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, Bastien Drouot et Jean Christophe Le Lann

ENSTA Bretagne

Lab-STICC UMR 6285

Joel.champeau@ensta-bretagne.fr

**Table des matières**

[1. INTRODUCTION 3](#_Toc447731682)

[1. Contexte 3](#_Toc447731683)

[2. Objectifs 3](#_Toc447731684)

[3. Cas d'usage 5](#_Toc447731685)

[2. Approche adoptée 6](#_Toc447731686)

[4. Présentation 6](#_Toc447731687)

[5. Les Outils 7](#_Toc447731688)

[1. PimCA 7](#_Toc447731689)

[2. Outil de modélisation logiciel 10](#_Toc447731690)

[3. Role4All 11](#_Toc447731691)

[4. Morphose 15](#_Toc447731692)

[6. Use Case 16](#_Toc447731693)

[3. Conclusion 19](#_Toc447731694)

[4. Références 19](#_Toc447731695)

# INTRODUCTION

L’étude traite de 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 maitriser les systèmes à sécuriser et par là même de caractériser cette méthodologie de découverte des systèmes embarqués.

La méthodologie de capitalisation des connaissances sur les systèmes embarqués ciblant l’interfaçage matériel-logiciel, restent à prototyper et à identifier. Ces méthodologies doivent s’appuyer conceptuellement sur les cycles de conception traditionnels, qui sont bien maitrisés, pour être transposée vers la sécurisation de systèmes connus et aussi à découvrir. Cet objectif doit permettre de réutiliser les connaissances 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 architectures MPSoC et/ou SCADA 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 puisque l’objectif est de partir d’une réalisation existante et d’analyser son contenu. Dans la découverte d'un tel système embarqué, l'articulation matériel-logiciel est cruciale car elle contient les informations de communication entre ces deux composantes. Ces informations sont deux niveaux d’abstraction différents avec la partie support matériel sur laquelle on peut se poser les questions : quels processeurs sont susceptibles d'intervenir dans la conception ? Quels bus ? Quels périphériques? De quelles marques ?

Et puis la partie symbolique du système avec les données échangées avec leur format sur laquelle nous pouvons nous poser les questions suivantes : Quel est le nombre et le format des données échangées ? Quelle est la structure de la carte mémoire de ces données ? Quel l’implantation de cette carte mémoire dans les registres de communication ?

Bien sur ces différentes questions sont également abordées lors des approches de conception descendante classiques où l’on doit identifier et fournir les données et leur représentation mémoire pour communiquer entre le logiciel et le matériel. Cette représentation évolue toujours au cours du temps selon les changements des spécifications liés à l’évolution du besoin suivant les performances offertes par le système mais aussi suivant les raffinements de la conception (introduction de détails architecturaux, etc) correspondant à autant d'explorations architecturales. Ces approches incrémentales sont particulièrement employées dans la conception  de SoC (system-on-chip), où deux 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 réelles 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. Cette démarche est tout à fait similaire à la conception d’un système où la conception est raffinée au court du temps pour intégrer les différentes contraintes architecturales.

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 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

Afin de développer notre approche de mise en relations de modèles et d’estimation d’architecture de notre système embarqué, nous nous reposons sur un cas d’étude développé au laboratoire qui servira d’exemple pour prototyper notre approche.

Supposons une application sensible de traitement d'un flux vidéo permettant la détection de formes (personnes ou objet). Cette application est un système cyber physique(CPS) [Lee] qui se compose d'un FPGA pour la récupération et le 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.

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. L’architecture cible du système embarqué est représenté dans le figure XX suivante.



Processeur ARM

FPGA

Soc

Caméra

Ethernet

Bus

Figure 1: Architecture embarquée du cas d’étude

# Approche adoptée

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 employées dans la conception où trois niveaux d'abstraction sont couramment utilisés : niveau comportemental, niveau architectural et niveau détaillé pour intégrer les détails de la plateforme finale. Nous envisageons donc 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 les différents langages définissant le système avec différents degrés de précision et aussi être capable d’assurer une mise en relation des modèles. Cette mise en relation est effectuée via une fédération de modèles évoluant au cours de la découverte du système, elle doit reposer conceptuellement et technologiquement sur une approche permettant une évolution dynamique de la définition de la mise en relation des modèles tout en restant cohérente au cours du temps. La fédération des modèles reposera sur une modélisation par rôle qui offre une capacité de modélisation dynamique 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 à des rôles.

