Systèmes artificiels programmés : conception et langage remis en question

Artificial programmed systems: design and language put into question

Mascia Antonio
Managing Director, Euro View Services S.A., Informatique Industrielle
Président Hainaut de IBRA (Institut Royal Belge de Régulation et d'Automatisme)
Expert en Informatique Industrielle agréé par l'ABEX (Web: http://www.mas1.be)
273, Chaussée de Lodelinsart, B-6060 GILLY, Belgique
E-mail: management@euro-view.com

Résumé : *Les catastrophes industrielles graves de Seveso, Bhopal, Mil Iland, Tchernobyl, et AZS Toulouse, furent causées soit par des erreurs humaines que les systèmes artificiels n'ont pas pu empêcher soit que les systèmes artificiels eux-mêmes ont provoqué. Pourquoi est-ce arrivé ? Cette présentation propose une approche globale nouvelle pour la conception des automatismes et la création de systèmes artificiels plus sûrs et plus performants. Un langage graphique permet, en une étape au lieu de sept, la conception et la déclaration des séquences (événements), la simulation, l'exploitation et le contrôle-commande d'un processus modélisé en logique booléenne. Ce concept systémique* (Generator of System Programs) *apporte une méthode de conception basée sur la modélisation des événements convertis en équations algébriques neuronales. Toute incohérence ou anomalie logique des processus est simulée de manière préventive ou détectée en production pour éviter toute action inattendue et ce afin d'aider l'opérateur à la prise de décision adéquate.*
Mots clés : Langage, programme, systèmes, sécurité

Abstract : *The heavy industrial disasters at Seveso, Bhopal, Mil Iland, Tchernobyl, and AZS Toulouse, were caused or by human errors that the artificial systems did not been able to prevent or which the artificial systems provoked themselves. Why did it happen? This presentation proposes a new global approach for the design of the automatisms and the creation of artificial systems more performing and safer. A graphic language allows, in one stage instead of seven, the design and declaration of sequences (events), the simulation, the exploitation and the control of a process modelled in Boolean logics. The systemic concept* (Generator of System Programs) *is based on the modelling of the events converted in neuronal algebraic equations. Any logical incoherence or logical abnormality of the processes is simulated in a preventive way or detected in production to avoid any unexpected action and for help the operator in the adequate decision making.*
Keywords : Language, program, systems, safety

1. INTRODUCTION

Cette présentation traite d'automatisation industrielle, en relation avec la Théorie des Systèmes Assistée par Ordinateur (Computer Aided Systems Theory) : CAST [Pichler et Schwärtzel, 1992], qui a permis de réaliser le logiciel prototype GENSYSPRO (Generator of System Programs) [Dubois et Mascia, 1995a, 1995b ; Mascia et Dubois, 1995]. Plusieurs problèmes existent dans les systèmes d'automatisation industrielle, avec l'Automate Programmable Industriel (API) appelé aussi Contrôleur Logique Programmable (en anglais PLC : Programmable Logical Controller).
Nous partons des divers constats suivants:

- Dans le monde, de part la conception même des systèmes artificiels, aucun programme n'est sûr à 100 %.
- De par l'explosion combinatoire de toutes les possibilités d'entrées multiples, il est pratiquement impossible de programmer toutes les possibilités par le manque de mémoire des systèmes artificiels ou encore tout simplement l'impossibilité d'imaginer toutes les possibilités.

- Nos systèmes artificiels programmés actuels fonctionnent en permanence par itérations successives. Le temps d'exécution dépend du temps de lecture du programme, de la manière dont il a été conçu et aussi des contraintes spécifiques imposées par chaque fabricant.
- La même application réalisée par plusieurs programmeurs donnera autant de programmes différents car il n'y a pas une seule manière ou méthode générique universelle.
- Beaucoup d'installations sont décomposées et distribuées à différents intervenants, dont la vue globale échappe souvent à chacun, il arrive parfois qu'une mauvaise coordination a des effets néfastes sur le résultat mais hélas toujours juste avant la mise en exploitation.

Tous ces éléments contribuent à des résultats insatisfaisants et parfois à des catastrophes.
Rappelons-nous les catastrophes très connues (sans compter celles dont on ignore l'existence) comme Seveso, Bhopal, Mil Iland, Tchernobyl, et AZS Toulouse et peut-être aussi celle de Fukushima Daiichi. Comment essayer de les éviter si ce n'est par la prévention et la sécurité ? Il faut donc construire des systèmes plus fiables et plus efficaces pour éviter les erreurs humaines.
C'est dans ce but qu'il est proposé une approche globale nouvelle pour la conception des systèmes artificiels qui gèrent les automatismes industriels.
Le concept GENSYSPRO propose :

- une méthode de conception (déclaration) des séquences d'un processus ;
- un langage graphique qui permet le dessin et la déclaration des séquences (événements) ;
- un simulateur qui permet la détection et la levée des illogismes,
- une modélisation des descriptions par des équations algébriques neuronales;
- une exploitation et un contrôle-commande complet allant du capteur à la supervision des processus.

La méthodologie présentée traite de la conversion de tables booléennes en modèles algébriques permettant la conception de réseaux neuraux. Les tables booléennes sont construites à partir d'un langage graphique représentant les étapes (séquences) d'un processus dans le domaine discret. La méthode de construction des équations discrètes, à partir des tables booléennes décrivant les événements, permet de créer automatiquement des systèmes neuronaux avec des neurones formels de McCulloch et Pitts [1943]. En effet, des équations algébriques non-linéaires peuvent être construites à partir des tables booléennes [Dubois et Resconi, 1993 ; Dubois, 1998, 1999]. Ensuite, ces équations peuvent être utilisées pour produire des réseaux neuronaux grâce aux neurones formels.

2. CONTEXTE INDUSTRIEL DE LA RECHERCHE

Depuis plus de trente ans, les coûts globaux d'automatisation d'une installation n'ont pas diminué alors que les équipements deviennent de plus en plus performants et moins chers.
La plupart du temps, la conception des projets est assez bien réalisée après une analyse sérieuse. Par contre, sur le terrain, on constate qu'il faut toujours autant d'efforts pour mettre au point les programmes d'automatisation afin qu'ils répondent aux exigences demandées. Les coûts de mise en route et les nombreuses mises au point n'ont pas diminué.
Voici trois raisons importantes :

- Il n'y a aucune méthode pour programmer un système. Après l'analyse du processus, chaque informaticien peut concevoir les programmes comme il le désire. Malgré des normes, il existe d'innombrables outils logiciels différents pour la programmation des systèmes.
- Pour les fabricants d'automatismes, bien que le principe fonctionnel général soit le même, les syntaxes et structures des programmes de bas niveau sont tous différents.
- La partie comportementale de l'installation est rarement simulée et validée préalablement à la mise en œuvre. La simulation prend du temps et des moyens trop coûteux (reprogrammation).

Dans l'industrie, il n'y a aucun droit à l'erreur, aux bogues ou bugs, sous peine de danger. La machine doit faire strictement ce qui est attendu et sans aucune place à l'incertitude. Imaginez l'impact dans le monde de l'industrie ou dans notre vie quotidienne, de machines incontrôlables ! Cà fonctionne, mais avec quels risques "acceptables" ? Tant qu'il n'y a pas d'accidents, les systèmes artificiels sont bons, mais pas encore assez, car ils sont faillibles et donc perfectibles.

3. LES SYSTÈMES ARTIFICIELS D'AUTOMATISATION

Le principe général de fonctionnement des systèmes d'exploitation des Automates Programmables Industriels (API), ou Programmable Logical Controller en anglais (PLC), est le suivant :

- lire les entrées physiques,
- mémoriser les entrées,
- exécuter le programme (avec les valeurs mémorisées et figées),
- rafraîchir la mémoire de sortie,
- mettre à jour les sorties physiques
- exécuter d'autres taches qui s'insèrent dans le cycle du processeur, comme par exemple la communication, les tâches rapides prioritaires, les tâches spéciales, etc.

Ces tâches se répètent à chaque cycle (en récursion permanente) qui dure un temps "dt" qui varie en fonction de nombreux paramètres comme le nombre de lignes à lire, la puissance du processeur ou le nombre d'entrées / sorties, etc. On pourrait dire que les performances et la fiabilité fonctionnelle des systèmes dépendent principalement de la puissance et de la vitesse d'exécution du processeur.
Pas seulement, cela dépend de beaucoup d'autres paramètres comme la qualité de conception physique et logicielle des systèmes, de la vitesse de lecture des entrées physiques, du temps d'échange entre la partie physique et la mémoire du système, de la manière dont le programme a été créé par l'automaticien, etc. Tous ces éléments contribuent aux faiblesses des systèmes car ils poussent à la course aux performances. La puissance n'est qu'un palliatif essayant de combler les imperfections de conception.

