Collaborative Role Modeling in a Cybersecurity Context

# Introduction

<http://www.isaca.org/Knowledge-Center/Blog/Lists/Posts/Post.aspx?ID=296>

<http://whatis.techtarget.com/definition/cybersecurity>

<https://www.dhs.gov/topic/cybersecurity>

Cybersecurity is the body of technologies, processes and practices designed to protect networks, computers, programs and data from attack, damage or unauthorized access. Cybersecurity is a complex concept including several elements like:

* [Application security](http://searchsoftwarequality.techtarget.com/definition/application-security)
* Information security
* Network security
* End-user education.

The diversity of elements involves the co- operation of the various services (Developer, Network manager, Managers, etc.). Each services use some specific tools adapt for their skills and others more common but for some specific tasks. Role4All allows simplifying this co-working with two main services:

* Specific point of view of a complex tools
* Synchronization of various tools

One of the most problematic elements of cybersecurity is the quickly and constantly evolving nature of security risks. Indeed today the most secure system without management is sure for a very short time (less than one year). The traditional approach has been to focus most resources on the most crucial system components and protect against the biggest known. Such an approach is insufficient in the current environment. The National Institute of Standards and Technology (NIST), for example, recently issued updated guidelines in its risk assessment framework that recommended a shift toward continuous monitoring and real-time assessments. Unlike the pivot model solution, Role4All is a dynamic federation tool and allows to update some elements of a system group without interact with the others.

The following parts present the role-based framework Role4All according to a cyber-attack example.

# Application to the cybersecurity context

A cyber terrorist wants infected a system named TEST with a homemade virus. To do his mission he needs some information about the system (conception, consumption, etc.). For our example we simplify the system to two elements: a platform (FPGA) and a processor (ARM) with an Ethernet connection. To collect information the terrorist hacks the mailbox of a member of the project TEST. 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 TEST is compose of two elements, a platform and a processor. The terrorist limits his investigation to two sources (Raspberry Pi and FPGA) and two processors (ARM and I7). Due to the worksheets of each product the terrorist can create an array to create a relation between some product name (Raspberry Pi, FPGA, ARM, and I7) and their consumptions.

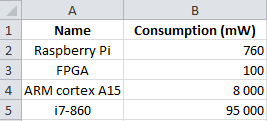


Figure 1: Excels file gathering some consumption

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

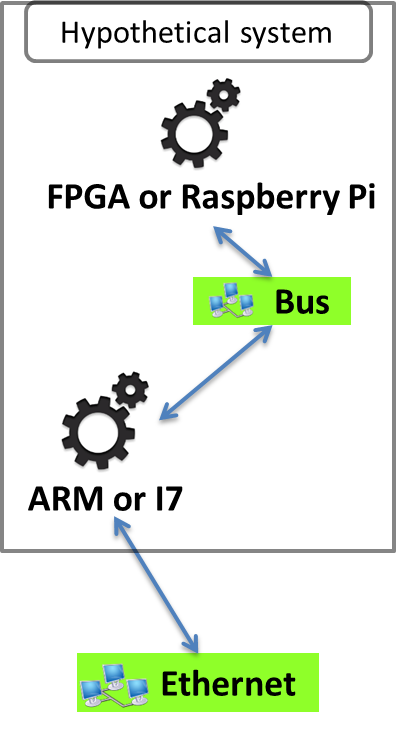


Figure 2: Pimca model of the hypothetical system

This model describes a simple system including two elements, a platform and a processor. But we have an undetermined about the 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.

# Federation Based on Roles

## The Necessity of Federation

In most projects we use several specific tools to model systems or manipulate data, and sometimes these tools use the same model or data. Therefore a collaborative process is necessary, it is called interoperability. Unification is a solution to create interoperability between two tools that consists on creating a pivot model shared by the tools. This solution is simple to implement if all the tools are linked to the same pivot model. But the unification creates two main problems, the first emerges when the pivot model is modified, indeed by definition of a pivot model all tool models linked with it need to be updated too. The second one comes when a new tool is added to the project, it is necessary to connect the new tool with the pivot model without modifying the pivot model (cf problem one). This task can be a real problem if the tool model and the pivot model are strongly discordant. Moreover the tool models progress faster than standards considering that we cannot use a pivot model for a dynamic interoperability [thesis of Maud RIO]. Therefore other solutions were imagined to create a dynamic interoperability, like federation systems. Role4All is an example of federation system, in Role4All a role model defines as a pivot model allows the interoperability between several tools.

