Convention de subvention DGA/DS - ENSTA Bretagne

N°2015.60.0091.00.470.75.01

Année 2015-2016- fourniture 1

Etude n°5 :

Modélisation et découverte de systèmes embarqués pour la cybersécurité

Joel Champeau, Jean Christophe Le Lann et Bastien Drouot

ENSTA Bretagne

Lab-STICC UMR 6285

Joel.champeau@ensta-bretagne.fr

Table des matières

[INTRODUCTION 4](#_Toc447095732)

[Contexte 4](#_Toc447095733)

[Objectifs 4](#_Toc447095734)

[Cas d'usage 5](#_Toc447095735)

[APPROCHE 6](#_Toc447095736)

[Présentation 7](#_Toc447095737)

[Scénario 9](#_Toc447095738)

[9](#_Toc447095739)

[CHAINE D'OUTIL DETAILLEE 9](#_Toc447095740)

[Les Outils 11](#_Toc447095741)

[PimCA 11](#_Toc447095742)

[Le graphe d’attaque 13](#_Toc447095743)

[Rhapsody 14](#_Toc447095744)

[Role4All(TODO) 14](#_Toc447095745)

[Morphose(TODO) 15](#_Toc447095746)

[Génération de code Json à partir d’un modèle de cybersécurité 15](#_Toc447095747)

[Génération de code à partir d’un modèle de cybersécurité 16](#_Toc447095748)

[Perspectives 17](#_Toc447095749)

[REF 17](#_Toc447095750)

# INTRODUCTION

L’étude aborde la définition de l’interface entre matériel et logiciel dans un cadre de cyberdéfense et se focalise sur la modélisation incrémentale des informations de systèmes embarqués issues de documentation et d’expérimentations. L’objectif général du projet est d’unifier et mettre en cohérence la modélisation du système issue de différentes sources et les résultats d’évaluation de l’architecture numérique par simulation. Cette unification des éléments de description de l’architecture permet d’établir une carte d’identité de l’architecture ce qui permet d’évaluer les vulnérabilités de cette architecture numérique. Cette carte d’identité est souvent partielle et se construit itérativement après les différentes simulations.

D’avance nous signalons que certaines parties de ce document seront rédigées en anglais pour anticiper la rédaction d’articles.

## Contexte

La cyber-sécurité des systèmes numériques est un enjeu critique et sociétal extrêmement prégnant tant pour les systèmes militaires et civils. De nombreux travaux sont menés sur les méthodes de conception pour assurer le « correct by design » soit sur le volet matériel (cryptographie, résilience, etc.) ou soit sur le volet logiciel (bonnes pratiques de programmation, validation formelle, etc.).

En revanche très peu de travaux ciblent l’interfaçage entre matériel et logiciel, pourtant présent dans tous les systèmes et notamment les systèmes embarqués incluant les systèmes SCADA. Renforcer la méthodologie de découverte des connaissances de cette interface matériel-logiciel permet de mieux caractériser cette méthodologie de découverte et par conséquent mieux maitriser les systèmes à sécuriser.

Des méthodologies de capitalisation des connaissances sur les systèmes embarqués à sécuriser ou à découvrir, notamment sur l’interfaçage matériel-logiciel, restent à identifier et prototyper. Ces méthodologies doivent s’appuyer conceptuellement sur les cycles de conception traditionnels et bien maitrisés pour être transposé vers la sécurisation de systèmes connus et aussi à découvrir. Cet objectif doit permettre de réutiliser les connaissances antérieures sur les flots de conception et la capacité à mettre en relation différents points de vue, ceci pour les adapter et chercher à couvrir les enjeux de la cyber-sécurité.

## Objectifs

Le projet a pour objectif d’adresser les aspects de modélisation de l’interface matériel-logiciel dans un contexte de cyberdéfense.

Pour atteindre cet objectif il est nécessaire de définir une méthodologie outillée de découverte d'information et de capitalisation du système embarqué susceptible d'être attaqué. Les systèmes embarqués de type SCADA ou les architectures MPSoC représentent des bonnes illustrations pour ce type de préoccupations. En ce sens, la méthodologie de découverte peut se présenter comme une activité de reverse-engineering. Dans la découverte d'un tel système embarqué, l'articulation matériel-logiciel est cruciale : quels processeurs sont susceptibles d'intervenir dans la conception ? Quels bus ? Quelle est la structure de la memory map ? Quels périphériques? De quelles marques ?

Bien sur ces différentes questions sont également abordées lors des approches de conception descendante classiques, où les spécifications subissent des changements incessants et où les raffinements (introduction de détails architecturaux, etc) correspondent à autant d'explorations architecturales. Ces approches incrémentales sont particulièrement employées dans la conception  de SoC (system-on-chip), où 2 à trois niveaux d'abstraction différents ont cours : comportemental, architectural et détaillé [1,2,9,10].

Les informations utiles pour la conception ou la découverte sont le plus souvent dispersées à travers différentes sources d'informations hétérogènes (différents formats, langages ou outils). Afin d'en extraire un modèle utile du système et de l'outiller nous avons besoin de mettre en relation ces différentes informations. De plus ces informations sont généralement disparates et incomplètes, il faut donc les raffiner au fur et à mesure de la découverte du système. Pour ces modèles constitués par incréments, nous essayerons le plus possible de les conserver exécutables, malgré l'imprécision dans la connaissance du système réel. Par exécutables, nous entendons, dans un premier temps, simulés puis potentiellement synthétisés sur matériel de type FPGA pour obtenir les propriétés du système embarqué ciblé. Que ce soit la simulation ou la synthèse, l’objectif est de confronter au plus tôt les hypothèses prises sur le système et reboucler sur les informations de départ.

