Vers une Implémentation Sûre et Cybersécurisée du protocole OPC UA

C. Metayer, P. Humbert, P.-A Brameret

Vers une Implémentation Sûre et Cybersécurisée du protocole OPC UA

Towards a safe and cybersecure implementation of the OPC UA protocol

C. Metayer et P. Humbert et P.-A. Brameret
SYSTEREL
Les Portes de l'Arbois, Bâtiment A
1090 rue René Descartes
13100 Aix-en-Provence

Résumé

Le but de cette communication est d'analyser comment il est possible de développer un composant logiciel en respectant à la fois une norme de développement de sûreté et une norme de développement de la sécurité de l'information (respectivement (CENELEC 61508, 2010) et les Critères Communs (ISO/IEC 15408, 2012)). Notre analyse se porte plus précisément sur l'implémentation du protocole OPC UA (IEC 62541) dans le cadre du développement S2OPC issu du projet INGOPCS.

L'analyse des normes de sûreté et de sécurité montre des philosophies orthogonales entre les deux normes. Les différentes stratégies employées pour développer le projet en respectant le protocole OPC UA et les contraintes normatives sont développées, en particulier les méthodes formelles.

Summary

The goal of this communication is to analyze how to accommodate the constraints taken from two French standards. The first standard is the (CENELEC 61508, 2010) which aims at safe systems, while the second standard is the Common Criteria (ISO.IEC 15408, 2012), which aims at cyber-secure systems. The analysis focuses on the implementation of the OPC UA protocol (IEC 62541) in the development of S2OPC, which follows the research project INGOPCS.

The analysis of the standards shows orthogonal philosophies between safety and cyber-security standards. The remainder of the communication details the different strategies that were applied to implement an OPC UA toolkit that respects both standards, with a focus on formal methods.

1. Introduction

Comme tous les acteurs du domaine du développement des systèmes critiques au sens de la sûreté de fonctionnement, Systerel est contraint de répondre à la problématique de la cyber-sécurité.

Cette composante nouvelle est une partie intégrante du développement du produit Systerel S2OPC, qui est une nouvelle implantation du protocole OPC UA.

L'objet de l'article est de décrire les différentes contraintes prises en compte pour ce développement et les réponses mises en œuvre pour ce développement.

Dans un premier temps nous présenterons le protocole OPC UA. Nous présenterons ensuite les différentes techniques de développement logiciel mises en œuvre avant de nous intéresser à des techniques particulières issues des méthodes formelles. Enfin nous nous intéresserons à la conciliation des contraintes issues des normes de sûreté de fonctionnement et de la norme critères communs.

Dans la suite du document, le terme sécurité correspond à la notion anglo-saxonne de « security » et le terme sûreté à celle de « safety ».

2. OPC UA

Le protocole OPC UA est un protocole basé sur le principe clients serveur pour les applications d'automatisation industrielle. Offrant un moyen de communication du niveau actionneurs/capteurs jusqu'au cloud en passant par les systèmes de supervision, il est fortement utilisé pour le déploiement d'installations industrielles. Le consortium allemand Industry 4.0 a choisi le standard OPC UA comme norme de communication entre équipements dans son modèle d'usine du futur. Il prend ainsi une part importante dans le rapprochement de l'IoT et du monde de l'industrie.

C'est un protocole ouvert, indépendant d'une technologie propriétaire. Plusieurs implantations sont disponibles en opensource ou non et pour différentes plateformes proposant une réponse à la problématique d'interopérabilité. La fondation OPC (http://opcfoundation.org) a pour mission de spécifier le protocole et d'en gérer les évolutions. Afin d'encourager l'interopérabilité, elle propose des certifications et met à disposition des adhérents à la fondation des moyens de tests.

Le protocole OPC UA intègre nativement la cyber-sécurité en incluant des mécanismes de chiffrement. Le BSI (bureau fédéral allemand pour la sécurité des systèmes d'information) a conclu en avril 2017, après avoir réalisé des analyses de sécurité en profondeur de la spécification du protocole, à l'absence de faille de sécurité (https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/OPCUA/OPCUA.html). Cependant, il suggère différentes améliorations de la norme et de l'implantation de référence en C.