Dans le cadre cette étude nous nous proposons donc d’utiliser ce concept de rôle, à travers le framework 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 mettre en relation différents modèles systèmes [10,11,12]. Le framework Role4All a été conceptualisé et mis en œuvre pour offrir la capacité d’unifiés différents langages de modélisation système sur un modèle dit « pivot ». Au cours de cette étude nous devons étendre ce framework afin de permettre une réelle fédération des éléments de modélisation de notre système.

Une fois le système suffisamment décrit, une étape de synthèse de haut niveau permet une production de circuits matériels à partir de ces spécifications modélisées, via l’outillage 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 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. Dans notre cas, elle permettra d’évaluer les hypothèses architecturales issues de la fédération de modèles de l’étape de précédente.

Enfin une interface d’analyse s'appuyant sur le modèle fédéré pourrait être développée pour faciliter la mise en œuvre de la méthodologie outillée.

Figure 2: L'approche générale Figure 3: L’approche retenue pour notre cas d’étude

Dans la suite du document, nous allons présenter les outils mis en œuvre dans notre cas d’étude et les premiers résultats obtenus dans l’adaptation du framework Role4All pour effectuer la fédération des informations des différents modèles.

## Les Outils

### PimCA

L'objectif du langage est de fournir un cadre pour guider l’utilisateur dans l’analyse de systèmes afin de résister aux attaques potentielles.

Le langage PimCA a été défini par une équipe de DGA-MI et un outil de modélisation basé sur cette définition a été développé au laboratoire 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.

