→ C. Neugnot, M. Kneppert,
Département Ingénierie
des équipements de travail,
Centre de Lorraine, INRS,
Vandœuvre-lès-Nancy

Logiciels applicatifs relatifs à la sécurité

Étude des problèmes liés à leur exploitation

SAFETY-RELATED APPLICATION SOFTWARE.
STUDY OF THE PROBLEMS LINKED TO THEIR USE

Programmable electronics are increasingly present in industrial processes, particularly for machine safety management. This preliminary study has contributed precise elements intended to optimise this management, and thus to limit potentially dangerous failures.
The safety elements present within a complex system of machines, which determine two levels of safety management, are firstly identified:
- machine level (sensors, operation, pre-actuators),
- global level: compatibility management (involving parameter settings), relevance and processing of the information provided by several machines.
Safety management in the application software is then situated in relation to the programming steps and their authors (manufacturer, integrator, operator). The hierarchical structure and limitation of actions (by passwords) on higher programming levels must distance the final operator from any possibility of action affecting safety, without however limiting the potential to upgrade the programmes.
On the basis of these observations, the authors analyse the development cycle of the application software. In particular, they examine problems linked to programming (fault, reliability, homogeneity, standards relative to languages, coherence of tools, etc.), simulations and the validation of programmes.
Finally, a field study yields a set of recommendations for each development step of the application software (design, validation, alteration, maintenance, documentation and training).

• *application software* • *machine safety* • *failure* • *programming* • *simulation* • *validation* • *maintenance*

L'électronique programmable est de plus en plus présente dans le processus industriel, notamment pour gérer la sécurité des machines. Cette étude préliminaire a permis d'apporter des éléments précis afin d'optimiser cette gestion, et ainsi de limiter les défaillances pouvant être dangereuses.
Dans un premier temps, sont relevés les éléments de sécurité présents au sein d'un système complexe de machines, qui déterminent deux niveaux de gestion de la sécurité :
- **niveau machine (capteurs, fonctionnement, préactionneurs),**
- **niveau global : gestion des compatibilités (impliquant des paramètrages), pertinence et traitement des informations fournies par plusieurs machines.**
La gestion de la sécurité dans le logiciel applicatif, en fonction des étapes de programmation et de leurs auteurs (constructeur, intégrateur, opérateur), est ensuite localisée. La hiérarchisation et la limitation des actions (par mots de passe) sur les niveaux supérieurs de programmation doit supprimer à l'opérateur final toute possibilité d'action sur la sécurité, sans cependant limiter les possibilités d'évolution des programmes.
À partir de ces constats, les auteurs analysent le cycle de développement du logiciel applicatif. Il aborde notamment les problèmes liés à la programmation (défaut, fiabilité, homogénéité, normes sur les langages, cohérences des outils…), aux simulations, et à la validation des programmes.
Enfin, une étude sur le terrain permet de donner un ensemble de recommandations pour le développement de chaque étape du logiciel applicatif (conception, validation, modification, maintenance, documentation et formation).

• logiciel applicatif • sécurité des systèmes • programmation • simulation • validation • maintenance

L'électronique programmable est devenue une alliée incontournable, que ce soit pour piloter les processus industriels, gérer les sécurités, ou superviser les ateliers de production.

Les concepteurs de composants de sécurité et de commande (automates programmables, commandes numériques) disposent généralement de moyens matériels et humains conséquents, ainsi que d'une expérience incontestable dans leur domaine, impliquant une maîtrise satisfaisante des logiciels systèmes ([1]). Par contre, ce constat s'applique dans une moindre mesure lorsqu'il s'agit des logiciels dédiés à l'application. Les utilisateurs de ces composants sont amenés à créer ou modifier le logiciel applicatif ([2]), pour adapter leur outil de production, sans souvent disposer d'un personnel suffisamment formé à la création, à la validation et à l'évolution de

([1]) Logiciel système : logiciel écrit par le constructeur du composant qui définit le fonctionnement du dispositif avec ou sans programme d'application. C'est un ensemble de sous-programmes qui convertit les instructions du programme d'application entré par l'utilisateur en code machine requis par le matériel du dispositif [1].

([2]) Logiciel applicatif : le logiciel applicatif d'un automate programmable désigne tous les logiciels développés ou modifiés pour un besoin de l'utilisateur spécifique (application).

tels logiciels. Ces modifications de programmation peuvent être à l'origine de dysfonctionnements, dont les conséquences peuvent s'avérer dangereuses pour les utilisateurs.

Ce constat montre que l'exploitation des systèmes programmés pour des applications liées à la sécurité demande des outils et des méthodes simples et efficaces. Ce domaine a fait jusqu'à présent l'objet de peu d'investigations, alors que c'est le lieu même de l'exposition aux risques.

L'INRS a donc décidé d'entreprendre une étude préliminaire sur ce sujet. Nos investigations ont consisté à recueillir des informations majoritairement sur l'utilisation des automates programmables et des commandes d'axes numériques pour :

■■ faire le point sur le logiciel applicatif, c'est-à-dire cerner les parties de ce logiciel à la charge des intervenants successifs ;

■■ recenser les problèmes et les difficultés rencontrés dans ses phases de développement, de validation et d'exploitation.

1. Recensement des dispositifs intégrant des systèmes programmables

1.1. Vue globale

Les installations automatisées peuvent être classées en deux catégories, selon qu'il s'agit de :

■■ Processus continu : il s'agit d'installations qui présentent un caractère inertiel. Ils ne sont maîtrisables qu'en fin de cycle opératoire ou en fin de processus, avec un temps de réponse qui peut aller de la minute à plusieurs jours. Dans ces deux cas, la sécurité est assurée par la disponibilité et l'éloignement des opérateurs.

■■ Processus discontinu ou séquentiel, maîtrisable à chaque phase de traitement du produit en cours de fabrication. Ici, la sécurité est assurée :

- par l'arrêt immédiat du processus (ou de l'entité de production) avec un ordre de grandeur de quelques centaines de millisecondes,
- ou par l'interdiction d'accès des personnes à la zone de danger du processus (ou à l'entité de production).

La différenciation en matière de sécurité se fait donc essentiellement au niveau du temps de réponse, suite à la détection d'une situation dangereuse.

Dans le cas de processus continus, on se donne les moyens de poursuivre celui-ci, fut-ce en mode dégradé. La poursuite de l'activité est nécessitée par les risques catastrophiques ou inacceptables (tant matériels qu'humains) qu'engendrerait un arrêt immédiat.

S'agissant d'un processus discontinu, il est possible, soit à tout moment, soit à la fin du cycle, de l'interrompre, sachant que cela suffit pour supprimer le risque. L'inertie liée au cycle ou aux masses en mouvements doit être compensée en empêchant toute intrusion dans la zone dangereuse avant l'arrêt définitif.

Ce document se limitera aux seuls processus discontinus, en lien direct avec la sécurité des machines.

Pour revenir à l'installation automatisée, celle-ci est constituée d'un ensemble de machines communicantes, toutes composées de 5 grandes parties [2] :

- la *commande :* dans laquelle se situe l'automate programmable, si tel est le cas, qui regroupe alors : UC + E/S ([3]) + logiciels de mise en œuvre + communication + fonctions métiers ;
- les *outils de contrôle-commande :* PC ([3]) industriels ;
- l'*interface H/M* ([3]) : outils de dialogues opérateurs (PC exclus) ;
- la *communication locale :* bus de terrain + E/S réparties ;
- les *dispositifs de protection* ou *moyens de prévention*.

1.2. Machines seules

1.2.1. Répartition des machines en familles

Les machines qui intègrent des systèmes programmables peuvent être classées de multiples façons. L'une d'elles consiste à les distinguer selon leur capacité à évoluer en fonction des besoins de l'utilisateur. La répartition proposée est faite selon les trois familles suivantes :

Machines d'usinage traditionnel

Machines-outils, tours, fraiseuses… : machines non évolutives.

Machines spéciales

Centres d'usinage à grande vitesse, robots… : machines qui évoluent au cours de la production de pièces pour s'adapter au besoin de l'utilisateur.

Cette flexibilité est d'ailleurs mise en avant par les fournisseurs de ces machines comme un atout. Or les utilisateurs l'utilisent sans systématiquement faire appel à un organisme de contrôle pour valider ensuite leurs adaptations.

Machines dangereuses

Ces machines sont répertoriées dans l'annexe IV de la « directive Machines » ([4]). En toute rigueur, elles ne doivent pas évoluer sans une nouvelle validation par un organisme agrée.

1.2.2. Sécurité dans la machine

Toute machine, siège d'un mouvement dangereux, nécessite la prise de mesures de sécurité, qui peuvent consister en :

- l'interdiction d'accès du personnel dans la zone de danger pendant l'existence du risque,
- l'élimination du risque par l'interruption du ou des mouvements dangereux.

La norme EN 292-1 [3] donne une représentation schématique générale d'une machine, permettant d'identifier les parties du système de commande ([5])(cf. (a), (b) et (c), ci-après) qui ont une influence sur la sécurité, et par ordre d'importance :

Capteurs et dispositifs de sécurité (a)

Placés en amont de la chaîne de sécurité, leur mission est de signaler ou détecter un passage ou une présence non prévue dans la zone de sécurité (humaine ou matérielle

([3]) UC : Unité centrale ; E/S : Entrées/Sorties ; PC : Personnal computer ; H/M : Homme-machine. Bus de terrain : système de transmission d'informations entre capteurs-actionneurs et systèmes de contrôle-commande.

([4]) Directive n° 98/37/CE du Parlement euopéen et du Conseil du 22 juin 1998 concernant le rapprochement des législations des États membres relatives aux machines. Journal Officiel des Communautés Européennes, *n° L. 207 du 23 juillet 1998.*

([5]) Système de commande [5] : Système qui réagit à des signaux d'entrée provenant du processus et/ou d'un opérateur et qui produit des signaux de sortie qui font que l'équipement commandé fonctionne de façon souhaitée.

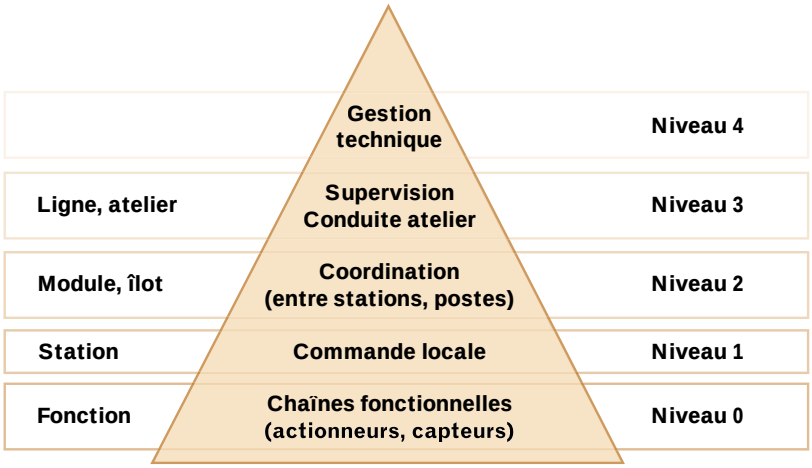


Fig. 1. Hiérarchisation d'un système automatisé
- *Hierarchical structure of an automated system*

par le biais d'un outil). Pour améliorer les performances de ces dispositifs, l'utilisation de technologie en électronique programmée s'impose, comme pour les barrages immatériels, planchers sensibles… Ces dispositifs comportent un logiciel embarqué où la fonction de base, qui est la détection, n'est pas accessible à l'utilisateur. Dans certain cas d'utilisation, un paramétrage est permis, sachant qu'il ne doit pas remettre en cause la pérennité de la fonction détection.

