Utilisation de méthodes formelles

La complexité des applications industrielles incluant des APIdS (Automate Programmable Industriel dédié à la Sécurité) est telle qu'il est nécessaire de mettre en pratique de nouvelles méthodes telles que celles proposées par la norme NF EN 62061 lorsqu'il s'agit de concevoir des fonctions de haut niveau de sécurité pour le domaine manufacturier. Cette norme ainsi que la norme CEI 61508, spécifiques à ces technologies, recommandent, par exemple, l'utilisation de méthodes formelles pour vérifier qu'un logiciel répond bien à son cahier des charges et contribuer ainsi à l'obtention d'un haut niveau de sécurité.

En phase de conception des logiciels, les méthodes formelles permettent, grâce à un langage particulier, d'exprimer très rigoureusement les propriétés issues du cahier des charges. Elles offrent ensuite la possibilité de prouver de manière automatisée que ces propriétés sont non ambiguës, cohérentes et non contradictoires. Elles garantissent par preuves mathématiques que ces propriétés sont respectées tout au long des étapes de conception. Les logiciels ainsi développés sont garantis sans défaut par rapport à ces propriétés. On cherche ainsi à limiter au plus tôt et en phase de conception les évènements dangereux qui pourraient apparaître en cas de comportement inapproprié du logiciel lors de l'exploitation de la machine.

A l'heure actuelle, de telles techniques ont été utilisées dans le domaine ferroviaire pour la partie sécuritaire de la ligne de métro Météor, dans le domaine énergétique, dans l'aéronautique, dans l'automobile pour la réalisation d'un régulateur de vitesse, pour des contrôles d'accès, mais pas encore dans le domaine manufacturier.

L'INRS a souhaité évaluer l'applicabilité et les contraintes d'utilisation de ces méthodes dans le cadre d'une application « machines » utilisant un logiciel développé avec un langage spécifique pour les automates programmables industriels.

Méthodes formelles employées

La démarche a consisté à utiliser deux méthodes de développement formel différentes :

Utilisation de la méthode B

La méthode B, inventée en 1980 par J.-R Abrial, utilise le langage B fondé sur les concepts mathématiques de la théorie des ensembles ; ce langage intègre des mécanismes de preuve et peut couvrir sans rupture tout un cycle de développement, jusqu'au code pour les éléments logiciels. Cette méthode est par ailleurs supportée par un outil logiciel.

Un développement selon cette méthode débute par la construction d'un modèle B reprenant toutes les descriptions du besoin. D'autres modèles sont ensuite élaborés par étapes successives, toujours à l'aide du langage B, jusqu'à l'obtention du programme exécutable. La cohérence des modèles obtenus à chaque étape et la conformité du programme au modèle initial sont garanties par des preuves mathématiques. Ce développement est illustré dans le schéma 1.

***Utilisation d'une méthode de vérification formelle* a posteriori**

La vérification formelle *a posteriori* consiste à décrire des propriétés (propriétés de sûreté en l'occurrence) qui devront être prouvées mathématiquement. Elle est basée sur le principe de « Model Checking » qui est un processus automatique consistant à vérifier l'équivalence entre un modèle formel des propriétés que doit vérifier le système et un modèle formel du système. Quelques travaux ont été effectués à l'aide de ce principe, appliqués aux langages de la norme CEI 61131-3 ainsi qu'aux langages de spécifications tel que le Grafcet.

Nous avons utilisé un logiciel de conception d'automatisme dans lequel un prototype de vérification formelle *a posteriori* a été intégré. L'avantage de ce prototype est de masquer, pour l'utilisateur, le formalisme mathématique de la vérification formelle. Ce type d'outil suppose dans un premier temps de concevoir le système puis d'insérer, à la fin de la conception, les propriétés que l'on souhaite voir respectées. Ce développement est illustré par le schéma 2.