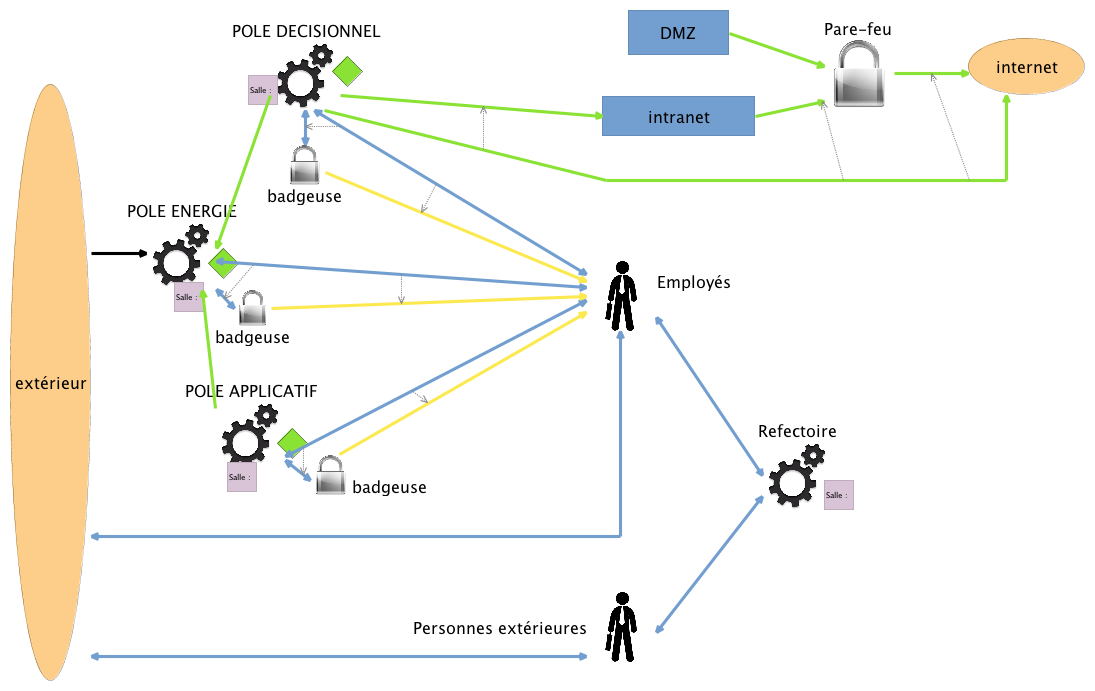


Figure 4: Un diagramme fait avec l'outillage PimCA

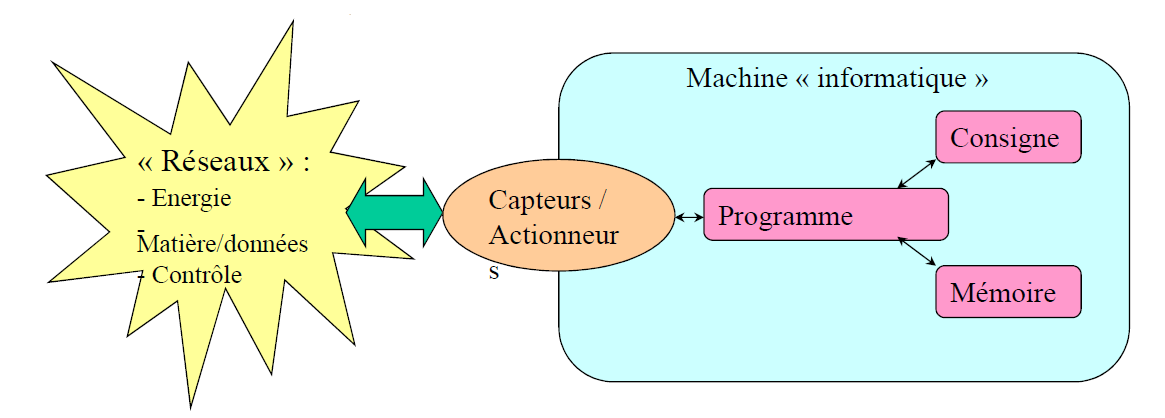
Le langage possède trois composantes :

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

Ce modèle d’attaque regroupe l’ensemble des possibles et représente un support de calcul pour la sélection du chemin d’attaque le plus sûr. Ce modèle est basé sur le séquencement des activités définissant un scenario, ou plusieurs, qui peut être séquentiel ou parallèle. Ces scénarii sont basés sur les éléments structuraux du modèle de connaissance associés aux activités qui sont : l’observation du système, raisonner sur le système et agir sur le système. Différentes actions pour agir sur le système étant possible. Sans détailler plus avant ces différentes actions, nous pouvons noter que dans un premier temps nous nous attacherons aux actions d’observation et de raisonnement, dans le sens où nous chercherons à évaluer les observations effectuées.

#### Outillage PimCA

L'outillage PimCA est basé sur le framework EMF(Eclipse Modeling Framework) avec un méatmodèle au format Ecore représentant les concepts et les relations du langage. Il existe un éditeur graphique réalisé à l’aide du composant open source Sirius permettant d'éditer un diagramme de connaissance. Ainsi les modèles de niveau système seront réalisés avec cet outillage dans la suite de nos expérimentations.

### Outil de modélisation logiciel

En complément des modèles systèmes, nous nous donnons la possibilité d’utiliser un modeleur UML qui permet de décrire la partie logicielle sous différents points de vue dont les parties comportementales avec par exemple des machines à états. Nous pouvons utiliser différents outils UML comme Rhapsody, Papyrus ou UMLDesigner qui fournissent tous une API Java permettant communiquer et manipuler le modèle en cours de conception. Sans avoir arrêté le choix définitif du modeleur, nous estimons qu’un modèle UML permettrait de capitaliser un certain nombre d’informations relatives à l’architecture logicielle indépendamment mais en cohérence avec le modèle système. A charge de définir en UML quels seraient le formalisme adéquat pour offrir la meilleure capitalisation.

### Role4All

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 c’est pourquoi il existe 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*.