4. LA PROGRAMMATION DES AUTOMATISMES

Aucun programme informatique n'est sûr ou complet à 100%, on l'a déjà dit.
Le nombre de paramètres donne une explosion combinatoire telle qu'il n'est pas possible d'imaginer toutes les combinaisons d'équations et encore moins de les programmer dans des systèmes généralement limités en capacité mémoire. Imaginez seulement deux essais qui sont rarement réalisés et pratiquement jamais à toutes les étapes d'un processus industriel en fonctionnement, en fait, à tout moment.

- Mettre à 1 (court-circuit), de manière accidentelle et à tout moment, toutes les valeurs d'entrées du système. Exemple potentiel : une pelle mécanique qui écrase un câble dont tous les fils seraient mis en contact (en court circuit).
- Mettre à 0 (coupure), de manière accidentelle et à tout moment, toutes ou une partie des entrées ou sorties du système, ce qui aura pour effet de rendre les feedback (rétroactions) complètement aléatoires. Exemple potentiel : une pelle mécanique qui arrache un câble. Tous les fils seraient sectionnés ou en cas de panne de courant sur les entrées uniquement où tout serait mis à 0.

Qui peut prédire pour chaque instant et à chaque ligne du programme ce qu'il va se passer dans les deux cas de figure cités ci-dessus? Personne ! Heureusement que ces deux types d'incidents arrivent rarement. L'argument complémentaire pour proposer une autre voie de recherche est que sur l'ensemble des programmes d'automatisation, 20 à 30% servent à réaliser les fonctions normales attendues et le reste sert aux autres fonctions qui gèrent les défaillances éventuelles externes au programme qui auraient été prévues. En effet, dans les programmes, il y a deux fonctions principales :

- les séquences du fonctionnement normal ;
- les séquences de replis ou modes dégradés pour gérer les défauts ou gérer les incidents.

La pratique montre que la plupart des programmes réalisent leurs fonctions avec assez bien de réussite, mais ce que l'on ne voit pas ce sont tous les incidents évités par la présence et les facultés de raisonnement de l'homme sur le terrain, qui réagit et corrige. Si l'homme commet une erreur et intervient au mauvais moment, non prévu par le programme, le système peut en être perturbé. On ne sait pas toujours comment il va réagir. Pour chaque intervention humaine suite à un incident, le système peut : accepter l'ordre, refuser l'ordre ou mal l’interpréter et réagir d'une mauvaise manière.

Un accident dépend souvent d'un cas non étudié, non pensé ou non programmé, donc imprévu ou imprévisible. Suite à ces situations insoupçonnées, improbables, voire déclarées impossibles par d'aucun, c'est l'accident ou la catastrophe. Là, c'est trop tard.

5. PROCESSUS DE REALISATION D'UNE AUTOMATISATION

SANS GENSYSPRO : dans le processus classique de réalisation d'un projet, différentes étapes permettent d'exprimer et de définir les objectifs et les moyens pour mener à bonne fin un projet.

1. L'analyse conceptuelle : traduction verbale vers un écrit (cahier général des charges : CdC)
2. L'appel d'offres et choix : traduction du CdC en coûts, avec la recherche d'adéquation technico-économique des fournisseurs et équipements à mettre en oeuvre.
3. L'analyse fonctionnelle : traduction des idées en plans électriques et schémas logiques.
4. La programmation : traduction en langages de praticiens (programmes aux syntaxes variables suivant les systèmes ou fabricants).
5. Les tests statiques à blanc ou par simulation : programmes spéciaux (rarement réalisés).
6. Les tests et mise en route sur site : tests dynamiques.
7. La supervision du processus : réalisation de son image animée sur écran via une connexion permanente en temps réel. Avec le superviseur, l'opérateur visualise l'installation et peut interagir sur elle par des commandes.

Que de chemin parcouru par l'information via des traductions successives de langages différents ! Il est souvent constaté qu'après chaque traduction, on perd, on déforme et on transforme les données du problème posé au départ. Ceci est normal et fait partie du processus, mais c'est aussi la cause des coûts qui ne baissent pas depuis si longtemps.

Pour améliorer l'ensemble de ce processus de conception, les performances ainsi que la sécurité des systèmes industriels, il faut :

- réduire les coûts globaux d'une installation d'automatisation ;
- réduire le nombre d'étapes, le nombre d'outils et le temps de conception des automatismes ;
- créer des programmes et des systèmes artificiels plus sûrs et plus performants ;
- créer un langage unique pour les opérations de conception, de simulation, de programmation, de contrôle-commande des systèmes artificiels, la surveillance et le pilotage des processus industriels ;
- augmenter les capacités des systèmes artificiels, la mémoire et la vitesse d'exécution ;
- définir une méthode de conception unique, pour avoir des programmes univoques et une formation unique.

AVEC GENSYSPRO : il n'existe pas encore aujourd'hui une seule méthode universelle de travail pour réaliser ces différentes étapes de mise en oeuvre et pour la programmation des systèmes. Il en est de même pour les différents outils nécessaires aux sept étapes qui seraient rassemblés en un seul outil. C'est ce que propose le concept général du logiciel GENSYSPRO (Generator of System Programs) qui est un générateur de programmes pour des systèmes artificiels qui possède trois fonctions principales :

1. dessiner et déclarer le processus à automatiser,
2. simuler et valider le processus ainsi que lever les illogismes,
3. superviser et contrôler le processus réalisé,

Le cadre de travail de GENSYSPRO contient une partie classique, comme tous les logiciels : son menu "ouvrir projet, fermer, nouveau, sauver, supprimer, ajouter, quitter, etc. Plusieurs modes d'utilisation sont possibles dont chacun représente un outil en soi :

1. Mode dessin (graphisme et paramétrisation des objets dynamiques)
2. Mode base de données "Orienté Objet"
3. Mode description des séquences (description fonctionnelle)
4. Mode simulation (instantanée avec visualisation des incohérences)
5. Mode exploitation en liaison temps réel avec le processus (non encore développé)

L'outil GENSYSPRO est un outil « tout en un » qui possède une méthode et un nouveau langage de création des automatismes. Via un exemple industriel, la description détaillée de GENSYSPRO vous est présentée ci-dessous.

6. LA MÉTHODE GENSYSPRO

Il faut d'abord rappeler deux aspects du concept :

- La description des séquences d'un processus est strictement du domaine discret (binaire)
- Le langage est graphique

La méthode GENSYSPRO repose sur deux points fondamentaux :

- Ne déclarer que ce que vous attendez du système car tout ce qui ne sera pas décrit sera réfuté et non exécuté.
- Décrire le processus non pas sur un classique aspect temporel mais sur un aspect purement séquentiel basé sur les événements. Chaque étape dépendra d'un changement attendu des états des d'entrées (paramètres).

7. EXEMPLE INDUSTRIEL DU "WATER SUPPLY"

Considérons une installation d'alimentation d'une usine en eau à partir d'une nappe phréatique disponible.

Un interrupteur marche/arrêt pilote l'installation "ST/SP" (start/stop), une cuve tampon "Tank" sera équipée d'un détecteur de niveau bas "Low" et d'un détecteur de niveau haut "High" qui piloteront deux pompes Pump1 qui remplit la cuve et Pump2 qui vide la cuve pour alimenter l'usine.

- La pompe Pump1 qui alimente la cuve est plus puissante que la pompe Pump2 qui alimente l'usine, donc on attendra le niveau bas de la cuve pour remettre en marche la pompe Pump1.
- Il s'agit ici de réaliser la description séquentielle du processus en mode graphique.
- L'objectif fonctionnel est le projet "WS" la fonction de mise en marche appelée "Started".

La Figure 1, montre le schéma de l'installation en mode marche (ST=1 et Pump1 en marche) pour la première fois cuve vide.

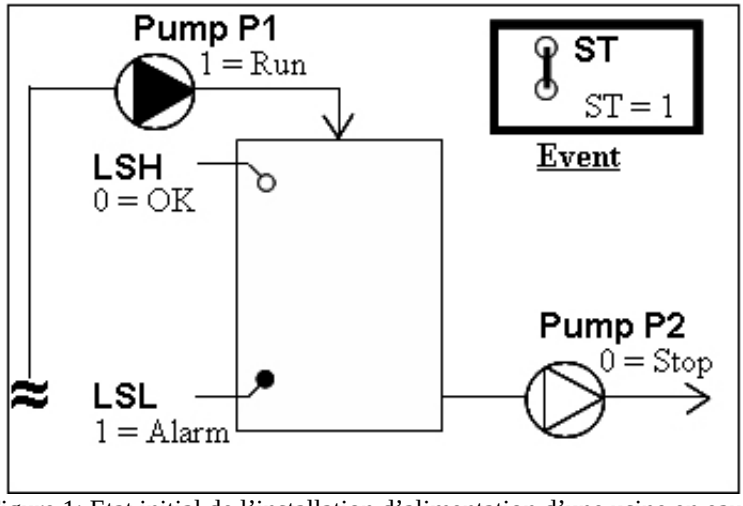

Figure 1: Etat initial de l'installation d'alimentation d'une usine en eau à partir d'une nappe phréatique (extrait de Dubois et Mascia, 2010)

Dans cette fonction, on définit une séquence de 4 étapes. Chaque étape correspond à un et un seul événement physique du processus :

1. k=1 - Etape 1 = remplir, l'événement = Niveau bas "Low Level"
2. k=2 - Etape 2 = remplir et vider, l'événement = Niveau normal "Normal Level"
3. k=3 - Etape 3 = arrêt remplir et vider, l'événement = Niveau haut "High Level"
4. k=4 - Etape 4 = vider, l'événement = Niveau normal "Normal Level"

L'étape 5 n'existe pas car elle devient alors l'étape 1 en boucle.
Le premier démarrage se fait en niveau bas : "Started" "Low Level", cuve vide (Figures 1 et 2).

