'humain au coeur des systèmes et de leur développement : quelle évolution en 20 ans dans le développement des systèmes ? Ergoia. Biarritz, ESTIA.

Sverdlov, A., D. Kopec, et al. (2005). Applied Software Synthesis. Third International Conference on Computer Science and its application, ICCSA 2005. San Diego, USA.

Thomas, D. (2003). "UML - Unified or Universal Modeling Language? UML2, OCL, MOF, EDOC - The Emperor Has Too Many Clothes." Journal of Object Technology **2**(1): 6.

Tran, H. N. (2007). Modélisation de procédés logiciels à base de patrons réutilisables. IRIT. Toulouse, Université Toulouse le Mirail - Toulouse II. **PhD thesis**.

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. LISMMA. Toulon, Université de Toulon et du Var. **PhD thesis**.

Vepsalainen, T., D. Hastbacka, et al. (2008). Tool Support for the UML Automation Profile - For Domain-Specific Software Development in Manufacturing. The Third International Conference on Software Engineering Advances, 2008. ICSEA '08. Sliema, Malta**:** 43-50.

Vicente, K. J. and J. Rasmussen (1992). "Ecological interface design: theoretical foundations." IEEE Transactions on Systems, Man and Cybernetics **22**(4): 589-606.

Annexe : support de l'étude

1.Introduction

Les travaux de recherche réalisés au cours de cette thèse ont nécessité la définition d'un support représentatif du sujet de la recherche. Nous avons donc défini un démonstrateur de système sociotechnique reconfigurable.

En rapport avec les activités navales de SEGULA Technologies sur Lorient, nous avons choisi de nous attacher à la conception d'un système de production, de stockage et de distribution de l'eau douce embarquée sur un navire que nous notons **EdS** dans la suite. Tout type de navire quel que soit son tonnage, s'il doit naviguer pour de longues périodes, dispose d'un tel dispositif.

Nous avons choisi de porter notre étude sur un système adapté pour un navire de moyen tonnage pouvant transporter une centaine de personnes. Ce type d'unité est très répandu et couvre un large spectre d'activités, de la croisière à la défense en passant par les ravitaillements. A titre d'exemple pour ces trois activités, nous pouvons citer respectivement le Ponant célèbre pour sa prise d'otage, le Marion Dufresne 2 chargé du ravitaillement des TAAF[1] et le Surcouf, une frégate de classe Lafayette.

Fig. 107 Exemple de navires

Ces types de navire disposent aujourd'hui d'une conduite centralisée qui permet le contrôle/commande de la plupart de leurs installations depuis la passerelle du navire ; ce type de dispositif est appelé IPMS[2] (MacLeod and Smeall 1999). Le système de production, de stockage et de distribution de l'eau douce fait donc partie d'un système plus vaste, la plate-forme propulsée du navire qui elle-même sert de support à la fonction principale du navire ; que cette dernière soit de plaisance, utilitaire ou de défense.

[1] TAAF : Terres Australes et Antarctiques Françaises
[2] IPMS : Integrated Platform Management System

2. Le caractère sociotechnique du système

Le terme « système » désigne un ensemble d'éléments interagissant entre eux. Accompagné du terme « sociotechnique », nous comprenons alors que les opérateurs sont des éléments du système à part entière. En effet au milieu de l'océan un équipage est indissociable de son navire. D'une manière plus précise nous nous intéresserons à un sous-système faisant partie d'un système de rang supérieur. Nous n'étudierons pas le navire dans son ensemble mais simplement une partie du navire, comme le système EdS. Avec une automatisation galopante et des ressources humaines qui diminuent, les facteurs humains prennent aujourd'hui une part importante dans la conception car ils doivent permettre d'appréhender au mieux ce que sera l'activité de conduite du système final. Le système conçu devra donc tenir compte de ces aspects.

La conception d'un système nécessite de considérer le système dans son environnement. Dans le domaine des interfaces graphiques, ce type d'approche est dite écologique. La conception écologique est basée sur une analyse du domaine de travail, matérialisée par un modèle de hiérarchie d'abstraction (Vicente and Rasmussen 1992). Pour de plus amples détails quant à la méthodologie de conception, nous renvoyons vers le livre Ecological interface Design (Burns and Hajdukiewicz 2004). En nous inspirant de cette démarche issue des travaux de Vicente et Rasmussen nous proposons une hiérarchie d'abstraction (Fig. 108) pour le type de navire que nous considérons.

Pour cette hiérarchie d'abstraction nous considérons les fonctions principales de la plate-forme propulsée du navire. La première fonction d'un navire reste « aller d'un point A à un point B » ; c'est avant tout un moyen de transport. La seconde fonction est « assurer la sécurité » du navire et de la cargaison. La troisième fonction est « assurer la survie » des passagers. A ces fonctions principales pourront s'ajouter celles issues de la spécificité du navire comme « assurer le confort des passagers » pour un navire de croisière.

Les fonctions abstraites sont les règles qui gouvernent les fonctions principales. Ainsi, le « droit maritime » a un impact sur la fonction « aller d'un point A à un point B » puisqu'il définit des règles de navigation. Il a également une relation avec la sécurité maritime. Les principes de l'hydrodynamique gouvernent la fonction de déplacement. Les normes de construction donnent des règles concernant la sécurité du navire mais également concernant la survie des passagers.