La modélisation outillée que nous nous proposons de mettre en œuvre repose sur une méthodologie agile basée sur des modèles hétérogènes (différents langages et différents niveaux d’abstraction) allant du niveau système jusqu’à la modélisation de la plateforme matérielle. Dès lors, il s'agit de trouver les moyens technologiques de reconstruire au plus tôt un modèle de système, même approximatif. Nous activons ensuite une pile logicielle avancée : fédération de modèles, génération de code, synthèse comportementale, estimation de paramètres, etc… permettant de simuler, estimer, prototyper une architecture de système embarqué et ainsi infirmer ou confirmer les hypothèses capitalisées dans les modèles. La manipulation de plusieurs niveaux d’abstraction et multipoints de vue dans des modèles nous permettra au final de renforcer la capitalisation des connaissances de l’interface matériel - logiciel.

## Cas d'usage

Supposons une application sensible de traitement d'un flux vidéo permettant la détection de formes(drones, personnes...). Cette application est un système cyber physique(CPS) [Lee] qui se compose d'un FPGA pour la récupération et pré-traitement du flux vidéo, lui-même relié à un processeur au travers de son banc de registres. Il est possible d'accéder au processeur par communication réseau via une liaison Ethernet ou WIFI.

Un assaillant envisage une attaque de ce système via le réseau Ethernet, afin notamment d'y récolter des informations précises sur le mécanisme qui entraîne un signal de détection de la part du FPGA, lui permettant ainsi de s'assurer que certaines formes seront indétectables par le système.

Pour pouvoir mener à bien son attaque l'assaillant doit récolter et capturer toutes les informations pertinentes concernant le système visé ainsi que sur son environnement, et ce à différents niveaux de détails. Parmi ces informations, on retrouvera des informations structurelles et comportementales. Les informations structurelles permettent de décrire les composants du système et leurs relations d'un point de vue statique, alors que les informations comportementales décrivent l'évolution du système et de ses composants au contact d'un environnement.

En s'appuyant sur ces informations l'attaquant peut ensuite, suivant sa propre expertise, concevoir différentes hypothèses de scénarios d'attaques. Un scénario est un ensemble d'activités automatisées ou manuelles qui permettent d'obtenir une réponse du système attaqué. A chaque mise en œuvre d'une attaque de nouvelles informations sont apportées par la réponse du système attaqué, et les hypothèses raffinées.

La réalisation de l'attaque suit donc une certaine méthodologie, en affinant progressivement sa représentation du système l'assaillant peut améliorer ses hypothèses d'attaques jusqu'à un résultat satisfaisant.



ARM

FPGA

Soc

Caméra

Ethernet

Bus

Figure 1: Cas d'utilisation

# APPROCHE

La méthodologie est selon nous en parfaite adéquation avec des approches de conception descendante (top-down) classiques, où les spécifications subissent des changements incessants et où les raffinements (introduction de détails architecturaux, etc) correspondent à autant d'explorations architecturales. Ces approches incrémentales sont particulièrement employées dans la conception  de SoC (system-on-chip), où 2 à trois niveaux d'abstraction différents ont cours : comportemental, architectural et détaillé [1,2,9]. Ce projet nous permet donc d'envisager une stratégie de capitalisation des connaissances d’un système, par analogie et transposition, selon une ingénierie (conceptuellement) traditionnelle : l'ensemble du cycle en V est alors applicable. La phase de remontée de ce cycle permet par exemple de confronter les hypothèses retenues lors de la constitution du modèle synthétique avec le système réel ciblé par une attaque. Cette phase peut par ailleurs constituer une étape d'incrément dans la connaissance du système : ce qu'il est et peut être surtout ce qu'il n'est pas.

## Présentation

Dans notre approche nous souhaitons modéliser les informations du système tant d'un point de vue structurel que comportemental et ce potentiellement au travers de différents langages (UML/Rhapsody, SysML, Pimca). D'autres informations pourraient être nécessaires(mot de passes, adresses IP...) qui pourraient être fournie par d'autre modèles ou sources d'informations(feuilles excel, bases de données).

Le système embarqué étant par nature multi-points de vue et intégrant différents niveaux d’abstraction, la modélisation du système repose sur une approche de modélisation hétérogène. Cette modélisation doit donc intégrer différents langages définissant le système avec différents degrés de précision et aussi être capable d’assurer une fédération entre ces modèles. Cette fédération de modèles évoluant au cours de la découverte du système, elle doit reposer conceptuellement et technologiquement sur une approche permettant une évolution dynamique des modèles et de leur fédération au cours du temps. Cette fédération reposera sur une modélisation par rôle qui offre cette capacité de modélisation dynamique de la fédération au cours du processus de modélisation.

La modélisation par rôle qui repose sur de nombreux travaux tant théoriques que technologiques [4,5] se propose de fournir une alternative pour la définition d’interfaces adaptables en allouant statiquement et dynamiquement des objets ou éléments de modèles par des rôles.

Dans le cadre cette étude nous nous proposons donc d’utiliser ce concept de rôle(Role4All) pour fédérer les différents modèles mis en jeu pour la capitalisation des connaissances des systèmes que nous cherchons à analyser dans un but de découverte de son architecture. Pour cela, nous nous baserons sur les travaux issus d’une thèse (DGA – Région Bretagne) menée au sein de l’équipe qui a défini et implanter un langage spécialisé pour la définition de rôle dans le but de fédérer différents modèles systèmes [10,11,12].

