

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

SOMMAIRE

1	INTRODUCTION.....	4
1.1	Contexte.....	4
1.2	Objectifs	4
1.3	Cas d’usage	6
2	APPROCHE DE MISE EN RELATION D’INFORMATION	7
2.1	Présentation	7
3	LANGAGES DE MODELISATION ET OUTILS	9
3.1	Le langage PimCA	9
3.1.1	Le modèle de connaissance.....	9
3.1.2	Les concepts principaux.....	10
3.1.3	Le modèle de scénarii d’attaque.....	11
3.1.4	Outillage PimCA.....	11
3.2	Langage de modélisation logiciel.....	11
3.3	Langage de modélisation matériel, Morphose	12
3.4	Langage de mise en relation de modèles : Role4All.....	13
4	AVANCEMENT DE L’ETUDE.....	15
4.1	Extension de Role4All pour la fédération.....	15
4.2	Use Case.....	17
5	CONCLUSION ET PERSPECTIVES	19
6	REFERENCES.....	21

1 INTRODUCTION

L'étude traite de la définition et de la découverte 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 principal 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 ou synthèse.

Cette unification des éléments de modélisation de l'architecture permet d'établir une carte d'identité de l'architecture afin 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.

1.1 Contexte

La cyber-sécurité des systèmes numériques est un enjeu critique et sociétal extrêmement prégnant tant pour les systèmes militaires que civils. De nombreux travaux liés à la sécurité 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 maîtriser les systèmes à sécuriser et par là même de caractériser une 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 maîtrisés, pour être transposées vers la sécurisation de systèmes connus et aussi à découvrir. Cet objectif doit permettre de réutiliser les connaissances sur les flots de conception.

Avant le prototypage qui permet de valider ou non les hypothèses, la capacité à mettre en relation différents points de vue de modélisation du système reste un point crucial à traiter. En effet, la construction des hypothèses de simulation cohérentes et incrémentales doit s'appuyer sur les points de vue mis en relation pour couvrir les enjeux de la cyber-sécurité.

1.2 Objectifs

Le projet a pour objectif d'adresser les aspects de modélisation de l'interface matériel-logiciel à partir de différents points de vue 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 de deux natures différentes avec :

- la partie de la plateforme matérielle sur laquelle on peut se poser les questions : quels processeurs sont susceptibles d'intervenir dans la conception ? Quels bus ? Quels périphériques ? De quelles marques ?
- la partie symbolique, ou applicative, du système avec les données échangées et le comportement 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 ? Quelle est l'implantation de cette carte mémoire dans les registres de communication ? Comment sont accédées ces données ?

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 classiquement employées dans la conception de SoC (system-on-chip), où deux à trois niveaux d'abstraction différents ont cours : comportemental, architectural et détaillé [1,2,9,10,11].

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 et assurer une cohérence forte ces différentes informations. Cette mise en relation reste actuellement un problème difficile car les solutions existantes restent très limitées. Les approches les mieux caractérisées sont :

- l'intégration de concepts en rassemblant tous les concepts dans un même langage. L'énonciation même de cette définition met en exergue la limitation de cette approche dès lors que de nouveaux concepts peuvent être intégrés.
- L'unification de concepts qui cherche à identifier des concepts communs entre différents langages, puis d'établir un alignement entre les langages et ce modèle souvent appelé « pivot ». La limitation majeure cette approche reste la définition de périmètre de l'ensemble de ces concepts qui doit être suffisant pour adresser les différents langages mais qui doit rester un sous-ensemble des concepts de l'intégration. Là encore ce périmètre est très difficile à identifier pour garantir une bonne couverture des concepts utiles.
- La fédération de concepts qui se focalise sur la modélisation des correspondances entre concepts et non plus des concepts eux-mêmes. Cette approche relâche la contrainte de l'identification des concepts communs ce qui est un apport capital. De plus l'introduction de nouveaux concepts ne remet pas en cause les modélisations déjà effectuée mais oblige à étendre celles-ci. Cependant comment définir cette fédération et comment gérer en cohérence cette extension, restent encore à définir.

Donc définir et maîtriser la mise en relation des informations issues de différents langages restent encore à bien caractériser et bien formaliser, même si la fédération semble apporter une réponse partielle à notre problème.

Mais dans notre cas, ces informations sont généralement disparates et incomplètes, il faut donc les raffiner au fur et à mesure de la découverte du système. Donc nous devons avoir une possibilité de gérer la dynamique et l'incomplétude de nos informations.

L'incomplétude sera nécessairement bornée car nous nous astreignons, dans la mesure du possible, à conserver nos modèles exécutables dans un but de simulation et d'évaluation au plus tôt.

La modélisation outillée que nous nous proposons de mettre en œuvre repose sur une méthodologie agile basée sur des modèles hétérogènes allant du niveau système jusqu'à la plateforme matérielle. Pour se faire, nous activons une pile logicielle avancée basée sur : de la fédération de modèles, de la génération de code, de la synthèse comportementale, pour estimer des paramètres de l'architecture embarquée ciblée. 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.

1.3 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 système embarqué 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) [6] 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'un scénario 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 la figure 1 suivante.

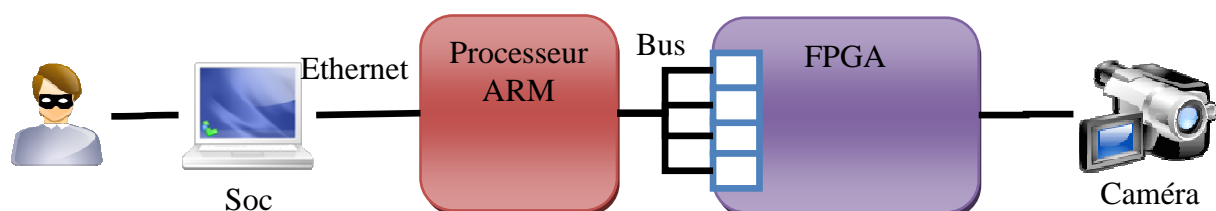


Figure 1: Architecture embarquée du cas d'étude

2 Approche de mise en relation d'information

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.