In our applicative example the cyber terrorist uses two tools (Pimca and Excel) and several concept common of two tools (FPGA, ARM, etc.). The terrorist uses Role4All to federate his tools (Pimca and Excel) through concepts that he defined himself (FPGA, ARM, Raspberry Pi and I7). In each tool, each model element can play roles defined in a role model (figure 3) in Role4All. Moreover all elements can play roles including elements of unrelated type, therefore with Role4All it is possible to federate all tool types.

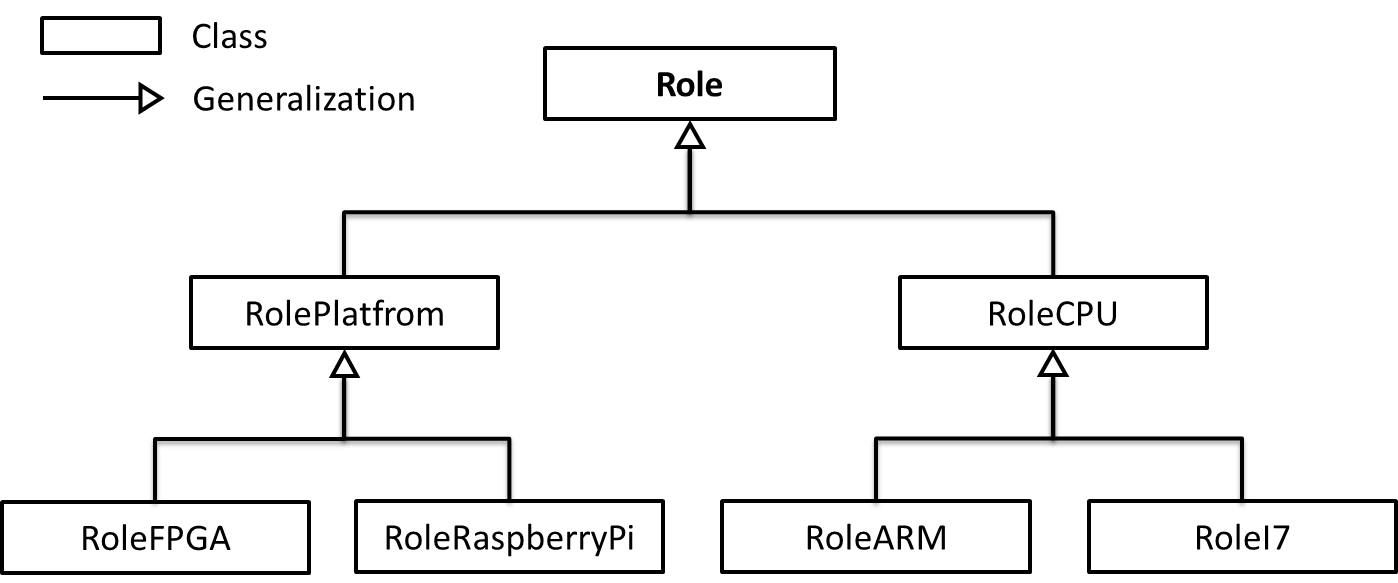


Figure 3: The role model create on Role4All by the terrorist

In our example the terrorist defines two main roles: *RolePlatform* and *RoleCPU* these roles are generalizations of the concept of Role 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*. 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 concepts like FPGA or ARM. The ability to extend or regroup roles provided by Role4All allows also creating different points of view about the same model according to the context [old article].

## Role4All a Dynamic Federation Tool

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

The elements extending the class *Player* are the model elements of tools, they are called “player”. In our example the terrorist uses two tools (Pimca and Excel) each tool has a meta-model whose complexity depends on the tool [ExcelMeta-model, PimcaMeta-model]. For example *PimcaMachinery* is an element of the Pimca’s meta-model and p*imcaMachinery0* is an element of a Pimca’s model (figure 2). The terrorist wants than the Pimca’s model element *pimcaMachinery0* play the role of *RoleFPGA* and he uses Role4All to create this relation. The connection between a role model element (*RoleFPGA*) and a tool element (*pimcaMachinery0*) is formatted by an adapter. Unlike a pivot model, a role model is independent of the tools, for example if a tool model is updated we need to adapt the adapter but not the role model himself.

The elements extending the class *Role* are the concepts created by the user through the role models, they are called “role”. Therefore, 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.