Ces produits offrent maintenant une garantie suffisante, dans la limite où ils entrent dans la catégorie des « composants de sécurité », lesquels sont soumis à des critères très stricts, tant dans leur conception que leur validation par des organismes habilités (CE de type).

Mémorisation et traitement des informations (b)

Essentiellement affectée au fonctionnel, elle pilote les différents mouvements qui participent à l'élaboration du produit. En plus, la plupart des machines gèrent également les sécurités lorsque celles-ci dépendent du cycle machine. Cependant, quel que soit l'état de fonctionnement de la machine, il est indispensable que la partie « sécurité » soit toujours suffisamment fiable, pour que la sécurité du dispositif soit assurée. Il est souhaitable que ces deux parties soient complètement séparées, ou que leur interaction soit aussi faible que possible.

Pour des raisons de productivité et de flexibilité, la gestion des sécurités se complexifie et s'accommode de moins en moins des technologies à relais. Ces traitements tendent à être effectués à partir de logiques programmables [4]. Par conséquent, pour s'assurer du bon développement des parties « commande » et « sécurité », réalisées en fonction de l'application en jeu, il faut passer par une étape de validation avant la mise en œuvre sur site et le démarrage de l'installation.

Préactionneurs (c)

La commande des moteurs et vérins est confiée à des variateurs de vitesse plutôt qu'à des contacteurs, pour accroître les performances et le rendement des machines récentes.

Ces dispositifs sont totalement gérés par des commandes en électronique programmable, à partir desquelles il est possible de choisir des caractéristiques de démarrage ou de freinage, et d'adapter le variateur au type de moteur par modification de quelques paramètres.

S'agissant de la sécurité, la défaillance de ces préactionneurs porte à conséquence, car ce ne sont pas des composants prévus pour assurer des fonctions de sécurité, notamment un arrêt sûr. Il convient donc, par des moyens complémentaires, de pallier à leur possible défaillance.

(6) Système automatisé [7] : système de commande dans lequel des configurations d'automates programmables sont incorporées par ou pour l'utilisateur, mais qui contient également d'autres éléments constitutifs, y compris leurs programmes d'application.

Les trois parties décrites précédemment contribuent toutes à la sécurité directe d'une machine. Elles font maintenant appel aux technologies électroniques programmables - devenues incontournables malgré une difficulté à prendre en compte leurs modes de défaillance - et sont à traiter avec la même rigueur pour obtenir un ensemble cohérent.

Bien entendu, les remarques contenues dans les paragraphes précédents ne doivent pas laisser croire que la sûreté de fonctionnement d'une machine dépend exclusivement du logiciel applicatif. Ce dernier doit bien entendu être supporté par un matériel adéquat, conçu à cette fin. De même, d'autres critères doivent être pris en considération, tels que la mise en œuvre, l'utilisation…

1.3. Installations automatisées

Précédemment, on s'est intéressé à la machine seule. En remontant d'un degré dans l'architecture des installations automatisées, on considère un ensemble de machines communicantes.

1.3.1. Structuration d'un système automatisé

Un système automatisé (6) peut être structuré selon le schéma de la *figure 1* [6], qui met en parallèle et hiérarchise les fonctions de commande et les fonctions opératives :

- niveau 0 : fonction opérative élémentaire *(fig. 2)*,
- niveau 1 : machine *(fig. 3)*,
- niveau 2 : combinaison de machines, concept de module ou d'îlot fonctionnel à considérer comme entité de production autonome *(fig. 4)*,
- niveau 3 : ligne ou atelier, constitué de plusieurs îlots (cf. fig. 4).
- niveau 4 : gestion technique.

Cette hiérarchisation fait apparaître un accroissement des communications existantes, du niveau le plus bas jusqu'à celui de l'atelier.

Plusieurs types de dialogues sont amenés à cohabiter dans un atelier pour assurer :

- la conduite du processus, en principe sans incidence sur la sécurité, sauf lors de travaux de maintenance avec une intervention humaine ;
- l'interface homme-machine, avec des fonctions de contrôle du processus et des sécurités, dont des données erronées pourraient entraîner des actions inadaptées de la part des surveillants ;
- la gestion de la sécurité, c'est-à-dire les échanges entre les capteurs de sécurité, les préactionneurs et les outils de paramétrage.

Tout ceci montre bien que la sécurité des personnes sera affectée par les défaillances de chaque bloc, mais aussi par une défaillance du système de communication.

Fort heureusement, la Directive Machines donne obligation aux constructeurs de machines et/ou de composants de sécurité, à mettre sur le marché des produits conformes aux exigences de sécurité qui leur sont applicables ; les normes les aident dans ce sens.

Fig. 2. Illustration du niveau 0, description d'une chaîne fonctionnelle de base -
Illustration of level 0, description of a basic functional chain

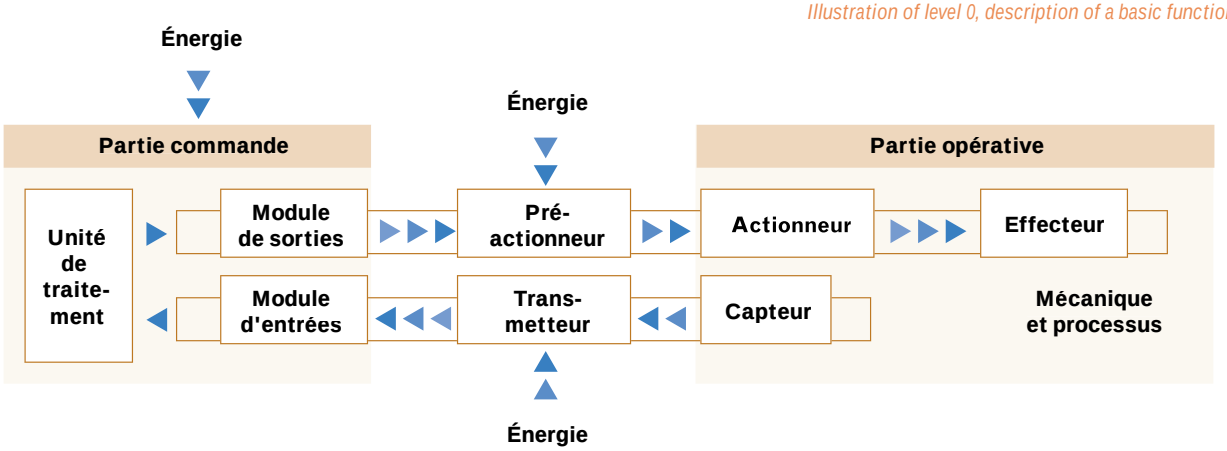
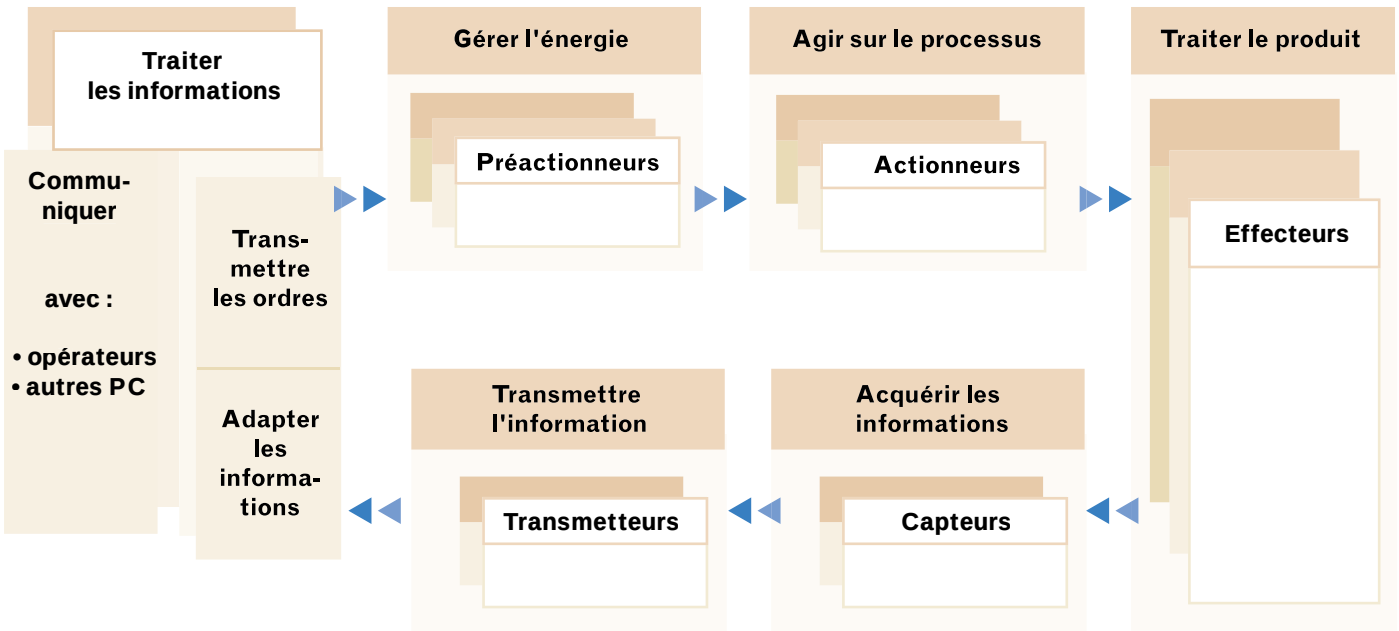


Fig. 3. Illustration du niveau 1, traitement d'une commande locale composée de n chaînes fonctionnelles -
Illustration of level 1, processing of a local control composed of n functional chains



