Formation à l'Industrie 4.0 par l'utilisation du Model-Checking et du Virtual Commissioning

Alexandre Philippot, Bernard Riera, Vinay Kunreddy, Serge Debernard

▶ **To cite this version:**

Alexandre Philippot, Bernard Riera, Vinay Kunreddy, Serge Debernard. Formation à l'Industrie 4.0 par l'utilisation du Model-Checking et du Virtual Commissioning. Colloque National S-mart/AIP-PRIMECA, Apr 2019, Montrichier-Albanne, France. hal-02151117

HAL Id: hal-02151117
https://hal.science/hal-02151117

Submitted on 7 Jun 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Formation à l'Industrie 4.0 par l'utilisation du model-Checking et du Virtual Commissioning

Alexandre Philippot, Bernard Riera
Research Centre for Science and Information Technology and Communication, CReSTIC (EA3804)
University of Reims Champagne-Ardenne
Reims, France
alexandre.philippot@univ-reims.fr
bernard.riera@univ-reims.fr

Vinay Kunreddy, Serge Debernard
Univ. Polytechnique Hauts-de-France, UMR 8201 – LAMIH
Laboratoire d'Automatique de Mécanique et d'Informatique Industrielles et Humaines,
F-59313 Valenciennes, France
VinayReddy.Kunreddy@uphf.fr
Serge.Debernard@uphf.fr

***Résumé*— Pour faire de l'industrie 4.0 un succès, il est indispensable de prendre en compte la composante humaine. Pour concevoir des contrôleurs sûrs de fonctionnement, les ingénieurs doivent disposer de méthodologies innovantes et adaptées à l'homme. Ce papier présente deux outils avancés – le Model-Checking et le Virtual Commissioning - qui pourraient modifier le travail des ingénieurs en automatique. La vérification de modèle est utilisée en tant que vérification hors ligne des propriétés structurelles d'une spécification de contrôleur. La mise en service virtuelle est utilisée en ligne pour tester la partie fonctionnelle.**

Keywords—Model-Checking, Virtual Commissioning, Automate Programmable Industriel, Industry 4.0

I. INTRODUCTION

L'usine du futur est un concept générique lié à la prise de conscience générale de l'importance de l'industrie manufacturière pour le développement des nations. Cette réflexion vise le maintien et le développement d'une industrie forte, génératrice de richesses et de création d'emplois. Ainsi, l'usine du futur doit prendre en compte à la fois les transitions énergétique, écologique, numérique, organisationnelle ainsi que sociétale, et se transformer progressivement pour devenir des industries plus durables et plus respectueuses de l'environnement [1]. En conséquence, le secteur est entré dans une phase de grand changement en considérant les technologies numériques comme un facteur clé d'avenir dans la conception de systèmes de production cyber-physiques. Ces derniers doivent permettre la mise en œuvre de nouveaux paradigmes d'automatisation afin d'améliorer le fonctionnement des installations en termes d'efficacité et de sécurité. Le Virtual Commissioning (ou mise en service virtuelle), la simulation de processus et des techniques telles que la vérification de modèles sont des approches clés visant à faire de l'usine du futur une réalité.

Ce papier présente deux outils avancés qui pourraient être utiles et utilisés par les ingénieurs en automatique : le Model-Checking et le Virtual Commissioning. Ces deux méthodes nécessitent des modèles de la partie opérative à différents niveaux d'abstraction. La première permet en premier lieu la vérification formelle hors ligne de certaines propriétés d'un modèle de contrôleur. La seconde consiste à utiliser un jumeau numérique pour effectuer une mise en service virtuelle et ainsi vérifier en ligne le programme de l'automate.

Cependant, ces méthodes modifient considérablement le travail de l'ingénieur. Il est montré comment ces méthodes avancées peuvent être utilisées et peuvent considérablement modifier le travail des ingénieurs en automatique. Pour réussir ce passage vers l'Industrie 4.0, il est donc nécessaire de prendre en compte la composante humaine et de proposer des méthodologies adaptées à l'Ingénieur pour concevoir des programmes automate sûrs de fonctionnement.

La section II de ce papier traite tout d'abors du principe de la vérification par Model-Checking, tandis que la section III traite du problème de modélisation. La section IV présente quant à elle le Virtual Commissioning comme un outil en ligne permettant de vérifier le programme de l'automate. Un exemple pédagogique illustre les propos du papier en section V avant de conclure et de proposer des perspectives de travail dans la section VI.

