INITIATION A LA QUALIMETRIE DE CODE D'AUTOMATE PROGRAMMABLE INDUSTRIEL

Alexandre Philippot[1], Bernard Riera[1], François Gellot[1], David Annebicque[1],
Raphaël Coupat[2], Eric Pierrel[3]
alexandre.philippot@univ-reims.fr

[1]CReSTIC (EA3804), UFR des Sciences Exactes et Naturelles, Université de Reims Champagne-Ardenne, Moulin de la Housse, BP 1039, 51687 REIMS Cedex 2, France
[2]Département des Installations Fixes de Traction Électrique, Section ZU 13 : Systèmes de Conduite et d'Automatismes, Direction de l'Ingénierie de la SNCF, 6 avenue François Mitterrand, 93574 La Plaine Saint-Denis
[3]ITRIS AUTOMATION SQUARE, 2 square Roger Genin, 38000 Grenoble, France

RESUME : Aujourd'hui, écrire un programme dans un Automate Programmable Industriel (API) selon un cahier des charges définit ne suffit plus. Assurer la qualité des programmes automates requiert de nouvelles solutions capables d'automatiser la vérification de la conformité avec les règles de codage et de réduire les coûts de maintenance. Dans cet article nous montrons comment des travaux de recherche menés au Centre de Recherche en STIC (CReSTIC EA3804) de l'Université de Reims Champagne-Ardenne (URCA) conduisent à des innovations pédagogiques dans le domaine de la formation à la commande des Systèmes à Evénements Discrets (SED). Dans ce cadre, un programme académique a été mis en place au sein du master Professionnel EEAII (Electronique, Electrotechnique, Automatique, Informatique Industrielle) de l'URCA et la société Itris Automation Square pour la vérification automatique de la qualité du code API au moyen de l'outil logiciel PLC Checker.

Mots clés : SED, API, qualimétrie, commande, simulation de PO.

1 INTRODUCTION

Au travers une spécialité « Systèmes Automatisés » au sein d'un master Professionnel Electronique, Electrotechnique, Automatique, Informatique Industrielle (EEAII), l'Université de Reims Champagne Ardenne (URCA) propose de former des étudiants à la conception des systèmes automatisés. Les diplômés de cette formation sont destinés à devenir des chefs de projets dans les domaines de l'automatique, de l'informatique industrielle, des systèmes d'informations de la production, de la robotique... Les étudiants passent donc de nombreuses heures à l'étude des différentes étapes du cycle en V (figure 1). Le cycle en V est un standard reconnu de l'industrie depuis des années, et l'une des étapes clés reste le codage [1].

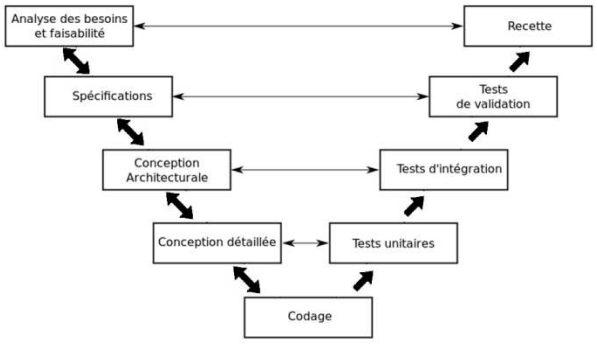

fig 1 : Cycle en V

La programmation des automates programmables industriels est une tâche que l'on peut qualifier de complexe car les programmes deviennent de plus en plus gros et évolués avec la nécessité de communiquer avec des applications de type MES (Manufacturing Execution System) et/ou ERP (Enterprise Ressource Planning). Au contraire de l'informatique où les outils d'évaluation de la qualité du code existent depuis de nombreuses années (ex : PMD, Sonar, Squale, …), on constate que dans le domaine des automatismes, les méthodes classiques de programmation ne permettent pas de relever ce défi. En effet, elles sont encore composées de nombreuses tâches manuelles et supportées par un faible nombre d'outils peu pratiques [2]. Par ailleurs, réutiliser des programmes automates existants est souvent impossible du fait de l'existence de nombreuses plateformes et langages incompatibles entre eux. Enfin, les approches qualité restent souvent informelles et, par conséquent, la qualité des codes est variable.

Dans cet article nous montrons comment des travaux issus de recherche menés au Centre de Recherche en STIC (CReSTIC EA3804) de l'Université de Reims Champagne-Ardenne (URCA) conduisent à du transfert « pédagogique » dans le domaine de la formation aux Automates Programmables Industriels (API).
En effet, fruit d'un partenariat recherche avec la SNCF, dans le cadre d'une convention CIFRE, sur la génération automatique de code API sûr de fonctionnement, un programme académique a été mis en place pour la vérification automatique de la qualité du code API au moyen d'un outil logiciel : PLC Checker, développé par la société Itris Automation Square (IAS).