Une fois le système suffisamment décrit, la synthèse de haut niveau permet la production de circuits matériels à partir de spécification modélisées(Morphose). La maitrise de cette synthèse de haut niveau dans l’équipe de l’ENSTA Bretagne permet de se reposer sur une expertise éprouvée pour améliorer la modélisation des systèmes sur différents niveaux d’abstraction. En effet, l’équipe de l’ENSTA Bretagne a développé en propre et en collaboration avec ses partenaires du Lab-STICC des outils logiciels permettant la synthèse de haut niveau [6,7,8]. La maitrise technologique de la phase de synthèse de haut-niveau est critique. Elle permet d’orienter la synthèse à façon, pour répondre à des problématiques ciblées.

Enfin un outil(ModSocTool) s'appuyant sur le modèle fédéré sera développé. Il permettra notamment de mettre en œuvre la méthodologie, et fournira des interfaces utilisateur adaptés(KPI's, Dashboard...).

Figure 2: L'approche Figure 3: Nos Outils

## Scénario

# C:\idm_modsoc\idm_modsoc\documentation\docTravail\Activite.jpg

# CHAINE D'OUTIL DETAILLEE

La fédération permet la connexion des différentes sources d'informations d'un point de vue sémantique.

D'un point de vue communication le moyen le plus efficace pour établir des connexion entre outils est de les faire communiquer sur le réseau. Nous avons choisit d'opter pour une architecture de type REST[]. REST à la particularité d'être extrêmement simple et adaptable, c'est aussi un type de web service qui a de plus en plus d'attrait pour l'intégration de logiciels notamment pour la conception de systèmes embarqués(REF[iFEST/OSLC]). Dans REST chaque ressource ou service proposé par l'outil est identifiée par une URI(ex http://monoutil.mesresources/resource1), et l'accès à cette ressource s'effectue par les commandes HTTP(GET, POST, PUT, DELETE...). REST est supposé Stateless ce qui impose que le serveur ne conserve pas d'informations relatives aux requêtes passées du client.

D'un point de vue format de données, les ressources transmises à travers l'architecture REST seront dans la mesure du possible au format JSON, afin d'optimiser le support au sein de l'outillage Role4All et d'obtenir un maximum de compatibilité avec les autres outils du web. Il faut noter par ailleurs que l'outillage Role4All est également capable de digérer d'autres format de données. JSON présupposant que l'on connaisse la structure de la ressource, il sera peut-être nécessaire de transmettre initialement cette information à role4All afin qu'il puisse interpréter par la suite les données JSON.

Pour chaque outil il faudra donc réaliser un adapter reposant sur l'architecture REST. Il devra fournir un ensemble de services qui seront ensuite découverts et utilisés par Role4All. Dans la mesure du possible ces adapter devront fournir une représentation de leurs ressources au format JSON.

WEB

PIMCA

Rhapsody

Adapter REST

UI

Morphose

Adapter REST

Adapter REST

Role4All

ModSocTool

Adapter REST

Adapter REST

QUESTION: est-ce que les adapter au niveau des outils doivent être spécifiques à role4All(requiere des services de role4All) ou bien pas besoin(dans ce cas c'est dans role 4All au niveau de l'adaptateur technique qu'on fait ça)? Dans tout les cas, il faudra surement un ensemble de services commun à tout nos outils?

TODO Diagramme de séquence

#### Technos possibles pour l'architecture REST

##### Vertx

Polyglotte:Java & Ruby), leger, HTTP et Rest. Bus d'évènements permettant de faire du synchonisme/asynchronisme entre des bout d'application(verticles), potentiellement utile pour des services qui seraient dépendant, ou qui seraient bloquant(chargement d'un modèle/versus modification), traitements long d'OBP)

##### Apache Tomcat/Servlet/JSP

Dans iFEST, lourds à mettre en place.

##### Teapot

Server HTTP smaltalk, à coupler avec le framework de l'étudiant pour le REST. Uniquement pour Role4All!

## Les Outils

### PimCA

L'objectif du langage est de fournir un cadre pour guider l’utilisateur dans la réalisation d’attaque de systèmes cyber-physiques.

Un outil de modélisation pour le langage PimCA (langage défini par DGA MI) existe et permet de modéliser des systèmes à différents niveaux d’abstraction. Ce langage contient la description de la structure du système mais aussi les graphes d’attaques possibles sur le système modélisé. Cet outil a été réalisé l’année passée.