II. VERIFICATION PAR MODEL-CHECKING

Dans les systèmes automatisés, la plupart des accidents survenus dans l'industrie résultent d'erreurs de programmation [2]. Par conséquent, la vérification du programme de l'automate programmable industriel (API) avant sa mise en œuvre reste une tâche très importante lors d'un projet d'automatisation et constitue un sujet brûlant de l'usine du futur. Cette vérification doit concerner à la fois les composants fonctionnels et les composants de sécurité, non seulement pour l'équipement mais aussi pour l'opérateur. Dans l'approche proposée, la première vérification permet de s'assurer que les programmes de l'API respectent les spécifications fonctionnelles liées à la partie opérative à piloter, alors que la seconde consiste à vérifier si le système commandé peut être ou non exposé à des états dangereux conduisant à des dommages humains et/ou matériels. De nos jours, certaines techniques de vérification et de validation, telles que les tests et la simulation, sont disponibles en utilisant par exemple un cahier de recettes. La vérification consiste à exécuter manuellement chaque instruction contenue dans le cahier de recettes lors de tests en usine, puis à comparer les résultats obtenus avec ceux attendus. Cette vérification n'est donc pas automatique et demande beaucoup de temps en raison de la longueur des tests (parfois plus de 100 pages d'instructions). De plus, le problème de ces techniques est qu'elles sont focalisées sur la partie fonctionnelle du système mais ne sont pas toujours exhaustives en termes de sécurité, et qu'il peut exister une condition dangereuse non prévue et donc

non testée. Les tests et la simulation ne permettent donc pas de vérifier formellement la partie de sécurité des programmes d'automate exhaustivement dans la mesure où ils vérifient uniquement si les programmes automate répondent aux spécifications requises.

Pour résoudre ce problème d'exhaustivité, l'approche par Model-Checking (MC) peut être utilisée dans une étape hors ligne. Ce concept consiste à vérifier une propriété sur toutes les branches d'un modèle du système (ou du code de l'automate). La méthodologie consiste en une vérification hors ligne de plusieurs propriétés de programmes automates via un model-checker [3]. Ces propriétés peuvent concerner à la fois la structure du code et la sécurité.

La génération de codes à partir de modèles est un objectif industriel existant depuis de nombreuses années. Cependant, cette phase d'un projet ne peut être réalisée que s'il existe un modèle de haut niveau et généralisé de la partie opérative. Il existe des approches de modélisation telles que MDD (Model Driven Development) et UML (Unified Modeling Language). Le développement dirigé par modèle (MDD) est une approche de génie logiciel qui utilise un modèle pour créer un composant. UML quant à lui est un langage de modélisation largement utilisé dans le domaine du génie logiciel. Les experts utilisent UML pour analyser, concevoir et mettre en œuvre des systèmes logiciels, ainsi que d'autres processus métier. Chacune de ces approches de modélisation propose des outils de génération automatique tels que l'architecture MDA (Model Driven Architecture) à partir d'UML et l'architecture DSM (Domain Specific Modeling) à partir de MDD. Le problème de l'utilisation de ces approches sur des systèmes manufacturiers ou des systèmes cyber-physiques (CPS) est que ces systèmes sont complexes et tentent d'être progressivement flexibles. Cette flexibilité implique de comprendre le système dans sa globalité par l'ingénieur automaticien.

Dans la démarche de conception, la première étape de l'ingénieur consiste, à partir du cahier des charges, à modéliser formellement le fonctionnel du système avec des outils adaptés tels que le GRAFCET ou les Réseaux de Petri [4]. Cette abstraction est ensuite traduite dans un langage API (IEC 61131-3) pour la mise en œuvre [5].

Dans une seconde étape, il convient d'interpréter la spécification GRAFCET afin de vérifier certaines propriétés structurelles et syntaxiques hors ligne par Model-Checking. Par exemple, le modèle GRAFCET doit respecter certaines règles de syntaxe et d'évolution qui sont décrites dans la norme IEC 60848 [6]. Tous les détails sur le GRAFCET sont décrits dans la norme IEC 60848 [6]. Les propriétés de sécurité sont ensuite identifiées par un expert grâce une analyse dysfonctionnelle (utilisant une AMDEC par exemple) permettant de définir tous les états dangereux pour les produits ou la partie opérative. Pour cela, le comportement de l'installation doit être modélisé pour devenir une entrée du model-checker.

Dans les deux cas, si une règle (propriété) n'est pas satisfaite, le model-checker peut fournir un contre-exemple pour aider l'opérateur à corriger le programme (fig. 1). La section suivante aborde plus particulièrement la modélisation pour Model-Checker.