Les fonctions généralisées identifiées sont, d'une part maîtriser l'attitude du navire et, d'autre part maîtriser l'itinéraire. Ces deux fonctions permettent le déplacement du navire en toute sécurité.

La première fonction physique identifiée est « contrôler la navigation », elle répond au besoin de contrôler l'itinéraire et pourra se décliner sous la forme physique d'interfaces de surveillance de l'espace maritime et de cartographie. L'ECDIS[1] est un exemple de forme physique répondant à cette

[1] ECDIS :Electronic Charts Display Information System

fonction. La seconde fonction vise le contrôle de la plate-forme propulsée du navire. Cette fonction physique répondra aux besoins de contrôler l'itinéraire et l'attitude du navire. Il se décline sous la forme physique d'un contrôle de la sécurité du bord, d'un contrôle de l'énergie, de la propulsion et des systèmes auxiliaires.

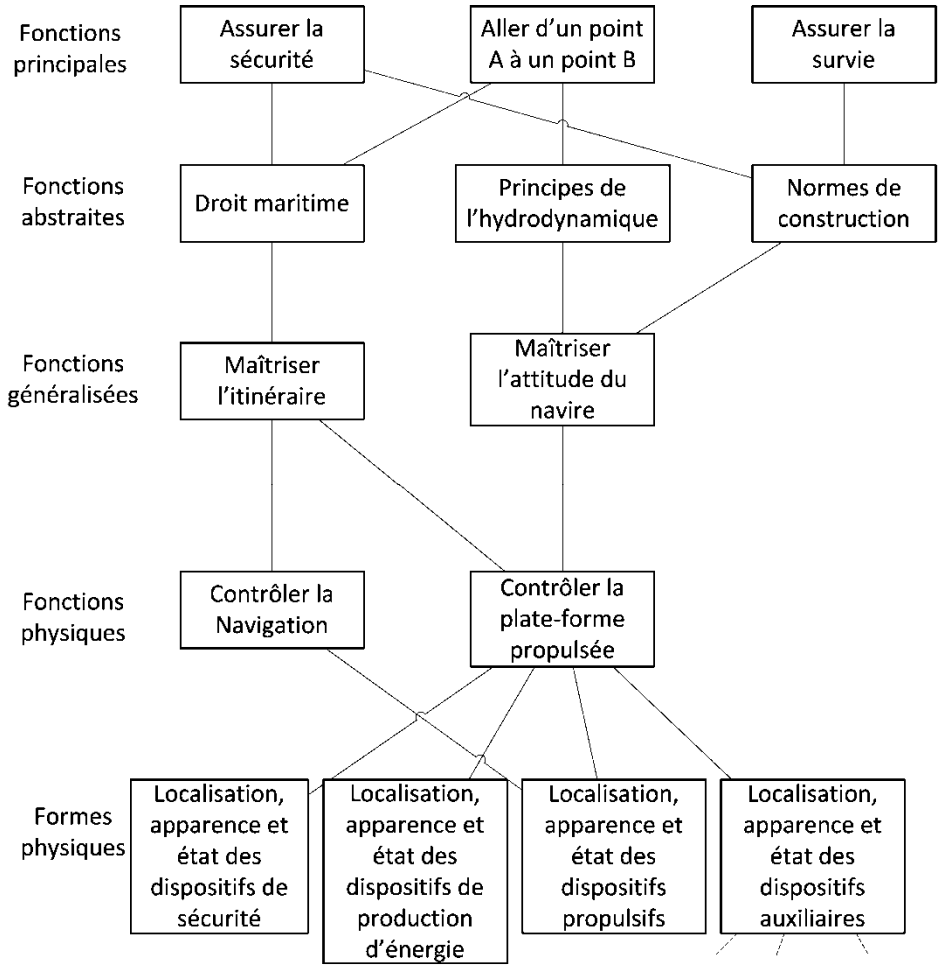


Fig. 108 Hiérarchie d'abstraction d'un navire

La place du système de production, de stockage et de distribution de l'eau douce dans la hiérarchie d'abstraction se situe dans la forme « localisation, apparence et état des dispositifs auxiliaires ». On peut considérer que ce système participe à la fonction principale « assurer la survie », en fournissant à bord une eau propre à la consommation.

3. Les caractéristiques induites du système

Nous avons déjà évoqué le système de production de stockage et de distribution de l'eau douce comme faisant partie d'un ensemble plus vaste. Cela le caractérise comme un système de système. Nous avons également évoqué son étroite relation avec l'équipage qui en fait un système sociotechnique.

A ces deux caractéristiques nous pouvons ajouter l'aspect reconfigurable. En effet, afin d'assurer la disponibilité du système lors de longues périodes en mer, le système est notamment doté d'équipements redondants qui permettent de faire fonctionner le système dans différentes configurations pour un même but.

Du point de vue de l'équipage qui exerce sur le système un pilotage macroscopique, le comportement de celui-ci est discret. Par exemple, pour une pompe de régulation de pression, l'opérateur en charge de la commande du système ne gère pas directement l'asservissement en vitesse de la pompe mais simplement son état (Marche/Arrêt) ou son mode de fonctionnement (Manuel/Automatique). Nous pouvons donc le classer dans les systèmes à événement discret.

Système EdS
▪ Système de système ▪ Système reconfigurable ▪ Système sociotechnique ▪ Système à événement discret

Fig. 109 Caractéristiques du sujet de l'étude.

4. Description du système EdS

La description du système de production, de stockage et de distribution d'eau douce constitue un document de conception générale servant de base à la conception détaillée, développée dans le corps de ce manuscrit.