2.1. Protocole OPC UA

Les échanges OPC UA sont organisés autour d'un fonctionnement asymétrique client/serveur, comme illustré en Figure 1. Le serveur porte les données que le ou les clients peuvent lire ou modifier. Les données et leurs métadonnées sont organisées en graphe, appelé « address space ».

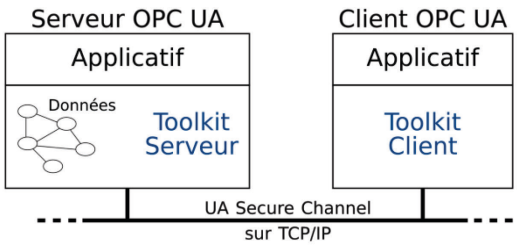

Figure 1. Aperçu d'un échange OPC UA

La partie protocolaire OPC UA, assurée par le « Toolkit », peut être séparée en deux familles de fonctionnalités : la messagerie et les services.

La messagerie assure les fonctions de connexions sécurisées, ainsi que de sérialisation/désérialisation des messages. Ces fonctionnalités font intervenir principalement des primitives bas niveau d'allocation et manipulation de mémoire, et des calculs mathématiques (chiffrement, signature).

Les services forment un ensemble de plus haut niveau, qui implément les services OPC UA. Les principaux services OPC UA sont la gestion des sessions utilisateurs, le parcours et la découverte de l'address space, la lecture ou l'écriture des données, la souscription aux données (notifications périodiques des changements de valeur des données), …

2.2. Projet INGOPCS et produit S2OPC

Si les travaux de la BSI n'ont pas révélé de faille de sécurité, l'ANSSI (Agence Nationale pour la Sécurité des Système d'Information), l'homologue français de la BSI, a conclu à la nécessité d'une nouvelle implantation du protocole. Les implantations existantes, n'ont bien souvent pas été développées avec un objectif de certification. La prise en compte a posteriori de contraintes normatives fortes est difficile et à coup sûr très couteuse.

L'ANSSI, conjointement avec Naval Group, est à l'initiative du projet INGOPCS (Initiative pour la Nouvelle Génération OPC-UA Sécurisée). Le projet a pour objectif la réalisation d'une implantation du protocole pouvant atteindre un haut niveau de certification (EAL4+ suivant les critères communs).

Systerel, leader du projet, s'est vu confier la réalisation de cette implantation. TrustinSoft et le CEA ont mis en œuvre les technologies basées sur Frama-C pour assurer l'absence de comportements indéterminés. Les autres partenaires, EOLANE, Schneider Electric, ATOS, EDF, couvrent l'ensemble de la chaîne d'approvisionnement des systèmes de contrôle-commande des fournisseurs de capteurs et automates jusqu'aux intégrateurs et aux utilisateurs finaux.

Le résultat du projet INGOPCS est industrialisé sous la forme d'un produit logiciel S2OPC. Ce produit répond aux contraintes suivantes :

- développement pour l'embarqué,
- qualification par la fondation OPC de l'implantation correcte du protocole,
- certification suivant les critères communs à un niveau EAL4,
- certification suivant la norme CEI 61508 à un niveau SIL2.

Cet ensemble de contraintes impose de développer un logiciel de grande qualité.

3. Processus de développement

3.1. Techniques standard de développement

S2OPC est publié en open source sur Gitlab. Bien sûr, ceci va pleinement dans le sens du principe de Kerckhoffs qui veut que la sécurité d'un cryptosystème ne doit reposer que sur le secret de la clef. C'est aussi la possibilité de mettre en œuvre différentes techniques, liées au monde du logiciel libre, qui visent à améliorer la qualité du logiciel et ce à moindre coût.

Gitlab offre aux projets open source différentes fonctionnalités. L'infrastructure permet de faire de l'intégration continue ; à chaque publication d'une modification, les tests et les vérifications sont rejoués. Elle contient également tous les moyens pour gérer un cycle de vie logiciel avec des systèmes de gestion des modifications et des relectures de code. Les utilisateurs du projet peuvent également remonter des problèmes, et suivre l'avancement des correctifs publiquement.