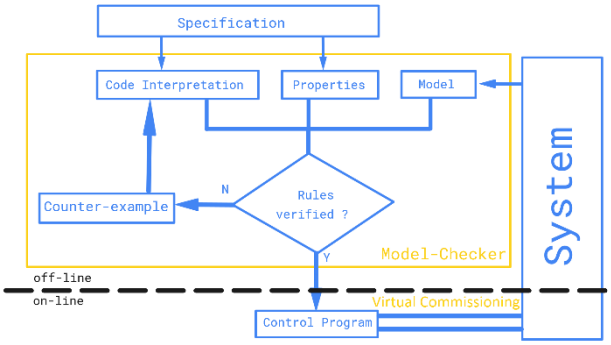

Fig. 1. Approches par Model-checking et Virtual Commissioning

III. Modélisation pour MC

A. *Interpretation du code API*

Un programme automate est défini classiquement dans un langage normalisé IEC61131-3. Cependant, un model-checker est un outil hors ligne qui ne peut recevoir un tel langage en entrée, mais une interprétation de celui-ci. Pour être considérée comme une entrée du model-checker Uppaal, la spécification GRAFCET est traduite dans un langage textuel ressemblant au langage Structured Text. Pour cela, on utilise la traduction classique du GRAFCET appelée programmation par auto-maintien [7] (Fig. 2).

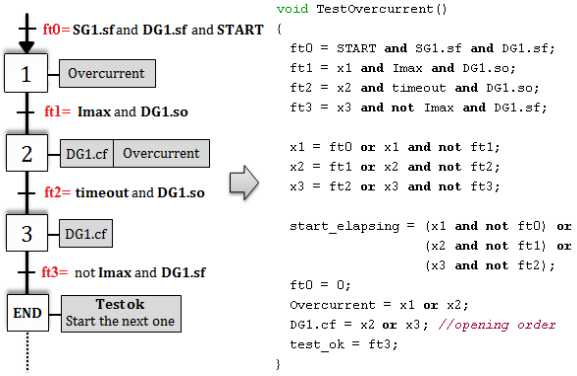

Fig. 2. Extrait d'une traduction d'un GRAFCET en un programme Uppaal

B. *Modélisation du système*

La modélisation de système nécessite quant à elle une analyse structurelle minutieuse des programmes d'automate, car les modèles doivent avoir le même comportement qu'un système réel. Si les ordres sont envoyés par le programme de l'automate, des événements tels que le changement de capteur ou l'interaction de l'opérateur avec le système peuvent se produire à tout moment. Par conséquent, un modèle qui génère des événements de capteur de manière aléatoire est proposé (Fig. 3). Ainsi, le capteur peut prendre deux valeurs booléennes possibles (vrai ou faux). Grâce à ce modèle, tous les états du système sont possibles.

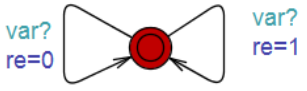

Fig. 3. Modèle d'une entrée booléenne

Par ailleurs, certains événements doivent être testés sur leur front montant (RE) ou front descendant (FE). En conséquence, un nouveau modèle est défini pour observer sur un cycle d'automate ce changement de valeur (fig. 4).

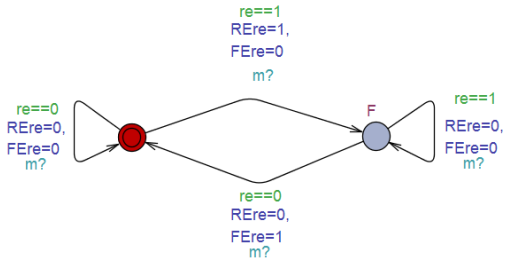

Fig. 4. Modèle de fronts.

Sur ces 2 modèles, on peut remarquer que 2 événements synchrones (*var* pour fig. 3 et *m* pour fig. 4) sont présents (symbole ?). Ces événements permettent d'introduire une évolution synchronisée au cours du cycle de l'API à modéliser.

C. *Modélisation du cycle API*

Le modèle de la fig. 5 synchronise tous les autres modèles du système grâce aux canaux de diffusion (symbole !). Le cycle de l'automate est modélisé et structuré comme une boucle. L'initialisation du programme et des paramètres est requise au cours du premier cycle de l'automate. Ensuite, le cycle est composé de 3 étapes dit « engagées » (symbol ©) permettant une priorisation de franchissement :

- Lecture des entrées (synchronisation par *var*) ;
- Lecture des fronts (synchronisation par *m*) ;
- Exécution du programme principal (GRAFCET mis à jour) suivie par une écriture des sorties.