2.1 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, Pimca, langage de programmation). D'autres informations pourraient être nécessaires (adresses IP, consommation de la plateforme...) qui pourraient être fournies par d'autres modèles ou sources d'informations (feuilles excel, métadonnées).

Le système embarqué étant par nature multipoints 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 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 mise en relation 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 des 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 rôles à des objets ou éléments de modèles.

En effet, un type attribue la nature intrinsèque d'un élément et le rôle est un élément spécifique associé à ces éléments typés selon les différents points de vue où l'on cherche à interpréter l'élément initial. Les rôles sont donc en relation avec l'élément initial relatif à un certain contexte de définition ou de modélisation. De part cette caractéristique la modélisation de point de vue en utilisant ce concept de rôle permet de fournir différentes interprétations à des éléments de modélisation.

De plus une des caractéristiques essentielles des rôles est de pouvoir s'associer et se dissocier dynamiquement des éléments de modélisation. Et cette caractéristique nous semble essentielle dans notre contexte pour prendre en compte les différentes itérations de modélisation et d'évaluation des hypothèses d'architectures.

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 [13,14,15]. Le framework Role4All a été conceptualisé et mis en œuvre pour offrir la capacité d'unifier différents langages de modélisation système sur un modèle dit « pivot » avec la capacité d'évoluer dynamiquement.

Un des objectifs de cette étude est d'étendre le framework afin de permettre la fédération des éléments de modélisation de notre système afin de fournir la nécessaire flexibilité pour notre modélisation.

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 qui fait partie des outils développés en propre et en collaboration avec ses partenaires du Lab-STICC [7,8,9]. La maîtrise technologique de la phase de synthèse de haut-niveau 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.

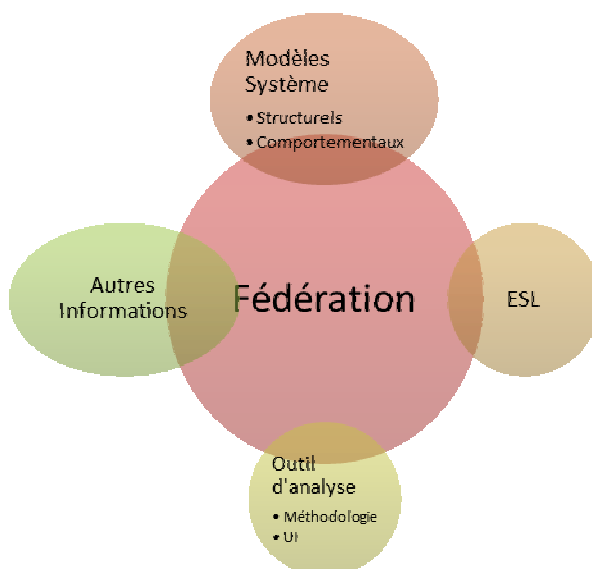


Figure 2: L'approche générale

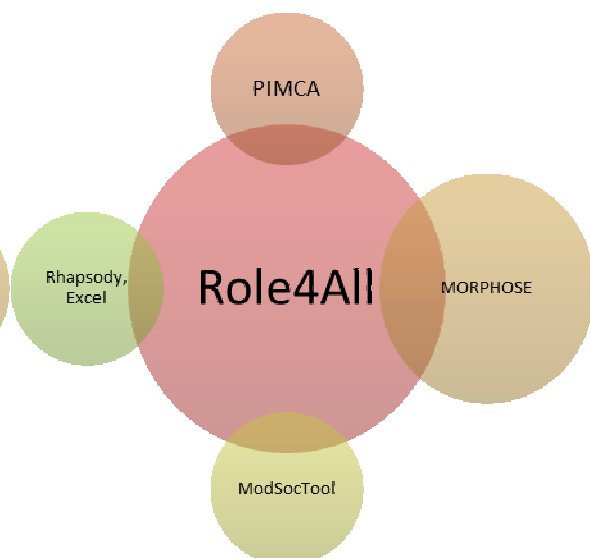


Figure 3: L'approche retenue pour notre cas d'étude

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

3 Langages de modélisation et outils

3.1 Le langage 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. Le langage PimCA permet de modéliser des systèmes à différents niveaux d'abstraction. La modélisation de la structure du système et aussi les graphes d'attaques possibles sont exprimables dans le langage.