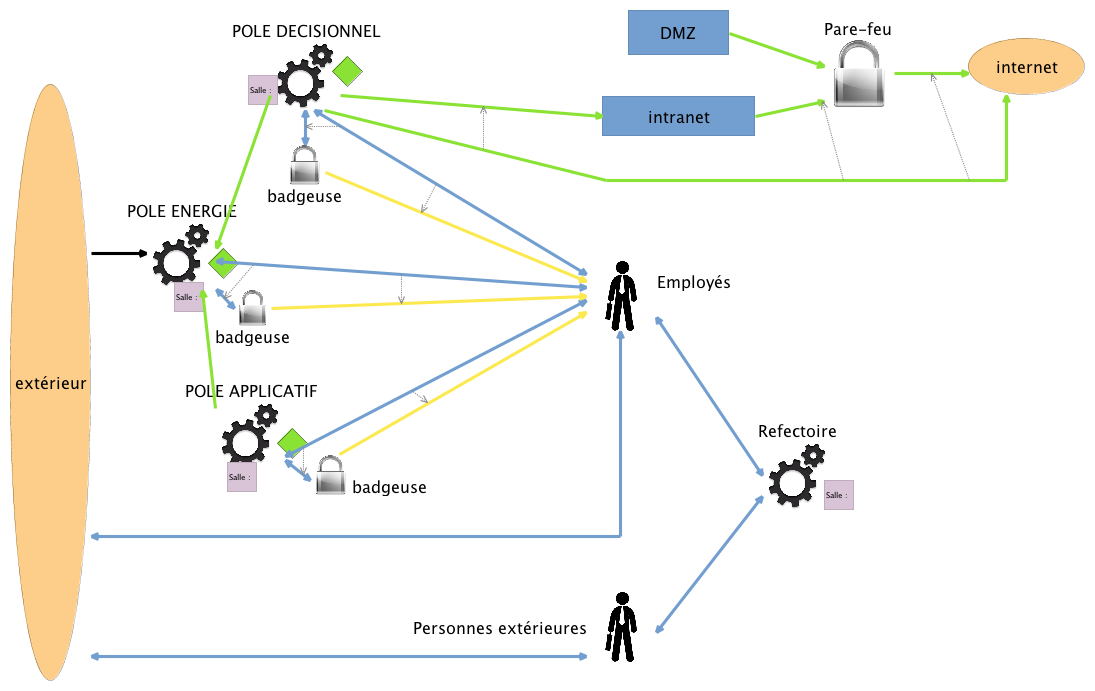


Figure 4: Un diagramme fait avec l'outil PimCA

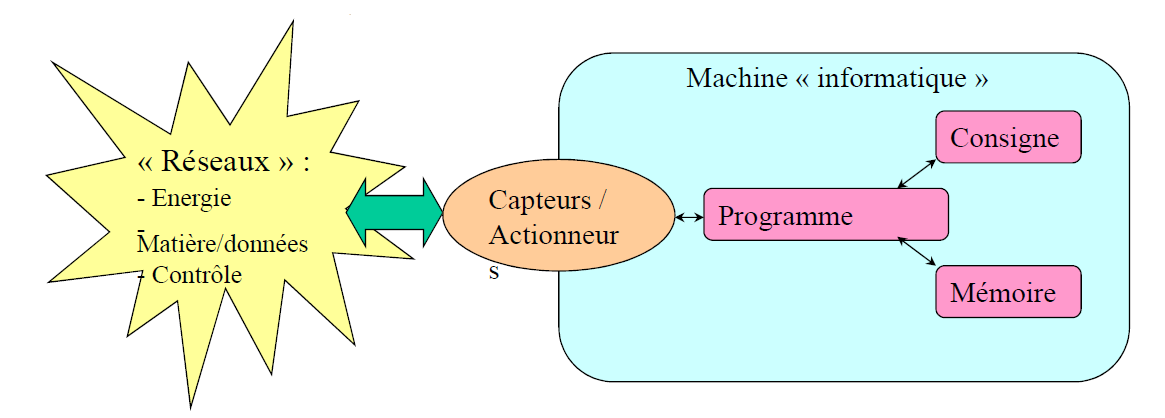
Le langage possède trois composantes :

* Un modèle de connaissance,
* Un graphe d’attaque,
* Un modèle de scénarii d’attaque.

##### 

##### Le modèle de connaissance

Le modèle de connaissance (Knowledge Model) représente un système faisant l’objet d’une (planification d’) attaque. Le système est composé (entre autre) d’un ensemble de machineries (Machinery), interconnectées entre elles par trois types de flux : énergie (Energy), matière/données (Matter/Data), et contrôle (Control).



La machinerie est caractérisée par trois points-clé : exécution – configuration – mémoire.

#### Les concepts principaux

Les « machines » :

|  |  |
| --- | --- |
| Icône-Concept | Description |
| machineMachinery | **Machinerie** : système manipulant des Ressources (regroupement particulier) : voiture, animal, PC, processus |
| C:\Users\creffst\AppData\Local\Microsoft\Windows\INetCache\Content.Word\executant.pngPerformer | **Exécutant (spécialise Machinerie)** : ce qui transforme la Ressource, e.g. UC/Programme, cerveau, régulateur. |
| reseauNetwork | **Réseau** **(spécialise Machinerie)** : zone d’échange de matière, d’information, d’énergie, etc. : câblage, tuyauterie, IPC Engine. |
|  |  |
| douaneCustoms | **Douane (spécialise Machinerie)** : fonctionnalité particulière mise en place par une Machinerie pour identifier & autoriser une autre Machinerie : cadenas, garde, login, crypto |
| interfaceInterface | **Interface (spécialise Machinerie)** : permet de passer d’une Machinerie à une autre, du monde physique au monde virtuel et inversement : NIC, caméra, clavier, écran. |
| regroupement Gathering (non réifié) | **Regroupement** : ensemble logique d’objets de tout type, entrepôt sans Ressource. Un regroupement ne possède pas les infos propres à une machinerie, c.-à-d. executant, configuration, mémoire. |
| C:\Users\creffst\AppData\Local\Microsoft\Windows\INetCache\Content.Word\entrepot.pngRepository | **Entrepôt** : zone de stockage de Ressource : armoire, bâtiment, disquette, database, file system |

Les « ressources » :

|  |  |
| --- | --- |
| Icône-Concept | Description |
| ressourceResource | **Ressource** : ce qui est transformé, manipulé par une Machinerie : matière, électricité, document, log, data |
| consigneInstructions | **Consigne (spécialise Ressource)** : La direction, les paramètres que l’exécutant suit : Fichier de configuration, ordre, politique de sécurité |
| C:\Users\creffst\AppData\Local\Microsoft\Windows\INetCache\Content.Word\passeport.pngPasseport | **Passeport (spécialise Ressource) :** élément à fournir à la Douane pour être identifié / autorisé : clef, carte d’identité, badge, login/password, clef de chiffrement |