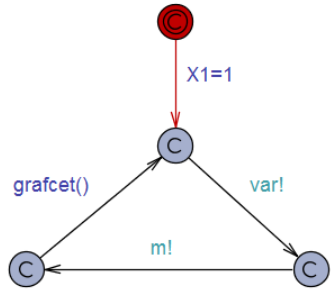

Fig. 5. Modèle Uppaal du cycle API.

Afin de réduire l'espace d'états, la plupart des états sont déclarés comme étant « engagés », de sorte que le temps ne peut s'écouler que pendant l'exécution du programme. Par conséquent, la durée d'acquisition des entrées et des sorties est considérée comme négligeable.

D. *Définition des propriétés*

Le langage de requête pour spécifier les propriétés est un sous-ensemble de CTL (computation tree logic) [8]. Les formules de chemin peuvent exprimer des propriétés d'atteignabilité, de sécurité et de vivacité à travers plusieurs syntaxes (E : existe éventuellement un chemin, A : inévitablement pour tous les chemins, <>: un état dans un chemin et []: tous les états d'un chemin) et le mot clé *deadlock*.

On peut noter que l'utilisation des techniques de Model-Checking modifie considérablement le travail de l'ingénieur. Il est absolument nécessaire de réfléchir à la manière de les intégrer au processus de conception du contrôleur. Il semble par exemple possible d'utiliser la trace du vérificateur de modèle pour donner des informations sur les erreurs. Des outils de coopération homme-machine doivent être conçus dans ce sens.

La section suivante aborde la seconde approche proposée en s'appuyant sur des outils de simulation sur lesquels la commande peut être évaluée.

IV. VIRTUAL COMMISSIONING

Le virtual commissioning [9] est un processus qui permet une évaluation complète du code de l'automate programmable, et de son éventuel déboguage, avant sa mise en service sur la partie opérative réelle. Un nombre croissant d'entreprises ont commencé à s'intéresser à cette technologie, car elle réduit le temps et le coût liés au lancement de nouveaux produits en permettant de tester différents scénarios pour valider les contrôleurs des systèmes manufacturiers. La mise en service virtuelle (virtual commissioning) est basée sur une simulation temps réel d'installation. Il semble possible d'utiliser des systèmes virtuels pour effectuer le virtual commissioning même si, théoriquement, un rendu 3D de haute qualité n'est pas nécessaire.

Le virtual commissioning peut être vu comme une partie du PLM (Product Lifecycle Management) qui est une approche intégrée et axée sur l'information de tous les aspects d'un produit depuis sa conception jusqu'à sa fabrication, son déploiement et sa maintenance [10]. Le PLM fournit aux entreprises un ensemble d'informations sur les produits pouvant être utilisé dans différentes applications, comme indiqué fig. 6.

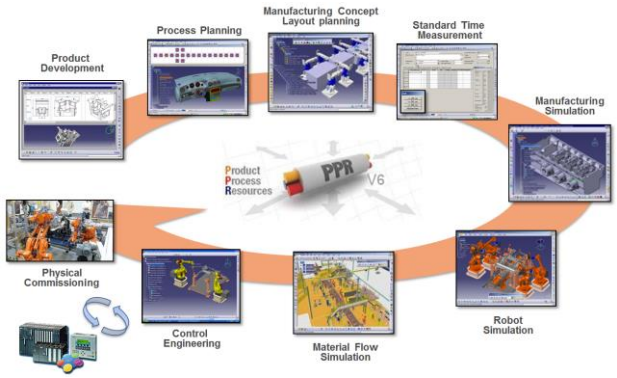

Fig. 6. Les phases d'un projet PLM chez DELMIA (Dassault Systèmes)

Le Virtual commissioning peut apporter plusieurs avantages :

- Vérification que l'équipement respecte le temps de cycle requis.
- Evaluation du code API.
- Développement rapide de standards.