Cette description fait référence au schéma du système et à la liste de ses éléments présentés dans les sections suivantes.

1. Objectifs

Le système EdS d'un navire doit répondre aux objectifs suivants :

- Produire de l'eau douce à partir de l'eau de mer.
- Embarquer/Débarquer de l'eau douce à terre.
- Traiter l'eau douce.
- Stocker l'eau douce.
- Distribuer l'eau douce.
- Chauffer l'eau douce.
- Refroidir l'eau douce.
- Produire de l'eau déminéralisée.
- Stocker l'eau déminéralisée.

2. Description technique

Il existe différentes manières de **produire** de l'eau douce à partir de l'eau de mer. Les bouilleurs sont restés pendant longtemps le principal moyen de produire de l'eau douce par distillation de l'eau de mer. Du fait de la forte consommation en énergie des bouilleurs, aujourd'hui, la technique privilégiée est celle de l'osmose inverse. Cette technique consiste à soumettre l'eau de mer à une forte pression de manière à lui faire traverser une membrane semi-perméable. Cette technique dite d'ultrafiltration permet d'éliminer toute particule d'une taille supérieure à la molécule d'eau. Pour des raisons de disponibilité nous prévoyons de produire l'eau douce au moyen de deux osmoseurs (Osm Av et Osm Ar).

Permettre d'**embarquer** ou de **débarquer** de l'eau douce à terre doit servir à assurer le besoin en eau en cas de panne ou de maintenance des osmoseurs. La débarquer permet d'assurer la maintenance des soutes de stockage.

Traiter l'eau douce à bord est une nécessité (OMS 2002) afin d'en assurer la potabilité lors du stockage et lors de la distribution. Deux moyens sont utilisés conjointement pour atteindre cet objectif. Un premier traitement est réalisé sur l'eau stockée par un dispositif de chloration (TR1). Une pompe doseuse injecte du chlore dans la ligne de remplissage des soutes. Une ligne de rebouclage permet au besoin de retraiter une eau stockée. Un deuxième dispositif est utilisé pour traiter l'eau distribuée. Il s'agit d'un stérilisateur UV (TR2) ; les ultra-violets ont une action bactéricide sur l'eau exposée. Ce traitement est instantané mais ne garantit pas la potabilité dans le temps contrairement au chlore. C'est pourquoi il est utilisé en complément sur la ligne de distribution.

Le **stockage** de l'eau douce est réalisé dans deux soutes du navire (St1 et St2). Le volume de ces soutes dépend directement du nombre de passager sur le navire. Nous estimons le besoin à 70m3 par soute. Cela permet une autonomie d'environ 5 jours pour 100 personnes (consommation de 250L par jour et par personne). Un asservissement sur les dispositifs de production permet de maintenir le niveau dans les soutes.

La **distribution** de l'eau est réalisée au moyen de groupes hydrophores. Chaque groupe se compose d'une pompe centrifuge dont la vitesse est asservie en fonction de la pression et d'une vessie permettant de maintenir la pression dans le réseau en aval du groupe. Afin d'assurer la disponibilité en cas d'avarie ou de maintenance, le système dispose de trois groupes hydrophores (H1, H2 et H3) qui assurent également le pompage de l'eau pour débarquement ou retraitement. La distribution de l'eau est également possible depuis le quai (raccordement au réseau du port) au moyen d'une ligne de tuyauterie (isolée par la vanne V2VM10). La ligne de distribution est protégée par un limiteur de débit.

L'eau est **chauffée** au moyen de ballons (CE1 et CE2) asservis en température et est distribuée par une boucle d'eau chaude sur laquelle deux pompes assurent la recirculation (PC1 et PC2). L'eau reste ainsi chaude en tout point de la boucle.

L'eau **froide** est maintenue en température grâce à un échangeur (E1) connecté au réseau du système « eau réfrigérée ». Une vanne de régulation

(V2VR01) asservie en température contrôle le débit de l'eau réfrigérée dans l'échangeur.

Les osmoseurs (comme les bouilleurs) produisent avant tout une **eau déminéralisée** qui est reminéralisée en sortie pour la rendre consommable. Cette capacité est utilisée pour fournir à bord, de l'eau déminéralisée servant à des opérations de maintenance (refroidissement des moteurs par exemple). L'eau déminéralisée est **stockée** dans une soute (St3) dont le volume nécessaire est estimé à $10m^3$.

3. Description fonctionnelle

A partir des objectifs du système et des moyens identifiés précédemment nous pouvons établir la liste des fonctions du système. En nous basant sur la taxonomie proposée par la norme Afnor X50-151 (AFNOR 2007) nous classons les fonctions par type. Les fonctions principales justifient la conception du système, les fonctions contraintes sont imposées par le contexte du système et les fonctions complémentaires résultent de la conception du système. La figure 110 représente la liste des fonctions retenues pour le système EdS.