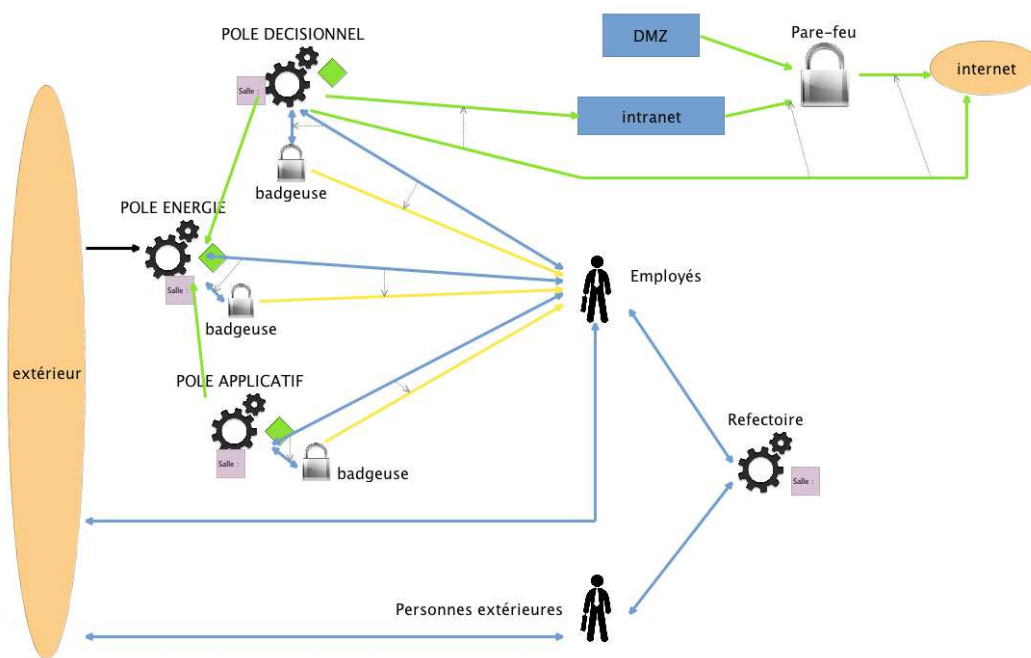


Figure 4: Un modèle réalisé avec l'outillage PimCA

Le langage possède deux grandes composantes :

- Un modèle de connaissance,
- Un modèle de scénarii d'attaque,

3.1.1 Le modèle de connaissance

Le modèle de connaissance (Knowledge Model) représente un système faisant l'objet d'une analyse de sécurité. 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), cf figure 5.

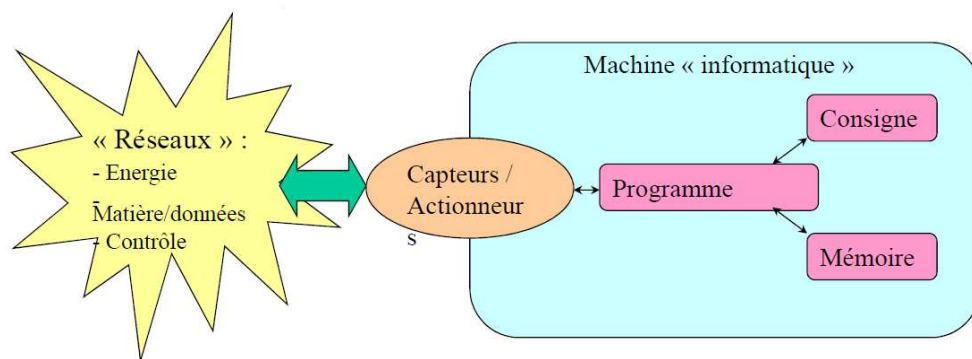









Figure 5 : Schéma des concepts principaux de Pimca




La machinerie est caractérisée par trois concepts clés : exécution – configuration – mémoire avec leur interface.

3.1.2 Les concepts principaux

Les « machines » :

Icône-Concept	Description
 Machinery	Machinerie : système manipulant des Ressources (regroupement particulier) : voiture, animal, PC, processus
 Performer	Exécutant (spécialise Machinerie) : ce qui transforme la Ressource, e.g. UC/Programme, cerveau, régulateur.
 Network	Réseau (spécialise Machinerie) : zone d'échange de matière, d'information, d'énergie, etc. : câblage, tuyauterie, IPC Engine.
 Customs	Douane (spécialise Machinerie) : fonctionnalité particulière mise en place par une Machinerie pour identifier & autoriser une autre Machinerie : cadenas, garde, login, crypto
 Interface	Interface (spécialise Machinerie) : permet de passer d'une Machinerie à une autre, du monde physique au monde virtuel et inversement : NIC, caméra, clavier, écran.
 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. exécutant, configuration, mémoire.
 Repository	Entrepôt : zone de stockage de Ressource : armoire, bâtiment, disquette, database, file system

Les « ressources » :

Icône-Concept	Description
 Resource	Ressource : ce qui est transformé, manipulé par une Machinerie : matière, électricité, document, log, data
 Instructions	Consigne (spécialise Ressource) : La direction, les paramètres que l'exécutant suit : Fichier de configuration, ordre, politique de sécurité
 Passeport	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

Un des intérêts de ce modèle de connaissances est la capacité de modéliser des systèmes à différents niveaux d'abstraction ou de raffinement. Dans la suite de l'étude, nous utiliserons le langage PimCA pour décrire le système numérique dans son contexte mais aussi les différentes architectures étudiées.

3.1.3 Le modèle de scénarii d'attaque

Ce modèle de scénarii d'attaque regroupe l'ensemble des possibles et représente un support d'analyse pour la sélection du chemin d'attaque le plus sûr. Ce modèle est basé sur le séquençement des activités définissant un scénario, 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 : observer le système, raisonner sur le système et agir sur le système. Différentes actions pour agir sur le système étant possibles. 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.

3.1.4 Outillage PimCA

L'outillage PimCA est basé sur le framework EMF(Eclipse Modeling Framework) avec un métamodèle 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. Cet outillage a été développé au cours de l'année 2015. Ainsi les modèles de niveau système et les modèles des architectures numériques seront réalisés avec cet outillage dans la suite de nos expérimentations.

3.2 Langage de modélisation logiciel

En complément des modèles systèmes, nous nous donnons la possibilité d'utiliser un modelleur UML qui permet de décrire la partie logicielle sous différents points de vue surtout 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 modelleur, nous estimons qu'un modèle UML permettrait de capitaliser un certain nombre d'informations relatives à l'architecture logicielle indépendamment mais qui devront être mis en cohérence avec le modèle système. A charge de définir en UML quels seraient le formalisme adéquat pour offrir la meilleure capitalisation.