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

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*.

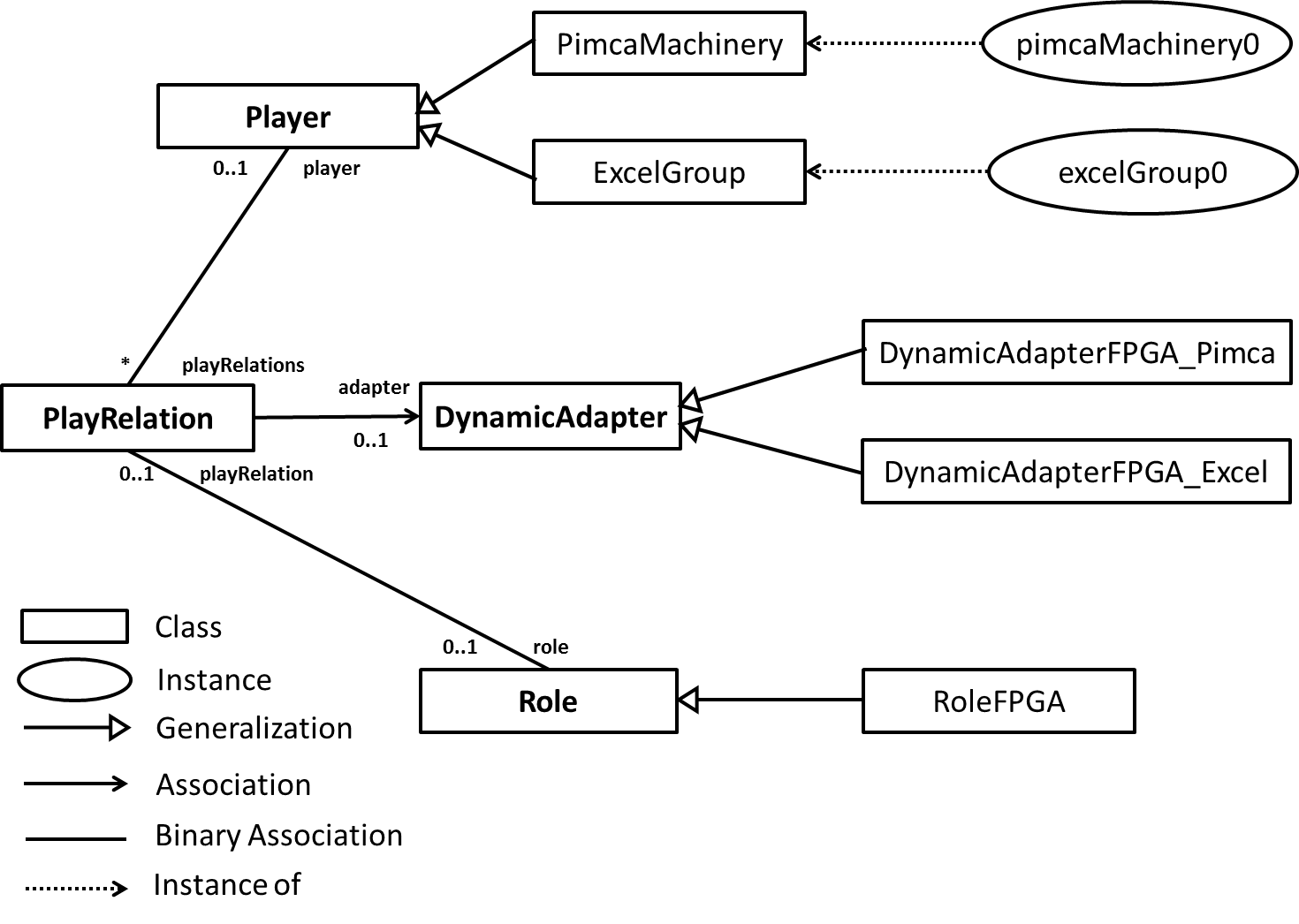


Figure 4 : Role4All’s meta-model illustrates with the example of the role of FPGA

According to the 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, the terrorist created a unique point of view on two different elements of two different tools. Now the terrorist can used the role *RoleFPGA* to manipulate the concept of FPGA instead of the couple of tools Excel and Pimca.

# Synchronization Based on Roles

## The Necessity of Synchronization

The previous part shows how to create a point of view of different elements form various tools. A point of view allows working in a specific environment on data provided by different tools (like Pimca and Excel in our example) and sometimes the same information are include in different tools. For example in our example the Excel file and the Pimca model provide the same information: the ARM’s model.