MODE 1 - DESSIN
La réalisation graphique passe par les opérations suivantes :

- Création de l'objet "Projet" (la base de données est vide). Cette base de données est réalisée directement par instanciation orientée objet.
- Création de l'objet "Fonction" (Ajouter un objet "Fonction" et donner un nom à l'objet WS (Water Supply). La base de données commence à se créer à partir du niveau 2 (projet = niv. 1)
- Développement de l'objet "WS" - Figure 2 (Projet : WS).
- Description graphique de l'application via un objet " Fonction"

On devrait appeler les variables WS.WS, pour simplifier, on ne tient pas compte du projet "WS".

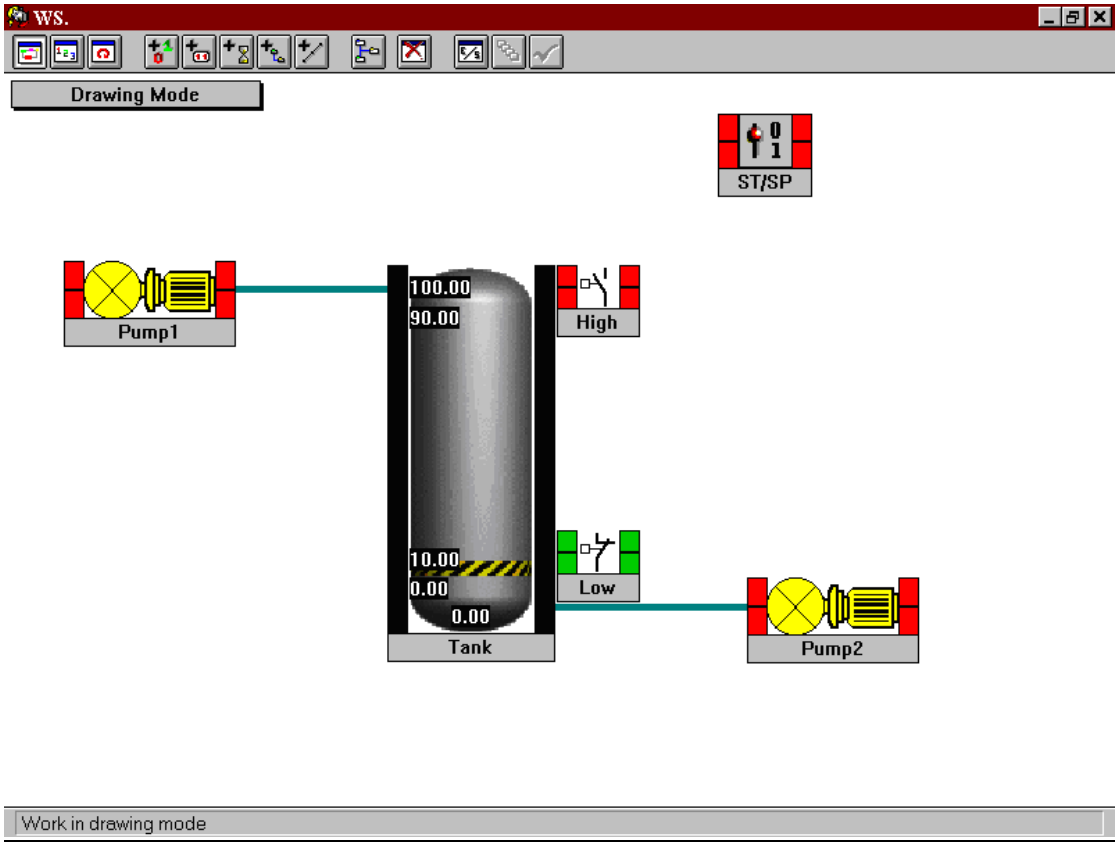

Figure 2 : dessin complet des objets de l'application

On réalise le dessin de l'application avec tous les objets : objets BIT associés à chaque image : WS.Low - WS.High - WS.ST/SP - WS.Pump1 - WS.Pump2.
L'objet valeur numérique du niveau de cuve est paramétré avec l'échelle, l'unité et les valeurs d'incrémentation et de décrémentation pour la simulation. Un troisième objet qui n'a pas été utilisé ici, le délai, le retard après une action (temporisation).

MODE 2 - BASE DE DONNÉES ( )
A partir de la création d'un projet et pour chaque description fonctionnelle, c'est le positionnement qui définit le nom de la variable de la base de données. Les noms deviennent une combinaison des instanciations successives.
Chaque fois qu'une variable est créée pour la première fois, c'est-à-dire que son nom simple "High" par exemple, il est généré un nom combiné qui représente une instanciation dans l'espace projet en cours, dans ce cas "WS" et cette variable devient "WS.High".
Les variables simples et les variables complexes.

Le projet "WS" est une variable complexe qui contient la déclaration du processus : fonction "WS".
Rappelons que GENSYSPRO possède trois objets de base :

- un objet bit
- un objet variable numérique
- un objet de temporisation

Chaque partie graphique (dessin) peut rester de type statique, ou peut être associé à un objet de base de type bit, variable numérique ou de temporisation.
Chaque objet devient par son nom une variable de la base de données de type "orienté objet".
Chaque objet possède un ensemble d'attributs paramétrables qui définiront trois parties aux objectifs différents :

- les informations statiques du dessin,
- les informations de lien avec la base de données,
- les valeurs de la partie comportementale en mode simulation ou en mode exploitation.

La figure 3 représente la base de données structurée en mode "orienté objet"