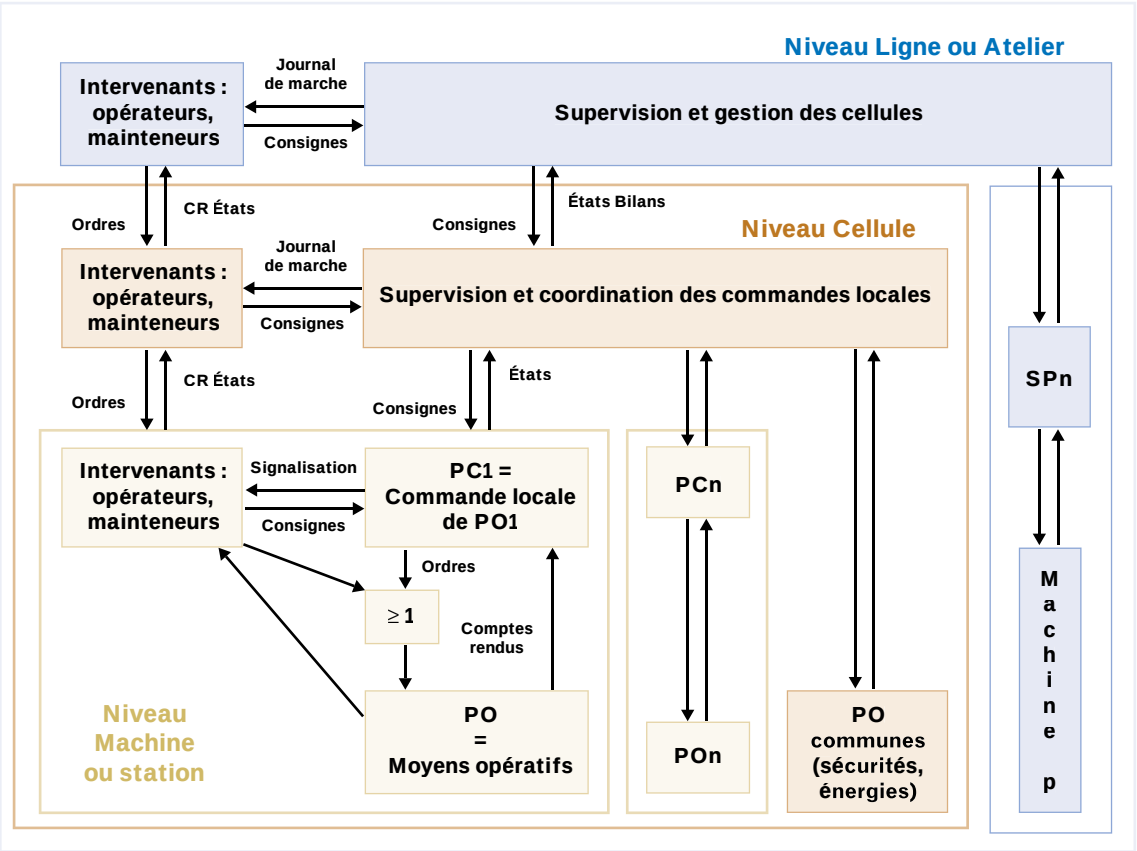


Fig. 4. Illustration des niveaux 2 et 3, structuration verticale d'un système automatisé - *- Illustration of levels 2 and 3, vertical structure of an automated system*

Néanmoins, il est nécessaire d'avoir une compatibilité entre ces produits pour les associer dans un même automatisme (barrages immatériels, systèmes de vision, APIdS, variateurs, bus de sécurité…). Dans ce cas, par l'intermédiaire d'un logiciel de gestion développé pour le bus de communication, l'utilisateur aura à paramétrer son application.

1.3.2. Sécurité à gérer dans une installation complexe (7)

A travers la description précédente, on s'aperçoit qu'en plus des facteurs propres à chaque machine, interviennent des notions relatives à l'ensemble, comme la communication entre plusieurs machines et la sécurité globale. Ces élements ont une influence forte sur la gestion de la sécurité.

Le problème de gestion de la sécurité se pose alors en termes de traitement et de transfert d'informations entre l'ensemble des machines.

De plus, compte tenu de la nécessité d'adaptation de la production à l'évolution du marché, des produits et de la technologie, l'organisation interne des installations n'est pas figée, pas plus que le processus au sein de chacune d'elles. L'automatisme doit évoluer ou être modifié sans dégrader la sécurité.

Remarque
sur la normalisation en cours

Au stade actuel, des normes de type C sont en cours d'élaboration pour fournir une aide au développement de machines simples. Pour certaines machines spéciales, résultant de l'avancée de la technologie, comme les machines à usinage grande vitesse, les problèmes sont différents. Ces cas sont particuliers et sont à considérer à part.
Il en est de même des machines dangereuses citées à l'annexe IV de la Directive « Machines », cas bien précis à distinguer selon les fonctionnements et les applications.

Par contre, peu de travaux concernent encore la sécurité dans les ensembles de machines.

(7) Installation complexe : groupe de machines fonctionnant ensemble de façon coordonnée [4].

La sécurité se retrouve au niveau de la communication, plus précisément, de la pertinence et du traitement des informations fournies par plusieurs machines, et dans ce cas, elle peut être mise en danger par d'autres facteurs.

2. Définition du logiciel applicatif

2.1. Description générale

La partie logicielle d'un système programmable peut être considérée de différentes façons, en fonction de sa position dans le cycle de vie de l'ensemble du logiciel. Chaque intervenant a un rôle de concepteur d'un des niveaux de logiciels et d'utilisateur d'une autre partie.

Le logiciel peut être décrit en 5 parties, définies comme suit :

① Logiciel système : développé par le constructeur du système programmable, il en gère les ressources internes ; celui-ci est aussi appelé logiciel embarqué (« *embedded software ou operating system* » en anglais).

② Atelier de programmation : développé par le constructeur du système programmable, il comprend *le langage de programmation et l'interface pour sa programmation* par un utilisateur.

③ Logiciel applicatif « constructeur », qui comprend *une partie dédiée « Fonctionnement » et une partie dédiée « Produit »* (programme pièce) : conçu par le constructeur de machines. Le responsable de l'application y programme :
- le fonctionnement de la machine,
- la maintenance en ligne,
- son cycle,
- la gestion des protections en prévision de leur utilisation.

Dans le cas d'une machine conçue pour fonctionner hors ligne de fabrication, une interface homme-machine est développée à ce niveau. Pour une ligne de machines, cette fonction sera plutôt assurée par un superviseur.

④ Logiciel applicatif « intégrateur », qui comprend *une partie dédiée « Fonctionnement » et une partie dédiée « Produit »* (programme pièce) : conçu par l'intégrateur de la machine au sein de son groupe de machines, de sa ligne ou de son îlot de machines. Ici, sont pris en compte :
- les modes de marche et d'arrêt de chaque machine,
- le réglage des outils après un choix parmi ceux disponibles sur la machine,
- la gestion des sécurités en fonction des protections à mettre en place,
- la maintenance.

Il faut ajouter l'assemblage des différentes machines, leur synchronisation, la gestion du flux et celle des protections du process.

Le programme pièces est développé à part, mais concerne le même niveau de développement, et nécessite les mêmes connaissances pour sa conception.

⑤ Paramétrage : effectué par le chef d'atelier ou le chef de production ou, plus généralement, l'utilisateur final.

Pour la production de pièces en grandes séries, le niveau ④ est développé par la même organisation que pour le niveau ③. Par opposition, pour les productions de pièces à l'unité comme dans l'industrie aéronautique, les niveaux ④ et ⑤ sont souvent réalisés par la même organisation.

2.2. Localisation de la sécurité dans les niveaux du logiciel

La gestion de la sécurité est répartie sur les différents niveaux (cf. p. précédente).

■■ Au niveau ①, c'est la sécurité du composant lui-même qui est gérée, sa défaillance ne devant pas avoir de conséquences en aval.

■■ Par contre, dès le niveau ②, la sécurité est influencée par le développement réalisé dans l'atelier de programmation. En effet, même si la sécurité n'apparaît pas explicitement, il est évident que des mesures prises à ce stade faciliteront le travail des utilisateurs de plus bas niveaux. Lorsqu'un outil est correctement développé et présente un confort d'utilisation, des efforts moindres restent à fournir par l'utilisateur. Par exemple, si les moyens sont donnés pour que l'éditeur et la configuration soient bien développés avec un affichage de messages d'alerte ou d'erreur, la validation ultérieure sera moins lourde et aussi plus rapide, moins de modifications restant à faire.

■■ Il est clair que la sécurité est principalement gérée aux niveaux ③ et ④, car ce sont les cycles de fonctionnement des machines et process qui sont directement développés. Selon les connaissances du développeur en ③ ou ④ sur l'application finale, la sécurité sera répartie plus ou moins sur l'un ou l'autre de ces deux niveaux.

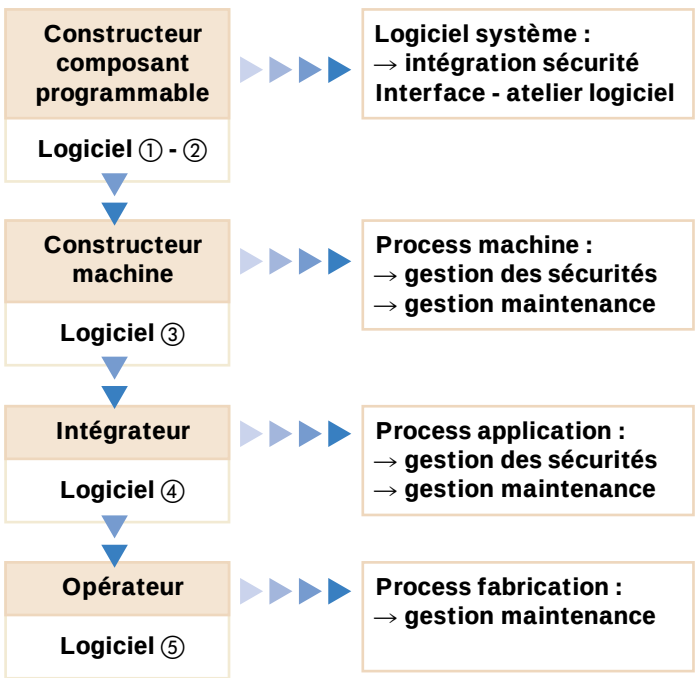
■■ Enfin, il est souhaitable qu'aucune intervention sur la gestion de la sécurité ne soit faite dans le paramétrage (⑤). En effet, on préférera que l'opérateur n'ait pas à choisir et à valider seul des paramètres directement liés à sa sécurité. Il faudra donc veiller à ce que celle-ci soit totalement verrouillée et entièrement définie dans les niveaux supérieurs.

2.3. Hiérarchisation des accès à la sécurité

Les exigences d'adaptation et d'évolution d'un système de production en fonction du marché sont favorisées, et simplifiées, par l'arrivée de commandes programmables, et plus précisément par la souplesse de modification du logiciel applicatif.

Dans ce cas, pour maîtriser la sécurité, plusieurs voies sont à considérer :
- limitation des accès au personnel habilité uniquement,
- séparation des parties fonctionnelle et sécuritaire,
- validation des sécurités après toute modification.