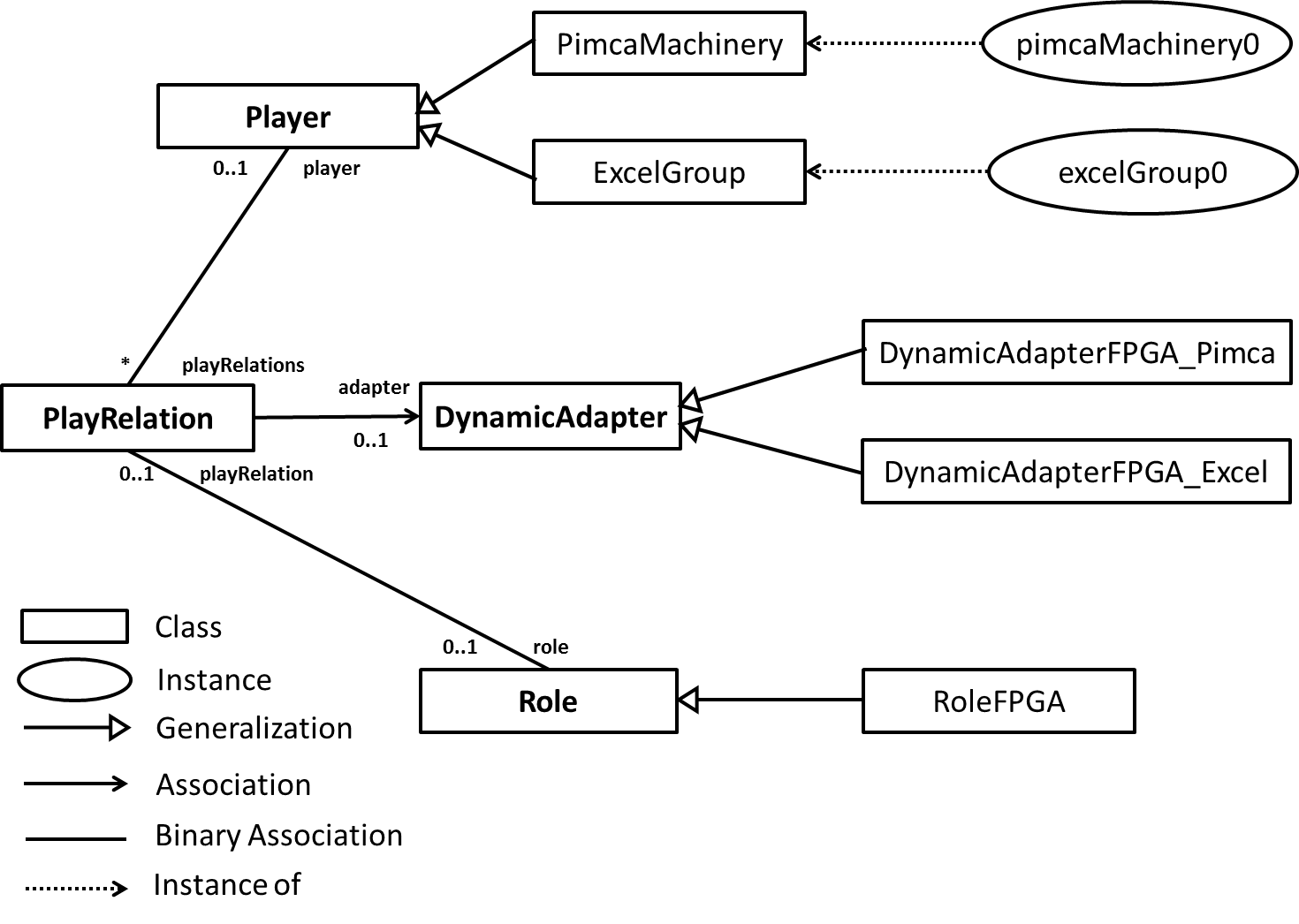


Figure 5 : Role4All’s meta-model illustrates with the example of the role of FPGA

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.

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

The last two parts present the main units of Role4All: the role model generation and the synchronization but Role4All includes one more unit, the simulation. The simulation is possible due to a specific tool: Morphose. Morphose is a simulation tool using the concept of activity, with Morphose, a system is a collection of activity. An activity is association of two entity a software behavior and the hardware restriction. The first one describes the software behavior through machines states. The second one describes the hardware features like the memory consumption, the energy consumption, the number of physical hearts, etc. In Morphose the user implements the behaviors of the system and not the system himself, so according to our example the cyberterrorist can implement the behavior of his hypothetical systems with Morphose. The terrorist focused on the hardware system therefore he generates 4 simulations with the same software behavior but with 4 different hardware configuration. As an example, one of the hypothetical systems is FPGA+ARM, so the hardware behavior is a platform with little memory and a low consumption and a processor with a low consumption and a low execution speed. Another hypothetical system is Raspberry Pi + I7, the hardware behavior is a platform with large memory and a significant consumption and a processor with a big consumption and a high execution speed. Role4All can generate a part of the Morphose code more or less substantial depending on the role model design. The figure 7 presents Morphose according to 3 levels: the system (1), the activities (2) and states machines/ hardware restrictions (3).