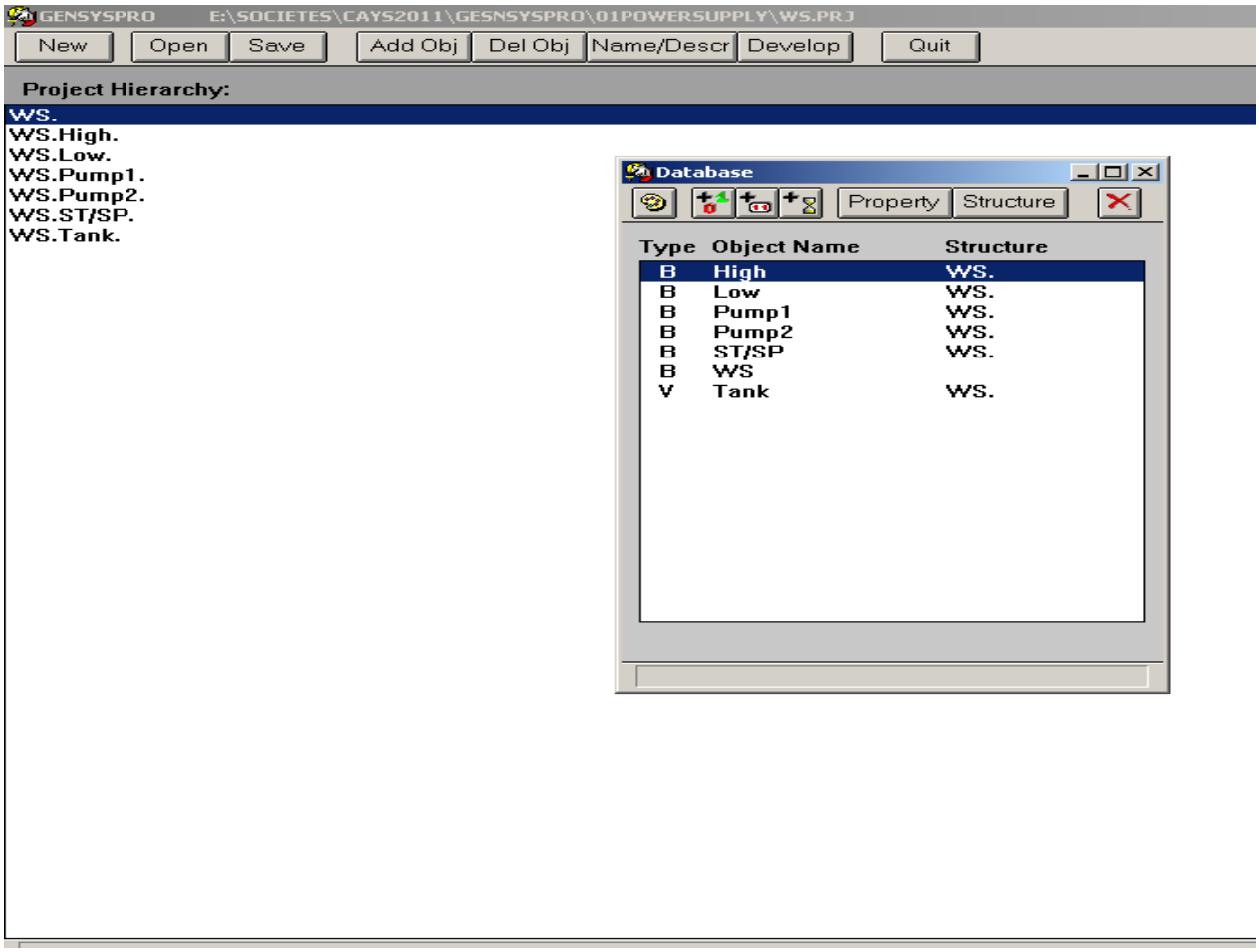

Figure 3 : base de données de l'application "WS"

MODE 3 - DESCRIPTION DES SÉQUENCES ( )

La figure 4 ci-dessous montre l'outil en mode de définition des séquences du processus.

- Click sur ( ) pour ajouter un objectif fonctionnel appelé "Started" du projet WS
- Numéro de l'étape = 1
- Le nom de l'étape est "Low Level"
- Configurer cette étape 1 Niveau bas
- Mettre chaque état dans la position 1 ou 0 suivant la valeur attendue dans cette étape physique "Niveau bas". Ce sont les valeurs reprises dans la table booléenne.

La couleur des états est modifiable. On ne s'occupe pas du tout de positionner la valeur numérique du niveau d'eau car elle dépendra soit de la simulation, soit de la valeur physique réelle.
k=1 Niveau bas = Remplir la cuve (Figure 4)
ST/SP = 1 = Vert
Low = 1 = Vert
Pump1 = 1 = Vert
High = 0 = Rouge
Pump2 = 0 = Rouge
On réalisera ainsi les 3 étapes suivantes de la séquence en donnant le nom et en positionnant la bonne valeur binaire à chaque étape suivant la table booléenne pour k=2, k=3 et k=4.

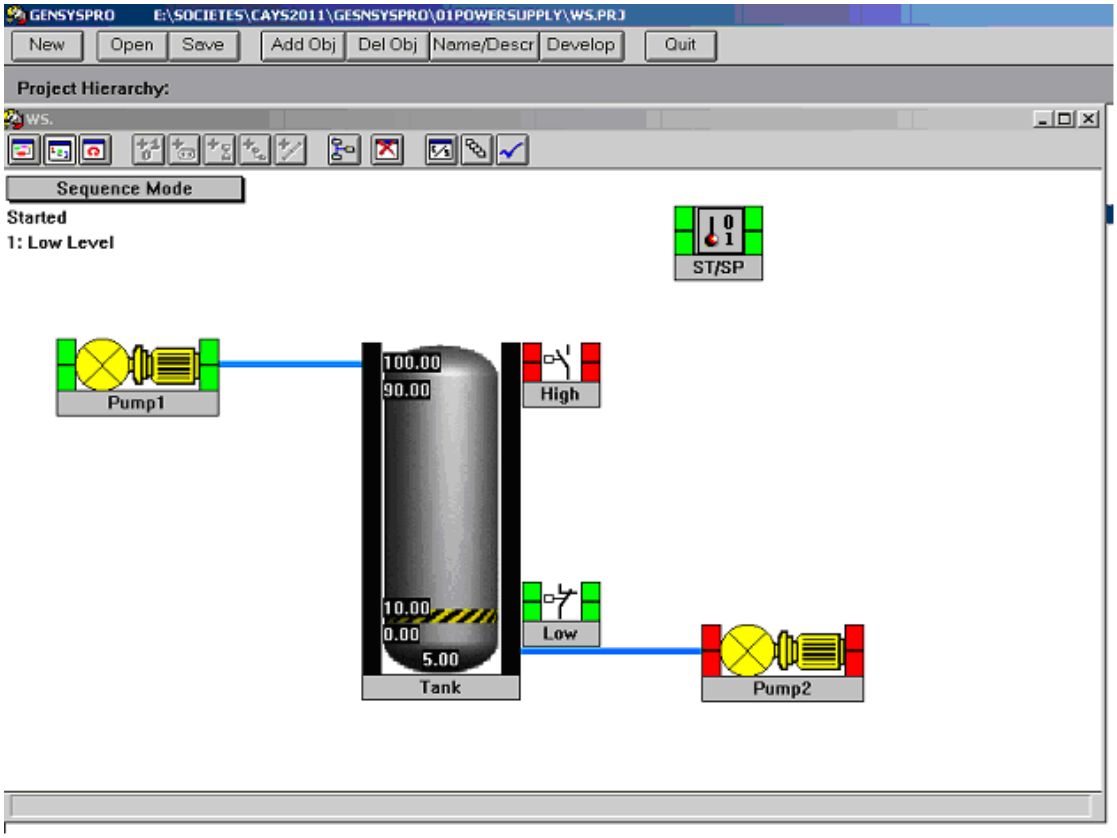

Figure 4 : description de la séquence à l'étape 1

La Table 1 donne la table booléenne du cycle des événements des 4 étapes décrites ci-dessus.

	Table 1	**Entrées**			**Sorties**	
Etape k	**Niveaux**	**ST**	**LSL**	**LSH**	**P1**	**P2**
1	**Low level**	1	1	0	**1**	**0**
2	**Normal level**	1	0	0	**1**	**1**
3	**High level**	1	0	1	**0**	**1**
4	**Normal level**	1	0	0	**0**	**1**

Par la suite, la construction graphique des séquences permet la représentation matricielle en table 2 ainsi que sa modélisation en équations algébriques ou en réseaux de neurones en figure 8.

LA PARTIE COMPORTEMENTALE
Simuler une partie opérative nécessite de connaître le temps de fonctionnement de chaque actionneur physique d'une installation. Il s'agit de connaître par exemple : le temps d'ouverture ou de fermeture, le temps de marche ou d'arrêt des équipements ou encore le temps entre deux événements d'un processus comme arriver à la bonne température ou à la bonne pression.

Ces temps doivent donc être connus et intégrés par l'outil pour être simulés, GENSYSPRO le permet. Si ces temps de fonctionnement sont inconnus, ils peuvent être imaginés et c'est lors de l'exploitation en temps réel qu'ils seront mesurés et mémorisés. Il en est de même pour les temps d'incrémentation et de décrémentation des valeurs numériques qui permettent de simuler le niveau d'eau dans notre exemple. Lors de la mise en exploitation, la valeur du niveau sera donnée alors par le capteur en valeur réelle du processus.
Dans la figure 5 ci-dessous représente une fiche de configuration de la partie comportementale d'un objet numérique qui dans notre exemple, devient le niveau d'eau.

Configuration de l'objet "TANK" (objet valeur numérique)
Quelque soit la valeur à présenter ou à simuler, il faut y mettre :

- L'échelle : 0-100
- L'unité : m^3
- Valeur d'incrément = 3 (m3)
- Pas d'incrément = 200 ms (temps de simulation)
- Valeur de décrément = 1 (m3)
- Pas de décrément = 200 ms (temps de simulation)
- Le bit de fonctionnement de l'incrémentation = ws.pump1
- Le bit de fonctionnement de la décrémentation = ws.pump2