Fig. 5. Résumé des accès des différents niveaux du logiciel
- Overview of access to the different levels of the application software



La première solution est déjà appliquée par les constructeurs-intégrateurs.

En partant du niveau ② où toutes les possibilités sont offertes en programmation de logiciel applicatif, et en évoluant vers le niveau ⑤, chaque intervenant développe sa partie (rôle de constructeur de programme) dans la configuration en place (rôle d'utilisateur) mise à disposition par l'intervenant précédent *(figs. 5 et 6)*.

Ainsi, niveau après niveau, les degrés de liberté se réduisent en matière de programmation, l'objectif étant de limiter les accès d'un niveau donné vers le précédent pour ne pas altérer les fonctions mises à disposition.

A la fin, l'opérateur ou utilisateur final (exposé aux accidents) ne dispose plus que du paramétrage et idéalement, n'agit plus sur la sécurité.

Le personnel de maintenance/dépannage est à prendre en compte, notamment aux niveaux ④ et/ou ③. Des procédures sont à mettre en place pour lui donner des possibilités d'action sans dégradation de l'automatisme et principalement des sécurités dont il est pourvu.

2.4. Influence de la maintenance à distance sur la sécurité des logiciels

La complexité croissante des automatismes, conjuguée au haut niveau de disponibilité exigé, ne permet plus aux équipes d'assurer un dépannage et une maintenance adaptés. Ce constat a amené des entreprises à recourir à la télémaintenance proposée par des constructeurs-intégrateurs, sous forme d'une ligne dédiée à une intervention directe.

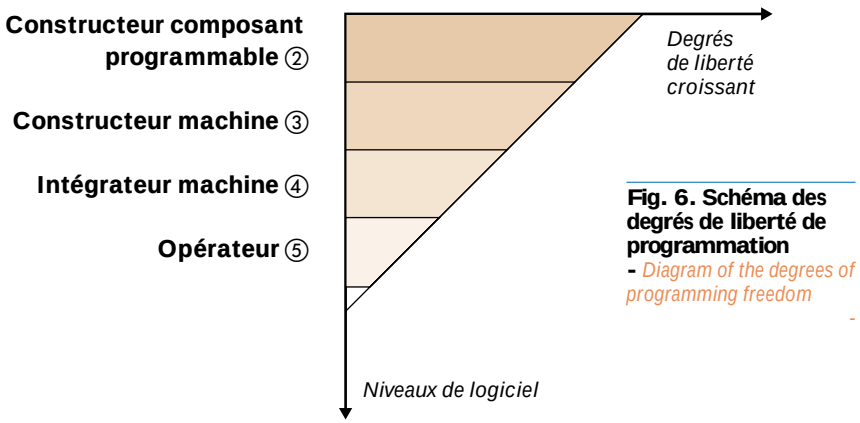


Fig. 6. Schéma des degrés de liberté de programmation
- *Diagram of the degrees of programming freedom*

Ainsi, à distance, sont réalisés le téléchargement de programmes, la lecture des messages, la vérification et modification de paramètres…

Cette avancée technologique peut avoir des effets bénéfiques sur la sécurité, dans la mesure où les intervenants et leurs actions sont bien définis, le constructeur-intégrateur n'étant pas obligé de divulguer ses mots de passe.

Par contre, pour des raisons de secrets de fabrication, des utilisateurs ne souhaitent pas que des intervenants extérieurs aient accès à certaines parties logicielles. Il en résulte, pour le constructeur de la machine, la nécessité de prévoir dans ses spécifications, la possibilité de séparer les parties commande et opérative de tout ce qui relève de la fabrication.

2.5. Exemple d'une machine spéciale : un centre d'usinage à grande vitesse (UGV)

Ce paragraphe illustre la classification proposée précédemment par l'exemple de l'architecture logicielle, d'un système programmable implanté sur un centre d'usinage à grande vitesse (UGV). La *figure 7* synthétise les différents types de logiciels et les intervenants qui leurs sont associés.

La partie sécurité est développée au niveau du constructeur de la machine et est concentrée dans un bloc dit de sécurité, verrouillé après validation par un mot de passe. Le bloc contient :

- activation des entrées/sorties,
- entrée des paramètres de sécurité (environ 20 pour une commande numérique),
- checksum des paramètres.

De ce fait, en principe, aucune sécurité n'est gérée dans le programme pièces.

Les paramètres liés à la sécurité sont fixés par l'intégrateur du système programmable. Le client ou utilisateur n'intervient pas dans ce choix. Si tel était le cas, c'est le client qui en prendrait la responsabilité.

Ces paramètres ne sont accessibles que par mot de passe, connu seulement de l'intégrateur.

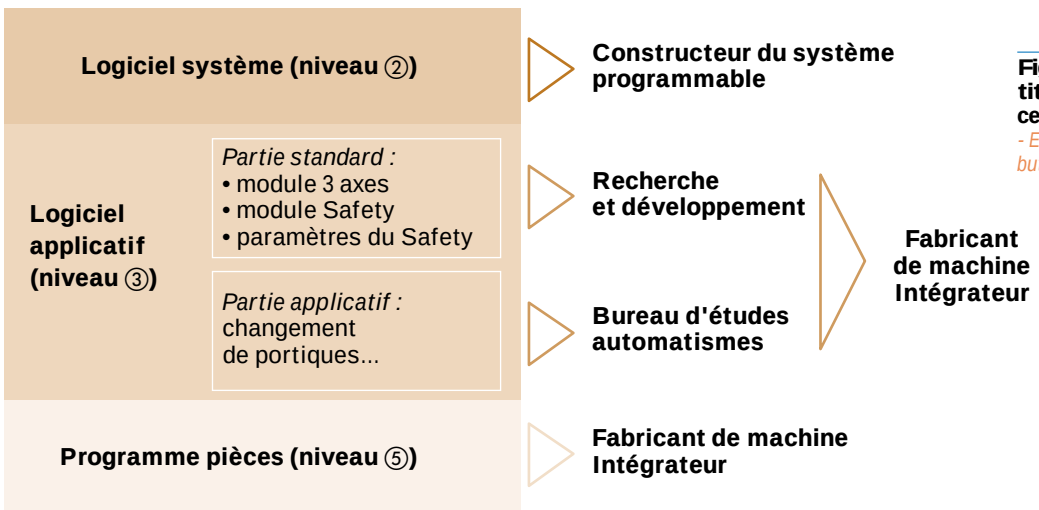


Fig. 7. Exemple montrant la répartition niveaux/intervenants pour un centre d'usinage à grande vitesse
- *Example showing the level/personnel distribution of a high-speed machining centre*

Ceci constitue seulement un premier verrouillage, pouvant être renforcé par d'autres : la modification d'une caractéristique (vitesse, par exemple) nécessite une action sur plusieurs paramètres. Au final, l'ensemble des paramètres est compilé et un checksum est calculé. A la mise en service de la machine, ce checksum est relevé et daté dans un fichier qui contient tous les paramètres, leur signification et leur valeur.

C'est seulement par une bonne connaissance du produit que l'on peut modifier des paramètres sans que le système ne se bloque.

L'affectation de ces paramètres sur chaque machine peut être réalisée de manière automatique par un exécutable lorsque plusieurs machines identiques sont en cours de construction. Le téléchargement des machines clonées sera effectué par un support matériel, de préférence au réseau.

3. Cycle de développement du logiciel applicatif

Les dépendances et fonctionnalités des logiciels et progiciels d'une application de contrôle-commande d'une installation automatisée peuvent être schématisées comme indiqué sur la *figure 8* [6].

3.1. Cahier des charges et spécifications

Un système de production automatisé doit répondre à plusieurs impératifs :

- la compétitivité des produits fabriqués : performances, qualité, coût…
- les objectifs assignés à l'outil de production : sécurité des biens, pérennité de l'installation…
- les objectifs en rapport aux interventions humaines en exploitation et en maintenance : sécurité des personnes, qualité des interventions…

Face à ces objectifs, une première réflexion consiste à caractériser les contraintes inhérentes à la production, à savoir :

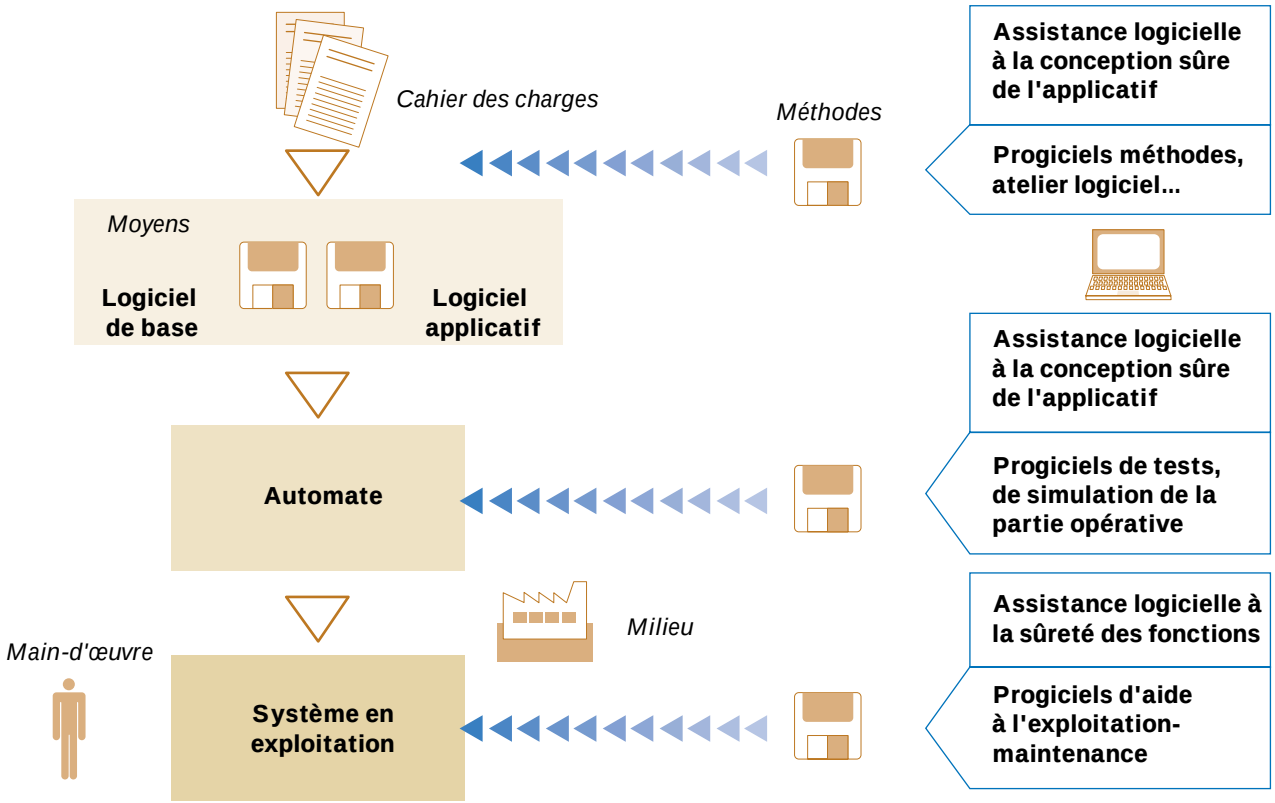
- délais et coûts de réalisation,
- technologies émergeantes,
- site de production,
- qualification du personnel,
- …

Une seconde étape concerne l'élaboration du cahier des charges fonctionnel, où sont définies d'une part les fonctions à assurer, leurs critères et contraintes, et d'autre part, les fonctions de sécurité, leurs critères et contraintes.