Les attributs :

|  |  |
| --- | --- |
| Attribut | Description |
| UsageArea | Zone d’usage ou la donnée est valide. |
| UsagePeriod | Période ou la donnée est valide. E.g. attaque possible uniquement lors de la maintenance du système. |
| UsageLevel |  |
| NbCopy | Nombre d’instances (n, {0, …} ou -1 si infini). Représente la notion de groupe générique ou de groupe / élément spécifiques à une activité donnée à un moment donné. Par exemple, lors d’une attaque, il peut être utile d’isoler et de cibler une machine donnée (ou d’une IP donnée), qui était jusqu’à lors perdue dans la masse de ses semblables. Définition d’un rôle à réaliser.  Outillage : fonction d’extraction d’un élément depuis un groupe à réaliser. |
| Value | Valeur |
| Comment | Commentaire (chaine de caractères) |
| Data |  |
| Type | Type de donnée. |
| Name | Identifiant |
| Validity | Degré de validité et de confiance dans la donnée, à une date donnée. (e.g. augmente si recoupement de sources indépendantes coïncides…). Pourcentage. |
| UpdatedBy | Précision sur la source de la modification (auteur). |
| ModifDate | Date de modification de l’information. |

Les relations :



|  |  |
| --- | --- |
| Verification | Vérificateur : Visualise le bon fonctionnement du processus |
| Swap | Echange : Communication d’égal à égal, utilisation / production interdépendante. |
| Control | Contrôle : Donne la direction, agit sur la consigne, administre |
| Use | Utilisation, Consommation, Client : Besoin pour fonctionner (ressource, énergie, exécutant, …) |
| Produce | Développeur, Producteur, Fournisseur : Réalise un objet, une ressource |
| Maintain | Maintenance : Agit partiellement sur la constitution d’objet, l’état, … |
| atd | Au travers de |

### Le graphe d’attaque

Un graphe d’attaque regroupe l’ensemble des possibles, il est un support au calcul (la sélection) du chemin d’attaque le plus sûr. Le séquencement choisi des activités définissant un scenario d’attaque, ou plusieurs, qui peut être séquentiel ou parallèle.

#### Adapter PimCA

L'outillage PimCA est basé sur le framework EMF(Eclipse Modeling Framework). Un méatmodèle au format Ecore représentant les concepts et les relations du langage peut être instancié en un modèle conforme au format stand XMI. Il existe un editeur graphique réalisé sous Sirius permettant d'éditer un diagramme de connaissance, et bientôt un diagramme d'attaque graphiquement.

L'adapteur Pimca reposera sur un adapteur générique Ecore, proposant par la même occasion une transformation de Ecore/Xmi->JSON

COMMENTAIRE: Il faudrait avoir un adapteur générique ECORE, auquel on rajoute une surcouche Pimca?

Les services exposés seront les suivants:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| URI | COMMANDE | PARAMETRES | DESCRIPTION |  |
| /models/ | GET | N/A | Récupérer la liste des modèles |  |
| /models/ | GET | ID ou Uri | Récupérer un modèle dans le liste des modèles présents |  |
| /models/model/ | GET | Type d'élement, ID | Requêter un sous ensemble d'un modèle |  |
| /models/model/Id | PUT | attribut, valeur | Modifier un élément de modèle |  |
| /models/model/ | POST | Json | Ajouter un élément de modèle |  |
| /models/model/Id | DELETE |  | Supprimer un élément de modèle |  |

##### Travail à effectuer:

1. Modèlisation des abres d'attaque
2. Transformation EMF vers JSON
3. Connecteur REST

### Rhapsody

Nous allons nous servir du modeleur UML Rhapsody, et plus précisément des machines à états. Rhapsody fournit une API Java permettant communiquer et manipuler le modèle en cours de conception. Un adapter Rhapsody sera réalisé, il permettera de récupérer et manipuler le modèle ouvert dans l'instance de Rhapsody.

### Role4All(TODO)

La majorité des projets, qu’ils soient industriels ou de recherche, demandent à l’heure actuelle l’utilisation de plusieurs outils métiers. Ces divers outils manipulent dans la majorité des cas un grand nombre de données non spécifiques, ce qui signifie qu’une même donnée est amenée à être manipulée par plusieurs outils ; on parle alors d’interopérabilité. La manipulation concurrente de données est l’un des problèmes sérieux des domaines du numérique, il existe alors de nombreux concepts solutionnant tout ou une partie de ces problèmes, regroupés en trois grandes écoles : l’intégration, l’unification et la fédération.

L’intégration consiste en la fusion des modèles de données utilisés, ce qui correspond à la création d’un modèle de donnée standard. L’intégration est une solution simple et adaptée au problème d’interopérabilité, elle permet de résoudre les problèmes de concurrences à la source en forçant l’utilisation d’un modèle standard. Une telle approche oblige par contre à adapter chaque outil utilisé à un standard unique, ce qui n’est plus envisageable sur les systèmes actuels, à cause de la complexité croissante des systèmes d’information, intégrant de plus en plus d’informations fondées sur des sémantiques différentes. L’intégration est donc une solution adaptée aux systèmes fixes, ne connaissant plus d’évolution.

L’unification consiste en la création de correspondances sémantiques entre différents modèles via un méta-modèle commun à tous les composants d’un système, appelé modèle “pivot”. Le modèle pivot permet d’interconnecter des outils métiers possédants des modèles différents, ce qui permet de se libérer des problèmes liés à la notion de modèle standard. L’augmentation de la complexité des applications métier, des processus, des demandes en terme d’échanges, etc., nécessite le développement de systèmes de plus en plus complexes. Il est alors nécessaire d’adapter le modèle pivot à l’évolution des systèmes, or la complexifications des systèmes ne permet pas la création d’un modèle pivot unique. Il est donc nécessaire de créer plusieurs modèles pivots pour créer l’interopérabilité entre plusieurs outils. La vitesse d’évolution actuelle des systèmes est trop importante pour permettre la gestion des modèles pivots pour un système complexe. L’unification est donc une solution adaptée aux systèmes à évolution lente ou à faible complexité.