Tous les outils utilisés sont mis en œuvre au moyen de conteneurs Docker. On peut rapprocher un conteneur Docker d'une machine virtuelle légère. Il permet d'exécuter une application indépendamment de l'installation de la machine sur laquelle il est exécuté. On obtient ainsi une grande maitrise des chaines de production du logiciel. Comme la virtualisation, ceci permet de répondre à la problématique de la pérennité des outils. De plus, l'utilisation de conteneurs Docker signés offre des moyens de contrôler l'intégrité des chaines de production ce qui est une exigence pour un développement sécuritaire et aussi de répondre aux exigences de sûreté relatives à la maitrise des outils de production et de tests du logiciel.

Sur le plan des tests, Systerel met en œuvre des tests de fuzzing. Si cette technique de tests est particulièrement utilisée dans le cadre de développements sécuritaires, à notre connaissance, elle n'est pas utilisée dans le cadre de développements liés à la sûreté de fonctionnement. Le principe consiste à tester automatiquement un programme en faisant varier ses entrées de façon plus ou moins aléatoire. Cette approche permet notamment de découvrir des failles de sécurité. Elle ne permet pas en revanche de justifier de l'atteinte d'objectifs fonctionnels.

Les normes de sûreté de fonctionnement ou de sécurité imposent la définition de règles de codage qui minimisent les erreurs de programmation notamment en évitant des constructions risquées ou indéfinies. S2OPC est développé en langage C version C99 (ISO/IEC 9899, 1999). Ce choix est guidé par la volonté d'obtenir un

produit logiciel portable sur le plus grand nombre d'architectures. Néanmoins ce langage n'est pas dénué de constructions mal définies ou laissées aux choix des implantations des compilateurs. Systerel applique des règles de codage, principalement issues de celles du CEI CERT. Différents outils sont utilisés pour vérifier leur bonne application. En premier lieu, des compilateurs, quatre différents sont utilisés (gcc 8, mingw, clang 6, Visual Studio par l'intermédiaire de AppVeyor). L'usage de plusieurs compilateurs permet d'obtenir des contrôles différents.

Au-delà de l'application de simples règles de codage, Systerel utilise des outils d'analyse de code dynamique ou statique. Les outils d'analyse dynamique (UBSan/ASan/TSan) permettent de détecter des comportements anormaux à l'exécution (lecture de mémoire non initialisée, débordement de tableau, …). Les outils d'analyse statique permettent de détecter des mauvaises constructions indépendamment d'une exécution.

Ces différents outils et méthodes sont classiques dans le monde de l'open source. Dans le cadre du projet INGOPCS, une autre méthode d'analyse statique a été mise en œuvre conjointement par le CEA et TrustInSoft. Le CEA a utilisé Frama-C et le plugin eva (anciennement value). TrustInSoft a mis en œuvre son outil propriétaire TrustInSoft Analyzer. Les deux outils ont des origines communes. L'analyse a pour objectif de se prémunir des comportements non définis du C. L'analyse est basée sur l'interprétation abstraite et relève des méthodes formelles. Le niveau de confiance est accentué par le fait que ces outils sont basés sur des modèles mathématiques de l'exécution d'un programme C.

3.2. Analyses formelles

Les techniques et outils cités jusqu'ici poursuivent un objectif : détecter les fautes de programmation qui amènent des comportements indésirables. Mais en aucun cas ils ne permettent de s'assurer que le programme répond à un attendu.

Frama-C et son plugin wp est utilisé par Systerel dans le cadre du développement S2OPC. Ce plugin permet de réaliser des preuves de bonne implantation de fonction. Cette utilisation se situe au même niveau que des tests unitaires.