La *figure 9* décrit l'expression fonctionnelle et contractuelle du besoin [6].

La réalisation de ces différentes étapes nécessite une large participation des acteurs du projet. Or, un des freins lié à ce type de démarche se situe au niveau de la nécessaire compréhension mutuelle qui doit exister entre les intervenants des différents horizons, lesquels ont des cultures et des langages différents.

Fig. 8. Dépendances et fonctionnalités des logiciels d'une installation automatisée *(d'après [6])*
- *Dependencies and functional features of application software of an automated plant (according to [6])*



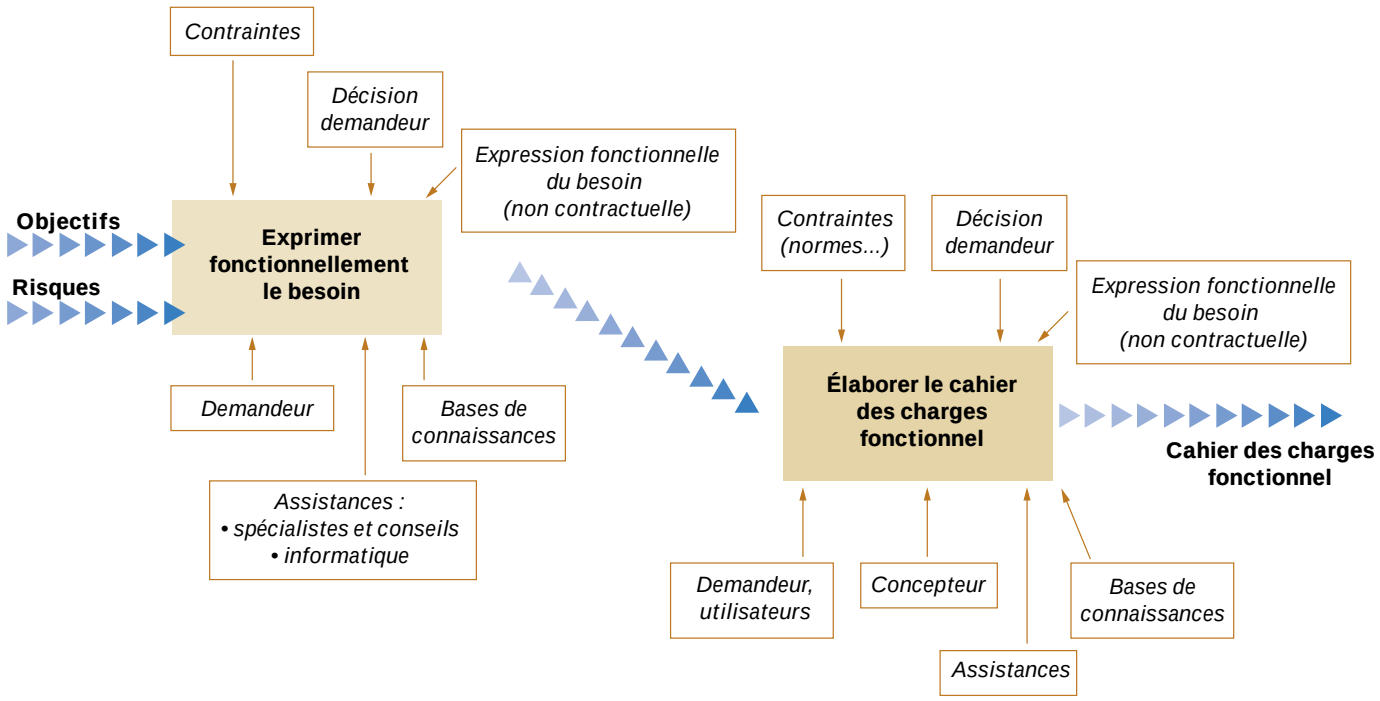


Fig. 9. Expression fonctionnelle et contractuelle du besoin *(d'après [6])*
- Functional and contractual expression of needs (according to [6])

Une des clefs de la réussite du projet consistera à choisir des outils permettant tout à la fois, de faciliter les échanges afin de bâtir un cahier des charges rigoureux, mais aussi de respecter toutes les phases du cycle de développement, tout en vérifiant et validant chacune d'entre elle par un processus interactif de tests et de révisions.

S'agissant des logiciels et progiciels

Dans l'expression du besoin, la sûreté de fonctionnement prend une part essentielle à l'intérieur de laquelle la partie consacrée aux logiciels et progiciels s'accroît de plus en plus. Avant d'aborder l'élaboration d'un cahier des charges, il est bon de rappeler les problèmes générés par le logiciel, dont il faudra tenir compte.

Des problèmes liés à la fiabilité humaine

D'une certaine manière, la fiabilité humaine et celle des logiciels sont liées. Une étude réalisée au profit de l'Armée de l'air américaine, à partir d'une trentaine de projets, a mis en évidence la génération dans le développement d'un logiciel, d'un défaut pour cent instructions en moyenne (pour des logiciels de 1 000 à 800 000 instructions) [6]. Elle a déterminé que les fautes, principalement issues de la conception, proviennent également de l'exploitation ou de la maintenance.

Des problèmes liés à la fiabilité du logiciel

Une autre difficulté à gérer est l'évolution du logiciel dans le temps. Contrairement aux a priori, un logiciel ne s'améliore pas obligatoirement en phase de maintenance : suite à la détection d'un défaut, soit la réparation est correcte, le défaut est éliminé, soit elle est mal faite et d'autres défauts sont introduits.

Des problèmes liés à l'approche probabiliste

Il n'existe pas de lien direct entre le nombre de défauts d'un logiciel et sa fiabilité. Ainsi, un logiciel comportant de nombreux défauts dans une boucle peu sollicitée a une bonne fiabilité, tandis que peu de défauts dans une boucle fortement sollicitée entraînent une mauvaise fiabilité.

Des problèmes liés aux défaillances

Aucune méthode pour prévoir les défaillances d'un logiciel à partir de ses composants n'est encore reconnue.

3.2. Conception du programme

Le programme est « l'âme » d'un système et la programmation représente l'activité centrale, donc la plus connue du cycle de développement du logiciel.

Cette activité ne se limite pas à la seule production d'un programme, car elle inclut d'autres activités essentielles à son exploitation. La programmation a donc pour mission de délivrer un produit qui contient, bien sûr, du programme mais aussi de la documentation et du test.

La programmation, par rapport à d'autres activités, se distingue par des éléments qui lui sont spécifiques, que l'on peut résumer ainsi :

■■ Un programme est un raisonnement dont les termes sont les données et les règles logiques, les instructions de la machine. A partir d'une certaine taille, données et instructions doivent être organisées afin de rester compréhensibles. Le programmeur devra donc garantir cette compréhension en organisant et en structurant données et instructions.

■■ L'acte de programmation est le produit de l'activité d'une seule personne (même si un programme peut avoir plu-

sieurs auteurs) qui lui imprimera sa marque.

■■ Lorsque le processus de programmation démarre, il n'y a aucune ligne de code, et le programmeur est face à un ensemble de documents papiers (cahier des charges, programmes antérieurs,…) qu'il doit interpréter et traduire en un programme qui comportera N lignes de code. Pour atteindre ce but, il aura :

- à déterminer toutes les données intermédiaires et les séquences d'instructions qui lui sont nécessaires,
- à gérer les ressources que le programme manipule,
- à organiser le texte explicatif,
- à prévoir des entités additionnelles permettant sa mise au point, son autocontrôle, sa maintenabilité, son évolution future.

■■ Enfin, il sera nécessaire de s'assurer que le programme respecte le cahier des charges et ses spécifications, à savoir que les données qui l'alimenteront, fourniront après traitement le résultat demandé, en respectant les conditions initiales.

L'activité de test est indissociable de l'activité de programmation. Elle met en œuvre différents moyens qu'il sera nécessaire d'optimiser en fonction du but recherché, sachant qu'en matière de sécurité, la suffisance reste l'objectif à atteindre (l'exhaustivité étant un objectif théorique).

3.3. Validation

La difficulté majeure des outils et méthodes développés jusqu'à présent est pour l'essentiel due au fait que l'on a oublié que ces moyens doivent être utilisés par des individus réels et non virtuels, œuvrant dans des projets concrets qui ont chacun leur spécificité, que de plus le personnel utilisateur n'a le plus souvent ni la formation suffisante, ni le temps pour mettre en œuvre ces moyens complexes. On assiste néanmoins au retour à un certain réalisme, dans un contexte où l'individu ou l'équipe est le cadre de référence, et que outils et méthodes ne peuvent s'exprimer qu'à travers eux.

S'agissant de sûreté de fonctionnement ou de sécurité des personnes, il devient nécessaire de tester beaucoup plus à fond ces logiciels, qui s'exécutent sur des architectures de plus en plus complexes. Les méthodes empiriques utilisées jusqu'alors s'avèrent mal adaptées, tant les combinaisons deviennent nombreuses. A titre d'exemple, on trouvera ci-dessous quelques-unes de ces approches :

■■ La validation du logiciel est faite grâce à un banc de simulation sur la machine par des tests, y compris du programme pièces pour vérifier ses interactions avec la sécurité. Elle est réalisée par le bureau d'études en automatismes.

■■ Lors de la mise au point, les « bugs » restants sont consignés dans des fiches de modification qui suivent le logiciel et remontent au bureau d'études des automatismes. C'est à lui seul que revient la responsabilité de mettre en œuvre les modifications. La version modifiée est remise sous test mais aussi archivée, pour servir à des futurs développements de produits.

■■ En principe, un procès-verbal de réception basé sur celui fourni par le constructeur du système programmable est réutilisé. Il est réalisé sur une machine de présérie (une seule fois pour toute la série de machines), puis consigné dans un document de référence ou stocké sur le serveur. Il contient la liste des tests à effectuer et l'échelle des valeurs de contrôle à obtenir, pour le module de base et les options. Ce procès-verbal est déroulé par des personnes habilitées uniquement.

■■ Sur l'ensemble des machines de la même série, tous les tests sont appliqués ; mais il s'agit alors d'un contrôle de validité des résultats par rapport à un seuil sans les relevés détaillés des valeurs réelles obtenues.