La fédération consiste en l’utilisation de « mapping » (cartes de correspondance codées) pour associer dynamiquement des modèles distincts. Les modèles fédérés partagent des concepts similaires ou équivalents, ils sont donc basés sur une ontologie commune (un même méta-méta-modèle). La mise en exergue des concepts communs à plusieurs modèles est complexe et fortement liée au contexte d’utilisation de l’outil, mais elle est indispensable à la fédération. Une relation basée sur les concepts permet d’être indépendant du méta-modèle d’un outil et donc de ses variations, de plus cela permet de relier entre eux tous types d’outils, quel que soit leur méta-modèle. Finalement la fédération est l’approche d’interopérabilité permettant le plus de flexibilité et de dynamisme, c’est pourquoi la fédération est une solution adaptée aux systèmes complexes évoluant rapidement.

Role4All est un outil architecturé autour de la notion de rôle, un rôle étant défini ici comme un élément de modèle dont les instances sont associées à des instances de types des langages de modélisation [thèse de Jean-Philippe SCHNEIDER]. Cette définition permet de créer un modèle de rôle séparé des modèles à manipuler puis d’associer ce modèle de rôles aux éléments des modèles de travail. Ainsi, il n’est pas nécessaire de modifier les éléments des modèles manipulés, de plus il est possible de faire évoluer le modèle de rôle indépendant des modèles outils, on a alors une grande flexibilité du rôle.

Role4All permet à un rôle de jouer un rôle c’est-à-dire de considérer une instance de rôle comme une instance de modèle et de lier cette instance à un rôle. Cela permet, entre autre, de créer des points de vue sur un modèle ou de centraliser dans un même point de vue des informations provenant de divers outils. Un rôle est donc un élément indépendant des modèles de travail associé à un ou plusieurs éléments de modèle, cette indépendance permet une allocation dynamique des rôles.

De ce fait Role4All est semblable à un outil de fédération car il permet de regrouper des informations provenant de plusieurs modèles en se basant sur un concept (un rôle) commun. Cependant la première version de Role4All (présenté lors de la thèse de Jean-Philippe SCHNEIDER) créait un lien unidirectionnel entre les modèles des outils et Role4All, ce qui est incompatible avec la notion de fédération (nécessitant une communication bidirectionnelle). De plus il n’existait pas de liens entre deux instances d’éléments de modèle jouant le même rôle, ce qui dans l’architecture de Role4All est nécessaire à la fédération. La partie suivante présente l’architecture de Role4All dans sa première version.

Role4All is structured around four main classes: *Player*, *Role*, *DynamicAdapter* and *PlayRelation*.

The figure 4 illustrates the relation between Role4All’s classes through the example of the role *RoleFPGA* played by the Pimca element *pimcaMachinery0* and the Excel element *excelGroup0*.