Application sur un cas réel

On distingue le logiciel embarqué ou système qui est le logiciel interne à l'automate,

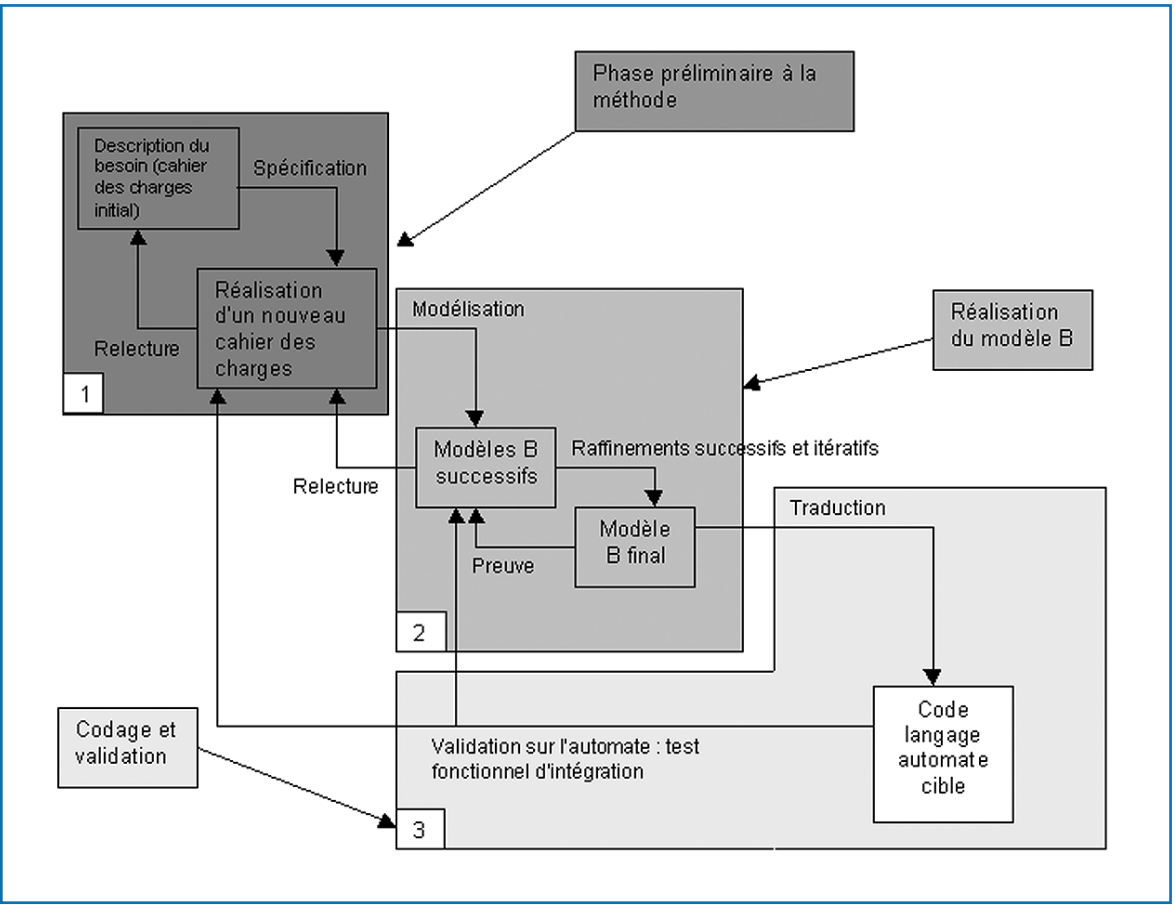


Schéma 1 – Représentation du cycle de développement mis en œuvre en utilisant la méthode B.

développé par le fabricant de l'APIdS, du logiciel applicatif plus communément appelé « programme automate », développé par l'utilisateur de l'APIdS par exemple le fabricant ou intégrateur d'une machine.

Pour cette étude c'est le logiciel applicatif d'une presse mécanique à embrayage-frein équipée d'un APIdS du marché qui a été développé à l'aide de ces deux méthodes.

Logiciel applicatif à l'aide de la méthode B

Réalisation d'un nouveau cahier des charges (cf. schéma 1, cadre n°1 « Phase préliminaire à la méthode »)

Cette partie a consisté à rédiger en langage naturel un nouveau cahier des charges de la presse qui rassemble toute l'information nécessaire pour le développement logiciel. Ce document doit décrire le système et il doit présenter de manière claire, non ambiguë et le plus complètement possible les exigences sur le logiciel sécuritaire contrôlant la presse.