Le moyen le plus couramment utilisé pour la validation consiste à simuler la partie opérative (PO) de l'application ([8]).

3.3.1. Objectifs de la simulation de partie opérative [10]

Le but recherché dans la simulation de partie opérative est de tester, le plus en profondeur possible, l'application configurée dans l'API, pour éliminer les défauts de conception et de réalisation, et réduire le temps de mise en service.

Ces tests concernent les phases situées essentiellement sur la branche droite du « cycle en V » : tests unitaires, tests d'intégration et validation ([9]).

Pour cela, il est nécessaire de modéliser la partie opérative (description structurelle de l'application) et de définir les modes de marche du dispositif pour faire la simulation.

Rappel
La partie opérative est constituée uniquement par l'ensemble des actionneurs et des capteurs.

L'automaticien et l'utilisateur cherchent, lors de cette simulation, à valider notamment :

- le mode manuel,
- le mode automatique,
- le fonctionnement des pupitres,
- les diverses séquences de démarrage de l'installation,
- les modes de marche particuliers : de repli ou dégradés, changement de mode de marche, récupération sur incident…
- les fonctions de sécurité, tests des arrêts d'urgence, des protections…
- la réaction des programmes applicatifs aux défauts de la partie opérative sur les cartes d'E/S, défauts de tête de filerie, des actionneurs de façon à vérifier la conformité :
  - de l'arrêt généré,
  - du mode de reprise,
  - du suivi et de la gestion des défauts par leur visualisation, traitement et archivage,
- les interactions avec l'environnement : échanges entre automates et/ou commande numérique,
- les valeurs limites des variables : débordement des compteurs et des temporisations,
- les temps de réponse des programmes des automates.

Ces progiciels doivent permettre de vérifier que l'application développée dans l'automate est conforme à ce qu'on attend, et d'aller plus loin dans la réalisation des essais sur site.

Ainsi, ils contribuent à garantir la sécurité de l'installation automatisée par la possibilité d'effectuer des tests dépassant les possibilités physiques du processus sans aucun risque comme :

- essayer de détecter toute situation potentiellement dangereuse,
- provoquer des conjonctions d'événements sur la partie opérative pour tester les réactions du programme applicatif : forcer des capteurs/actionneurs à 1 ou à 0, simuler des rebonds de contacts, créer des coupures ou des courts-circuits,…

([8]) Partie opérative [6] : ensemble des moyens (matériels) opérant physiquement sur les matières d'œuvre ou les utilités (énergie, outils, fluides, outillages...), en vue d'assurer la production.

([9]) Cf. par exemple : CHARPENTIER P. et coll. - Comment construire les tests d'un logiciel. Cahiers de notes documentaires - Hygiène et Sécurité du Travail, *2000, 181, ND 2140, pp. 65-77.*

Il est utile d'ajouter que la simulation de partie opérative permet non seulement de faire de la validation (c'est-à-dire la vérification de la conformité du fonctionnement par rapport aux spécifications du cahier des charges), mais également de mettre en évidence des comportements que les concepteurs n'avaient pas imaginés. Dans un système complexe, la prévision de dysfonctionnements au niveau partiel d'un composant ne permet pas toujours d'imaginer les dysfonctionnements qui en résultent au niveau global [11].

3.3.2. Principe de fonctionnement d'un simulateur de partie opérative [10]

Le simulateur est composé d'une partie logicielle implantée sur PC et, selon les cas, d'une partie matérielle autorisant le raccordement au système de contrôle-commande (automate programmable ([10]) ou commande numérique principalement). Le simulateur va pouvoir lire les informations de sortie de l'automate et écrire les informations en entrée. En fonction de la commande faite à un actionneur, le simulateur va renvoyer des changements d'état des entrées, images du procédé qui évolue.

La liaison entre le PC et l'automate est assurée [12] :

■■ soit *par une carte spécifique* accédant directement au bus automate : le fonctionnement de l'automate n'est pas perturbé et cela ne demande aucun ajout dans le programme application. Le principal inconvénient est la nécessité de disposer d'un type de carte pour chaque gamme d'automates ;

■■ soit *par un coupleur de communication* : cela nécessite l'adjonction dans le programme application d'un module gérant les échanges avec le coupleur, le programme testé n'est alors pas rigoureusement celui implanté sur site et sur des installations à temps critique, cela pose des problèmes ;

■■ soit *par une liaison traditionnelle en fil à fil* : ici, le programme original est non modifié. Cependant, la mise en œuvre du câblage n'est pas immédiate, ce qui explique l'abandon progressif de ce type d'architecture.

Les simulateurs de partie opérative sont actuellement en pleine phase de croissance. Cependant, ce ne sont que des outils de conception, ce qui implique que :

- leur mise en œuvre nécessite autant de soin et de rigueur et une application des mêmes règles que pour le développement du logiciel lui-même,
- un écart entre le modèle et la partie opérative est quasi inévitable et il faut veiller à ce qu'il soit le plus faible possible.

En conclusion, il est bien entendu qu'ils n'apportent pas une solution à tous les problèmes. Leur utilisation doit s'intégrer dans une démarche globale de développement, à adopter dès le début d'un projet.

3.4. Les langages d'automatismes

Sous l'impulsion de grands utilisateurs souhaitant avoir une vue plus unifiée de la programmation de leurs automatismes, la norme CEI 61131-3 sur les langages ([11]) pour automates programmables a vu le jour en 1993 [8].

La norme CEI 61131 s'étend de la spécification aux architectures [13]. Elle comporte actuellement 8 parties. Les 4 premières parties portent, dans l'ordre, sur : introduction, matériel, langages de programmation, manuel et conseil à l'utilisateur. Les parties 5 et 6 concernant les communications et communications par le bus de terrain sont à l'état de projet.

La partie 7 porte sur la logique floue et a été finalisée en août 2000.

La partie 8 donne les lignes directrices pour l'application et la mise en œuvre des langages de programmation et a été publiée en janvier 2000.

La partie 3 définit la syntaxe et la sémantique de 4 langages de programmation pour API, hormis le SFC ([12]). Celui-ci est à considérer comme un surlangage qui permet de structurer l'organisation interne des programmes et des blocs fonctionnels.

([10]) Automate programmable [10] : Système électronique fonctionnant de manière numérique, destiné à être utilisé dans un environnement industriel, qui utilise une mémoire programmable pour le stockage interne des instructions orientées utilisateur, aux fins de mise en œuvre de fonctions spécifiques, telles que des fonctions :
• de logique,
• de mise en séquence,
• de temporisation,
• de comptage,
• de calcul arithmétique,
pour commander au moyen d'entrées et de sorties TOR ou analogiques, divers types de machines ou de process.

([11]) langage : ensemble de règles qui permet de décrire les instructions à exécuter, la structuration des données et celles des programmes eux-mêmes, et d'organiser l'ensemble de ces informations dans la mémoire du processeur.

([12]) SFC : Sequential Function Chart.

Il est issu du Grafcet, défini dans la norme CEI 60848 [14] et a été conçu pour la description de comportements séquentiels. Il lui ressemble beaucoup, même s'il présente de grandes différences au moins au niveau de l'exécution car le SFC intègre la notion de priorités.

Les langages textuels

■■ Liste d'instructions *(Instruction List - IL) :* issu du langage AWL très utilisé Outre-Rhin, il est proche de l'assembleur pour la programmation des instructions de saut et boucles itératives.

■■ Texte structuré *(Structured Text - ST) :* issu du domaine informatique.

Les langages IL et ST sont plus proches du monde de l'informatique que du monde de l'automatique, et donc d'utilisation risquée pour une personne non-avertie, en charge des aspects sécurité et maintenabilité des applications [15].

Les langages graphiques

■■ Schémas à relais *(Ladder Diagrams - LD)* : dans l'esprit de la programmation à base de relais, il est fondé sur la présentation graphique de schémas constitués de contacts et de bobinages pour décrire le cheminement du courant.

■■ Diagrammes de blocs fonctionnels *(Functional Block Diagrams - FBD) :* issu de la régulation, il exprime le comportement de fonctions, blocs et programmes comme un ensemble de blocs graphiques interconnectés par le flot de signaux entre éléments de calcul. Il est destiné à résoudre les tâches de régulation et de commande, et permet de réaliser le traitement des signaux nécessaires à la commande d'un procédé.

Remarques

■ Dans les applications d'automatismes, deux voies principales se sont précisées progressivement [16] :
- les langages séquentiels pour les automatismes,
- les schémas fonctionnels pour la régulation des procédés continus.

Elles ont cependant en commun le concept de bloc fonction. Ce bloc est un élément de programme autonome catalogable et réutilisable.

Les deux approches diffèrent cependant sur le contenu et l'exécution du bloc fonction. Pour les langages séquentiels, un bloc est une opération exécutable une fois dans l'ordre du schéma.

Pour l'approche régulation, un bloc fonction est une fonction de régulation complète, telle qu'un régulateur PID.

■ La norme est un outil très important pour uniformiser les langages. Elle rentre dans un niveau de détails suffisamment bas mais qui contient, pour l'instant encore, des imprécisions, des incohérences dues à l'ampleur du projet. Elle n'est qu'une étape dans l'évolution des langages pour l'automatique, et ne constitue pas encore un état définitif [17]. Elle prévoit la possibilité aux offreurs de proposer des extensions non normalisées pour prendre en compte ces évolutions. En effet, les technologies utilisées dans le monde de l'automatique sont de plus en plus alignées sur celles de l'informatique.

■ De plus en plus, l'application ne sera plus programmée à partir d'instructions de base, mais par assemblage et configuration de composants métiers déjà spécifiés, programmés, testés, et intégrant non seulement l'aspect commande mais également des aspects supervision et maintenance. L'intérieur de ces composants n'est pas accessible au niveau de l'application.

3.5. Ateliers et outils de développement

Un environnement de programmation nécessite souvent plus d'une dizaine d'outils différents [18]. Ces outils se doivent d'être cohérents entre eux (il faut éviter la duplication de l'information). Cette cohérence doit porter sur les IHM (interface hommes-machines), les commandes, la présentation de l'information et les formats de données.

Des ateliers logiciels sont actuellement disponibles sur le marché et réunissent divers outils qui permettent aux intervenants d'atteindre un niveau élevé de qualité de programmation et une efficacité suffisante [19]. Ils servent surtout pour :
- spécification des fonctions,
- la description et la structuration des modules,
- la production du programme devant être exécuté,
- la simulation,
- la mise au point (test),
- la gestion de projet,
- la production des documents.

Au travers du recensement des outils informatiques d'aide à la conception de commande, il apparaît que la plupart des constructeurs proposent des outils dédiés à leurs équipements [20].

L'ADELI (Association française de génie logiciel) dénombre 22 ateliers de génie logiciel actuellement disponibles ([13]). Cet organisme note un déclin des méthodes de conception structurées au profit des méthodes objet.