3.3 Langage de modélisation matériel, Morphose

The simulation is possible due to a specific tool: Morphose. This simulation tool has a dedicated language based on the concept of activity and a system is modeled by a collection of interconnected activities. An activity is an association of two entity a software behavior and the hardware platform specification. The first entity describes the software behavior through sequential code. The second one describes the hardware features like the memory consumption, the energy consumption, the number of physical hearts. In Morphose the user implements the behaviors of the system and not the system himself, so according to our example the analyst can implement the behavior of his hypothetical systems with Morphose.

The analyst focuses on the hardware system therefore he generates 4 simulations with the same software behavior but with 4 different hardware configurations. As an example, one of the hypothetical systems is FPGA associated with a ARM processor, so the hardware behavior is a platform with little memory and a low consumption processor with a low frequency. Another hypothetical system is Raspberry Pi + I7. In this case, the hardware platform includes a large memory and a processor with a high consumption with a high frequency.

Role4All is used to generate the Morphose code based on the role model of the viewpoint. The figure 6 presents Morphose according to 3 levels: the system (1), the activities (2) and control flow states machines/ hardware specifications (3).

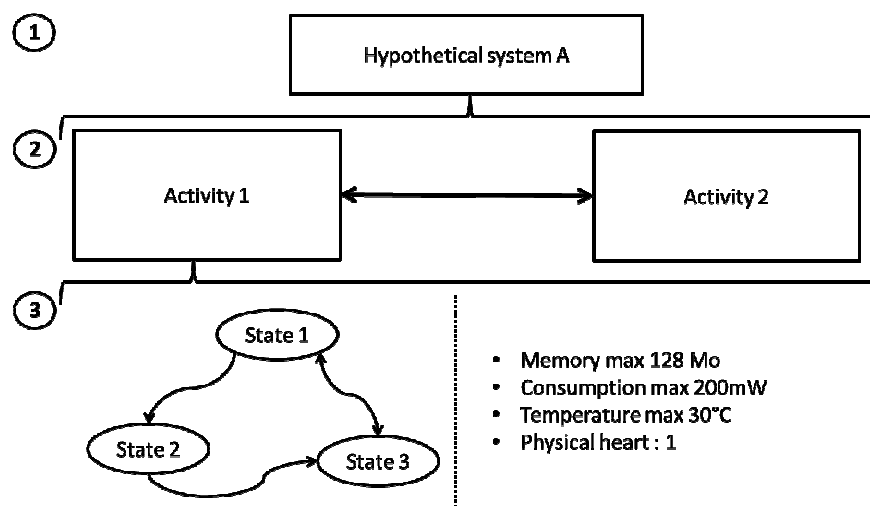


Figure 6 : Morphose design

Morphose returns two types of information, a target code and simulation measures (local or global consumption, memory consumption, etc.). The code produced by Morphose is executable on the target system and also produces evaluations by simulation for the execution time, memory

consumption. The analyst can use the produced data to compare the simulated architecture to the real system or to detect a difference with the real system.

3.4 Langage de mise en relation de modèles : 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 d'éléments, et de plus une même information est amenée à être manipulée par plusieurs outils. Entre ces différents outils, une interopérabilité tant technologique que sémantique doit être assurée pour échanger et propager les informations. Cet échange d'information est l'un des importants problèmes dans les chaînes d'ingénierie nécessairement basées sur un nombre important d'outils. C'est pourquoi il existe principalement en trois grandes approches pour traiter ce problème : l'intégration, l'unification et la fédération.

L'intégration consiste en la fusion des modèles d'éléments utilisés, ce qui correspond à la création d'un modèle 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, intégrant de plus en plus d'informations fondées sur des sémantiques différentes. L'intégration est donc une solution adaptée aux systèmes basés sur un faible nombre de concepts et fixe, 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 possédant 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 termes d'échanges nécessitent le développement de modèle pivot de plus en plus complexes. Il est alors nécessaire d'adapter le modèle pivot à l'évolution de l'environnement d'ingénierie. Sinon plusieurs modèles pivots peuvent être créés pour assurer une interopérabilité entre plusieurs outils. Dans ce cas, la gestion et surtout l'évolution des modèles pivots devient un obstacle important à l'interopérabilité. 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 métamodèle). 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 du concept 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 [15]. 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 en ajoutant dynamiquement un nouveau rôle ce qui permet de tirer parti de la grande flexibilité du modèle 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 [15]) créait un lien unidirectionnel entre les modèles des outils et Role4All, ce qui est incompatible avec la notion de fédération (nécessitant une relation conceptuelle 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.
(par la suite nous réutilisons un texte en anglais)

Role4All language is mainly based on four main concepts: *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*. This example is fully described in the next chapter.