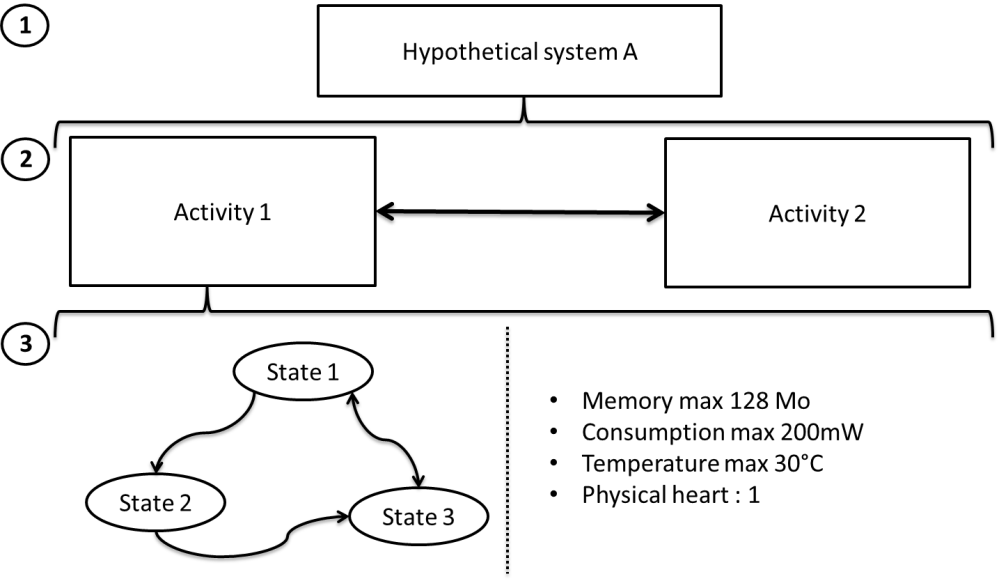


Figure 7 : Morphose design

Morphose returns two types of information, an executable code and many simulation measures (local or global consumption, memory consumption, etc.). The code produced by Morphose is executable in the real system and produced the same behavior than the real code (the same execution time, memory consumption, etc.). The terrorist can use this code to create a replica of the real system. The data returned by Morphose are an indicator of the real system, the user can use them to compare his simulation and the real system or to detect a failure in the real system.

Finally Morphose is a simulation tool based on the concept of activity, it use software behavior and hardware restriction to return executable code and hardware observations.

## Use Case

A cyber terrorist wants infected a system named CSUC with a homemade virus. CSUC is an object detection system composes of a camera, a FPGA and various processors/CPU. For our example we simplify the system to two elements: a platform (FPGA) and a processor (ARM) with an Ethernet connection. To do his mission the terrorist needs some information about the system (conception, consumption, etc.). To collect information he hacks the mailbox of a member of the project CSUC. The terrorist catch some important information: the global consumption of the system (2 750 mW/h) and a photo of the system. According to the photo, the terrorist detect that the system CSCU is compose of two elements, a platform and a processor. The terrorist limits his investigation to two platforms (Raspberry Pi and FPGA) and two processors (ARM and I7). Due to the worksheets of each product the terrorist can create an array to connect some product name (Raspberry Pi, FPGA, ARM, and I7) and their consumptions.

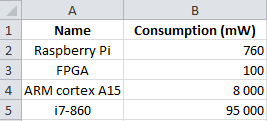


Figure 8: Excels file gathering some consumption

The second source of information (the photo) allows modeling a hypothetical system with Pimca.

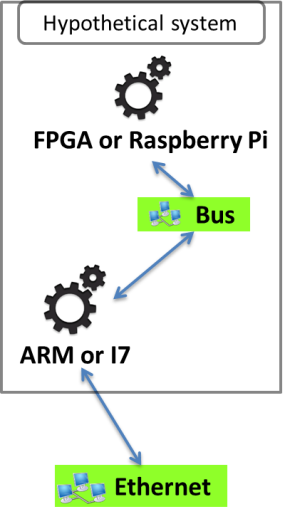


Figure 9: Pimca model of the hypothetical system

This model describes a simple system including two elements, a platform and a processor. But we have an undetermined about these elements, the processor is an ARM or an I7 and the platform is a FPGA or a Raspberry Pi. To solve our problem we simulate our systems (FPGA-ARM, FPGA-I7, etc.) and compare the consumption of the simulated systems and of the real system. Role4All is one solution to specify a hypothetical system and to run simulations required.