Il existe plusieurs logiciels de simulation pour gérer la mise en service virtuelle. Cependant, les solutions proposées sont souvent très coûteuses et difficiles à utiliser car nécessitant un effort important de la modélisation du comportement de la partie opérative. En d'autres termes, la mise en service virtuelle est une solution intéressante, mais aujourd'hui risquée pour les PME (petites et moyennes entreprises). Il semblerait que les technologies de jeux vidéo puissent devenir une solution plus facile à utiliser et moins coûteuse pour la mise en service virtuelle. Cependant, cela implique certaines exigences :

- La physique (objets, actionneurs, capteurs) doit être certifiée. Dans le cas d'applications de virtual commissioning, il est nécessaire de garantir le comportement de l'installation simulée. Les progrès dans le domaine de la physique vont permettre dans un futur proche d'étendre les possibilités de simulation (température, fumée, liquide…). Par conséquent, la validation des systèmes virtuels sera une étape nécessaire.
- Des installations complexes avec des dizaines de points d'E/S sont généralement nécessaires pour mener des applications de programmation et de contrôle d'automates dans l'industrie. Les systèmes virtuels doivent devenir des installations synthétiques, qui doivent être modélisées en tant que systèmes distribués, permettant aux utilisateurs de visualiser et d'interagir avec différentes sections de l'installation, sur différents ordinateurs ou sur différents écrans.
- L'inclusion d'une plus grande variété de capteurs et d'actionneurs dans la prochaine génération d'environnements synthétiques est une autre exigence compréhensible pour la plupart des applications de formation dans les futures usines. Par exemple, bien que les capteurs et les actionneurs binaires soient très courants dans les applications industrielles, les capteurs et les actionneurs analogiques industriels se retrouvent souvent dans des applications pratiques. Aujourd'hui, les systèmes synthétiques disponibles n'incluent pas les applications échangeant des données analogiques avec l'automate de commande.

Les logiciels d'émulation deviennent courants dans la programmation d'API. Une autre exigence serait la connexion aux logiciels de d'émulation de l'automate programmable. Les systèmes virtuels, parce qu'ils intègrent une simulation de partie opérative, peuvent également être un outil intéressant pour la formation des opérateurs humains à l'aide d'interface homme-machine (IHM). Cependant, il est nécessaire de développer des outils et des méthodes spécifiques pour utiliser le virtual commissioning. En effet, le système virtuel n'est qu'une simulation. Il semble nécessaire de concevoir des outils homme-machine spécifiques permettant de coopérer avec l'ingénieur et de fournir des explications sur les erreurs et problèmes rencontrés. Le logiciel de simulation doit offrir de plus la possibilité de concevoir ses propres scénarios de formation. Il est donc intéressant que les futurs logiciels de formation incluent une bibliothèque de machines, pièces et équipements industriels à partir de laquelle les éducateurs pourraient développer leurs propres ateliers de formation. De plus, il est intéressant de noter que les modèles développés pourraient être exportées et importées, ce qui permettrait aux formateurs d'échanger de cette manière leurs scénarios de formation.

V. Illustration sur un Systeme Pedagogique

Pour illustrer les sections précédentes, nous proposons d'utiliser comme étude pédagogique, une porte de garage sectionnelle. Ce système est composé des éléments présentés dans la table I. La spécification est classique et la porte de garage est considérée comme sans inertie. Le bouton de la télécommande permet d'ouvrir, de fermer ou d'arrêter le mouvement de la porte en fonction de son énoncé précédent. Cependant, lors de la fermeture, si le capteur infrarouge détecte un obstacle, la porte doit être arrêtée. Une proposition de GRAFCET d'un étudiant est présentée sur la fig. 7. À partir de cette spécification, certaines propriétés sont d'abord vérifiées hors ligne avant d'implémenter le code de l'automate sur un modèle virtuel.

TABLE I. Entrees/Sorties de la portre de garage

Entrées	***Sorties***
os: open sensor	OP: Open order
cs: closed sensor	CL: Close order
ir: presence infrared sensor	
re: remote control button	

A. *Verification hors ligne par Model-Checking*

L'interprétation du GRAFCET en méthode auto-maintien est donnée fig. 8. Les modèles pour le Model-checker Uppaal sont présentés fig. 9. Les informations d'entrées booléennes *os*, *cs*, *ir* et *re* sont modélisées par 4 modèles. Le GRAFCET n'utilisant qu'un seul front montant sur la télécommande, il n'existe alors qu'un seul modèle de fronts. Le modèle de cycle de l'API est initialisé à l'étape X1. Plusieurs propriétés peuvent être vérifiées comme:

- E<>deadlock: Y a-t-il un chemin avec un état de blocage? (réponse : propriété non satisfaite);
- E<> OP and CL: Est-il possible d'activer en même temps les ordres *OP* et *CL*? (réponse : propriété satisfaite)

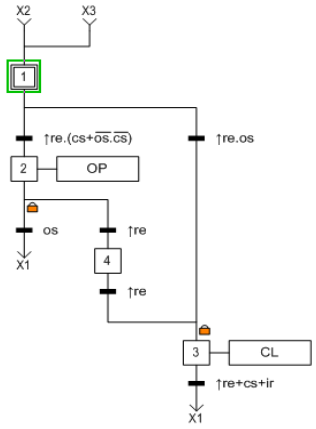

Fig. 7. Proposition étudiante du GRAFCET.