Le nouveau cahier des charges contient à la fois une partie descriptive du fonctionnement de la presse et une partie définissant précisément les exigences fonctionnelles et sécuritaires du logiciel. Ces descriptions servent de référence au développement du logiciel contrôlant la presse, à la fois lors des phases de spécification et conception logicielle et lors des phases de test d'intégration et de validation.

La rédaction par l'expert (ou la personne ayant en charge la modélisation) B de ce nouveau cahier des charges lui permet de s'approprier le fonctionnement de la presse et de reformuler les exigences initiales. L'utilisation du langage naturel assure d'une part que ce nouveau cahier des charges peut être relu par les experts de l'INRS, et d'autre part un consensus entre les experts INRS et le modélisateur B sur le nouveau cahier des charges.

Modèle B : spécification et conception (cf. schéma 1, cadre n°2 « Réalisation du modèle B »)

Ce travail de spécification et de conception à l'aide de la méthode formelle B consiste à décrire les aspects fonctionnels du système à l'aide d'une succession de points de vue allant du plus général au plus détaillé : c'est le raffinement. Un modèle B constitué de plusieurs niveaux de raffinement entièrement prouvés a ainsi été réalisé. Ce modèle décrit l'ensemble de la presse et de son environnement en formalisant les exigences du cahier des charges rédigé à l'étape précédente.

Les premiers raffinements décrivent le système en détaillant les parties observables de l'extérieur : la presse, les mains de l'opérateur, la direction du mouvement du coulisseau, les protecteurs, l'arrêt au point mort haut.

Les derniers niveaux de raffinement décrivent le fonctionnement du système de manière très détaillée. Ils formalisent notamment le détail de chaque traitement unitaire du logiciel et de chaque action physique observable.

Le mécanisme de preuve effectué lors des raffinements successifs assure la cohérence d'une nouvelle itération vis-à-vis des modèles précédents, mais aussi au sein du raffinement en cours de création. L'intérêt de la méthode B est d'obtenir un modèle final prouvé et cohérent par rapport au modèle initial et à ses itérations.

Codage et validation (cf. schéma 1, cadre n°3 « Codage et validation »)

Afin de pouvoir importer le modèle B sur l'automate, il faut définir les principes de traduction du dernier raffinement B en langage de programmation de cet automate cible. Cette étude de traduction établit la faisabilité d'un outil de traduction automatique de B en langage automate. Dans notre cas, la traduction a été faite manuellement à l'aide de ces principes.

Ensuite vient la validation sur l'automate. Cette tâche consiste à charger le logiciel généré à l'étape précédente dans l'automate dédié à la sécurité de la presse et à passer des scénarios de test sur un banc de simulation permettant d'émuler les capteurs et actionneurs de la presse. Il est aussi possible de réaliser des défaillances de composant ou des incohérences de comportement.

L'avantage d'une telle validation est de réaliser une réelle validation du logiciel dans son environnement matériel (automate).

Cette validation reste nécessaire, malgré l'usage de la méthode B, afin de valider l'interface du logiciel avec le matériel, les conditions d'initialisation et les aspects temps réel. Plusieurs problèmes ont effectivement été détectés en phase de validation. Il s'agit de problèmes d'interface, d'un problème de cohérence d'initialisation du modèle B, de problèmes de validation des principes de traduction et de problèmes de traduction manuelle.

Logiciel applicatif à l'aide de la vérification formelle a posteriori

Modélisation du logiciel applicatif et conception détaillée (cf. schéma 2, cadre n° 1 « Modélisation »)

Le logiciel applicatif de la presse a été modélisé sous forme d'un ensemble hiérarchique et structuré de fonctions. Ces fonctions (également appelées composants génériques) peuvent être élémentaires ou composées. Une fonction élémentaire est décrite par un langage de la norme CEI 61131-3. Une fonction composée est décrite dans un éditeur d'assemblage de fonctions. Chaque fonction peut être utilisée plusieurs fois dans l'application. Les fonctions sont représentées dans l'outil sous forme de boîtes noires.

Validation du modèle (cf. schéma 2 cadre n°2)

L'outil utilisé pour la modélisation met en oeuvre un certain nombre de ressources pour aider l'utilisateur dans le développement de son application. Ainsi, il est possible par des mécanismes de simulation et de visualisation dans des Interfaces Homme Machine représentatifs de l'environnement de la presse de valider les composants élémentaires, donc de réaliser des tests unitaires, mais aussi l'intégration des différents composants logiciels dans les assemblages (équivalent des tests d'intégration).