One data repeated in various tools can create inconsistencies, two different values for the same data. Inconsistencies are a major problem because to solve them the user needs to compare all the occurrences of the data in each tool and select one value. To avoid the inconsistencies the simplest solution is to repeat all the updates in all the tools (the equivalent of the copy and paste). The problem is the user needs to work in all the different tools at the same times, so the point of view provided by Role4All becomes useless, so Role4All and the federation become useless. Therefore it is necessary to automate the “copy and paste” this is called synchronization.

Different types of synchronization exist, the simplest one is a check-out/check-in synchronization without security. After a save all changes are immediately reflected in all the synchronized tools. Due to this synchronization the user can use only one tool (Role4All, Pimca, etc.) and synchronize all the other. With a synchronization without security system, a mistake (delete an element, etc.) is directly impacted on the synchronized elements. Therefore it is necessary to implement some security measures. By default Role4All provides two security systems: a temporary save and a specification system. The temporary save exist just during the development phase, this system is a simple backup system. For the production phase Role4All will allow a Git system to secure the synchronizations and easiness the team work. The specification system allows to the user to create synchronization rules. This rules aim to hierarchize tools compared to a synchronization. As an example the terrorist can 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 user uses Excel to update an Excel element synchronized with a Pimca model element through a role (roleFPGA0).

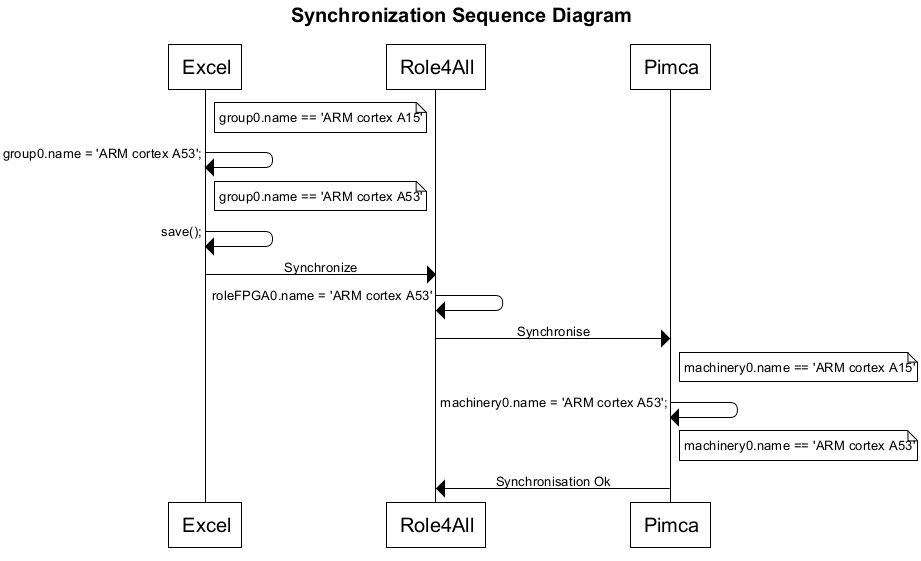


Figure 5 : A sequence diagram of a synchronization using Role4All

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 last example) 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 user works with this point of view instead of tools (Excel and Pimca in our example).

## Role4All or how dynamic federation implies dynamic synchronization

In addition to the federation Role4All allows tools synchronization through the concept of role. With Role4All all model elements play a role therefore all instances are strongly linked with a role instance, this relation between role and player was explained in the part 3. As an example (figure 4) the Pimca model element *pimcaMachinery0* plays the role *RoleFPGA* therefore the instance *pimcaMachinery0* is linked with an instance of *RoleFPGA* call r*oleFPGA0*. So in Role4All it has a bidirectional link between model instance and role instance consequently to synchronize model instances is equivalent to synchronize role instances. In other words to synchronize tools we need to synchronize role instances.

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 was connected the role instances, the instance *roleFPGA0* contain 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* known in which instances they are contained through the variable *containerRoles*. So we are 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, this allow 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 depends of the context (Network management, Cybersecurity, etc.) therefore Role4All allows to personalize each synchronization. A user can define various rules and checks when he creates synchronization between tools with Role4All, as examples 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.