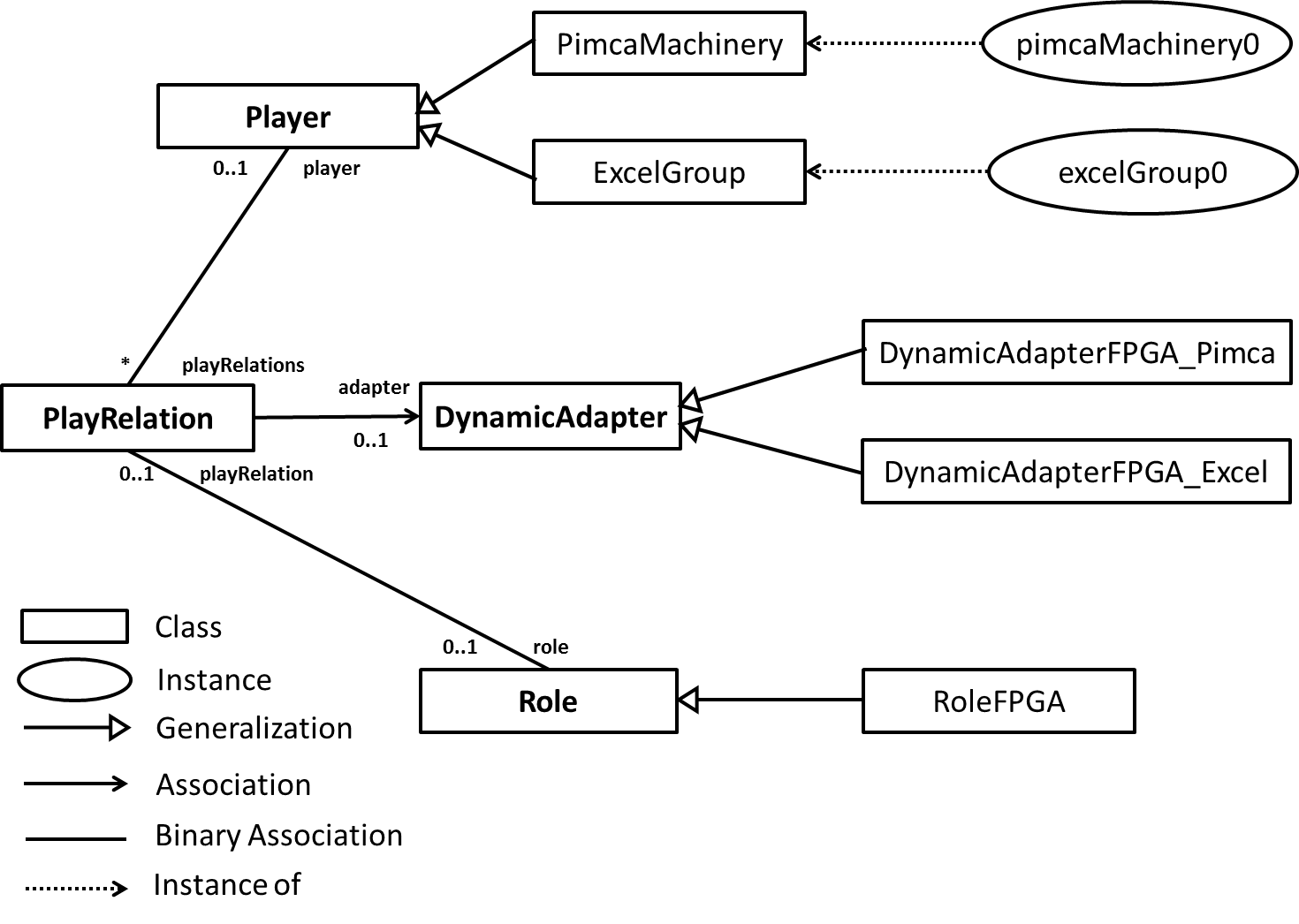


Figure 4 : Role4All’s meta-model illustrates with the example of the role of FPGA

The elements extending the class *Player* are the model elements tools, they are called “player”. For example *PimcaMachinery* is an element of Pimca’s meta-model and p*imcaMachinery0* is an element of Pimca’s model (figure 4). We want Pimca’s model element *pimcaMachinery0* to play the role of *RoleFPGA,* therefore the model element *PimcaMachinery* extends the class *Player*. The connection between a role model element (*RoleFPGA*) and a tool element (*pimcaMachinery0*) is formatted by an adapter.

The elements extending the class *Role* are the concepts created by the user through the role models, they are called “role”. As an example in the figure 4 *RoleFPGA* is a role therefore the class *RoleFPGA* extends the class *Role* played. We can notice that stating “In Role4All a role can play a role” is equivalent to: “a role can be a player” or “the class *Role* extends the class *Player*”.

The elements extending the class *DynamicAdapter* allow adapting a model element for its role, they are called “adapter”. The adapters define the behavior of the relations between players and roles. In the figure 4 *DynamicAdapterFPGA\_Pimca* is the adapter between the instances of *PimcaMachinery* and the role *RoleFPGA* and *DynamicAdapterFPGA\_Excel* is the adapter between the instances of *ExcelGroup* and the role *RoleFPGA*

The elements extending the class *PlayRelation* are connectors without behavior between three elements: a role, a player and an adapter.

So according to figure 4, the player *pimcaMachinery0* plays the role of *RoleFPGA* and the behavior of this relation is defined in the adapter *DynamicAdapterFPGA\_Pimca*. Finally a Pimca element and an Excel element play the same role, so we created a unique point of view on two different elements of two different tools. Now we can use the role *RoleFPGA* to manipulate the concept of FPGA instead of the tools Excel and Pimca.

Mettre les différents points :

* Positionnement de l’approche par role par rapport aux aspects unification, integration et federation.
* Apport de la modélisation par role -> aspect dynamique
* Présentation de role4All avec son metamodèle et son outillage
* Donner les limites au jour de la soutenance de JP avec notamment la fédération.

#### Extension de Role4All pour la fédération

Nous avons mis en avant la nécessité de mettre à jour Role4All afin d’en faire un outil de fédération. En effet, actuellement Role4All permet de récupérer des informations dans divers modèles et de créer un point de vue sur ces informations. Cependant aucun lien n’est gardé entre le point de vue ainsi généré et la source des informations, ainsi si l’utilisateur modifie des informations via un point de vue, aucune de ces modifications n’est effective sur les sources. C’est pour remédier à ce manque qu’il est nécessaire d’apporter deux mises à jour majeures à Role4All :

* Création d’une synchronisation entre les instances de rôles
* Mise en jours du lien modèle-rôle afin de le rendre bidirectionnel

La première étape vise à créer un lien entre deux instances de rôles afin de permette la communication entre ces instances. Cela permettrait de répercuter les modifications apportées sur une instance de rôle à une autre. La deuxième étape répercuterait alors les modifications effectuées sur une instance rôle à l’instance d’élément de modèle outils qui lui est liée. Finalement nous aurions un lien entre instances de modèle et instances de rôle et un lien entre deux instances de rôle, nous aurions alors un lien entre deux instances de modèles différents. La synchronisation des instances de rôles est abordée dans le chapitre suivant, la mise à jour du lien modèle-rôle ne sera pas abordée dans ce rapport.

Description du problème et de la solution adoptée.

Actuellement Role4All est capable d'importer un modèle, le lien vers la source n'est ensuite plus gardé (duplication de l'information). Il n'y a fédération que si le lien vers la source est gardé, et que l'on est capable de faire autant de la lecture/écriture vers les informations.

In Role4All all model elements play a role therefore all instances are strongly linked with a role instance, as it was explained in the previous part. As an example, in figure 4 the Pimca model element *pimcaMachinery0* plays the role *RoleFPGA* therefore the instance *pimcaMachinery0* is linked with an instance of *RoleFPGA* called *roleFPGA0*.

A role has two instance variables called *containedRoles* and *containerRoles* that are collections of role instances. So, a role instance can contain several role instances and be contained in various ones. We use this relation to connect together the role instances that will be synchronized.



Figure 6 : Roles synchronization in Role4All

The figure 6 presents how the role instances were connected, the instance *roleFPGA0* contains two role instances*: roleFPGA0.1* and *roleFPGA0.2*. The instance *roleFPGA0* knows which instances it contains through the variable *containedRoles* and the instances *roleFPGA0.1* and *roleFPGA0.2*, whichknow in which instances they are contained through the variable *containerRoles*. So we have a bidirectional relation between role instances through a role container (*roleFPGA0*). Furthermore the container role is a facade to the contained ones, therefore it is possible to work directly with the containers, which allows to use a specific point of view.