Objectif	Nom de la fonction	Type de fonction
Produire de l'eau douce à partir de l'eau de mer	Production	Principale
Produire de l'eau déminéralisée		Complémentaire
Stocker l'eau douce	Stockage	Principale
Stocker l'eau déminéralisée		Complémentaire
Embarquer/Débarquer de l'eau douce à terre	Embarquement	Principale
	Débarquement	Principale
Traiter l'eau douce	Chloration	Contrainte
	Stérilisation	Contrainte
	Brassage	Contrainte
Distribuer l'eau douce	Distribution	Principale
Chauffer l'eau douce	Chauffage	Principale
	Cirulation	Contrainte
Refroidir l'eau douce	Refroidissement	Principale
Transférer de l'eau douce	Transfert	Complémentaire

Fig. 110 Liste des fonctions du système EdS

Les objectifs **produire de l'eau douce à partir de l'eau de mer** et **produire de l'eau déminéralisée** sont réalisés par la fonction de *production*. Cette fonction met en œuvre les osmoseurs (Osm Av et Osm Ar) ainsi que les vannes du réseau de tuyauterie du système. Cette fonction est principale pour la partie production de l'eau douce qui est la raison d'être du système ; elle est complémentaire pour la partie production d'eau déminéralisée qui se décline du choix des osmoseurs pour la production.

De la même manière, les objectifs **stocker l'eau douce** et **stocker l'eau déminéralisée** sont réalisés par la fonction *stockage*. Cette fonction est principale pour l'eau douce et complémentaire pour l'eau déminéralisée ;

elle met en œuvre les soutes de stockage (St1 et St2 pour l'eau douce et St3 pour l'eau déminéralisée).

L'objectif **embarquer/débarquer de l'eau douce à terre** est réalisé par deux fonctions. La première, *embarquement*, met en œuvre les vannes du réseau de tuyauterie du système essentiellement en amont des soutes. La seconde, *débarquement*, met en œuvre les vannes du réseau de tuyauterie essentiellement en aval des soutes ainsi que les groupes hydrophores (H1, H2 et H3) utilisés pour le pompage et l'évacuation de l'eau des soutes. Ces deux fonctions sont des fonctions principales.

L'objectif **traiter l'eau douce** est quant à lui réalisé par trois fonctions. La fonction de *chloration* met en œuvre le module de chloration (TR1). La fonction de *stérilisation* met en œuvre le module de stérilisation UV (TR2). La fonction de *brassage* met en œuvre les vannes du réseau de tuyauterie du système ainsi que les groupes hydrophores (H1, H2 et H3). Ces trois fonctions sont contraintes par les normes de traitement des eaux potables.

L'objectif **distribuer l'eau douce** est réalisé par la fonction de *distribution* qui met en œuvre les vannes du réseau de tuyauterie du système ainsi que les groupes hydrophores (H1, H2 et H3). Cette fonction répond à l'une des raisons d'être du système ; il s'agit d'une fonction principale.

L'objectif **chauffer l'eau douce** est réalisé par deux fonctions. La première, *chauffage*, met en œuvre les chauffe-eau du système. Il s'agit d'une fonction principale. La seconde est la fonction de *circulation*. Elle met en œuvre les pompes de recirculation ainsi que la boucle d'eau chaude du réseau de distribution de l'eau. Il s'agit d'une fonction contrainte qui permet de limiter les risques de prolifération de bactéries.

L'objectif **refroidir l'eau douce** est réalisé par une fonction de *refroidissement*. Elle met en œuvre l'échangeur thermique du système ainsi que sa vanne de régulation. Il s'agit d'une fonction principale.

L'analyse fonctionnelle du système fait apparaître une fonction complémentaire de *transfert* de l'eau douce entre deux soutes. Cette fonction ne répond pas à un objectif justifiant la conception du système, mais peut s'avérer utile en exploitation pour assurer des opérations de maintenance sur une soute par exemple.

4. Exploitation des fonctions

L'implémentation des fonctions permet de préciser comment elles seront exploitées par l'opérateur. Nous faisons apparaître ici une autre typologie des fonctions extraite de (Niel and Craye 2002) qui distingue les fonctions génériques intrinsèques à un élément du système des fonctions contextuelles qui tiennent compte de l'environnement d'utilisation d'un ou plusieurs éléments.

Nous définissons également un mode d'exploitation des fonctions. Ce dernier peut être automatique si la fonction est lancée sans intervention de l'opérateur, semi automatique si la fonction est lancée par l'opérateur, ou manuel si l'opérateur doit assurer l'enchaînement de plusieurs commandes pour réaliser la fonction.

La figure 111 présente les modes d'exploitation des fonctions définies précédemment.

Nom de la fonction	Nature de la fonction	Mode d'exploitation
Production	Contextuelle	Automatique, Semi-automatique, Manuel
Stockage	Générique	/
Embarquement	Contextuelle	Semi-automatique, Manuel
Débarquement	Contextuelle	Semi-automatique, Manuel
Chloration	Générique	Automatique
Stérilisation	Générique	Automatique
Brassage	Contextuelle	Semi-automatique, Manuel
Distribution	Contextuelle	Automatique, Manuel
Chauffage	Générique	Automatique
Circulation	Contextuelle	Automatique
Refroidissement	Contextuelle	Automatique
Transfert	Contextuelle	Semi-automatique, Manuel

Fig. 111 Exploitation des fonctions du système EdS

La fonction de *production* est contextuelle car elle met en œuvre différents éléments qui doivent être coordonnés entre eux. Elle est exploitée préférentiellement sur un mode automatique. Si un niveau bas est détecté dans l'une des soutes de stockage, les vannes du réseau de tuyauterie sont configurées et les osmoseurs sont démarrés. Une détection de niveau haut dans les soutes entraîne la fin de la fonction. Cette fonction peut également être déclenchée par l'opérateur suivant un mode semi-automatique. Elle peut enfin être mise en œuvre par l'opérateur qui peut configurer lui-même le réseau et démarrer les osmoseurs. La fonction de production nécessite le développement d'une **commande globale** qui coordonne les différents éléments du système pour son exploitation.