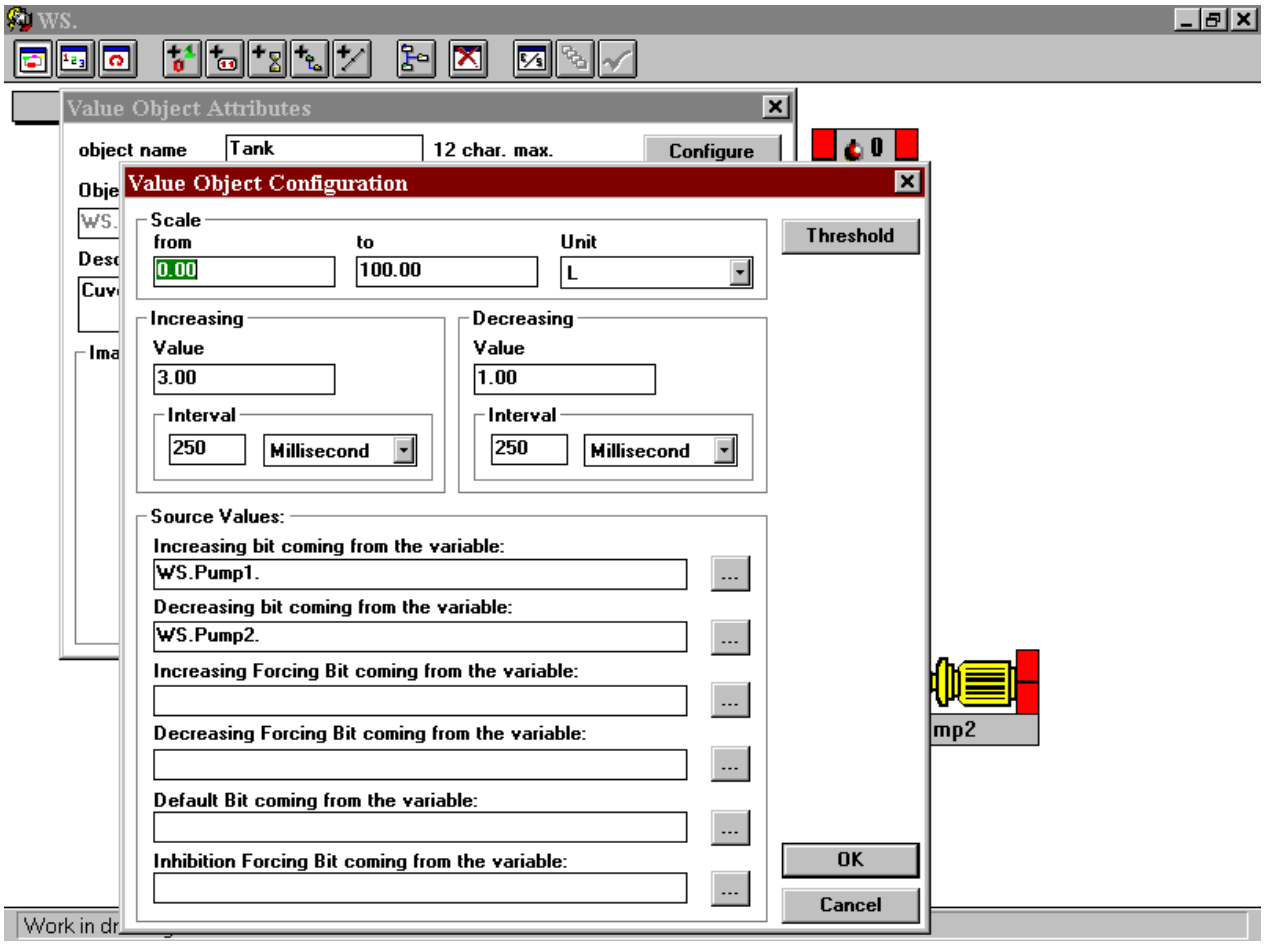

Figure 5 : configuration de la partie comportementale de l'objet niveau d'eau

MODE 4 - SIMULATION ( )
La figure 6 montre l'outil en mode simulation qui permet d'actionner (forcer) les valeurs pour tester le fonctionnement complet du processus déclaré.
Le mode simulation est instantané sans aucune programmation.

Pour que la simulation s'exécute, il faut positionner l'état ST/SP à 1 = marche et le niveau en position "Low" = 1 soit en état d'alarme (contact fermé) tel que décrit dans la table 1 ci-dessus. Les autres états à 0.
Voici un cas de discordance. Après le démarrage en étape 1, la cuve commence à se remplir, le niveau devient normal. A l'étape 2, au lieu d'avoir le niveau haut attendu LSH=1, il apparaît le défaut, du niveau d'eau bas LSL=1. Cela provoque immédiatement un arrêt des 2 pompes et une alarme de discordance sur l'état "Low" à l'étape 2 nommée Niveau normal "Normal Level". L'état "Low" est en couleur d'alarme de discordance "vert-rouge-jaune", ceci permet non seulement un dépannage rapide mais aussi une sécurité immédiate par le refus d'exécution des sorties. Si on veut donner l'alarme et garder les états en cours, par exemple les pompes en service, il suffit de l'indiquer dans la configuration du système.

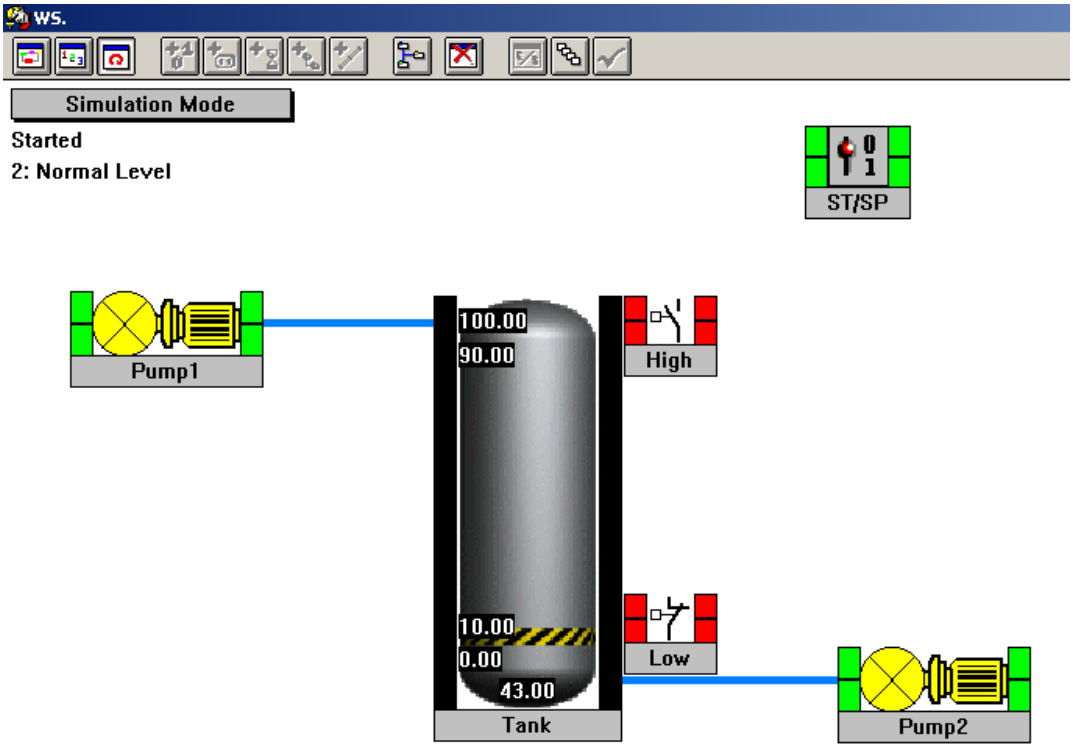

Figure 6 : la simulation du processus se déroule automatiquement et se trouve en étape 2

Pour ce mode très particulier, il est nécessaire d'émettre quelques remarques :

- Chaque objet "projet", "fonction" ou bit simple ont les mêmes propriétés
- GENSYSPRO permet la création graphique, la création de la base de données, la simulation du processus et sa validation, la création automatique des équations algébriques ou réseaux de neurones que l'on peut envoyer dans un système d'exploitation pour exécution. Relié en temps réel avec ce système, GENSYSPRO devient alors le superviseur / contrôleur de l'application.
- Dans GENSYSPRO, on peut intégrer la partie comportementale en paramétrant tous les temps de réponse de tous les équipements du processus comme par exemple le temps d'ouverture ou de fermeture d'une vanne, etc.
- Un ensemble de possibilités de configuration permet des choix fonctionnels adaptés aux différents modes d'utilisation parfois très différents comme par exemple entre l'industrie et la production électrique où l'inversion des couleurs vert/rouge ou encore continuer en cas de défaut au lieu de l'arrêt.

Remarques importantes :

- Un seul langage graphique et en une seule étape suffit pour concevoir, simuler et valider un processus séquentiel sur base d'évolutions binaires.

- Toutes les séquences seront décrites comme un film où chaque image correspond à un changement, à un événement binaire.

MODE 5 - EXPLOITATION

Bien que ce mode ne soit pas encore développé, il va de soi que cette dernière étape sera la plus intéressante car elle permettra de boucler les sept phases de conception avec le même outil. Le mode d'exploitation représente l'outil de supervision des installations appelé aussi en anglais SCADA (Supervisory Control And Data Acquisition). La supervision consiste à visualiser l'installation pour réaliser son suivi et son contrôle/commandes. Avec GENSYSPRO, tout redessiner n'est plus nécessaire et les variables d'entrées (objets d'état binaire) et de sorties (commandes des objets pompes) ainsi que la variable numérique du niveau du tank seront rafraîchies par les données réelles venant du processus. Pour la conduite du processus, l'opérateur doit disposer d'une série d'outils complémentaires.