The synchronization methods are developed in the container role, by default it is a check-out/check-in synchronization [Thesis synchro]. After a save all changes are immediately reflected in all the synchronized tools according to the synchronization rules. The synchronization needs to depend on the context (Network management, Cybersecurity, etc.), therefore Role4All allows to personalize each synchronization. A user can define various rules and check when he creates synchronizations between tools with Role4All, for example a prioritization of the synchronizations according to the role of the user. Moreover it is possible to change the default synchronization method for another one, like the Long Transaction Model designed to support the evolution of whole systems as a series of apparently atomic changes [thesis synchro]. The main idea with the synchronization in Role4All is to be independent of the tools and easily editable.

### 

### Morphose(TODO)

Mettre la description de Morphose sans la description du use case.

Morphose est une suite d'outils de l'ESL dédiés à l'exploration d'architecture, la synthèse architecturale(HSL,SLS), la simulation ainsi que la vérification.

### Génération de code Json à partir d’un modèle de cybersécurité

Le domaine de la cybersécurité cherche à établir des modèles au niveau système pour étudier les stratégies d’attaque de ses systèmes. Actuellement les modèles de scénarios servent de support à l’analyse.

D’autre part, nous avons un framework, Role4All, développé en langage Smalltalk dans l’environnement Pharo qui permet de mettre en relation plusieurs types de modèles pour notamment faire des transformations et des générateurs de code.

Dans ce projet, nous souhaitons coupler la modélisation qui offre un support d’analyse avec l’environnement de transformation et de génération basé sur le format JSON.

Un outil de modélisation pour le langage PimCA (langage défini par DGA MI) existe et permet de modéliser des systèmes à différents niveaux d’abstraction. Ce langage contient la description de la structure du système mais aussi les graphes d’attaques possibles sur le système modélisé. Cet outil a été réalisé après un sujet UV5.4 de l’année passée.

L’intégration des modèles PimCA dans Role4All doit donc être basé sur un générateur de code JSON à partir des modèles et aussi un importateur JSON/PimCA dans Role4All.

Le travail va se décomposer dans les étapes suivantes :

Prise en compte du modeleur PimCA

Etudes des importateurs Role4All

Réalisation du générateur de code JSON

Réalisation de l’importateur de code JSON/PimCA

### Génération de code à partir d’un modèle de cybersécurité

Le domaine de la cybersécurité cherche à établir des modèles au niveau système pour étudier les stratégies d’attaque de ses systèmes. Actuellement les modèles de scénarios servent de support à l’analyse.

D’autre part, il existe des simulateurs de système d’information comme hynesim qui utilisent la technologie de virtualisation de machine pour offrir une capacité de simulation d’un réseau de machines intégrant tout type de support (ordinateurs, routeurs, switchs) et d’OS (windows, Linux,…). Une fois le réseau virtualisé, nous pouvons exécuter toutes les applications possibles sur chacune des machines et ainsi dérouler tous les scénarios d’attaques possibles.

Dans ce projet, nous souhaitons coupler la modélisation qui offre un support d’analyse avec le simulateur de systèmes d’information.

Un outil de modélisation pour le langage PimCA (langage défini par DGA MI) existe et permet de modéliser des systèmes à différents niveaux d’abstraction. Ce langage contient la description de la structure du système mais aussi les graphes d’attaques possibles sur le système modélisé. Cet outil a été réalisé après un sujet UV5.4 de l’année passée.

Le simulateur Hynesim fournit une API en Python qui permet dynamiquement de créer un réseau de machine à partir d’une bibliothèque de machines. Ainsi plutôt que de créer à la main le système d’information, un programme permet cette création dynamiquement.

L’objectif du projet sera fournir un générateur de code pour Hynesim à partir de l’outil de modélisation

PimCA.

Le travail va se décomposer dans les étapes suivantes :

· Prise en compte du modeleur PimCA

· Etude de l’API Hynesim et prototypage de scripts Python

· Modélisation de l’API Python

· Réalisation du générateur de code Python

· Utilisation du générateur de code sur un exemple modélisé pour simuler le modèle sous Hynesim

# Perspectives

# REF

[Lee] Edward A. Lee - Position Paper for NSF Workshop On Cyber-Physical Systems: Research Motivation, Techniques and Roadmap October 16-17, 2006, Austin, TX

http://www.hynesim.com/

http://www/diateam.net/

##### Openflexo

1. Z. J. Jia, A. Núñez, T. Bautista, and A. D. Pimentel. 2014. A two-phase design space exploration strategy for system-level real-time application mapping onto MPSoC. Microprocess. Microsyst. 38, 1 (February 2014).
2. Mark Thompson and Andy D. Pimentel. 2013. Exploiting domain knowledge in system-level MPSoC design space exploration. J. Syst. Archit. 59, 7 (August 2013), 351-360.
3. M. Seifert, C. Wende, and U. Aßmann, “Anticipating unanticipated tool interoperability using role models,” in Proceedings of the First International Workshop on Model-Driven Interoperability. ACM, 2010.
4. F. Steimann, “On the representation of roles in object-oriented and conceptual modelling,” Data & Knowledge Engineering, vol. 35, no. 1, pp. 83–106, 2000.
5. T. Kühn, M. Leutäuser, S. Götz, C. Seidl, and U. Aßmann, “A metamodel family for role-based modeling and programming languages,” in Software Language Engineering, ser. Lecture Notes in Computer Science. Springer International Publishing, 2014,