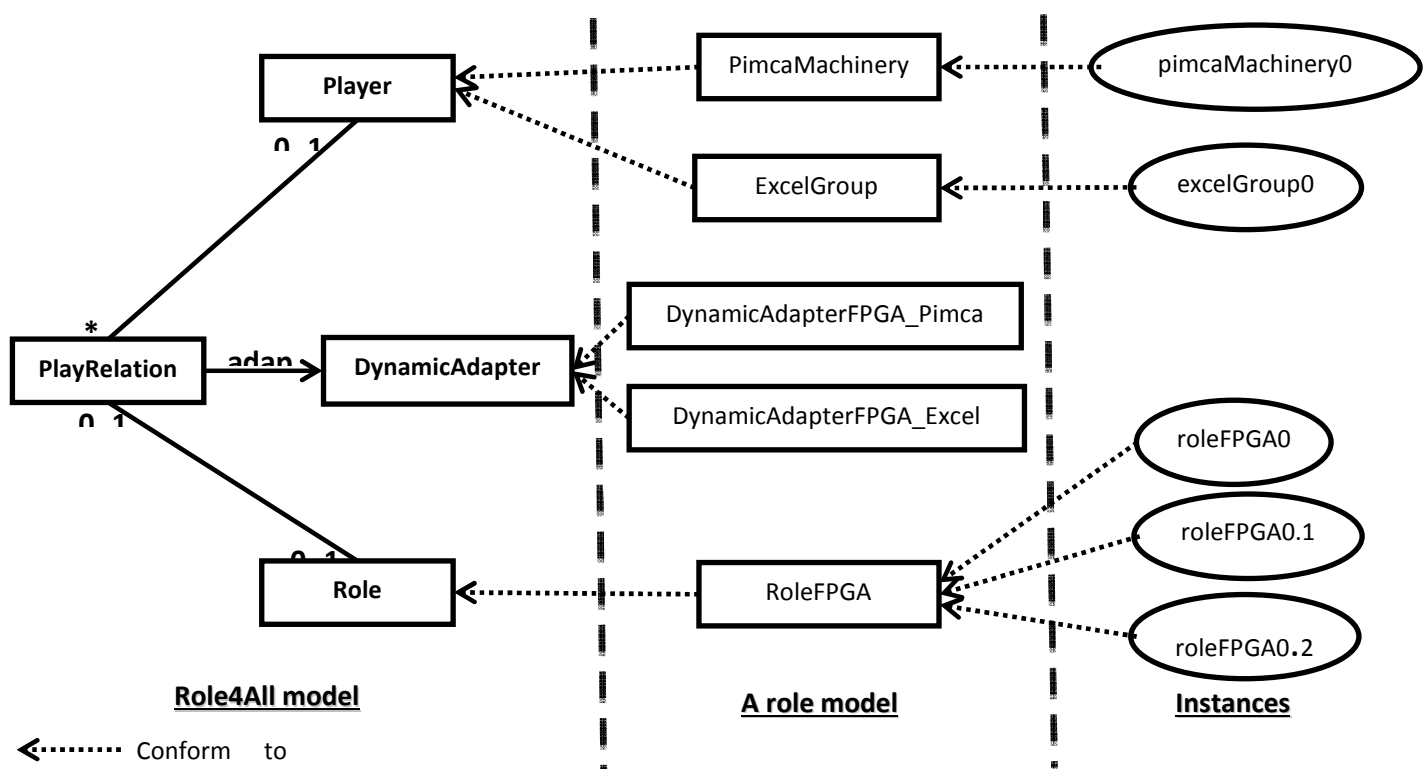


Figure 7: Role4All's meta-model illustrates with the example of the role of FPGA

The elements extending the class *Player* are the model type of the language elements; they are called “players”. For example *PimcaMachinery* (cf figure 7 for all element names) is an element of Pimca’s meta-model and *pimcaMachinery0* is an element of Pimca’s model (detailed after in the figure 10). We want Pimca’s model element *pimcaMachinery0* to play the role of *RoleFPGA*, therefore the model element *PimcaMachinery* is defined as a *Player*. The connection between a role model element *RoleFPGA* and a tool element *pimcaMachinery0* is transformed by an adapter.

The concepts conformed to the Role class, are called “roles”. As an example in the figure 7 *RoleFPGA* is a role.

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 created through the class *DynamicAdapter* allow transforming 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 7 *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 from the class *PlayRelation* are connectors without behavior between three elements: a role, a player and an adapter.

So according to figure 7, the player *pimcaMachinery0* plays the role *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 from the two different tools. Now we can use the role *RoleFPGA* to handle the concept of FPGA instead of the tools Excel and Pimca.

4 Avancement de l’étude

Afin de prendre en compte un cycle itératif de découverte, évaluation et prise en compte des résultats des simulations, il nous a fallu étendre le framework Role4All pour y intégrer un mécanisme de fédération. Cette extension a pour objectif de fournir au modèle de rôle une capacité de constituer la base de référence pour interconnecter les différents formalismes mis en œuvre pour la découverte des systèmes.

Donc nous allons présenter cette extension et ensuite son utilisation sur le cas d’usage du système présenté à la section 1.3.

4.1 Extension de Role4All pour la fédération

Nous avons mis en avant la nécessité d’étendre 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 cette limitation qu’il est nécessaire d’apporter deux mises à jour majeures à Role4All :

- Création d’une bijection entre les instances de rôles
- Mise à jour du lien modèle-rôle afin de le rendre bidirectionnel

La première étape vise à créer un lien entre deux instances de rôles afin de permettre la synchronisation entre ces instances. Cela permettrait de propager les modifications apportées d’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, et en revanche la mise à jour des éléments de modèles des outils et les instances de rôles ne sera pas traitée dans ce rapport.

In Role4All all model elements play a role therefore all model elements are strongly linked with a role instance, as it was explained in the previous part. As an example, in figure 8 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 relations called *containedRoles* and *containerRoles* that are collections of role instances. So, a role instance can contain several role instances and be contained by several ones. We use this relation to connect together the role instances that must be synchronized.