La fonction de *stockage* est générique à l'élément soute. Elle ne dispose pas de mode d'exploitation puisque l'opérateur ou le système ne peut pas agir sur cette fonction. En conséquence, il n'y a pas de commande relative à la fonction de stockage

L'*embarquement* est une fonction contextuelle. Elle met en œuvre les vannes du réseau de tuyauterie en amont des soutes, nécessaires à leur remplissage. Cette fonction est déclenchée sur demande de l'opérateur qui doit coordonner le déroulement avec le personnel à quai. En effet pour cette fonction, les moyens de pompage sont extérieurs au navire. Pour ce mode semi-automatique, une **commande globale** doit être développée. L'opérateur garde aussi la possibilité de réaliser la fonction manuellement.

Le *débarquement* est une fonction contextuelle qui met en œuvre les vannes du réseau et les groupes hydrophores. Ces derniers sont utilisés pour pomper l'eau des soutes et l'évacuer du navire. La fonction est déclenchée par l'opérateur en supervision. Le déroulement du débarquement doit être coordonné car du personnel à quai doit assurer la réception de l'eau dans des moyens de stockage. Pour ce mode d'exploitation semi-automatique, une

commande globale doit être développée. L'opérateur peut également réaliser la fonction manuellement.

La fonction de *chloration* est générique au module de chloration. Son fonctionnement est complètement automatisé et ne nécessite pas d'intervention de l'opérateur en exploitation.

La fonction de *stérilisation* est générique au module de stérilisation UV. Son fonctionnement est également entièrement automatisé et ne nécessite pas d'intervention de l'opération en exploitation.

La fonction de *brassage* est contextuelle. Elle met en œuvre les vannes du réseau ainsi que les groupes hydrophores. L'opération est déclenchée sur demande de l'opérateur en supervision. Elle se termine automatiquement lorsque l'intégralité du volume de la soute à brasser est passée par les groupes hydrophores. Une **commande globale** doit être développée pour assurer ce mode d'exploitation semi-automatique. L'opérateur garde néanmoins la possibilité de dérouler la fonction manuellement.

La *distribution* est une fonction contextuelle qui met en œuvre les vannes du réseau et les groupes hydrophores. Cette fonction est activée par défaut. Les vannes 3 voies sont positionnées pour mettre la sortie des groupes hydrophores en communication avec le réseau de distribution. Une chute de pression dans le réseau de distribution entraîne le démarrage des pompes. Une **commande globale** doit être développée pour assurer un mode d'exploitation automatique. L'opérateur garde la possibilité de configurer le système manuellement pour cette fonction.

La fonction de *chauffage* est générique aux chauffe-eau. Son fonctionnement est complètement automatisé et ne nécessite pas d'intervention de l'opérateur en exploitation.

La fonction de *circulation* est contextuelle dans le sens où elle nécessite une architecture particulière du réseau de distribution d'eau chaude, qui doit former une boucle. Néanmoins son fonctionnement est entièrement automatisé et ne nécessite pas d'intervention de l'opérateur en exploitation.

La fonction de *refroidissement* est contextuelle. Elle met en œuvre l'échangeur thermique E1 et la vanne de régulation V2VR01. Son fonctionnement est automatisé et ne requiert pas d'intervention de l'opérateur en exploitation.

La fonction de *transfert* est une fonction contextuelle qui met en œuvre les vannes du réseau de tuyauterie et les groupes hydrophores. Elle est déclenchée par l'opérateur en supervision qui donne une consigne du volume à transférer. Une **commande globale** doit être développée pour assurer ce mode d'exploitation semi-automatique. L'opérateur garde néanmoins la possibilité de dérouler la fonction manuellement.

La figure 112 résume la liste des commandes globales du système pour le système EdS.