Techniquement, on exprime sous la forme de commentaires d'une fonction C :

- les préconditions des fonctions, conditions qui doivent être respectées par l'appelant et qui jouent donc un rôle d'hypothèse sous lesquelles l'exécution a lieu, et
- la post-condition, description de l'attendue de la fonction,

Cette technique est considérée comme peu intrusive d'un processus existant. Le programme est écrit en C puis dans un second temps, il est possible d'ajouter des commentaires permettant de faire de la preuve. Il faut noter cependant un prérequis : les fonctions sur lesquelles on souhaite faire de la preuve doivent être « bien écrites ». Elles doivent être petites, avec peu de structures de contrôle, sans « astuce » de programmeur C. Dans les faits, la preuve peut nécessiter des reprises de code, qui restent minimes.

Cette technique permet d'amener une première réponse au fait de s'assurer que le programme répond à son attendu. Elle reste cependant partielle et Systerel aujourd'hui n'envisage pas d'étendre son utilisation.

3.3. Développement formel

Systerel a utilisé une autre méthode formelle pour implanter une grande partie du protocole : la méthode B (Abrial, 1996). Cette méthode permet de produire le logiciel correct par construction. Elle définit un langage de spécification et de programmation ainsi que toutes les preuves à réaliser. Elle est entièrement outillée. Le programme en B est ensuite traduit en langage impératif standard. Pour S2OPC, nous générons du C. La traduction est possible vers un langage pour peu que l'on puisse définir des variables entières, des tableaux et des fonctions avec des instructions simples (affectation, condition, boucle, …).

On fait la distinction entre traduction et génération de code. Dans le cadre d'une traduction, il y a une association directe entre le code initial et le code traduit. Chaque variable est traduite par une variable similaire, chaque instruction par son instruction correspondante. Contrairement à une génération de code où l'on peut avoir un saut sémantique important, la traduction étant directe, le code obtenu est lisible et l'utilisation de debugger facilitée.

Le développement de la ligne 14 du métro parisien a permis l'émergence de cette méthode. Beaucoup en ont conclu que c'est une méthode « ferroviaire » pour logiciels « ferroviaires » et qu'elle ne peut être utilisée pour faire d'autre type de développement.

Le développement de S2OPC montre qu'il est possible de faire un usage de la méthode dans un autre cadre. Les logiciels sûrs sont en général cycliques sans allocation dynamique de mémoire et sans pointeur. Au contraire, S2OPC est événementiel, alloue dynamiquement sa mémoire et fait un usage important de pointeurs.

La mise en œuvre est aujourd'hui effectuée sans changer les outils utilisés pour le développement de logiciel sûr. A l'avenir, des modifications minimes des traducteurs sont envisagées pour optimiser l'écriture des modèles.

La méthode B autorise différentes preuves :

- la manipulation correcte de pointeur,
- l'absence de comportements non définis,
- la preuve de propriétés invariantes sur le programme,
- la preuve de bonne implantation des services de la norme OPC UA au regard de leur spécifications.

Elle n'est pas utilisée sur l'ensemble du programme car elle est peu adaptée aux manipulations bas niveau (codage et décodage de message binaire par exemple) ou aux interactions avec l'OS. Son utilisation dans ce cadre ne serait pas pertinente d'un point de vue industriel.

Sans vouloir être exhaustif, nous avons pu voir jusqu'ici que le développement de S2OPC met en œuvre un ensemble de méthodes et d'outils qui permettent d'obtenir un logiciel de qualité.

Cependant, cela ne fait pas du logiciel ainsi obtenu un logiciel qui soit sûr ou de sécurité. Il faut noter que ces caractéristiques ne sont pas des qualités intrinsèques du logiciel.

4. Contexte normatif

Que ce soit la norme CEI 61508 ou les critères communs, les deux normes incluent des contraintes sur l'ensemble du processus de réalisation et l'organisation qui portent le développement. L'accent est particulièrement mis sur la qualité et la démonstration de l'atteinte des objectifs normatifs.