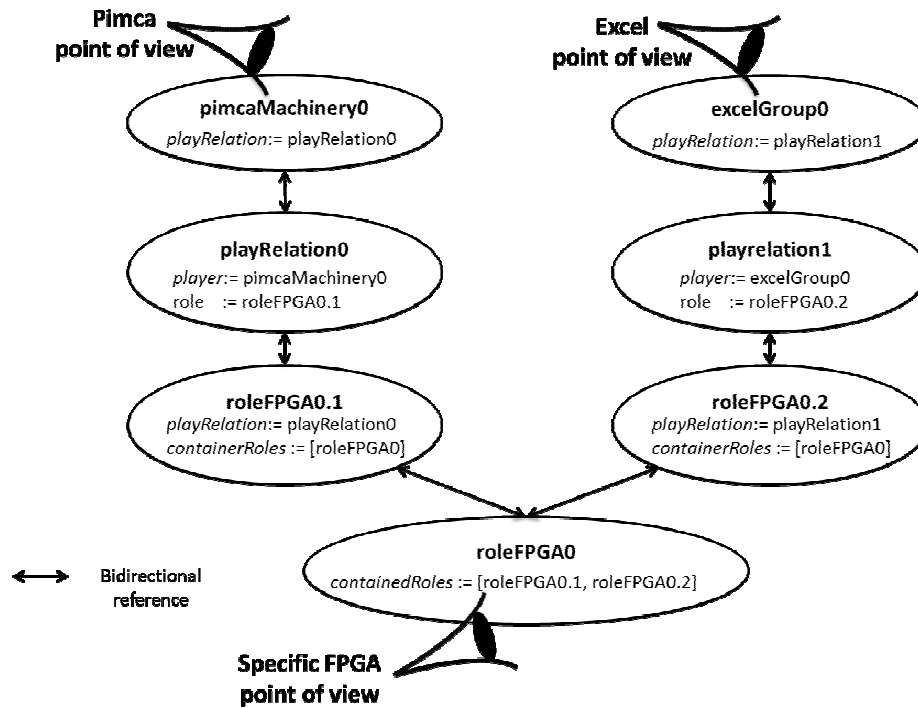


Figure 8: Roles synchronization in Role4All

The figure 8 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 *containedRoles* and the instances *roleFPGA0.1* and *roleFPGA0.2*, which know in which instances they are contained through the *containerRoles*. So we have a bidirectional relation between role instances through a role container (*roleFPGA0*). Furthermore the container role is an aggregation of the contained roles, therefore the container role allows to define a specific point of view.

The synchronization methods are developed in the container role, by default it is a check-out/check-in synchronization. All the changes are immediately reflected in all the synchronized tool elements according to the synchronization rules. The synchronization is related to the context (Network management, Cybersecurity, etc.), therefore Role4All allows to customize the synchronization. A user can define various rules and check them when he creates synchronization between tool elements with Role4All, for example a prioritization of the synchronizations according to the skill of the user. Moreover the default synchronization mechanism can be changed for another one, like the Long Transaction Model designed to support the evolution of whole systems as a series of

apparently atomic changes [16]. The main idea with the synchronization in Role4All is to be independent of the tool languages and easily customizable.

4.2 Use Case

A cyber analyst wants to understand an object detection system composed by 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 achieve his mission the analyst needs some information about the system (conception, consumption, etc.). He collects information from the mailbox of a member of the system designer. The analyst catches some important information: the global consumption of the system (2750 mW/h) and a picture of the system. According to the picture, the analyst detects that the system is composed of two elements, a accelerator platform and a processor. The analyst limits his investigation to two accelerator platforms (Raspberry Pi and FPGA) and two processors (ARM and I7). Due to the worksheets of each product the analyst can create an array to connect some product name (Raspberry Pi, FPGA, ARM, and I7) and their consumptions, see figure 9.

	A	B
1	Name	Consumption (mW)
2	Raspberry Pi	760
3	FPGA	100
4	ARM cortex A15	8 000
5	i7-860	95 000

Figure9: Excels file gathering some consumption

The second source of information (the picture) allows modeling a hypothetical system with Pimca, presented in the next figure 10.

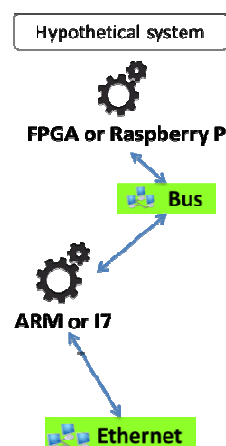


Figure 10: Pimca model of the hypothetical system

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 want to simulate our systems (FPGA-ARM, FPGA-I7, etc.) and compare the consumption between the simulated systems and the real system. Role4All is used to specify a view point dedicated to the required simulation.

The analyst uses two tools (Pimca and Excel) and several common concepts of these tools (FPGA, ARM, etc.). So he uses Role4All to federate elements through roles that he defined himself (FPGA, ARM, Raspberry Pi and I7). Each model element can play roles defined in a role model (figure 11). The analyst defines two main roles: *RolePlatform* and *RoleCPU*, by using the *Role* class defined in the framework. In our example the analyst extends the role *RolePlatform* to two roles *RoleFPGA* and *RoleRaspberryPi* and the role *RoleCPU* to two roles *RoleARM* and *RoleI7*.