Citons quelques-uns de ces outils :

- Pour simuler le processus, l'opérateur doit pouvoir décider d'une marche manuelle, automatique ou semi automatique et définir d'un temps de pas à pas suffisant à une analyse.
- Pouvoir simuler les valeurs numériques en fonction de paramètres variables.
- Liaison complète à toutes les variables du processus : états binaires (st/sp), commandes binaires (Pump1), valeurs de mesures numériques (niveau), valeurs de commandes numériques (position d'une vanne) qui n'est pas utilisé dans l'exemple ci-dessus.
- Pouvoir lier ou non chaque variable au processus, forcer des valeurs, forcer la désactivation des fonctions ou encore inhiber ou sauter les pas des séquences.
- A chaque coupure de courant, il faut pouvoir mémoriser des données et donc décider lesquelles.
- A chaque retour sous tension, les système doit se resynchroniser avec le processus, on doit pouvoir reconnaître automatiquement tous les états du processus, les analyser et les comparer aux fonctions pré établies. Proposer alors à l'opérateur le choix univoque ou lui demander de choisir en lui présentant les choix qui ne sont pas univoques et l'aider à la décision.
- A chaque redémarrage, le système doit reconnaître les fonctions actives ou non, les étapes actives ou non mais, s'il y a doute, le système propose les choix à l'opérateur qui peut alors agir sur les différents paramètres de démarrage (fonctions, séquences, étapes et valeurs).
- A chaque incohérence, discordance ou disfonctionnement, l'opérateur est averti immédiatement de la fonction, des étapes et/ou des paramètres en cause, ce qui permet la prévention des accidents et une maintenance préventive efficace.

Toutes ces fonctionnalités sont réparties en deux endroits :

- dans le système GENSYSPRO qui fonctionne sur un ordinateur bureautique avec les outils de supervision et de contrôle.
- dans les systèmes industriels d'automatisation Contrôleur Logique Programmable (PLC).

La construction et le fonctionnement des systèmes industriels actuels doivent être revus complètement pour arriver aux résultats de sécurité et de performances espérées par cette recherche.

LA MAINTENANCE PRÉDICTIVE ET L'ANTICIPATION DES RISQUES

Nous pouvons également mémoriser le déroulement d'un processus de référence avec tous les temps des séquences et étapes (temps entre chaque événement). Comparer ensuite en permanence les temps de référence avec le temps réel du processus en cours. On peut ainsi déterminer toutes les dérives des capteurs et actionneurs (pannes, usure ou vieillissements). Cette fonctionnalité correspond à une maintenance prédictive et anticipative qui prévient les risques.

8. LE MODÈLE ALGÉBRIQUE DU "WATER SUPPLY"

Afin de pourvoir transformer la description graphique en modèles algébriques neuronaux, il faut passer par une traduction des états binaires des séquences en matrice booléenne.

Chaque fonction comporte une séquence d'étapes. Comme pour un film, chaque étape de la séquence correspond à événement. Chaque événement est une photo instantanée du processus (image graphique) où chaque état binaire est dans sa position 0/1 de la description fonctionnelle. Rappelons les 4 étapes de la séquence :

- k=1 - Niveau bas "Low Level"
- k=2 - Niveau normal "Normal Level"
- k=3 - Niveau haut "High Level"
- k=4 - Niveau normal "Normal Level"

Ces 4 étapes sont représentées ci-dessous en figures 7 A-B-C,D

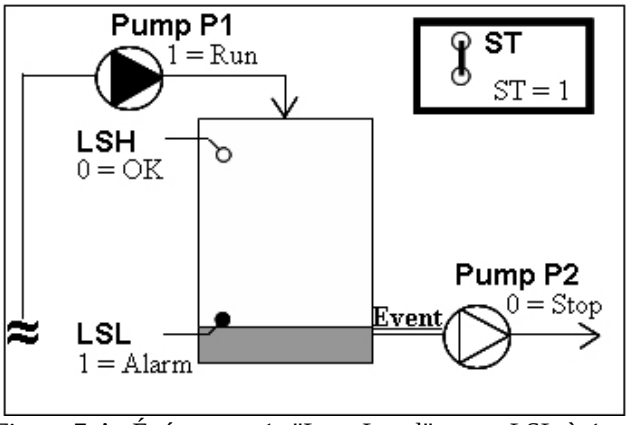

Figure 7-A: Événement 1, "Low Level", avec LSL à 1 = Alarm, et la pompe P1 à 1 = Run (extrait de Dubois et Mascia, 2010).

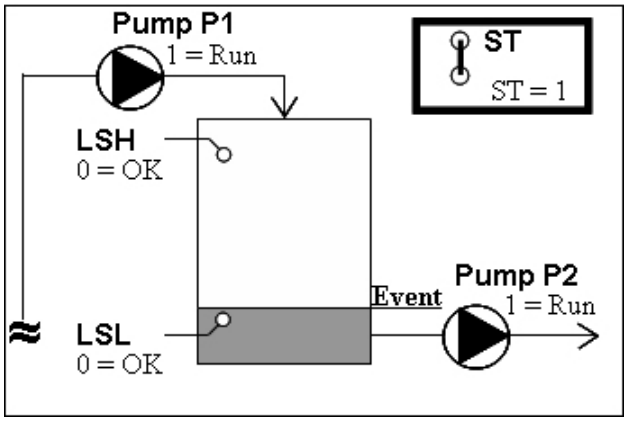

Figure 7-B: Evénement 2, "Normal Level", avec LSL à 0 = OK, et la pompe P2 à 1 = Run (extrait de Dubois et Mascia, 2010)

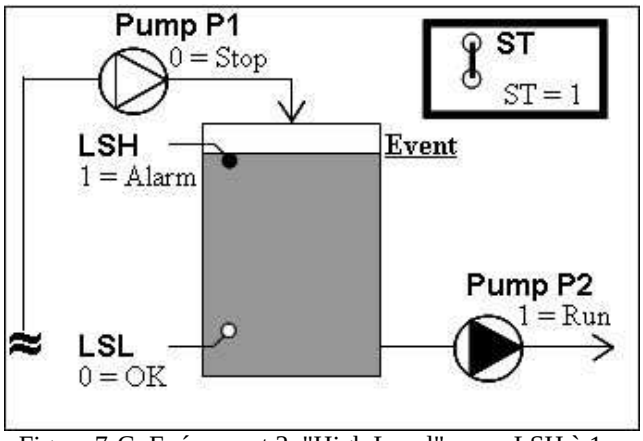

Figure 7-C: Evénement 3, "High Level", avec LSH à 1 = Alarm, et la pompe P1 à 0 = Stop (extrait de Dubois et Mascia, 2010)

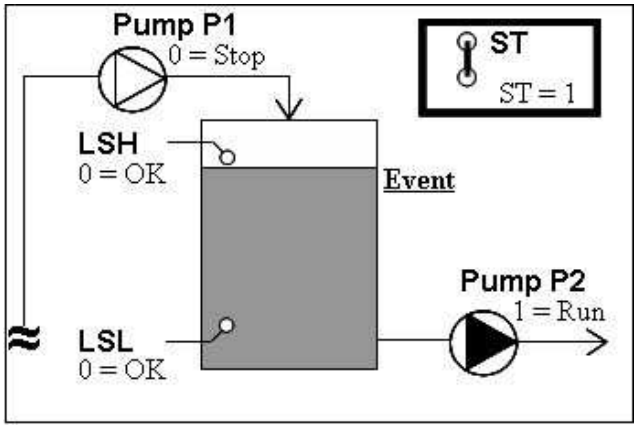

Figure 7-D: Evénement 4, "Normal Level", avec LSH à 0 = OK (extrait de Dubois et Mascia, 2010)

La Table 2 donne la table booléenne du cycle des événements correspondant aux Figures 7-ABCD.