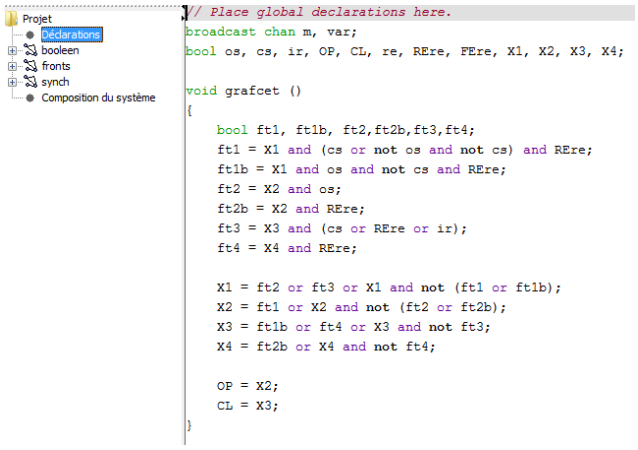

Fig. 8. Interpretation Uppaal du GRAFCET fig.7.

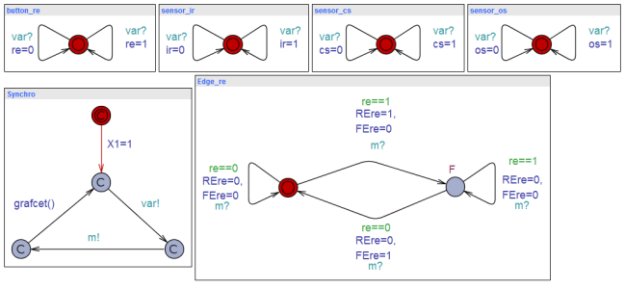

Fig. 9. Modèles de la porte de garage.

Nous pouvons noter que la deuxième propriété identifie la possibilité d'avoir 2 ordres opposés (OPEN et CLOSE) dans un même cycle API. Ce comportement n'est pas sûr et peut provoquer des perturbations ou des défauts. Pour informer le concepteur du problème, une trace de diagnostic est utilisée pour renvoyer un contre-exemple de la propriété et rejouer la simulation (fig. 10). Ainsi, l'utilisateur peut analyser la trace et trouver que l'origine de ce problème ne vient que lorsque l'étape X2 est active et que le vecteur d'entrée est (os, cs, ir, re) = (1, 0, 1, 1). Cette «mauvaise» proposition permet d'allumer la télécommande dans le même cycle automate que lors du changement de valeur du capteur d'ouverture. Pour résoudre ce problème, la condition de transition entre X2 et X4 doit être"*re* and not *os*" pour être sûr de l'exclusivité avec la transition entre X2 et X1.

La partie fonctionnelle doit ensuite être vérifiée grâce au Virtual Commissioning.

B. Vérification en ligne par Virtual Commissioning

L'utilisation de technologies basées sur les jeux vidéo (rendu graphique et audio, interactivité et attractivité) est considérée comme un moyen de promouvoir la « prise de conscience de la situation ». Des simulations peuvent être utilisées pour réduire les risques, réduire les coûts et optimiser le temps passé à acquérir de l'expérience. Une coopération scientifique et technique entre le CReSTIC de l'Université de Reims Champagne-Ardenne (URCA) et la société portugaise Real Games (www.realgames.co) va dans ce sens depuis 2008.

HOME I/O (https://realgames.co/home-io/) est le résultat de cette coopération dans le cadre du projet de recherche et développement de trois ans «DOMUS» (2011-2014), et partiellement financé par le MENESR [11]. HOME I/O est un logiciel de simulation en temps réel ou accéléré (fig. 11) d'une maison intelligente et de son environnement [12]. Ce logiciel a été conçu pour couvrir un grand nombre d'applications éducatives en technologie et en sciences de l'ingénieur. HOME I/O a été construit pour étudier la maison avec différents points de vue (automatisation, efficacité énergétique, maison intelligente…), dans son intégralité ou en sous-systèmes.