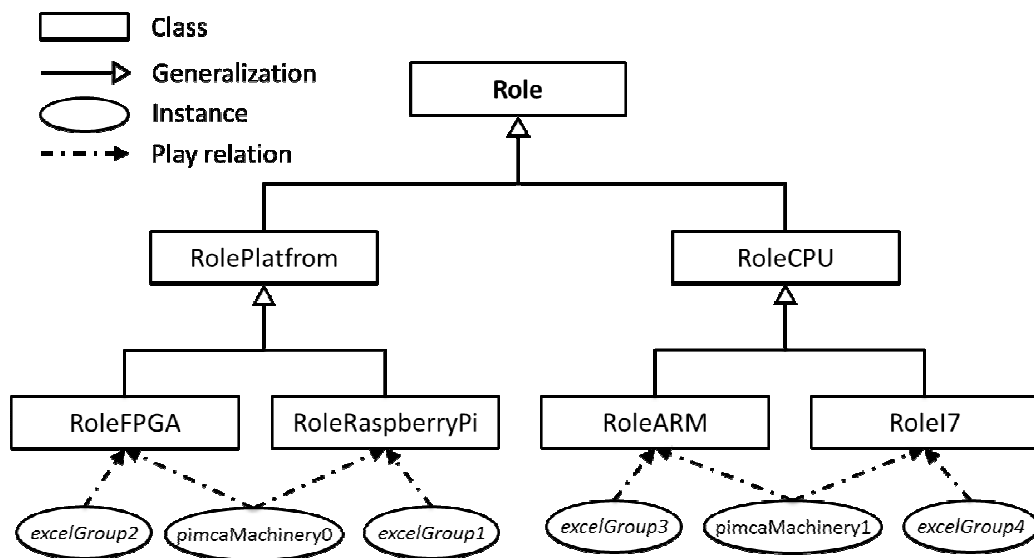


Figure 11: The role model create on Role4All by the analyst

The role model of the figure 11 allows to the analyst to define platform and processor's concepts and to specify them to create new roles like FPGA or ARM. With this role model the analyst can create a view point on his tools and associate elements form various tools. As an example the Pimca's element *pimcaMachinery0* 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 analyst can create view points on different elements of various tools. As example, due to the role model, the analyst can manipulate the concept of FPGA instead of the couple of tools Excel and Pimca.

A viewpoint is dedicated to a specific environment provided by different tools (like Pimca and Excel). Sometimes the same information is included in different tools. In our example the Excel sheet and the Pimca model store the same concept with or without the same name but definitively the same data (e.g the ARM's model). In this case, Role4All provides synchronization through the role model. So the analyst defines some synchronization rules such as we must synchronize the ARM model value from the model Pimca according to the value from the Excel model. In this case the reference value remains the value from the Excel sheet.

Of course the user can create other synchronization rules adapted to the context. The figure 12 illustrates a synchronization example where the analyst uses the Excel tool to update an element, synchronized with a Pimca element through the role, roleFPGA0.

Synchronization Sequence Diagram

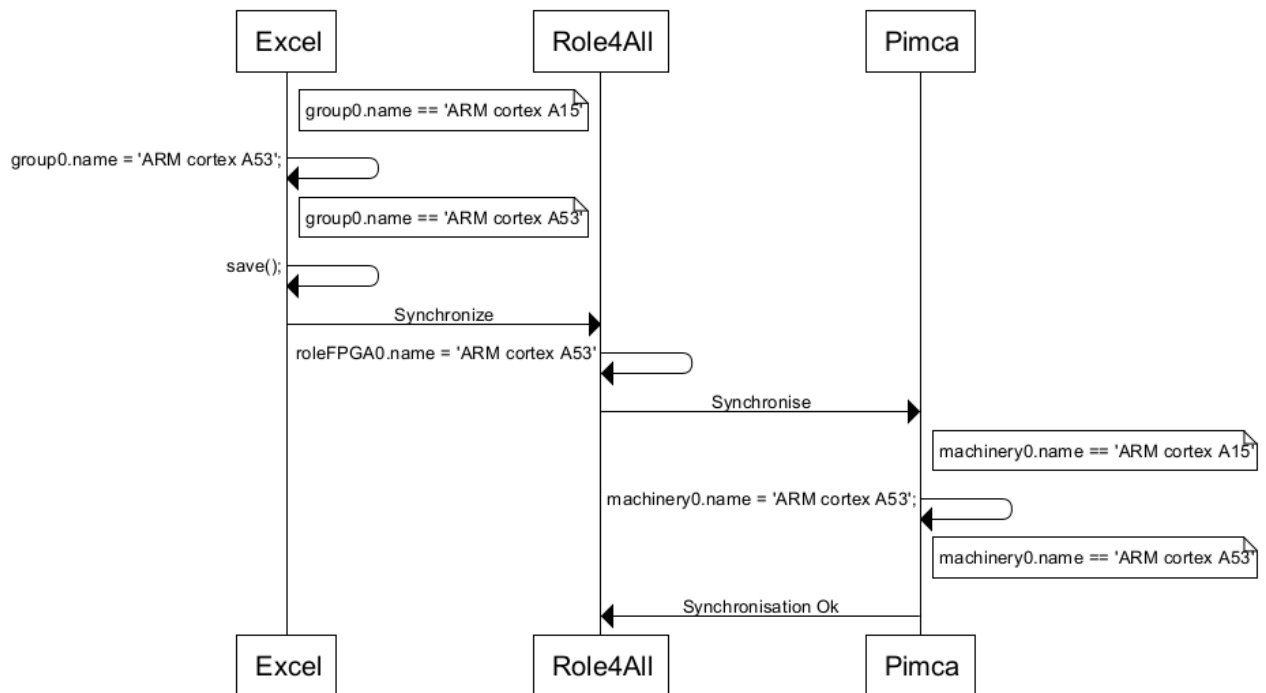


Figure 12: A sequence diagram of a synchronization using Role4All

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 viewpoints. Therefore the common use case of the synchronization in Role4All uses the viewpoints. As an example to update the name of the role FPGA (the update performed in the figure 12) 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 analyst works with this viewpoint instead of tools and synchronizes these tool models with the support of Role4All.