Le papier présente plus en détail le contexte des travaux et l'application PLC Checker en section 2. La section 3 décrit la procédure de mise en place d'une

séance de TP autour de la qualimétrie de code API pour les étudiants de Master Professionnel EEAII. Nous conclurons ensuite ce papier autour du retour d'expérience.

2 VERIFICATION AUTOMATIQUE DE CODE API

En novembre 2011, une thèse CIFRE a démarré avec la Direction de l'Ingénierie de la SNCF autour d'une problématique de « synthèse de la commande sûre de fonctionnement et génération automatique de code API pour les systèmes distribués reconfigurables pour la gestion des installations fixes de traction électrique ». La première étape de ce travail de recherche passe par l'analyse fine des méthodologies et outils de conception des programmes API existants au sein de la SNCF, mais aussi par l'évaluation de la qualité du code API aujourd'hui développé par la SNCF. La recherche bibliographique qui a été menée a permis de mettre en évidence l'existence d'outils logiciels de vérification de la qualité du code API comme PLC Checker de la société Itris Automation Square (IAS) (http://www.automationsquare.com/fr/). Une version Online de PLC Checker est disponible en Free trial (http://www.plcchecker.com/).

2.1 La vérification de code API et PLC Checker

Cette société est un éditeur de logiciel, situé à Grenoble, qui fournit des outils de développement avancés pour les API permettant d'améliorer la qualité du code et de le générer plus rapidement. IAS propose différents outils reposant sur les techniques du génie logiciel : PLC Metrics, PLC Checker, PLC DocGen et PLC Converter. Ces logiciels acceptent en entrée des programmes écrits pour la plupart des automates du marché et les transforment en des abstractions permettant des analyses de haut niveau.

PLC Checker est un outil d'analyse de code API. Il analyse automatiquement des programmes automates [3] et vérifie de façon exhaustive leur conformité avec des règles génériques et si besoin, des règles spécifiques à un secteur d'activité. A l'issue de la vérification un fichier "résultat" est généré dans lequel les non-respects des règles sont consignés (figure 2).

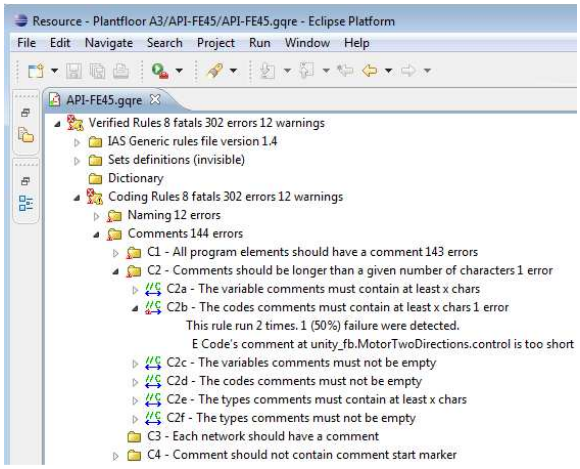

fig 2 : Exemple de fichier résultat PLC Checker

PLC Checker aide ainsi à répondre aux exigences des normes génériques telles qu'ISO/CEI 9126 [4], CMMI (Capability Maturity Model + Integration) ou des normes sectorielles telles que celles de la FDA (Food and Drug Administration) et de l'industrie du nucléaire.

PLC Checker supporte plusieurs familles d'API : Siemens Step 7, Siemens Step 5, Schneider-Electric PL7 Pro, Schneider-Electric Unity Pro, Rockwell Automation RsLogix 5000, PLC Open et Codesys v3.2. Les développeurs de programmes automates ont ainsi à leur disposition un environnement de travail qui permet le lancement des analyses et l'exploration des résultats. Il est important de noter que des outils comme PLC Checker ne sont pas conçus pour tester la qualité fonctionnelle du programme API développé. En d'autres termes, l'adéquation du programme API au cahier des charges n'est pas vérifiée.

2.2 Règles de codage des programmes API

IAS fournit un guide [5] proposant des règles de codage des programmes communes à tous les API. Les règles contenues dans ce document sont classées dans plusieurs catégories : Nommage, Commentaire, Ecriture, Structuration et Informations utiles qui, si elles sont respectées, contribuent à améliorer la lisibilité, la fiabilité et la modularité des programmes API.