Les deux normes ont des objectifs bien différents. La norme CEI 61508 concerne le développement de systèmes électriques et/ou électroniques et/ou électroniques programmables (E/E/PE) qui réalisent des fonctions de sûreté. Son application vise à rendre acceptables les risques résiduels liés à l'utilisation d'un système. Elle couvre l'ensemble du cycle de vie du système, depuis les premières phases amont de conceptualisation en passant par la conception, l'installation, l'exploitation et la maintenance, jusqu'à la mise hors service. Elle intègre les activités d'analyse des risques. Ces activités définissent les niveaux d'intégrités de sécurité SIL (Safety Integrity Level) qui doivent être atteints et leur allocation aux différentes fonctions du système. Le développement du logiciel fait partie intégrante du développement du système.

Les processus de développement de sûreté sont fortement appuyés sur des aspects documentaires. On cherche à montrer que le système est correctement conçu, testé, déployé, qu'il réalise ses objectifs techniques, et que les processus utilisés sont conformes aux niveaux d'exigences d'intégrités requis.

Les critères communs sont très différents de l'approche des normes de sûreté de fonctionnement. La norme a pour objectif de définir un cadre d'évaluation de la sécurité des systèmes d'information. Elle est structurée en 4 parties. Aujourd'hui les critères communs ne sont pas envisagés pour des systèmes complexes. Ils sont appliqués sur un équipement réseau, un petit équipement matériel, un logiciel ou une simple fonction. La norme définit un document essentiel dans une certification critères communs : la cible de sécurité. C'est l'objet de la première partie de la norme. Le contenu de ce document est normalisé. Il spécifie :

- les menaces contre lesquelles le produit ou système doit offrir une protection,
- des objectifs de sécurité à atteindre,
- le niveau d'assurance EAL (Evaluation Assurance Level) à acquérir dans la sécurité du système,
- les critères suivant lesquels l'évaluation sera faite.

Si la certification d'un système au regard d'une norme de sûreté de fonctionnement minimise les risques liés à son utilisation, les garanties d'une certification critères communs sont à lire dans la cible de sécurité, et vise l'intégrité des biens à protéger identifiés.

Les critères communs listent des exigences techniques qui contraignent la réalisation du système (objet de la partie 2 de la norme). Elles peuvent être interprétées comme des exigences portant sur un système type traitant de la sécurité de l'information. Il y a des exigences, par exemple, sur les mécanismes de journaux d'exécution, sur l'administration des fonctions de sécurité, sur l'authentification des utilisateurs, ou bien sur les mécanismes d'autodéfense. Ces exigences sont interdépendantes. La cible de sécurité doit sélectionner un sous-ensemble de ces exigences et assurer que les exigences interdépendantes sont toutes sélectionnées.

La norme contient un autre ensemble d'exigences qui est comparable aux exigences des normes de sûreté de fonctionnement. Elles portent sur la qualité du développement. L'ensemble du cycle de vie du système est concerné. L'environnement de développement au sens large (organisation, locaux, outils, …) est un des points auquel s'attachent les critères communs. Le risque d'attaque existe tout au long de développement y compris lors des phases amont. On peut noter, par exemple, le risque d'introduction de code malicieux dans des logiciels utilisés pour le développement.

Les exigences applicables au développement sont sélectionnées en fonction du niveau d'assurance qualité choisi (EAL). Il existe 7 niveaux correspondants à des niveaux de maturité.

Niveaux correspondant à des systèmes courants de bonne qualité et à la mise en œuvre de bonnes pratiques :

- EAL1 : testé fonctionnellement
- EAL2 : testé structurellement
- EAL3 : testé et vérifié méthodiquement
- EAL4 : conçu, testé et vérifié méthodiquement

Niveaux correspondant à des systèmes conçus avec une démarche et des méthodes de sécurisation particulièrement poussées :

- EAL5 : conçu de façon semi-formelle et testé
- EAL6 : conception vérifiée de façon semi-formelle et testée

Niveau répondant à des problématiques de stratégie nationale de sécurité :

- EAL7 : conception vérifiée de façon formelle et testée

Un niveau peut être complété d'un '+' qui indique que le niveau est augmenté par rapport à la définition normative.

Si les niveaux de SIL sont déterminés sur la base d'analyse de risque au niveau système, les niveaux d'assurance qualité ne sont pas sélectionnés en fonction de menaces. L'ANSSI propose 3 niveaux de qualification de produits.