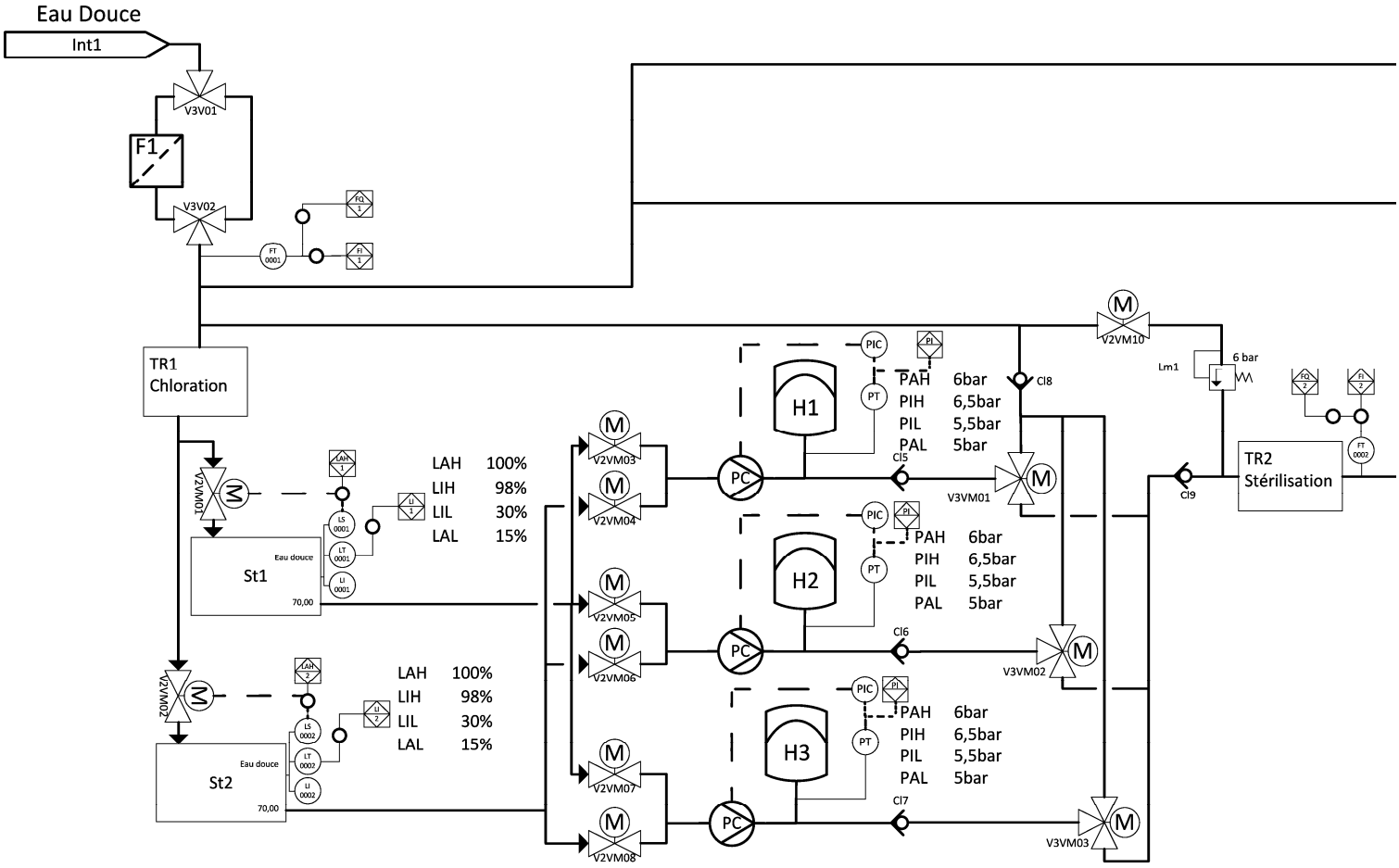
Commande globale	Priorité
Production	Haute
Embarquement	Basse
Débarquement	Basse
Brassage	Moyenne
Distribution	Haute
Transfert	Basse

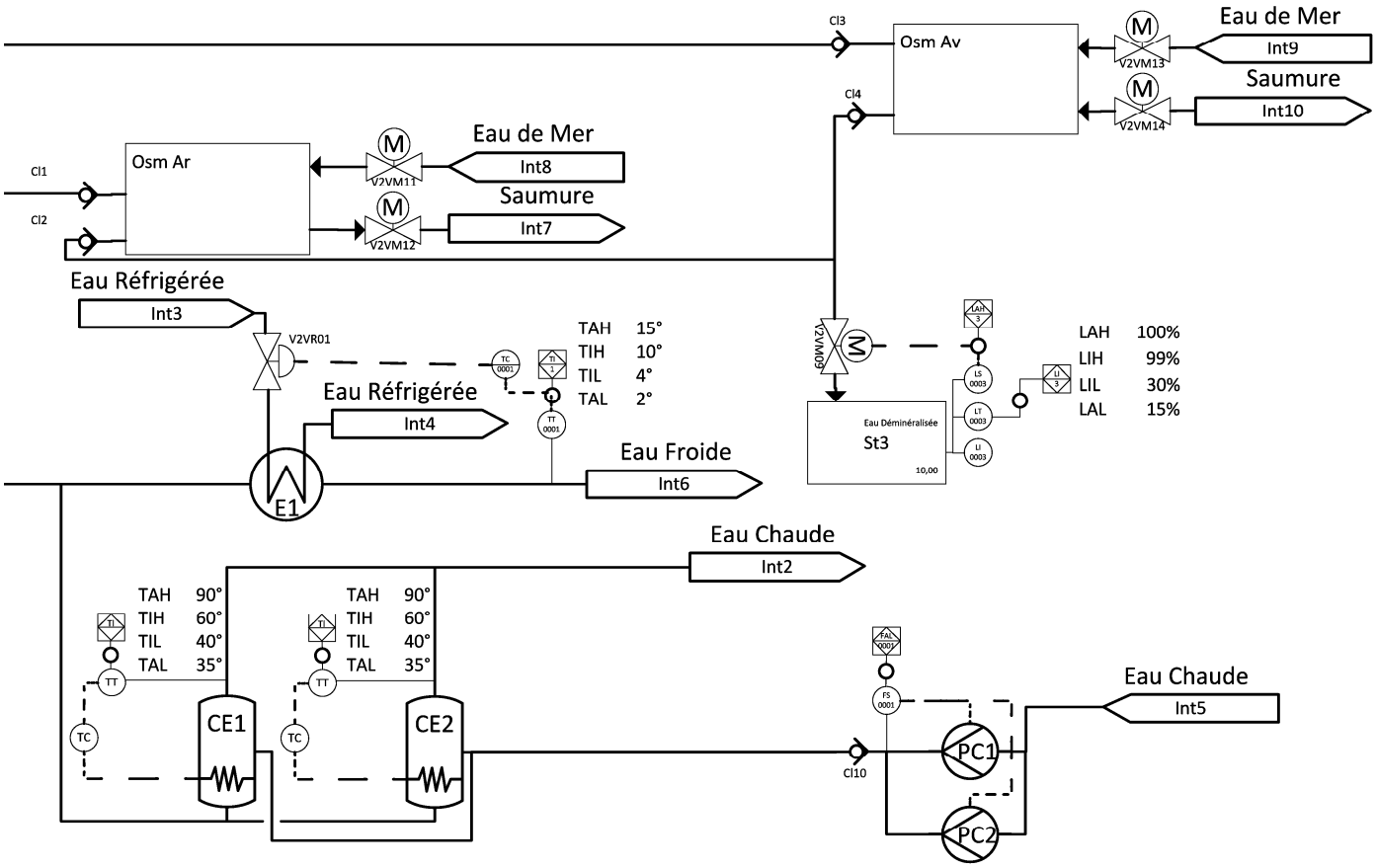
Fig. 112 Les commandes globales du système EdS