2.2.1 Règles de Nommage

Les variables, routines (FC), blocs fonctions (FB) utilisés dans les ateliers automates portent des mnémoniques qui doivent suivre des règles assurant la lisibilité, la maintenabilité et permettant une plus grande pérennité du code. Parmi les règles existantes, on peut citer que :

- Tous les éléments constituant le programme doivent être nommés,
- Les noms des éléments du programme doivent avoir une taille d'au moins 4 caractères et d'au plus 30 caractères,

- Les mnémoniques des variables ne doivent pas faire référence à leur emplacement physique,
- …

2.2.2 Règles de Commentaire

La bonne utilisation des commentaires facilite la compréhension du code. Parmi les règles proposées dans le guide, on peut citer :

- Tous les éléments constituant le programme doivent être commentés,
- Les commentaires doivent comporter un minimum de caractères (7 pour les variables, 15 pour les codes) pour être significatifs,
- Chaque réseau doit être commenté,
- …

2.2.3 Règles d'Ecriture du code

Le code doit respecter des règles d'écriture qui vont permettre d'éviter des problèmes lors de l'exécution du programme. La règle la plus élémentaire concerne le fait que toutes les variables, à l'exception des Entrées et des variables système, doivent être écrites avant d'être lues.

2.2.4 Règles de Structuration

La structuration du programme est importante car la maintenabilité du code en dépend. Parmi les règles préconisées dans le guide, nous pouvons citer :

- Les sauts en arrière sont interdits,
- Une variable doit être écrite au sein d'une seule routine, ou d'une seule tâche,
- Une sortie physique ne doit être écrite qu'une seule fois par cycle API. Cette erreur est d'ailleurs très classique chez nos étudiants.
- …

2.2.5 Informations utiles

Les informations utiles ne sont pas tout à fait des règles mais plutôt des « bonnes pratiques » qui peuvent contribuer à l'amélioration de la qualité des programmes API. On peut citer parmi les informations utiles :

- Le taux de commentaires,
- La présence de code mort,
- La présence de code dans les commentaires,
- Les indicateurs de la complexité du code comme le nombre cyclomatique ou la complexité essentielle,
- Le nombre d'étapes du SFC
- Le ratio de code dupliqué sur l'application,
- …

L'utilisation de PLC Checker en enseignement se traduit par une meilleure formalisation des exigences qualité, des critères d'acceptation objectifs et un meilleur partage des bonnes pratiques de codage.

3 TRANSFERT PEDAGOGIQUE

Trop souvent négligé dans nos formations en EEA, les aspects liés à la qualité du code API sont primordiales lorsque l'on parle d'Ingénierie des Systèmes ou de maintenabilité à long terme. Les étudiants n'ont pas assez conscience de son importance dans l'industrie. Il nous semblait donc assez naturel dans nos formations d'utiliser des outils comme PLC Checker. Cela nous a conduit d'une part à développer courant 2012 un partenariat académique avec la société IAS dont tous les établissements d'enseignement supérieur peuvent maintenant bénéficier et, d'autre part, à introduire dans nos formations à l'URCA, tant au niveau des cours que des travaux pratiques, la qualimétrie des programmes API.

IAS met ainsi à la disposition de l'URCA 40 droits d'utilisation de PLC Checker pour la vérification des programmes automates développés par les étudiants lors de leurs travaux pratiques et projets. Ils pourront ainsi faire évoluer leurs programmes à travers différentes analyses.

3.1 Utilisation pédagogique de PLC Checker

Les étudiants de Master Professionnel en 2nde année qui suivent le module « Méthodologie de conception des systèmes » utilisent PLC Checker depuis la rentrée 2012. Ce module a pour objectif de former les étudiants à la conception des systèmes dans l'industrie à travers notamment une vue globale de l'Ingénierie des Systèmes et du suivi du cycle en V.

Dans une première phase, un cours magistral de 2 heures est donné sur la qualimétrie de code API et les différentes règles est introduit aux étudiants selon leur catégorie :

- Lisibilité - Les commentaires sont présents et bien formatés,
- Fiabilité - Absence de code mort et de sous-routine non appelée
- Modularité - Les variables sont manipulées correctement et au bon endroit

La seconde phase intervient sur 2 séances de travaux pratiques de 3 heures chacune. Les étudiants sont répartis en binôme sur 6 postes (2 groupes). Lors de la première séance, un système de gestion d'écluse leur est présenté en simulation afin de bien comprendre le fonctionnement en mode aval vers amont et amont vers aval d'une écluse, mais également afin d'identifier par auto-apprentissage les contraintes sécuritaires de ce type de système (figure 3). Pour cela, les étudiants utilisent le simulateur d'écluses disponible à l'adresse suivante :
http://ressources2.techno.free.fr/mecanique/ecluse/Index.htm.