The cyber terrorist uses two tools (Pimca and Excel) and several concept common of these tools (FPGA, ARM, etc.). So he uses Role4All to federate his tools through concepts that he defined himself (FPGA, ARM, Raspberry Pi and I7). Each model element of each tool can play roles defined in a role model in Role4All (figure 3). The terrorist defines two main roles: *RolePlatform* and *RoleCPU,* who by definition of “role” extend the *Role* class defined in Role4All. In Role4All a role can play a role; this ability allows extending a role in order to adapt the role model without modifying the tool model or the role model. In our example the terrorist extends the role *RolePlatform* to two roles *RoleFPGA* and *RoleRaspberryPi* and the role *RoleCPU* to two roles *RoleARM* and *RoleI7*.

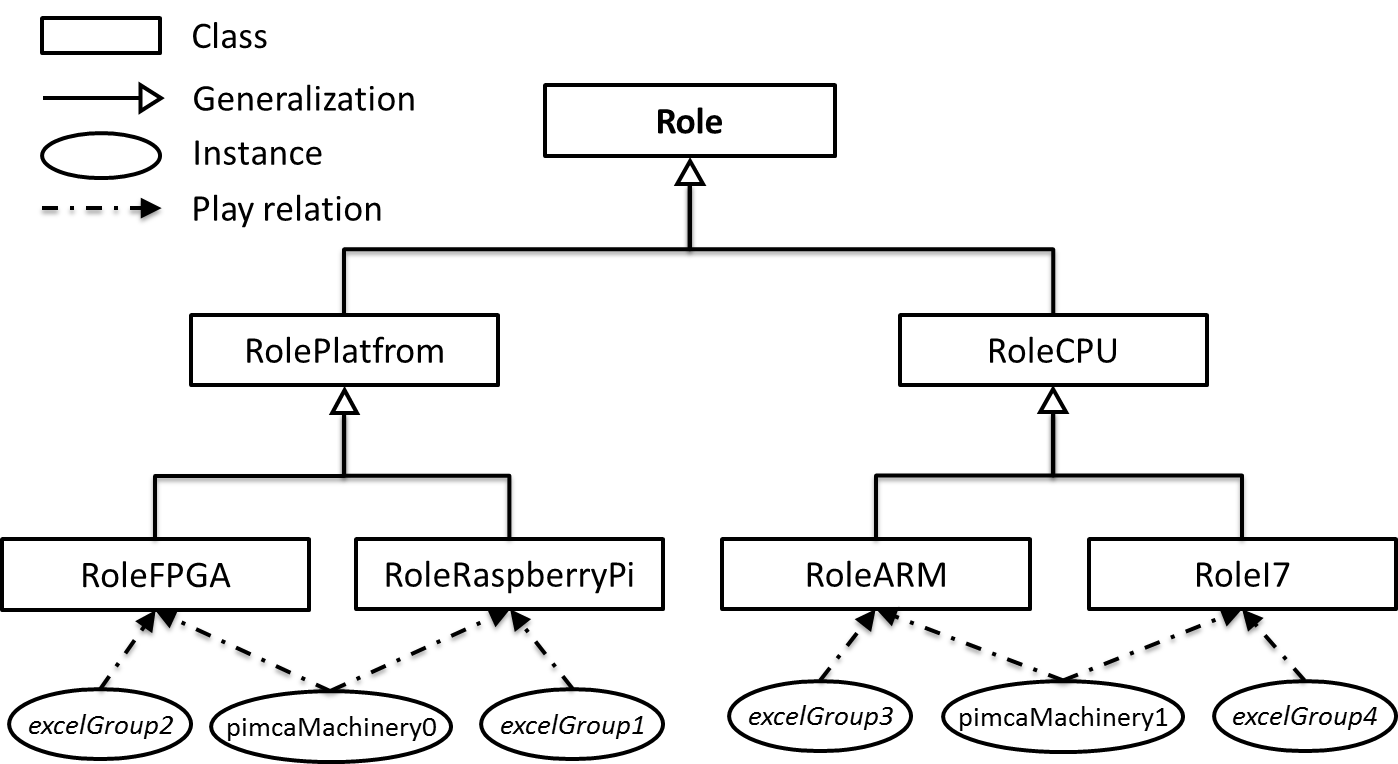


Figure 10: The role model create on Role4All by the terrorist

The role model of the figure 3 allows to the terrorist to define platform and processor’s concepts and to specify them to create new roles like FPGA or ARM. With this role model the terrorist can create a point of view on his tools and associate elements form various tools. As an example the Pimca’s element p*imcaMachinery0* and the Excel’s element *excelGroup2* (the equivalent of the line 3 of the Excel file) play the same role: *RoleFPGA*. Finally Pimca’s elements and Excel’s elements are linked through roles therefore the terrorist can create points of view on different elements of various tools. As example, due to the role model, the terrorist can manipulate the concept of FPGA instead of the couple of tools Excel and Pimca.