Cette interface permet d'une part, d'agir sur des boutons représentant les différentes possibilités d'actions envisageables pour l'opérateur et d'autre part, de visualiser l'état dynamique des différentes variables et de simuler des défauts.

Réalisation des preuves

Une propriété à vérifier est modélisée par une variable correspondant à l'événement redouté. Cette variable porte sur les entrées ou les sorties et peut être simple, comme par exemple s'assurer que deux sorties du modèle ne seront pas vraies en même temps ou plus complexe en s'assurant qu'une combinaison logique de variables ne pourra jamais arriver.

Le moteur de preuves va automatiquement vérifier que la propriété est respectée et en cas de non vérification de la preuve, un exemple amenant à cette non vérification de la preuve est proposé et peut être simulé. Toutefois, cet exemple n'est pas exhaustif car l'outil fournit le premier exemple qu'il trouve et nécessite une interprétation afin de trouver la cause du problème qui peut-être soit une erreur dans la modélisation, soit un cas dont les conditions d'obtention ne sont pas réalistes.

L'outil utilisé pour réaliser les preuves est un prototype non commercialisé qui n'est pas encore suffisamment mature. Lors du passage des preuves, plusieurs problèmes ont été rencontrées, comme des contraintes fortes liées à l'utilisation du moteur de preuve qui peuvent perturber l'utilisateur dans sa modélisation et lui imposer des choix de modélisation qui ne sont pas forcément judicieux ; la difficulté à faire une preuve sur plus de deux variables ; la difficulté à faire des preuves sur des combinaisons temporelles entre variables ; la saturation de l'outil lors de la réalisation de certaines preuves, phénomène connu sous le nom d'explosion combinatoire et lié à la technique utilisée qui consiste à parcourir l'ensemble des états atteignables.