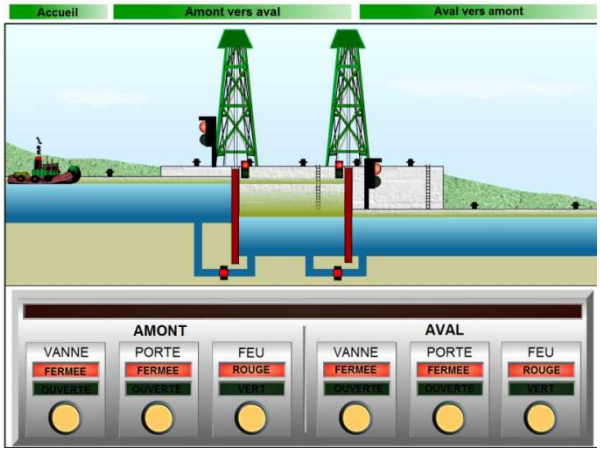

fig 3 : Exemple d'une écluse

Par la suite, un programme automate de ce système leur est fourni sans autre document. Ce programme est donné sous l'environnement Unity Pro de Schneider Electric et est défini sous différentes sections de programmation avec différents langages de la norme IEC 61131-3 (essentiellement en Structured Text et Ladder Diagram).

Durant la première séance, il leur est demandé de réaliser du effectuer du « reverse engineering ». Ils doivent proposer une analyse fonctionnelle (AF) qui aurait pu amener à ce résultat, et ce à travers les différents outils d'ingénierie des systèmes vus en cours (au choix modèles UML, SysML, SADT, GRAFCET, etc. ou un ensemble de ceux-ci). Le document devra ainsi permettre une lecture et compréhension simple du système et ainsi de guider l'utilisateur/technicien/automaticien dans la structure du programme.

Le deuxième temps consiste à utiliser PLC Checker afin d'analyser ce programme. Ils doivent alors retrouver de cette analyse les différentes règles non-respectées, corriger le programme et réitérer une analyse afin de voir l'évolution de la qualité du code à travers les indicateurs graphiques de PLC Checker (figure 4).

fig 4 : Exemple d'indicateurs graphiques avec PLC Checker

Ils complètent ainsi leur AF afin d'améliorer le suivi d'évolution du programme. Ils doivent s'apercevoir que la lisibilité et la maintenabilité du code automate obtenu au final sont améliorées par rapport au programme initial.

3.2 Retour d'expérience

Au regard des étudiants et de l'évaluation de ce module, ils comprennent tout à fait que même si quelques règles paraissent parfois triviales ou simpliste, elles restent tout de même cohérentes et nécessaires à tout utilisateur. Par exemple, la figure 5 retour une information de type C1 sur le fait qu'une variable du programme API ne dispose pas de commentaire. De même, si une adresse est utilisée sans être nommée, la règle N1 sera introduite (figure 6).

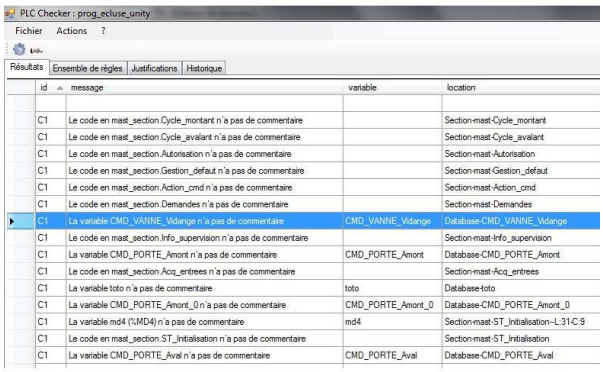

fig 5 : Exemple d'interprétation C1 de PLC Checker

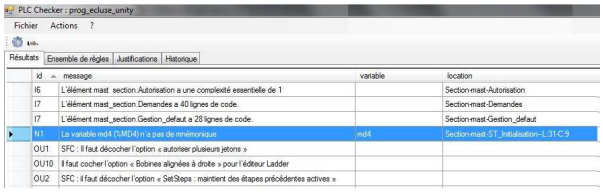

fig 6 : Exemple d'interprétation N1 de PLC Checker

Néanmoins, certaines règles et résultats sont encore assez vagues pour eux, et ils ne parviennent pas toujours à délimiter la frontière entre l'erreur et la notion de recommandations. Par exemple, l'erreur E1 de la figure 7 insiste sur le fait « qu'une variable est lue avant d'être écrite ». Il faut alors vérifier si la variable concernée est une entrée API où sinon cela signifiera qu'au premier cycle API, il est impossible de connaître avec certitude la valeur de celle-ci. L'opération consiste donc à double cliquer sur la ligne concernée qui permettra alors d'avoir un retour direct sur la liste des mnémoniques du code.