4. Constats chez les intégrateurs et les utilisateurs

Les remarques citées ci-dessous ont été faites lors de réunions avec :
- des constructeurs de centres d'usinage à grande vitesse,
- des organismes de contrôle et membres de l'Institution,
- des comités techniques de l'EXERA (Association des exploitants d'équipements de mesure, de régulation et d'automatismes, Verneuil-en-Halatte).

4.1. Conception : méthode et langages

Constats

■■ Pour le développement de la partie applicative des logiciels (niveaux ③ et plus), aucune procédure particulière n'est mise en œuvre. Par contre, le personnel est spécialisé, il suit des stages de perfectionnement sur les nouvelles technologies comme par exemple les commandes de sécurité intégrée. Les compétences des concepteurs sont également utilisées, notamment pour les premières réalisations qui se font sous l'assistance du constructeur du système programmable.

■■ Dans les années 1980, chaque programmeur développait des applications en fonction de ses acquis. Sur certains produits, 50 % des programmes ont été refaits après mise en exploitation, en phase de maintenance.

■■ Imposer des règles strictes de programmation peut constituer un frein à l'innovation, mais aussi entraîner une baisse de technicité du personnel. Si ces règles sont bien conçues et adaptées, ce point de vue est discutable. En contrepartie, une personne « géniale » peut coûter très cher à une société, économiquement, ou d'un point de vue sécurité.

■■ Les outils et méthodes à disposition ne permettent pas de passer rapidement de la phase de spécification à la phase de validation. A ce titre, il faut tout de même relever la mise en œuvre d'approches théoriques innovantes comme les méthodes formelles, dans des domaines soumis à de fortes contraintes en matière de sécurité comme les transports. Cependant, elles n'ont pas encore percé celui des machines, dans lequel l'absence de méthodes de spécification ne permet pas d'automatiser les phases suivantes. Il s'ensuit un nombre important de documents générés dans les différentes étapes (analyse fonctionnelle, analyse organique, listing programme, liste de tests…). L'hétérogénéité des outils qui a permis l'élaboration de ces documents ne favorise pas la prise en compte des modifications qui surviennent inévitablement au cours d'un projet.

■■ Il est difficile d'obtenir une qualité de programmation satisfaisante. Quelques sources de non-qualité sont identifiables pour chacune des phases du projet :
- difficulté à spécifier de façon précise et complète le fonctionnement des machines lors de l'analyse fonctionnelle,
- difficulté à faire appliquer les règles de l'art,
- difficulté à capitaliser le savoir-faire,
- absence de standard sur la structuration des programmes,
- difficulté à gérer les modifications des spécifications après avoir dépassé la phase d'analyse fonctionnelle,
- manque de rigueur des développeurs dans la gestion des modifications à chaud, à la mise en route de l'installation.

Certains organismes de contrôle constatent également une baisse de la qualité logicielle, ainsi qu'une augmentation du temps de dépannage.

■■ En ce qui concerne les langages, on constate souvent une opposition de culture : les jeunes embauchés sont formés au Grafcet tandis que la génération précédente est plutôt orientée Ladder [21]. En France, les langages d'automatismes choisis majoritairement sont le SFC et le Ladder [22].

■■ Certains langages d'automatismes disposent de possibilités de programmation trop étendues qui rendent la maintenance plus difficile. Ainsi, une limitation de certaines d'entre elles a été réalisée chez quelques utilisateurs [23].

Recommandations

■■ L'élaboration du programme doit être fondée sur les principes suivants :
- rechercher une bonne séparation des fonctions,
- viser la simplicité, en évitant notamment les structures complexes de données ou de programmes,

([13]) Pour plus de renseignements, consulter le site Web : www.adeli.com

• s'assurer, dès la conception, de l'existence des moyens de test et de qualification,
• s'interdire toute modification « on-line » de programme après la mise en service,
• vérifier et tester les programmes établis.

■■ La rigueur du processus de développement est le fruit d'une bonne organisation, basée sur :
• la compétence des personnes,
• l'organisation humaine et technique du projet (plan de développement, procédure de maintenance),
• le formalisme poussé des opérations, en particulier la traçabilité de toutes les interventions,
• les mesures de gestion de projet (configurations, documentation,…), mais aussi planning et budget.

4.2. Simulation et validation

Constats

■■ La simulation est très utile pour le déverminage des logiciels et peut couvrir 90 % des problèmes. Les 10 % restants sont liés au fait que le modèle n'est jamais totalement représentatif. Il subsiste des représentations fausses ou incomplètes qui ne permettent pas de valider complètement les logiciels, d'où la nécessité de recourir à des tests réels sur la machine.

■■ Les méthodes de validation ne sont pas standardisées au niveau des organismes de contrôle.

■■ La vérification interne ou sur site par un organisme de contrôle n'est pas forcément sollicitée par l'intégrateur.
L'utilisateur peut toutefois entreprendre seul la démarche.

Recommandations

■■ La simulation d'une pré-installation s'effectue sans connecter l'automate programmable à la partie opérative - ensemble des moyens matériels opérant physiquement sue les matières d'œuvre ou les utilités (énergies, outils, fluides, outillages…) en vue d'assurer la production. Dans le cas d'un système complexe, un simulateur de procédé peut être requis.

■■ Il convient que les essais du logiciel d'application utilisateur comprennent :
• la procédure de chargement, de rechargement, de copie de sauvegarde et d'archivage du logiciel,
• la vérification des modules d'application et l'interaction des modules séparés,
• l'initialisation du système et des routines de redémarrage (à froid, à chaud, immédiat),

• la simulation de conditions d'erreurs envisageables (lignes de communication, erreurs provenant des opérateurs),
• les conditions limites,
• l'essai de critères.

■■ Il ne faut cependant pas négliger le test réel (incontournable) sur l'installation en grandeur nature pour éviter des modifications de dernière minute par l'équipe de mise en route.

4.3. Modifications

Constats

■■ Les mots de passe sur les logiciels applicatifs posent problème en terme de flexibilité des programmes.

■■ Certains constructeurs de machines sont amenés à délivrer leur mot de passe pour accélérer les opérations de maintenance et éviter des temps d'immobilisation trop longs des machines. Il s'ensuit que les paramètres de sécurité ne sont plus protégés. Malgré ce nouvel accès à ces paramètres, la complexité pour introduire une modification du logiciel est telle qu'à notre connaissance, aucun utilisateur n'a pris jusqu'à présent le risque de l'entreprendre, de peur de mettre la machine hors service.

■■ Le choix de délivrer les mots de passe à l'utilisateur en phase de maintenance n'est pas une généralité.

■■ Plusieurs niveaux d'accès existent et des mots de passe leur sont associés : exemple d'une commande numérique avec 7 niveaux et 4 clés *(cf. encadré 1)*.
Les paramètres liés à la sécurité sont entrés au niveau 6.

■■ Chez d'autres fabricants de machine, le client possède 3 clefs. C'est l'utilisateur qui les gère (codification des clefs par couleur et limites d'action dans le commutateur) sachant que le programme applicatif n'est accessible ni à l'opérateur ni au régleur, tandis que le programme pièce leur est accessible.

Recommandations

■■ Quand le logiciel d'application d'un AP est sujet à modification, il convient de prendre en compte les éléments suivants [9] :
• avant de modifier le logiciel, le programme courant doit être archivé,
• après avoir réinstallé le logiciel modifié, un essai de réévaluation doit être effectué. Il peut être nécessaire de procéder à nouveau à l'essai de réception de la première mise en service.

■■ Il convient de soumettre les modifications du logiciel d'application à des procédures équivalentes à celles qui sont appliquées à l'ingénierie, la documentation et aux essais du logiciel d'origine.

■■ Il convient que seul le personnel autorisé et qualifié ait le droit d'effectuer les modifications logicielles, en fonction des critères établis au départ.

■■ Toute modification nécessite de procéder à un essai de réévaluation afin de s'assurer que la sécurité/protection n'est ni compromise, ni mise en échec, ni contournée. L'utilisateur doit conserver un enregistrement de toute modification.

■■ Un logiciel applicatif critique peut être rendu non modifiable à l'installation en enregistrant cette partie de programme sur la mémoire morte ou en la rendant inaltérable, par l'utilisation de systèmes de verrouillage par le fabricant d'AP, le cas échéant.

ENCADRÉ 1

- **niv. 0 -** visualisation uniquement (pas pour certains menus), sans modification du programme pièce
- **niv. 1, 2, 3 -** définis par l'intégrateur
- **niv. 4 -** utilisateur (mot de passe utilisateur)
- **niv. 5 -** maintenance (mot de passe intégrateur)

Pas d'accès à la sécurité

- **niv. 6 -** intégrateur (mot de passe intégrateur)
- **niv. 7 -** fournisseur commande numérique

SÉCURITÉ

4.4. Maintenance

Constats

■■ Les logiciels d'exploitation évoluent trop vite (évolution des versions de Windows), ce qui génère un problème de suivi. Sur un parc de machines, il existe une trop grande diversité de ces logiciels d'exploitation.

■■ Le suivi des versions pose souvent problème : soit on choisit d'être à jour dans ses versions, ce qui implique un coût et une maintenance périodique des logiciels, soit les versions sont figées, ce qui implique un risque de devoir changer tout le parc au moment d'une défaillance non récupérable d'un des logiciels (« plantage »).

■■ Certains équipements, non encore stabilisés, évoluent en permanence et sont cependant commercialisés. C'est le cas de certaines commandes numériques à sécurité intégrée (2 à 3 changements de versions en 6 mois). Le suivi des changements de versions est subi et freiné par l'utilisateur autant que possible, mais il devient obligatoire lorsque l'apport est nécessaire à l'amélioration des performances de la machine. Les nouvelles versions sont intégrées, en général, pour répondre à des problèmes rencontrés.

■■ Il peut arriver qu'il y ait des changements de langages en cours de projet chez le fournisseur du système programmable. Cela peut générer des problèmes de compatibilité résolus par le constructeur du système programmable.

■■ Dans l'ensemble, les changements de versions posent le problème de suivi des versions lors de la conception jusque chez le client, lorsque la machine a déjà été livrée. A cela s'ajoute également une difficulté de remise à niveau permanente du personnel.

■■ La sauvegarde des logiciels applicatifs n'est pas encore réalisée correctement chez certains utilisateurs. Elle commence à se faire sur des serveurs, en réseau, avec un accès aux nouvelles versions des programmes pour les utilisateurs. La tâche leur est facilitée car ils accèdent à une boîte aux lettres pour la mise à jour des machines.

■■ Lorsque les modifications prennent de l'importance, dans certains cas, un comité statue sur la mise à jour d'une machine ou d'un parc complet chez le client. Du télédiagnostic par modem peut être mis en place.

■■ Le fabricant possède en général un fichier avec toutes les machines et tous les numéros de versions correspondant aux machines, ainsi qu'une sauvegarde de l'ensemble des logiciels au cas où le client les perdrait.