Table 2: Table logique du cycle des événements du "water supply"

	Table 2	**Inputs**					**Outputs**	
Step k	**Levels**	**ST**	**LSL**	**LSH**	**LSL(k–1)**	**LSH(k–1)**	**P1**	**P2**
1	**Low level**	1	1	0	0	0	**1**	**0**
2	**Normal level**	1	0	0	1	0	**1**	**1**
3	**High level**	1	0	1	0	0	**0**	**1**
4	**Normal level**	1	0	0	0	1	**0**	**1**

Voici les équations algébriques du "water supply" pour chaque sortie P1 et P2 (d'après Dubois et Mascia, 2010):

$P1 = ST.LSL.(1-LSH).(1-LSL(k-1)).(1-LSH(k-1))$

$$+ ST.(1-LSL).(1-LSH).LSL_{(k-1)}.(1-LSH_{(k-1)}) \quad (1a)$$

$$P2 = ST.(1-LSL).(1-LSH).LSL_{(k-1)}.(1-LSH_{(k-1)})$$

$$+ ST.(1-LSL).LSH.(1-LSL_{(k-1)}).(1-LSH_{(k-1)})$$

$$+ ST.(1-LSL).(1-LSH).(1-LSL_{(k-1)}).LSH_{(k-1)} \quad (1b)$$

9. CONCEPTION DU RÉSEAU NEURONAL DU "WATER SUPPLY"

Le réseau neuronal est réalisé à partir du modèle de neurone formel de McCulloch and Pitts [1943], définit comme suit : $y = \Gamma(w_1.x_1 + w_2.x_2 + w_3.x_3 + ... - \theta)$ où les x_1, x_2, x_3, …, sont les valeurs d'entrée du neurone formel, y, la valeur de sortie du neurone formel, les w_1, w_2, w_3, …, sont les poids synaptiques, θ est le seuil, et Γ est la fonction d'activation du neurone représentée par la fonction de Heaviside définie par $\Gamma(x) = 0$ si $x \leq 0$ et $\Gamma(x) = 1$ si $x>0$.

A partir des équations algébriques 1ab, les équations d'un réseau neuronal du "Water Supply" a été construit (voir les détails de la méthode dans Dubois et Mascia, 2010). Les neurones formels d'entrée sont identifiés par les variables suivantes:

$ST, LSL, LSH, LSL_{(k-1)}, LSH_{(k-1)}$

Les fonctions de Heaviside des neurones formels cachés pour la sortie P1 sont données par

$$H11 = \Gamma(ST + LSL - LSH - LSL_{(k-1)} - LSH_{(k-1)} - 1) \quad (2a)$$

$$H12 = \Gamma(ST - LSL - LSH + LSL_{(k-1)} - LSH_{(k-1)} - 1) \quad (2b)$$

Les fonctions de Heaviside des neurones formels cachés pour la sortie P2 sont les suivantes :

$$H21 = \Gamma(ST - LSL - LSH + LSL_{(k-1)} - LSH_{(k-1)} - 1) \quad (3a)$$

$$H22 = \Gamma(ST - LSL + LSH - LSL_{(k-1)} - LSH_{(k-1)} - 1) \quad (3b)$$

$$H23 = \Gamma(ST - LSL - LSH - LSL_{(k-1)} + LSH_{(k-1)} - 1) \quad (3c)$$

Les fonctions de Heaviside des neurones formels de sortie, P1 et P2, sont

$$P1 = \Gamma(H11 + H12) \quad (4a)$$

$$P2 = \Gamma(H21 + H22 + H23) \quad (4b)$$

La figure 8 montre le réseau neuronal de ce "Water Supply", correspondant aux équations 2ab, 3abc et 4ab des neurones formels de McCulloch and Pitts .

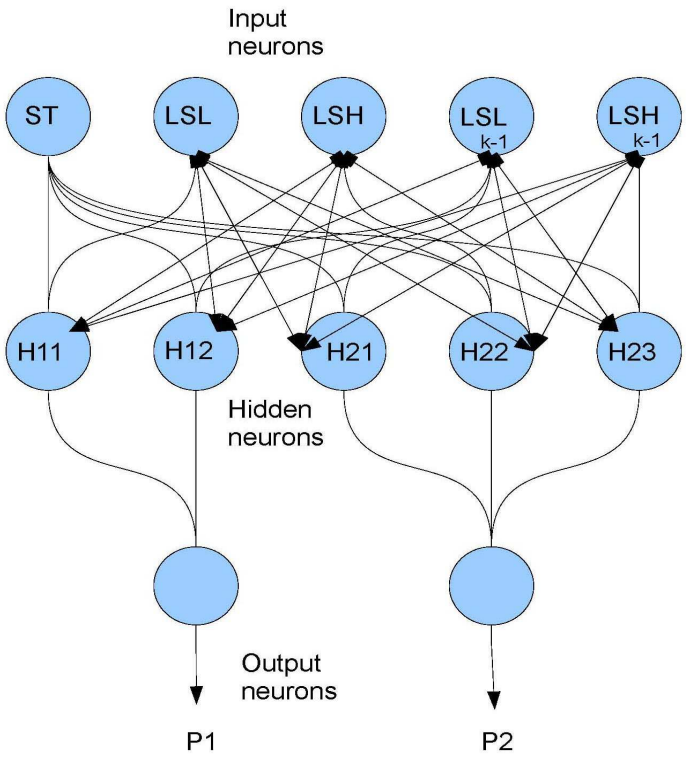

Figure 8. Réseau neuronal du Water Supply : les neurones d'entrée (Input neurons), les neurones cachés (Hidden neurone)cet les neurones de sortie (Output neurons). (extrait de Dubois et Mascia, 2010)

10. NOUVELLE ORGANISATION DES SYSTEMES

Lorsque la déclaration graphique des séquences est terminée, que la simulation a permis de valider l'ensemble des séquences et d'éliminer les incohérences, on envoie les résultats dans le système industriel de production pour exploiter les résultats.

Actuellement, on peut procéder de deux manières :

1. on envoie, dans les systèmes existants, les formules algébriques issues des modélisations
2. on envoie, les matrices booléennes résultantes des déclarations dans un nouveau type de système dont le moteur (système d'exploitation) transformera celles-ci en formules algébriques avant exécution. Le moteur du système (SE) aura un certain nombre de nouveautés dont ce convertisseur des matrices actives en modèles algébriques neuronaux en est le principal.

Dans la première hypothèse, pour chaque système industriel, il faudra traduire les noms et les équations dans la syntaxe imposée par le fabricant, de ce fait, rien ne change au niveau sécurité et performances. Dans la seconde hypothèse, il est proposé une nouvelle conception des systèmes afin de recevoir et d'exploiter de manière optimum les résultats du concept global présenté.

Comme tous les systèmes industriels actuels, celui-ci possède également des entrées/sorties physiques, un processeur, une mémoire, une interface de communication.

Par contre, une nouvelle organisation de la mémoire doit être structurée en matrices binaires issues des objectifs fonctionnels décrits de manière graphique par GENSYSPRO (exemple "WS").

Chaque objectif fonctionnel contenant la matrice, possède un bit d'état d'activation ou de désactivation. Dans l'exemple de la fonction "WS", c'est le bit ST/SP de marche arrêt qui active à l'état 1 ou désactive à l'état 0 la fonction. L'activation ou désactivation des objectifs fonctionnels sont surveillés par un XOR, ce qui permet à tout moment d'observer tout changement de directives fonctionnelles données par l'opérateur (contrôle de la différence).

Pour chaque matrice active, les entrées physiques concernées (exemple "Low" et "High") sont informées qu'elles doivent elles même avertir la fonction XOR de leur changement d'état.

Prenons notre exemple "WS", dès que ST/SP est à mis à 1, la fonction « WS » devient active et les entrées sont "Low=1" et "High=0". En 1 seul "top" ou cycle interne CPU, on lit la matrice binaire

active "WS", on la transforme en équation algébrique et on exécute le résultat d'équation "Pump1= 0" et "Pump2= 1" pour ensuite affecter les résultats 0/1 aux sorties physiques.
Ceci sans aucune syntaxe programmée dans le système.
Tant que la fonction "WS" est active, il faudra que non seulement qu'il y ait changement d'état mais que ce soit les bons états et les bons changements au bon moment, c'est-à-dire les états attendus à l'étape décrite par l'outil graphique GENSYSPRO.
La figure 9 ci-dessous décrit de manière simple les différentes parties du système proposé :

- Le module "XOR"(OU exclusif) qui surveille les changements des objectifs fonctionnels (ici Start/stop) qui active la fonction "WS"
- Le module "XOR"(OU exclusif) qui surveille les changements des entrées physiques concernées par la fonction "WS" (ici "Low" et "High")
- Le module des sorties physiques qui gère les "Outputs"
- La mémoire contenant les tables binaires, les états et les valeurs historiques des états.
- Le CPU qui va piloter l'ensemble et contenir toutes les fonctionnalités décrites ci-dessus ainsi qu'un certain nombre de fonctions élémentaires classiques connues du monde électronique comme la gestion des interfaces de communication, d'auto test "hardware control and watch dog", de gestion des erreurs, d'interface avec la mémoire, batterie de conservation heure et mémoires spéciales, etc.