Le premier niveau (niveau élémentaire) est basé sur une évaluation CSPN (certification de sécurité de premier niveau) qui est une certification propre à l'ANSSI.

Les deux autres niveaux (standard et renforcé) sont basés respectivement sur une évaluation critères communs à un niveau EAL3+ et EAL4+.

Le lien entre ces niveaux de qualification et un niveau de menace n'est pas explicite.

Par ailleurs, une différence notable est que les critères communs concernent directement l'évaluateur. Plusieurs exigences lui sont allouées et un pan entier de la norme lui est réservé : les analyses de vulnérabilité.

L'évaluateur intervient de deux manières. Il doit s'assurer :

- que le développement est conforme à la norme,
- que les objectifs de sécurité en regard des menaces et du niveau EAL sont atteints.

Le contrôle du respect de la norme est comparable à ce qui est réalisé par un évaluateur dans le cadre d'une certification SIL.

En revanche, le deuxième aspect n'a pas d'équivalent. La norme impose au concepteur de fournir à l'évaluateur des moyens de tests de l'objet soumis à l'évaluation. Il va ensuite déterminer des scénarios d'attaque pour évaluer les vulnérabilités du produit. Les niveaux EAL augmentant, l'évaluateur a de plus en plus d'information sur le produit, y compris l'ensemble des codes sources et moyens de tests. Suivant les niveaux EAL, les attaquants sont considérés plus ou moins puissants.

La méthodologie d'identification et de classification des scénarios d'attaque est décrite dans la quatrième partie des critères communs qui décrit plus généralement la méthodologie d'évaluation.

5. Orthogonalité du monde de la sûreté et de la sécurité

Avant de discuter de différences importantes entre le monde de la sûreté et celui de la sécurité, on peut noter que deux normes telle que la CEI 61508 et les critères communs sont miscibles. Bien sûr, la prise en compte de deux normes ajoute des activités supplémentaires et donc entraine une augmentation des coûts. Cependant, elles imposent globalement les mêmes activités et les mêmes documents :

- Les deux imposent, par exemple, de contrôler les outils mis en œuvre.
- Le déploiement doit assurer l'intégrité des objets et leur bonne configuration.
- L'ensemble du cycle de vie est concerné, des premières phases de spécification jusqu'au démantèlement.
- …

On peut alors envisager des plans de développement, ainsi que l'organisation associée, qui puissent être compatibles d'une norme de sûreté et d'une norme de sécurité.

Il existe cependant des différences essentielles entre les deux mondes. Une concerne le caractère même des systèmes : un système sûr le reste dans le temps. Alors qu'il suffit d'attendre pour qu'un système, considéré de sécurité à un instant donné, ne le soit plus.

C'est une des difficultés essentielles pour développer un système ou un composant logiciel comme S2OPC. La sécurité amène à des évolutions permanentes qui remettent en cause une certification SIL. C'est là sans doute l'orthogonalité la plus importante entre les deux mondes. Les systèmes ont des cycles de vie incompatibles.

On peut ainsi trouver dans les installations industrielles des équipements informatiques avec des systèmes d'exploitation obsolètes qui n'ont pas eu de mise à jour depuis des dizaines d'années. Du point de vue de la sécurité des systèmes d'information c'est tout à fait impensable.

Vis-à-vis d'un développement de sûreté SIL, même si la maintenance et les corrections sont intégrées aux normes, la moindre modification du système conduit à une reprise de cycle complète et une nouvelle certification. Vis-à-vis d'un développement sécuritaire critère commun, la découverte de nouvelles failles de sécurité et leur corrections fait partie intégrante de la maintenance du système et ne remet pas nécessairement en cause un certificat.

Une autre différence semble résider dans le niveau de maturité des approches entre les développements SIL et la cyber-sécurité.