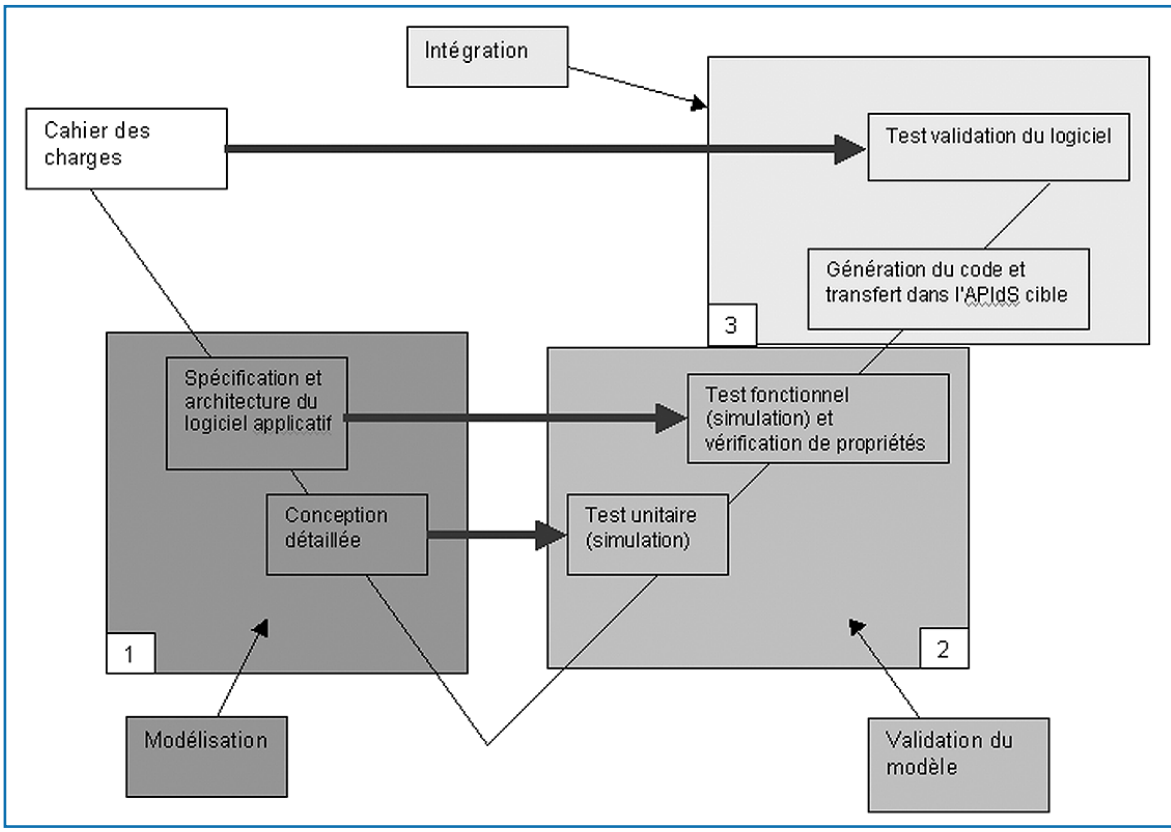


Schéma 2 – Représentation du cycle de développement mis en œuvre en utilisant une méthode de vérification formelle *a posteriori*.

La partie non formelle de l'outil a permis de poursuivre le travail afin de générer le logiciel et de procéder à son intégration dans l'automate cible (cf. schéma 2, cadre n° 3 « Intégration ») mais les problèmes rencontrés n'ont pas permis d'obtenir un modèle prouvé du programme de la presse.

Quelques constatations

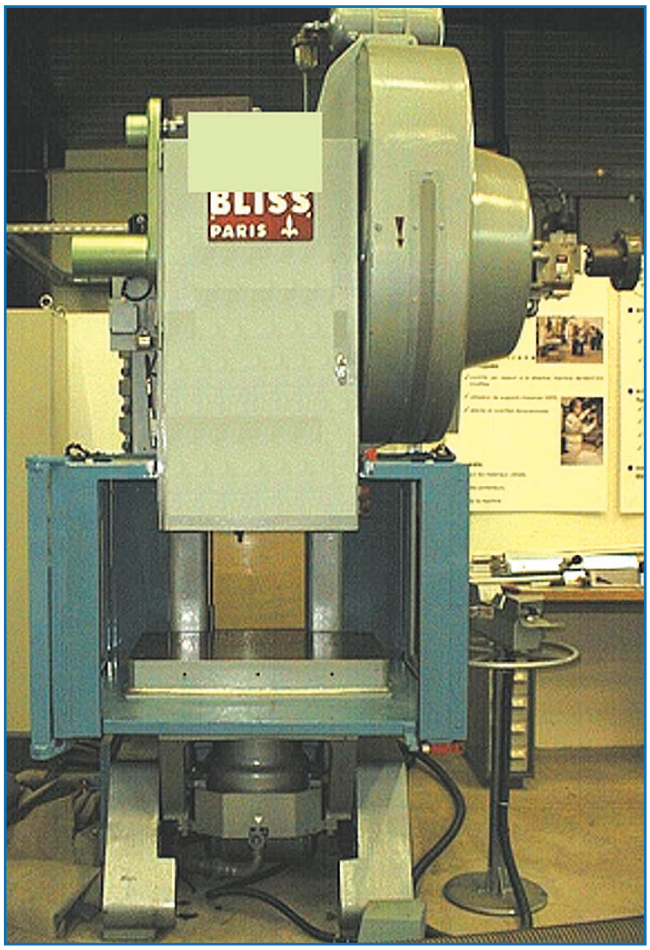
L'utilisation de la méthode B a amené plusieurs constatations :

– la place de l'expert en B est très importante pour la réalisation du modèle puisque de nombreux choix de modélisation ont été réalisés. Par contre, l'utilisation de la méthode B et de la preuve ont permis de s'assurer que ces options de modélisation sont bien cohérentes avec les règles de sécurité modélisées dans le modèle B,

– la présence d'un expert en B est nécessaire pour comprendre et mener à bien la démarche suivie d'où un coût de formation et d'utilisation de cette méthode,

– la traduction manuelle du dernier raffinement vers le langage de programmation de l'automate a entraîné des erreurs dans le code généré. Il est bien évident que dans le cadre d'un processus industriel, un traducteur automatique validé devrait être réalisé d'où un coût et des délais supplémentaires,

– le principal problème détecté lors de la validation est venu d'une mauvaise définition, en phase de conception, de l'interface entre les variables logicielles et l'implémentation matérielle (définition des entrées/sorties),



Presse mécanique à emboutir à embrayage-frein.