De par cette organisation, il y aura automatiquement optimisation du temps CPU où le processeur n'aurait rien à faire si :

- Aucun changement d'objectif fonctionnel n'intervient (premier XOR)
- Aucun changement d'état (événement) n'arrive aux étapes des fonctions actives (second XOR)

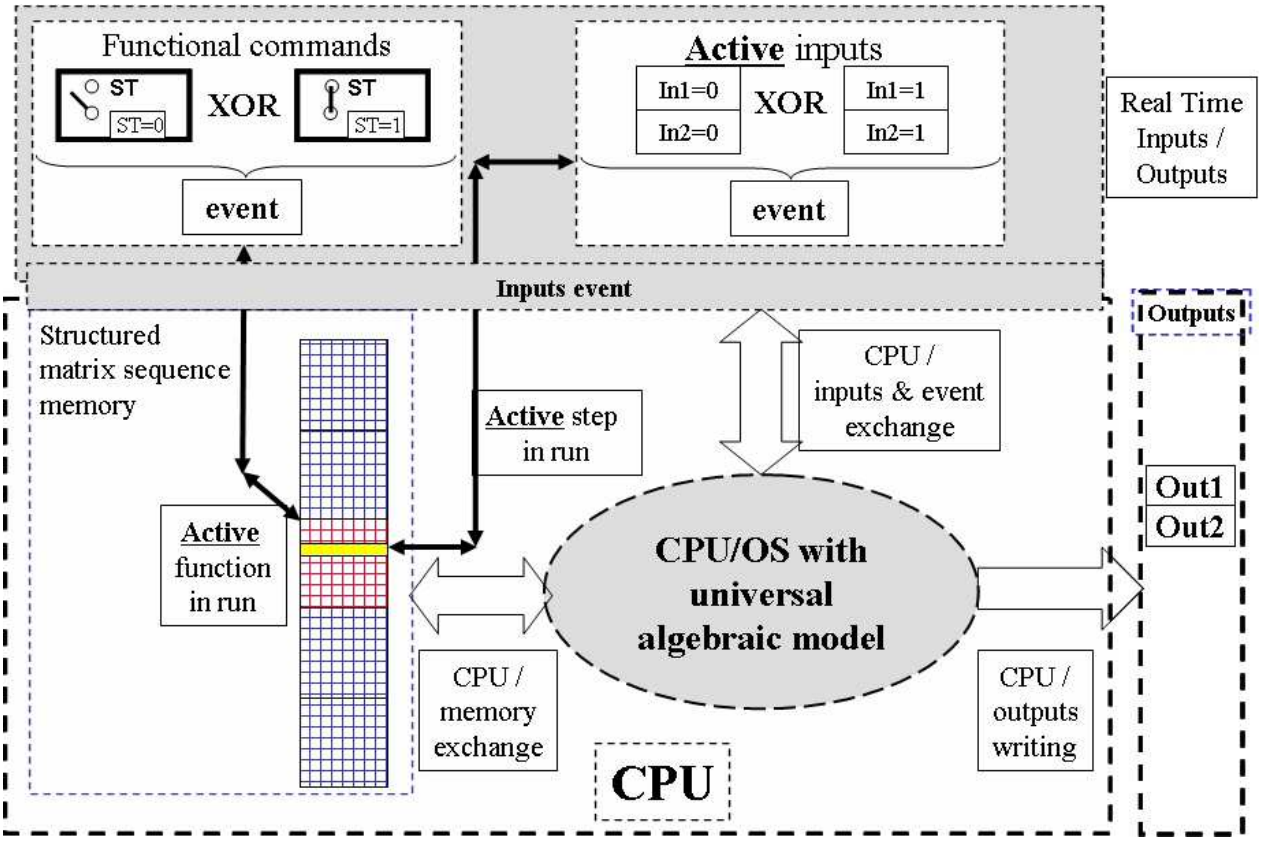

Figure 9 : structure synthétisée de l'organisation du système d'exploitation proposé (extrait de Dubois et Mascia, 2010b).

11. CONCLUSION

Un nouveau concept, un nouveau langage, une méthode et de nouveaux modèles pour rendre les systèmes artificiels plus fiables et plus performants. Pour aussi rendre les processus mieux

contrôlés, prévenir les risques et anticiper les défaillances techniques et humaines, ce sont les objectifs de cette recherche. Les systèmes artificiels actuels ne sont pas assez fiables et les programmes insuffisamment sûrs. Ils demandent des processeurs de plus en plus puissants et la mémoire n'est jamais suffisante pour y mettre toutes les combinaisons de l'explosion combinatoires des possibilités. L'homme n'est pas parfait et les systèmes artificiels devraient l'aider à palier ses défaillances afin d'éviter les accidents.

Le langage pour parler à nos machines est affaire de spécialiste et le langage naturel s'impose de plus en plus comme la manière "normale" d'échanger avec les machines. GENSYSPRO est un concept global qui propose un langage graphique et une méthode pour l'automatisation des processus du domaine discret. Pour résumer l'ensemble de ce qui a été décrit ci avant, voici quelques avantages du concept GENSYSPRO :

- La sécurité totale depuis le capteur/actionneur jusqu'au contrôle.
- Plus d'inconnues sur une exécution non attendue et aléatoire des systèmes.
- Maintenance préventive, prédictive d'anticipation sur toute dérive des processus.
- Tous les programmeurs feront le même programme et auront une formation unique.
- Tous les systèmes seront génériques.
- Un seul outil de conception, de simulation et de contrôle/commande.
- Plus de syntaxe de programmation.
- Augmentation très importante des performances et capacités des systèmes artificiels.

On peut aussi envisager une communication différente entre l'homme et la machine comme par exemple le langage graphique proposé mais aussi par le langage naturel en formalisant chaque objet et chaque opération de conception, pour former un système multi-agents (voir, par exemple, Wooldridge Michael, 2009).

REMERCIEMENTS

Je remercie particulièrement mon ami et mentor scientifique Daniel M. Dubois, sans qui, cette importante réflexion sur le monde des automatismes industriels n'aurait jamais été possible. Cet article est basé sur le projet de recherche et développement, GENSYSPRO proposé par la société Euro View Services SA . Ce projet a été financé à 50% par la Région Wallonne de Belgique que je voudrais remercier ici. Grâce à ces financements, le partenariat avec l'ASBL CHAOS a été possible. CHAOS a assuré la partie scientifique (Centre for Hyperincursion and Anticipation in Ordered Systems, Institut de Mathématique de l'Université de Liège, Belgique (http://www.ulg.ac.be/mathgen/CHAOS))

REFERENCES

1. Dubois D. M., Resconi G. (1993), Mathematical Foundation of a Non-linear Threshold Logic: a new Paradigm for the Technology of Neural Machines, ACADEMIE ROYALE DE BELGIQUE, Bulletin de la Classe des Sciences, 6ème série, Tome IV, 1-6, pp. 91-122.
2. Dubois D. M. (1998), "Boolean Soft Computing by Non-linear Neural Networks with Hyperincursive Stack Memory". In *Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications*, Edited by O. Kaynak, L. A. Zadeh, B. Türksen, I. J. Rudas, NATO ASI Series, Series F: Computer and Systems Sciences, volume. 162, Springer-Verlag.
3. Dubois Daniel M. (1999) Hyperincursive McCulloch and Pitts Neurons for Designing a Computing Flip-Flop Memory. Computing Anticipatory Systems: CASYS'98 - Second International Conference. Edited by Daniel M. Dubois, Published by The American Institute of Physics, AIP Conference Proceedings 465, pp. 3-21.
4. Dubois D. M., A. Mascia (1995a), Méthode Générale pour l'Analyse et la Programmation des Systemes Discrets, Proceedings of the 14th International Congress on Cybernetics, edited by the International Association for Cybernetics, pp.571-576.

5. Dubois D. M. and A. Mascia (1995b), Computer Aided Generator of Models for Designing Automation Devices, in Advances in Computer Cybernetics, volume III, edited by G. E. Lasker, published by The International Institute for Advanced Studies in Systems Research and Cybernetics, University of Windsor, Canada, pp. 23-27.
6. Dubois Daniel M. and Mascia Antonio (2010). Design of an Algebraic Neural Operating System in Programmable Logical Controller for Simulation and Execution of Sequential Operations. *International Journal of Computing Anticipatory Systems, VOLUME 25, pp. 3-20*
7. Dubois Daniel M., Mascia Antonio (2010b). A General Method for the Analysis and the Logical Generation of Discrete Mathematical Systems in Programmable Logical Controller. In: CYBERNETICS AND SYSTEMS 2010, Robert Trappl (editor), Austrian Society for Cybernetic Studies, pp. 65-70
8. Mascia A., Dubois D. M. (1995), Générateur de Modèles Algébriques Appliqué à la Conception d'un Automatisme Industriel, Proceedings of the 14th International Congress on Cybernetics, edited by the International Association for Cybernetics, pp.577-582..
9. McCulloch W. S., Pitts W. (1943), A logical calculus of the ideas immanent in nervous activity, Bulletin of mathematical Biophysics, vol 5, pp. 115-133.
10. Pichler F., H. Schwärtzel (Eds.) (1992), CAST Methods in Modelling, Computer Aided Systems Theory for the Design of Intelligent Machines, Springer-Verlag, Berlin, Heidelberg.
11. Wooldridge Michael (2009). An Introduction to MultiAgent Systems, John Wiley & Sons, 2nd Edition