Nous associons à chaque commande une priorité relative à l'importance de la fonction réalisée par la commande.

Ainsi les fonctions de production et de distribution qui garantissent la disponibilité de l'eau dans le navire, sont de priorité haute. La fonction de brassage qui permet de maintenir la potabilité de l'eau stockée, est de priorité moyenne. Les fonctions d'embarquement, de débarquement et de transfert qui sont plus éloignées de la raison d'être du système, sont de priorité basse.

5.Synoptique du système EdS




Cl3
Osm Av
Eau de Mer
Int9
V2VM13
Saumure
Int10
V2VM14
Cl4
Cl1
Osm Ar
Eau de Mer
Int8
V2VM11
Saumure
Int7
V2VM12
Cl2
Eau Réfrigérée
Int3
V2VR01
TAH 15°
TIH 10°
TIL 4°
TAL 2°
Eau Réfrigérée
Int4
Eau Froide
Int6
E1
V2VM03
LAH 100%
LIH 99%
LIL 30%
LAL 15%
Eau Déminéralisée
St3
10,00
Eau Chaude
Int2
TAH 90°
TIH 60°
TIL 40°
TAL 35°
CE1
CE2
Cl10
PC1
PC2
Eau Chaude
Int5

6. Liste des éléments du système EdS

Élément	Description	Utilité
V2VM01	Vanne 2 voies motorisée	Isolation remplissage St1
V2VM02	Vanne 2 voies motorisée	Isolation remplissage St2
V2VM03	Vanne 2 voies motorisée	Configuration St1 H1
V2VM04	Vanne 2 voies motorisée	Configuration St2 H1
V2VM05	Vanne 2 voies motorisée	Configuration St1 H2
V2VM06	Vanne 2 voies motorisée	Configuration St2 H2
V2VM07	Vanne 2 voies motorisée	Configuration St1 H3
V2VM08	Vanne 2 voies motorisée	Configuration St2 H3
V2VM09	Vanne 2 voies motorisée	Isolation remplissage St3
V2VM10	Vanne 2 voies motorisée	Isolation distribution depuis le quai
V2VM11	Vanne 2 voies motorisée	Isolation Entrée eau de mer Osm Ar
V2VM12	Vanne 2 voies motorisée	Isolation Sortie saumure Osm Ar
V2VM13	Vanne 2 voies motorisée	Isolation Entrée eau de mer Osm Av
V2VM14	Vanne 2 voies motorisée	Isolation Sortie saumure Osm Av
V3V01	Vanne 3 voies manuelle	Vanne By-pass F1
V3V02	Vanne 3 voies manuelle	Vanne By-pass F2
V3VM01	Vanne 3 voies Motorisée	Configuration Distribution Retour H1
V3VM02	Vanne 3 voies Motorisée	Configuration Distribution Retour H2
V3VM03	Vanne 3 voies Motorisée	Configuration Distribution Retour H3
V2VR01	Vanne 2 voies de régulation	Contrôle du débit d'eau réfrigérée dans E1
PC1	Pompe Centrifuge	Recirculation de l'eau chaude
PC2	Pompe Centrifuge	Recirculation de l'eau chaude
H1	Groupe Hydrophore	Maintient de la pression dans le réseau de distribution
H2	Groupe Hydrophore	Maintient de la pression dans le réseau de distribution
H3	Groupe Hydrophore	Maintient de la pression dans le réseau de distribution
TR1	Module de chloration	Traitement de l'eau stockée
TR2	Module de stérilisation UV	Traitement de l'eau distribuée
Osm Av	Osmoseur	Production de l'eau douce et de l'eau déminéralisée (Avant)
Osm Ar	Osmoseur	Production de l'eau douce et de l'eau déminéralisée (Arrière)
St1	Soute	Stockage de l'eau douce
St2	Soute	Stockage de l'eau douce
St3	Soute	Stockage de l'eau déminéralisée
E1	Échangeur	Maintient en température de l'eau froide
CE1	Chauffe eau	Production et maintient en température de l'eau chaude
CE2	Chauffe eau	Production et maintient en température de l'eau chaude
F1	Filtre	Filtration de l'eau embarquée ou distribuée depuis le quai
Lm1	Limiteur de pression	Protection du réseau de distribution contre les surpressions
Int1	Interface	Connexion du réseau d'eau douce du bord au quai
Int2	Interface	Distribution eau chaude
Int3	Interface	Arrivée eau réfrigérée
Int4	Interface	Départ eau réfrigérée
Int5	Interface	Retour eau chaude pour recirculation
Int6	Interface	Distribution eau froide
Int7	Interface	Rejet saumure
Int8	Interface	Aspiration eau de mer
Int9	Interface	Rejet saumure
Int10	Interface	Aspiration eau de mer
Cl1	Clapet anti-retour	Protection sortie eau douce osm Ar
Cl2	Clapet anti-retour	Protection sortie eau déminéralisée osm Ar
Cl3	Clapet anti-retour	Protection sortie eau douce osm Av
Cl4	Clapet anti-retour	Protection sortie eau déminéralisée osm Ar
Cl5	Clapet anti-retour	Protection H1