A point of view allows working in a specific environment on data provided by different tools (like Pimca and Excel) and sometimes the same information are included in different tools, in our example the Excel file and the Pimca model provide the same data: the ARM’s model. In this case, synchronization is necessary and Role4All provide a synchronization system through roles. So the terrorist defines some synchronization rules as an example he choose to synchronize the ARM model value on Pimca according to its value on Excel. In this specific case the “good” value will always be the value on Excel. Of course the user can create other synchronization rules adapted to the context. The figure 5 is an example of synchronization, the terrorist uses Excel to update an Excel element synchronized with a Pimca model element through a role (roleFPGA0).

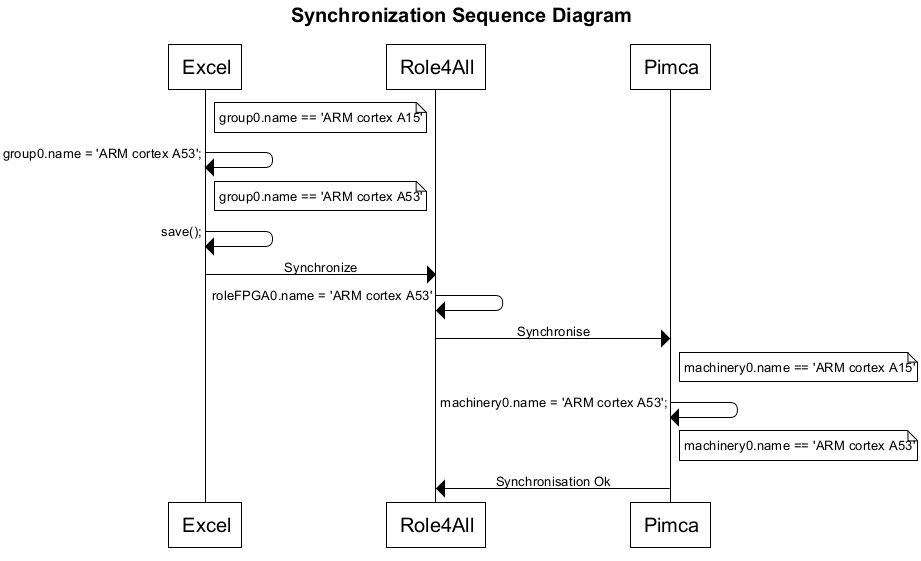


Figure 11 : A sequence diagram of a synchronization using Role4All

When the user saves his Excel file he sends a request to Role4All. Role4All detects the changes between the new model elements and the last ones and applies the suitable modifications to the Pimca model elements. The suitable modifications are defined due to the adapter between Role4All and Pimca associate with the role *roleFPGA0* (*dynamicAdapterFPGA\_Pimca0*). In this example the user use Role4All only to synchronize tools, but the main feature of Role4All is the creation of dedicated point of views. Therefore the common use case of the synchronization in Role4All uses the points of view. As an example to update the name of the role FPGA (the update performed in the figure 5) the user can use a point of view dedicated to the concept of FPGA or another one which gathering only the name of the elements. So the terrorist works with this point of view instead of tools and synchronize them due to Role4All.

# Conclusion

# Références

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

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,