Recommandations

■■ Pour qu'un programme soit maintenable, il faut mettre en place des standards de programmation. Ceux-ci facilitent l'intervention des personnels de maintenance, puisque tous les logiciels d'un site obéissent aux mêmes règles de base. Cette standardisation limite aussi les règles de régression lors des modifications.

■■ Pour que cet investissement reste pérenne, il faut pouvoir mesurer le respect des obligations imposées par la démarche. Pour ce faire, le recours à un outil informatique de contrôle statique du code de chaque application en fin de réalisation est utile. Cet outil permet d'identifier les écarts par rapport aux obligations qu'impose l'application des standards. Cependant, un programme réalisé avec des règles standards peut être complètement « buggé ». Les tests sur plate-forme restent incontournables.

4.5. Documentation et formation

Constats

■■ Il persiste toujours des problèmes de langues et de gestion de la documentation fournie par les constructeurs de systèmes programmables. En plus, à la mise en service du dispositif, les commentaires sont fournis en français, mais après une intervention d'assistance, les programmes sont rechargés dans le langage du fournisseur puis laissés comme tels.

■■ En principe, les fabricants de machines délivrent un dossier technique complet sur la machine en plusieurs langues (souvent, français, anglais, allemand et espagnol) car un client ou un confrère peut avoir à gérer des modifications sur la machine. Les exigences prioritaires sont la concision et la précision de la documentation, indispensables lorsqu'une production en grande série est en jeu, les temps d'intervention étant très limités.

■■ La formation des intégrateurs de systèmes programmables sur le langage d'automatisme utilisé, et sur le système lui-même, est bien réalisée chez le constructeur de ces systèmes puis reconduite en interne chez le fabricant de la machine. Par contre, les intégrateurs sont souvent déconnectés de l'évolution des normes et sur ce point, ils souhaiteraient être conseillés ou épaulés par des organismes nationaux.

■■ Les utilisateurs finaux et le personnel de maintenance du client sont formés et sensibilisés aux risques induits par le fonctionnement des machines dangereuses, soit chez le fabricant, soit chez le client. Une habilitation leur est délivrée en fin de stage pour l'utilisation de la machine.

Recommandations

■■ L'acquisition de tout produit doit inclure un dossier d'exploitation-maintenance (manuel d'exploitation-maintenance) rédigé en français, dans lequel doivent être mentionnés les renseignements et prescriptions suivants :

- les caractéristiques techniques et opérationnelles du produit,
- les contraintes d'utilisation et d'installation,
- les conditions permettant d'assurer sa pérennité,
- les attestations, certificats confirmant le respect des normes ou règlements le concernant,
- la formation requise pour son application,
- le champ couvert pour la garantie.

■■ Il faut également insister sur l'importance de la formation du personnel, laquelle doit être générale (connaissances techniques générales, méthodes de diagnostic, d'aide à la maintenance, de remise en cycle…), mais aussi spécifique (dossier technique de l'installation, conduite et maintenance des équipements).

CONCLUSIONS

Cette étude fait apparaître de nombreux problèmes en matière d'exploitation de logiciels applicatifs. Ils sont d'ordres différents, et sont générés par des acteurs et facteurs distincts. Par exemple, certains problèmes sont liés au décalage entre l'offre du constructeur et le besoin de l'utilisateur, dû dans certains cas à la situation de monopole dans laquelle se trouve ce constructeur, qui n'est pas conscient des difficultés rencontrées lors de la mise en œuvre de son dispositif (changements de

versions de logiciels, interface homme-machine peu satisfaisante, documentation, maintenance,…). D'autres sont le résultat d'un manque de savoir-faire de l'utilisateur (absence de méthode de développement), et de considération dans la prise en compte de la sécurité chez l'utilisateur (modifications puis absence de procédures de validation).

Plus précisément, les sources de problèmes apparaissent à différents niveaux :

■■ La conception des logiciels applicatifs faite par des personnes autres que les concepteurs du système programmable (pour des applications liées de près ou de loin à la sécurité, mêmes très simples) est très délicate. C'est une problématique éminemment complexe. Même pour des programmeurs confirmés, il est nécessaire de s'entourer d'un certain nombre de méthodes, de règles, de précautions pour développer, tester et valider de tels programmes. Le recensement des moyens employés pour éliminer les erreurs de conception et de programmation a mis en évidence l'existence de normes, de méthodes et d'outils de validation. Mais ils ne satisfont pas l'ensemble des utilisateurs, car ils sont, pour la plupart, trop éloignés de leurs besoins. Il n'y a pas de solution simple pour mettre en œuvre du logiciel, notamment lorsqu'il s'agit d'une application à la sécurité.

■■ Le manque de méthode et de rigueur, à tous les stades de la conception du logiciel applicatif (développement, test, validation), se traduit par une diversité de pratiques non reconnues ni approuvées. A ce titre, il parait pertinent de faire porter prioritairement de futurs travaux sur des aspects du logiciel applicatif, depuis son développement jusqu'à son exploitation. Considérant que les systèmes programmables sont suffisamment encadrés (norme, validation…), il est préférable de privilégier les domaines concernés par les interventions des utilisateurs.

■■ Pour prévenir les conséquences de programmations ou de paramétrages non méthodiques, l'interface homme-machine conçue par les intégrateurs de systèmes programmables doit être adaptée aux besoins de l'utilisateur. Sur ce sujet, il faut s'attendre à ce que les exigences sur la conception de ces interfaces diffèrent en fonction des applications sur lesquelles elles sont mises en œuvre, mais aussi des individus qui l'utiliseront.

■■ Jusqu'à présent, il n'existe aucune méthode systématique, applicable à tous les systèmes programmables, permettant de s'assurer que le logiciel applicatif correspond parfaitement aux exigences de sécurité exprimées par les utilisateurs. L'étude de cet aspect ne pourra pas conduire à une méthode, ou procédure unique et exhaustive, valable pour tous les systèmes. Il est probable qu'elle se décline en plusieurs variantes en fonction des systèmes programmables considérés, des dispositifs sur lesquels ils sont implantés et de leurs applications.

Néanmoins, la taille et la complexité des logiciels applicatifs qui gèrent des sécurités (hormis les composants de sécurité) restent modérées, si on a pris la peine de les séparer du fonctionnel. Ainsi, dans la mesure où les événements non désirés sont identifiés, un niveau de confiance suffisant dans la gestion des sécurités par le logiciel peut à terme être obtenu.

Article reçu en septembre 2001,
accepté en février 2002.

BIBLIOGRAPHIE

[1] CEI 61511-1 - Sécurité fonctionnelle - Système instrumenté de sécurité pour le secteur des industries de transformation - Partie 1 : sommaire définitions, recommandations pour le logiciel et le matériel. *Genève, CEI, 1999, 121 p.*

[2] GIMELEC. Le recensement des automatismes. *J'Automatise, 1999, n° 6, pp. 49-52.*

[3] EN 292-1 - Sécurité des machines - Notions fondamentales, principes généraux de conception. Partie 1 : Terminologie de base - Méthodologie. *Paris, AFNOR, 1991, 36 p.*

[4] CT5 - Note relative à l'acceptation de certains automates programmables pour gérer des fonctions de sécurité sur machines. *Paris, ministère de l'Emploi et de la Solidarité, Document e23/APIDS/, 26 mai 1998, 5 p.*

[5] CEI 61508-4 - Sécurité fonctionnelle des systèmes électriques/électroniques/ électroniques programmables, relatifs à la sécurité - Partie 4 : Définitions et abréviations. *Genève, CEI, 1998, 26 p.*

[6] APAVE - TELEMECANIQUE. La sûreté des machines et installations automatisées. *Paris, SADAVE - CITEF, 1992, 333 p.*

[7] CEI 61131-1 - Automates programmables - Partie 1 : Informations générales. *Genève, CEI, 1992, 63 p.*

[8] CEI 61131-3 - Automates programmables - Partie 3 : Langages de programmation. *Genève, CEI, 1993, 411 p.*

[9] CEI 61131-4 - Automates programmables - Partie 4 : Directives pour l'utilisateur. *Genève, CEI, 1993, 62 p.*

[10] EXERA - Projet de guide de choix de simulateurs de partie opérative et de processus. Rév. G. *Verneuil-en-Halatte, EXERA, 2000,19 p.*

[11] CEA - Compte-rendu du groupe « Sûreté de Fonctionnement des Automates ». *Vandœuvre, Réunion du 30/06/2000, Document interne DCS/SPACI/SSI/00-458, 13 p.*

[12] BOUILLANE L. - La simulation de partie opérative : état de l'art, principaux avantages pour les concepteurs et utilisateurs d'automatismes. *In : Journée technique « Pour concevoir des logiciels de qualité », CETIM, 4 novembre 1992, pp. 9-21.*

[13] CLUB AUTOMATION - Journée d'information « Les langages dans les systèmes automatisés ». (09/2000). *Vandœuvre, INRS, 2000, Doc. interne IET–S/00CRR-069.*

[14] CEI 60848 - Langage de spécification GRAFCET pour diagrammes fonctionnels en séquence. *Genève, CEI, févr. 2002, 101 p.*

[15] CEA - Compte-rendu du groupe « Sûreté de Fonctionnement des Automates ». *Vandœuvre, Réunion du 09/12/1999, Document interne DCS/SPACI/SSI/00-21, 16 p.*

[16] NOURY P. - Généralisation des blocs fonctionnels dans un environnement distribué. *Revue des Electriciens et Electroniciens, 2000, 3, pp. 54-59.*

[17] GUEGUEN H. - Les nouveaux langages en automatismes. *Revue des Électriciens et Électroniciens, 2000, 3, pp. 36-37.*

[18] PRINTZ J. - Génie Logiciel. *Paris, Techniques de l'Ingénieur, 1997, pp. H 3208-11.*

[19] SOURISSE C., BOUDILLON L. - La sécurité des machines automatisées. Tome 2 : Techniques et moyens de prévention opératifs - Systèmes de commande - Utilisation des machines. *Institut Schneider Formation, collection Technique, 1997, 301 p.*

[20] BOUAZDI S. - Outils d'aide à la conception de commandes automatiques, rapport d'étude. *Senlis, CETIM, 1991, 25 p.*

[21] EXERA - Compte-rendu du CT « Groupe d'Evaluation des Langages d'Automates ». *Réunion du 24/05/2000. Vandœuvre, INRS, 2000, Doc. interne IET–S/00CRR-041, 3 p.*

[22] EXERA - Compte rendu du CT « Groupe d'Evaluation des Langages d'Automates ». *Réunion du 19/06/2000. Vandœuvre, INRS, 2000, Doc. interne IET–S/00CRR-044. 2 p.*

[23] EXERA - Compte-rendu du CT « API/ Superviseurs ». *Réunion du 18/01/2000. Vandœuvre, INRS, 2000, Doc. interne IET–S/00CRR-014, 3 p.*

■