Le développement d'un système au regard de la sûreté de fonctionnement est rationnel. Il peut s'appuyer sur des modèles mathématiques pour l'étude des défaillances. Les activités sont organisées en phases visant à contrôler l'absence d'oublis ou d'erreurs. La sûreté de fonctionnement introduit des exigences dans les phases de spécification pour que le risque soit correctement traité. La prise en compte des exigences et l'atteinte des objectifs du système sont contrôlées tout le long des phases remontantes. Finalement, rien n'est laissé au hasard, une fois fait l'hypothèse que la malveillance n'est pas dans le champ des analyses.

A l'opposé, le développement d'un système de sécurité s'appuie sur des techniques et méthodes qui sont beaucoup plus discutables et dont l'efficacité est difficile à démontrer. Comme illustration, on peut citer le test fuzzing qui est reconnu comme efficace pour le développement de système de sécurité. Mais on ne sait pas caractériser ou mesurer l'efficacité de l'approche.

Certains seront tentés d'expliquer la différence en arguant du fait que la sûreté de fonctionnement est une pratique plus ancienne et donc plus mature.

On peut aussi imaginer que la sûreté de fonctionnement concerne jusqu'à présent des systèmes dont on maîtrise relativement bien l'environnement. Ou encore, qu'on s'autorise à développer uniquement des systèmes dont on saura maitriser le développement. Sans doute cela a-t-il

conduit les normes de sûreté de fonctionnement à écarter la malveillance.

On peut se poser la question si, à l'avenir, la sûreté de fonctionnement ne verra pas des développements moins maitrisés de systèmes plongés dans des environnements beaucoup plus complexes. Par exemple des véhicules autonomes. Dans ce cas alors, la sûreté de fonctionnement utilisera peut être des techniques irrationnelles de la sécurité à l'image du test fuzzing.

6. Conclusion

Le produit S2OPC est à la convergence :

- de l'émergence du protocole OPC UA comme standard industriel,
- de la prise en compte du risque cyber dans les installations industrielles,
- du mariage forcé des contraintes de sûreté et des contraintes de sécurité,
- de la nécessité de conduire des certifications suivant une norme de sûreté de fonctionnement et une norme de sécurité.

Systerel a exposé dans cet article différentes méthodes et techniques de développement logiciel visant à augmenter la qualité du logiciel, afin de mieux maitriser les risques. Ces techniques se diffusent au travers de la communauté du logiciel libre.

D'autres techniques et méthodes plus confidentielles sont utilisées. Elles proviennent des méthodes formelles et permettent de montrer que leurs utilisations sont pertinentes économiquement et techniquement pour le développement d'un logiciel industriel standard.

S2OPC étant publié en open source, Systerel espère faciliter la diffusion des méthodes qui ont été mises en œuvre pour son développement.

S2OPC est aujourd'hui déployé sur des équipements industriels qui sont des passerelles réseaux SIL2.

Toutefois, le développement actuel ne concilie pas efficacement les deux mondes de la sécurité et de la sûreté. Systerel continue ses travaux visant à factoriser les analyses de sécurité et à optimiser des processus ou les ensembles documentaires pour répondre aux deux systèmes de contraintes normatives.

Néanmoins, même si il est possible d'optimiser les processus et les différentes analyses, ceci ne résout pas la problématique de la perte de la certification SIL dans le cas de mises à jour liées à la sécurité.

Deux pistes sont aujourd'hui envisagées. La première se place sur le plan contractuel. Un organisme de certification pourrait proposer des mises à jour rapides des certificats dans le cas de modifications mineures qui minimisent les délais et les coûts.

Une autre approche consiste à imaginer une évolution des normes de sûreté de fonctionnement. On pourrait imaginer, par exemple dans le cas d'une mise à jour d'un système d'exploitation, sous réserve de repasser l'ensemble des campagnes de tests sans encombre, le certificat serait maintenu.

7. Références

Abrial, J.-R. (1996). « The B-book - assigning programs to meanings ». Cambridge University Press.

ISO/IEC 9899: 1999 « Programing languages – C »

CENELEC IEC 61508: 2010 « Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité »

ISO/IEC 15408: 2012 V3.1 R4 « Common Criteria for Information Technology Security Evaluation »

IEC 62541: « OPC unified architecture »