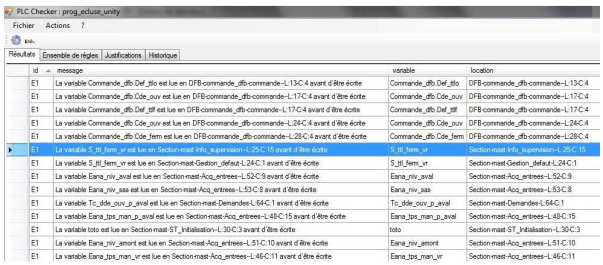

fig 7 : Exemple d'interprétation E1 de PLC Checker

En figure 8, On retrouve également un code à surveillé de type S8 concernant le chevauchement d'adresse d'une variable par une autre. Il faut alors double cliquer sur cette ligne pour avoir un retour à la liste des mnémoniques du programme, figure 9, pour vérifier s'il s'agit d'une opération volontaire ou non.

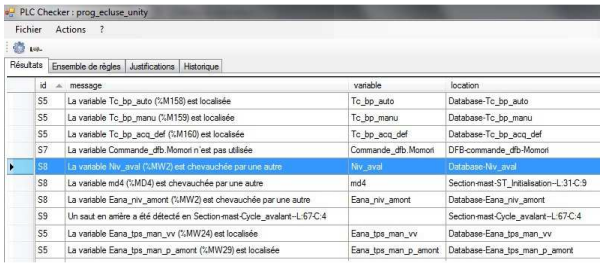

fig 8 : Exemple d'interprétation S8 de PLC Checker

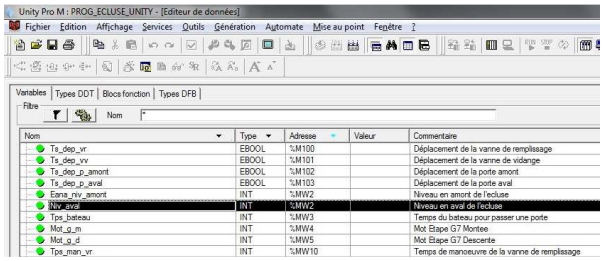

fig 9 : Retour de code après une interprétation S8 de PLC Checker

Par ce listing et la génération d'une nouvelle analyse de code par PLC Checker, les étudiants voient disparaître au fur et à mesure certaines lignes d'interprétations.
En outre, les indicateurs graphiques n'expriment pas toujours selon les étudiants l'analyse attendue. En effet, le pourcentage d'erreurs et de commentaires donnés ne sont reliés à aucune donnée exprimée quantitativement (10% d'erreurs sur quelle quantité d'information ?).

4 CONCLUSION

Cet article montre un exemple de transfert pédagogique suite à des activités de recherche de l'enseignant-chercheur. PLC Checker, en premier lieu utilisé pour la recherche, est devenu outil pédagogique dans le cadre de l'initiation à la qualimétrie de code d'automate programmable industriel pour la formation master Professionnel EEAII (Electronique, Electrotechnique, Automatique, Informatique Industrielle), spécialité Systèmes Automatisés de l'URCA.

Il est à noter que PLC Checker sera systématiquement utilisé lors de l'évaluation des projets nécessitant le développement d'un programme API. Par ailleurs, un questionnaire anonyme leur sera remis afin qu'ils puissent exprimer librement leur ressenti.

Bibliographie

[1] J. McDermid and K. Ripken. Life cycle support in the ADA environment. University Press, 1984.

[2] S. Biallas, S. Kowalewski, S. Stattelmann and B. Schlich. "Efficient handling of states in abstract interpretation of industrail programmable logic controller code", 12th Workshop On Discrete Event Systems (WODES'14), Cachan, France, May 14-16, 2014.

[3] IEC 61131-3. International Electrotechnical Commission, PLCs – Part 3: programming languages. Publication 611131-3, 1993.

[4] ISO/CEI 9126-1:2001. Software engineering -- Product quality -- Part 1: Quality model. Revised by: ISO/IEC 25010:2011.

[5] "Guide de codage des programmes automates", *Rapport interne, Itris Automation Square, Novembre 2011*, http://www.automationsquare.com.