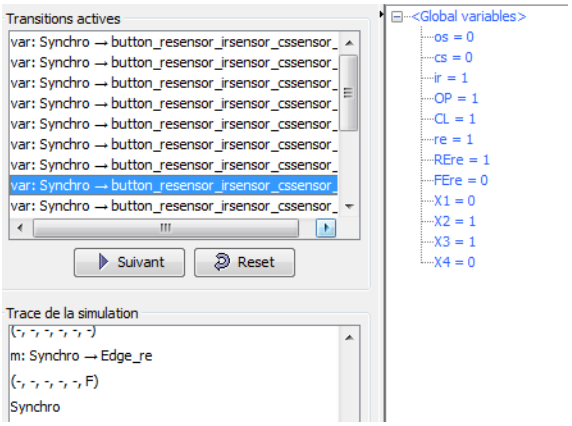

Fig. 10. Trace de diagnostic avec évolution des variables.

Fig. 11. Images illustratives de HOME I/O et de sa porte de garage

Pour contrôler la porte de garage, les étudiants proposent une implémentation du GRAFCET de la figure 7 sur un automate Schneider Electric programmée dans le langage de schéma à contacts Ladder Diagram (CEI 61131-3), qui est peut-être le langage le plus populaire en raison de sa correspondance avec les schémas électriques. Le programme sous le logiciel Unity Pro est présenté en fig. 12.

Grâce à une communication Modbus TCP/IP entre HOME I/O et le simulateur d'API de Unity Pro, le Virtual Commissioning peut être utilisé. L'opérateur peut tester plusieurs scénarios grâce à un cahier de recettes. Par exemple, un scénario consiste à arrêter l'ouverture de la porte avant sa fin de course par une première pression sur la télécommande (de X2 à X4). La deuxième pression doit alors induire la fermeture de la porte (étape X3). Cependant, grâce à la mise en service virtuelle, les étudiants peuvent noter que la porte est toujours dans une position intermédiaire. C'est une troisième pression du bouton qui provoque un mouvement d'ouverture. Après analyse,

les étudiants doivent pouvoir diagnostiquer l'erreur de programme et voir que sur la ligne 6 du programme LD (Fig. 12), un front montant manque sur *re*. De plus, on peut noter que la condition de transition entre X1 et X2 peut être simplifiée à « *re and not os* ».

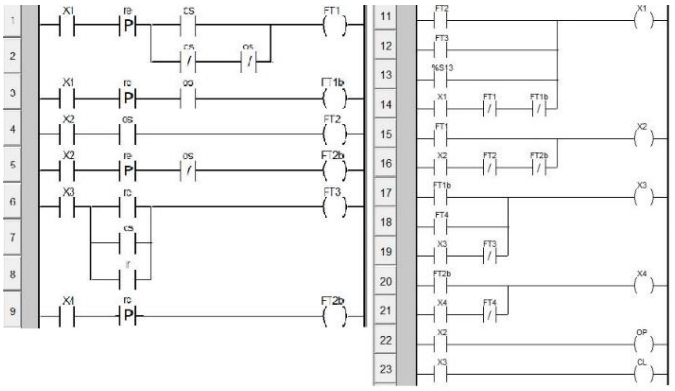

Fig. 12. Ladder Diagram sur Unity Pro (Schneider Electric).

Le retour d'évaluation de cette expérimentation effectuée sur un panel d'environ 50 étudiants de cycle ingénieur et master a permis de montrer les failles d'une analyse fonctionnelle sans et cela même sur un système simple. Il convient cependant de pousser l'expérimentation sur des personnes expérimentées issues du monde industriel.

VI. CONCLUSION

Ce papier propose aux ingénieurs automaticiens d'utiliser lors de la conception d'une commande, le Model-Checking pour une vérification hors ligne des propriétés structurelles d'une spécification et le Virtual Commissioning pour tester le fonctionnelle en ligne. Ces deux outils permettent de s'assurer que les programmes ont été suffisamment validés avant leur implémentation sur une partie opérative réelle. Ces méthodes avancées peuvent néanmoins considérablement modifier dans le futur, le travail des ingénieurs en automatisme, et il est donc essentiel de faire évoluer la formation de ces futurs ingénieurs aux exigences de l'Usine du Futur et plus généralement de notre société.