– une autre erreur de modélisation a été découverte pendant la validation ; cette erreur était due à un compromis entre l'investissement en temps, coût d'utilisation de la méthode et la volonté d'obtenir un modèle complet du système. La validation aura ainsi permis de valider certaines hypothèses de modélisation.

Une réflexion sur le développement de logiciel pour presse de façon classique et par l'utilisation de la méthode B peut être menée. Deux éléments peuvent la guider :

– Ce type de logiciel est réalisé par de petites entreprises. De plus, l'utilisation des Automates Programmables Industriels dédiés à la Sécurité orientent le concepteur vers l'utilisation de l'atelier logiciel spécifique à cet automate. Cet atelier logiciel peut contenir des blocs fonctionnels validés et certifiés (la gestion de la commande bimanuelle est par exemple assurée par un seul bloc). A l'heure actuelle, ces entreprises disposent de peu de temps pour développer et valider le logiciel. Il se pose alors le problème du niveau de sûreté du logiciel atteint. Apprécier ce niveau de sûreté reste difficile et constitue une des préoccupations de l'INRS dans son travail de validation de systèmes de commande.

– L'utilisation de la méthode B a nécessité environ quatre semaines pour le développement logiciel complet par des personnes non expertes de la presse (sans utiliser les blocs fonctionnels certifiés) et une semaine supplémentaire pour exécuter des tests fonctionnels. La preuve couvre de manière exhaustive les tests unitaires et les tests d'intégration et permet d'obtenir un niveau de sûreté élevé.

L'utilisation d'une méthode de vérification formelle *a posteriori* mène à un résultat plus mitigé. L'objectif était de pouvoir combiner les pratiques de l'automaticien avec l'utilisation d'un moteur de preuves, ceci de manière transparente pour l'utilisateur. Cet objectif n'a pas été atteint. En effet, les différents problèmes rencontrés avec le prototype de vérification formelle n'auront pas permis de construire des preuves pertinentes. L'utilisation de la vérification formelle *a posteriori* semble, à ce jour, difficilement applicable du fait du manque d'outil approprié.

Conclusion

Cette étude de faisabilité réalisée par l'INRS a permis d'appliquer deux types de développement formel du logiciel au cas d'un système manufacturier à base d'Automate Programmable Industriel dédié à la Sécurité.

Elle a montré que l'utilisation de la méthode B est très intéressante dans le cas d'applications sécuritaires puisque les modèles B ont été très peu modifiés pendant la phase d'intégration. Cette méthode a permis de s'affranchir de la validation des différents modules logiciels (tests unitaires). La phase d'intégration du logiciel et du matériel reste quant à elle une étape nécessaire dans tout développement logiciel, même utilisant des méthodes formelles, afin de s'assurer de l'adéquation avec le matériel.

Un des points qui pourrait être pénalisant dans le secteur manufacturier est la nécessité de faire appel à du personnel formé et compétent, voire expert en B, au départ mais aussi pour toute évolution ou modification du logiciel. Une telle contrainte destine l'utilisation de cette méthode à des entreprises disposant de projets d'envergure de conception de machines neuves dont l'investissement initial pourra être rentabilisé sur plusieurs projets. Il faut donc une politique volontaire de l'entreprise pour favoriser l'utilisation de cette méthode de conception puisqu'elle entraîne un changement de mentalité vis-à-vis du développement logiciel.

Concernant l'utilisation des méthodes de vérification formelle *a posteriori*, bien que cette solution soit plus abordable pour l'automaticien, l'outil utilisé dans notre cas n'est pas finalisé et cette technique n'est donc pas encore exploitable. L'utilisation de cette technique de vérification *a posteriori* nécessite, tout comme la méthode B, une nouvelle approche de la part du concepteur afin de déterminer les propriétés de sécurité qu'il voudra vérifier. De plus, il ne sera pas dispensé de réaliser des tests et de la simulation.

Pascal LAMY, Jean-Christophe BLAISE
Institut National de Recherche et de Sécurité