Élément	Description	Utilité
Cl6	Clapet anti-retour	Protection H2
Cl7	Clapet anti-retour	Protection H3
Cl8	Clapet anti-retour	Protection ligne de distribution
Cl9	Clapet anti-retour	Protection ligne de distribution
Cl10	Clapet anti-retour	Protection pompes de recirculation
LI0001	Monture de niveau	Affichage mécanique du niveau de St1
LI0002	Monture de niveau	Affichage mécanique du niveau de St2
LI0003	Monture de niveau	Affichage mécanique du niveau de St3
LT0001	Transmetteur de niveau	Mesure analogique du niveau de St1
LT0002	Transmetteur de niveau	Mesure analogique du niveau de St2
LT0003	Transmetteur de niveau	Mesure analogique du niveau de St3
LS0001	Détecteur de niveau	Détection trop plein St1
LS0002	Détecteur de niveau	Détection trop plein St2
LS0003	Détecteur de niveau	Détection trop plein St3
LI1	Mesure de niveau	Affichage en supervision du niveau de St1
LI2	Mesure de niveau	Affichage en supervision du niveau de St2
LI3	Mesure de niveau	Affichage en supervision du niveau de St3
LAH0001	Alarme de niveau haut	Affichage en supervision de l'alarme de niveau haut de St1
LAH0002	Alarme de niveau haut	Affichage en supervision de l'alarme de niveau haut de St2
LAH0003	Alarme de niveau haut	Affichage en supervision de l'alarme de niveau haut de St3
FT0001	Transmetteur de débit	Mesure analogique du débit sur la ligne d'embarquement
FT0002	Transmetteur de débit	Mesure analogique du débit sur la ligne de distribution
FI1	Mesure de débit	Affichage en supervision du débit sur la ligne d'embarquement
FI2	Mesure de débit	Affichage en supervision du débit sur la ligne de distribution
FQ1	Totaliseur de débit	Comptage du volume d'eau embarquée
FQ2	Totaliseur de débit	Comptage du volume d'eau distribuée
FS0001	Détecteur de débit	Détection sous débit dans la ligne de recirculation d'eau chaude
FAL0001	Alarme de débit bas	Affichage en supervision de l'alarme de débit bas dans la ligne de recirculation d'eau chaude

Génération conjointe de commandes et d'interfaces de supervision pour systèmes sociotechniques reconfigurables

La conception de grands systèmes se heurte souvent à des problèmes de communication au sein des équipes de concepteurs et à l'incohérence de certains documents de conception. Nous avons identifié deux approches de conception concurrentes mais complémentaires. L'une est ascendante et voit la conception réalisée par agrégation de composants standardisés, l'autre est descendante et voit la conception réalisée par raffinages successifs d'un modèle de base.

Dans ces travaux, nous proposons un flot de conception intégré, tirant partie des deux approches et permettant la génération conjointe de commandes et d'interfaces de supervision pour un système sociotechnique reconfigurable. Conformément à notre retour d'expérience industriel, notre approche est basée sur un modèle métier appelé synoptique, et sur une bibliothèque d'éléments standards.

Nous proposons également un modèle d'analyse des configurations du système. Il peut être utilisé pour une analyse hors ligne, afin de valider les spécifications de l'expert en charge de la conception. Il peut également être utilisé lors d'une analyse en ligne pour rechercher des configurations alternatives, en cas d'aléas ou de commandes conflictuelles.

Nous présentons enfin l'outil Anaxagore, qui implémente le flot proposé et qui permet de générer, en quelques minutes, un programme de commande et une interface de supervision pour un système de gestion des fluides.

Mots clés : Conception, Contrôle, Commande, IHM, Supervision, IDM, Objet, Patron

Joint generation of controls and user interfaces for reconfigurable sociotechnical systems

The design of large systems suffers from communication problems inside design team and inconsistencies in design documentation. We have identified two concurrent but complementary approaches for design. The first is a bottom-up approach where the design is made by aggregation of standard components. The second is a top-down approach where the design is made by successive refining of a model.

This works offers an integrated design flow for the joint generation of controls and users interfaces for reconfigurable sociotechnical systems. In accordance with our industrial feedback our approach is based on a business model called the synoptic, and on a library of standard elements.

We also offer a model for configuration analysis. It can be used for offline analysis to validate the specification of the expert in charge of the design. It can also be used in an online analysis to find alternative configurations, in case of unexpected or conflicting orders.

Finally, we describe the tool Anaxagore, that implements this approach and with which one can generate, in few minutes, a command and an interface for two simple input models. The results of the generation are consistent with the specifications.

Keywords: Design, Control, Command, HMI, Supervision, MDE, Object, Pattern