La sensibilisation, la formation au domaine du contrôle en automatisme doivent apporter d'autres compétences aux apprenants. Par exemple, d'autres outils avancés peuvent être utilisés dans le futur. La normalisation et l'analyse qualimétrique du code de l'automate doivent être par exemple développées. Un outil tel que PLC Checker proposé par Itris Automation Square, analyse automatiquement les programmes des automates et vérifie de manière exhaustive leur conformité aux règles génériques (ISO 9126 [13]). Cette norme décrit les exigences de: (i) lisibilité (commentaires, désignation de variable), (ii) fiabilité (toutes les entrées sont lues, toutes les sorties sont écrites, tous les défauts sont évalués, toutes les sections sont présentes et dans l'ordre spécifié), (iii) Modularité (pas de code mort et de sous-programme non appelé, les variables sont gérées correctement et au bon endroit). Un ensemble de règles de bonnes pratiques de programmation doit compléter les approches de model-checking et de virtual commissioning qui sont proposées dans ce papier. Des directives de codage sont disponibles par exemple sous PLCopen (www.plcopen.org). Il couvre la dénomination, les commentaires et les pratiques de codage pour améliorer la qualité et la cohérence des programmes d'un automate. Enfin, l'ensemble de ces outils ou méthodes doivent être complétés par des systèmes d'aide à l'ingénieur permettant de rendre les informations plus lisibles et plus compréhensibles [14]. Cela semble être un nouveau domaine très prometteur pour la recherche sur les systèmes homme-machine.

REMERCIEMENT

Les travaux présentés dans cet article sont réalisés dans le cadre du programme HUMANISM N° ANR-17-CE10-0009, financé par l'ANR ''Agence Nationale de la Recherche".

REFERENCES

[1] F. Lamnabhi-Lagarrigue, A. Annaswamy, S. Engell, A. Isaksson, P. Khargonekar, R. Murray, H. Nijmeijer, T. Samad, D. Tilbury, P. Van den Hof, « Systems & Control for the future of humanity, research agenda: Current and future roles, impact and grand challenges», *Annual Reviews in Control 43 (2017) 1-64.*

[2] H.B. Mokadem, B. Bérard, V. Gourguff, O. De Smet and J.M. Roussel, « Verification of a timed multitask system with UPPAAL », *IEEE Transactions on Automation Science and Engineering, Institute of Electrical and Electronics Engineers, 7 (4), pp.921-932, 2010.*

[3] G. Behrmann, J. Bengtsson, A. David, K.G. Larsen, P. Pettersson and W. Yi, « Uppaal implementation secrets », *7th International Symposium on Formal Techniques in Real-Time and Fault Tolerant Systems, Oldenburg, Germany, 9 -- 12 September 2002*

[4] H. Alla and R. David, « Du GRAFCET aux Réseaux de Petri, 2nd édition », *Hermes Science Publication, 1992.*

[5] IEC 61131-3 (2003). Programmable controllers – Part 3: Programming languages, Reference number CEI/IEC 601131: 2003.

[6] V. Carré-Ménétrier and J. Zaytoon, « GRAFCET: behavioural issues and control synthesis », *European Journal of Control, vol. 8, n°4, p.375-401, 2002.*

[7] William S. Levine. « The Control Handbook, Second Edition ». *CRC Press, December 2010. ISBN 9781420073669.*

[8] E.M. Clarke, E.A., Emerson, and A.P. Sistla, « Automatic verification of finite-state concurrent systems using temporal logic specifications », ACM Transactions on *Programming Languages and Systems 8 (2): 244–263, 1986.*

[9] A. Heidari and O. Salamon, « Virtual Commissioning of an Existing Manufacturing Cell at Volvo Car Corporation Using DELMIA V6 », *Master's Thesis, CHALMERS University of Technology, Goteborg, Sweden 2012, Report No. EX023/2012*

[10] W. Kuhn W, « Digital factory – integration of simulation rom product and production planning towards operative control », *20th European conference on modeling and simulation, ECMS 2006, Germany, 2006*

[11] M.J. Mayo. « Video Games: A Route to Large-Scale STEM Education? », *Science. 2009 Jan 2; 323(5910):79-82. doi: 10.1126/science.1166900.*

[12] B. Riera, F. Emprin, D. Annebicque, M. Colas, B. Vigario. « HOME I/O: a virtual house for control and STEM education from middle schools to Universities ». *11th IFAC Symposium on Advances in Control Education ACE 2016, Bratislava (Slovakia), 1-3 June 2016.*

[13] ISO/IEC 9126 Software engineering — Product quality was an international standard for the evaluation of software quality, 2001.

[14] P. Marangé, S. Debernard, F. Gellot, M. Pacaux, A. Philippot, T. Poulain, B. Riera et J.-F. Pétin. Approche de détection et d'explication d'erreur de commande par filtrage robuste. Hermès/Lavoisier, Journal Européen des Systèmes Automatisés, 48(4):339-372, décembre 2014.