5 Conclusion et perspectives

Pour conclure ce rapport, nous pouvons dire que nous avons étendu le framework Role4All pour prendre en compte la nécessité d'effectuer la fédération de modèles. Cette fédération permet de nous focaliser sur le point de vue de modélisation de l'analyste du système permettant une évaluation de l'architecture à modéliser. La fédération a été mise en œuvre sur le cas d'étude modélisé à l'aide de Pimca, et complétée avec des données (ou meta-données) récoltées et stockées dans un fichier de type Excel. La fédération permet donc d'assurer des différents modèles d'entrée pour obtenir un point de vue facilitant l'analyse du système. Les différentes architectures candidates

pourront donc être simulées et évaluées à l'aide de l'outillage Morphose comme le décrit la figure 13, à suivre.

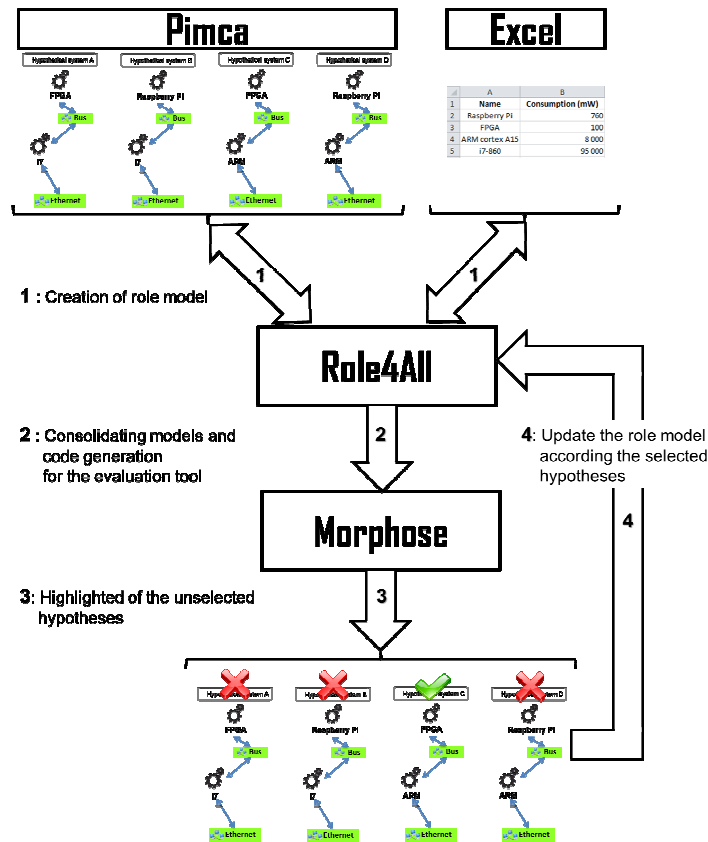


Figure 13 : Schéma général de l'approche

La poursuite de l'étude va exploiter ce mécanisme de fédération pour prendre en compte les analyses effectuées par l'outil Morphose et maintenir en cohérence les modèles de description du système (étape 1 de la figure 13) et les résultats de Morphose (étape 4 de la figure 13).

Pour mener à bien cette étape nous allons, dans un premier temps, intégrer une phase de génération de code pour l'outillage Morphose à partir du modèle de rôles (étape 2 de la figure), puis utiliser Morphose pour évaluer les différentes architectures candidates.

L'étape de sélection de l'architecture retenue sera dans un premier temps effectuée par l'analyste. Une modélisation des résultats de Morphose devra être effectuée pour mener à bien l'étape 4 de la figure 13 pour consolider les modèles de rôles et par là même les modèles d'entrées, en se reposant sur le mécanisme de fédération de Role4All.

6 Références

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,
6. Edward A. Lee - Position Paper for NSF Workshop On Cyber-Physical Systems: Research Motivation, Techniques and Roadmap October 16-17, 2006, Austin, TX
7. Lagadec L., Le Lann Jean-Christophe, Bollengier T. A Prototyping Platform for Virtual Reconfigurable Units. Recosoc 2014 - May, Montpellier France.
8. Julien Heulot, Karol Desnos, Jean-François Nezan, Maxime Pelcat, Mickaël Raulet, Hervé Yviquel, P.-L. Lagalaye, J-C Le Lann. An experimental toolchain based on high-level dataflow models of computation for heterogeneous MPSoC. DASIP'12, 2012
9. Jean-Christophe Le Lann, Joël Champeau, Papa Issa Diallo, Pierre-Laurent Lagalaye. From system-level models to heterogeneous embedded systems, RITF 2012 - Recherche et Innovation pour les Transports du Futur, Paris : France.
10. Jean-Christophe Le Lann, Philippe Dhaussy, Pierre-Laurent Lagalaye. Modélisation algorithmique et synthèse d'architectures assistées par model-checking, CAL 2012, Montpellier : France.
11. Ali Koudri, Joël Champeau, Jean-Christophe Le Lann and Vincent Leilde. MoPCoM Methodology: Focus on Models of Computation. ECMFA'2010, Paris
12. J. Champeau, V. Leildé, and P. I. Diallo, Model federation in toolchains. Workshop “Semantic Information Modeling for Federation “ in conjunction with MODELS 2013.
13. Jean-Philippe Schneider, Joël Champeau, Ciprian Teodorov, Eric Senn and Loïc Lagadec. A Role Language to Interpret Multi-Formalisms System of Systems Models. IEEE International System Conference, Vancouver, April 13-15, 2015.
14. Jean-Philippe Schneider, Joël Champeau, Loïc Lagadec and Eric Senn. Role Framework to Support Collaborative Virtual Prototyping of System of Systems. WETICE Conference June 15-17 2015.
15. Jean-Philippe Schneider, Les rôles : médiateurs dynamiques entre modèles système et modèles de simulation. Thèse de doctorat, Université de Bretagne Occidentale, Novembre 2015.
16. Peter H Feiler, Configuration Management Models in Commercial Environments, Carnegie Mellon University Technical Report, 1991.