# Simulation through Role4All

## 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).

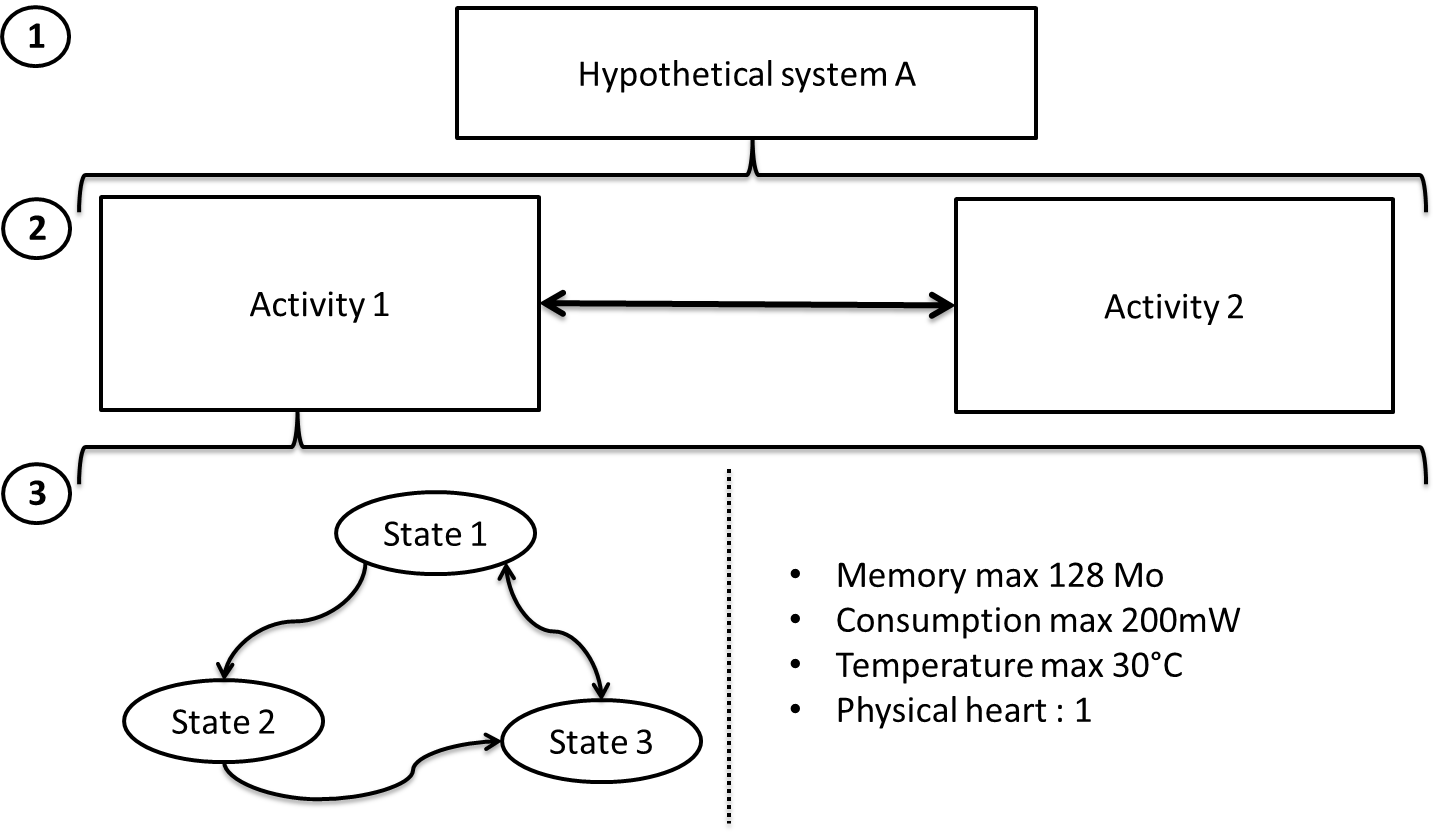


Figure 7 : Morphose design

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.

## Application in in a Cybersecurity Context

Part 2 describes a situation which a cyberterrorist wants infected a system therefore he collect data about the system and model these data with two tools, Pimca and Excel. Role4All allows him to create specific points of view on the system and to synchronize his tools due to the concept of role. Until now the specific points of view allows considering and using data independently of a tool but it is possible to create a point of view specialize for a tool, for example a simulation tool like Morphose. So Role4All formats the data from various tools to generate a simulation (figure 7). Consequently the terrorist can simulate his hypothetic systems.

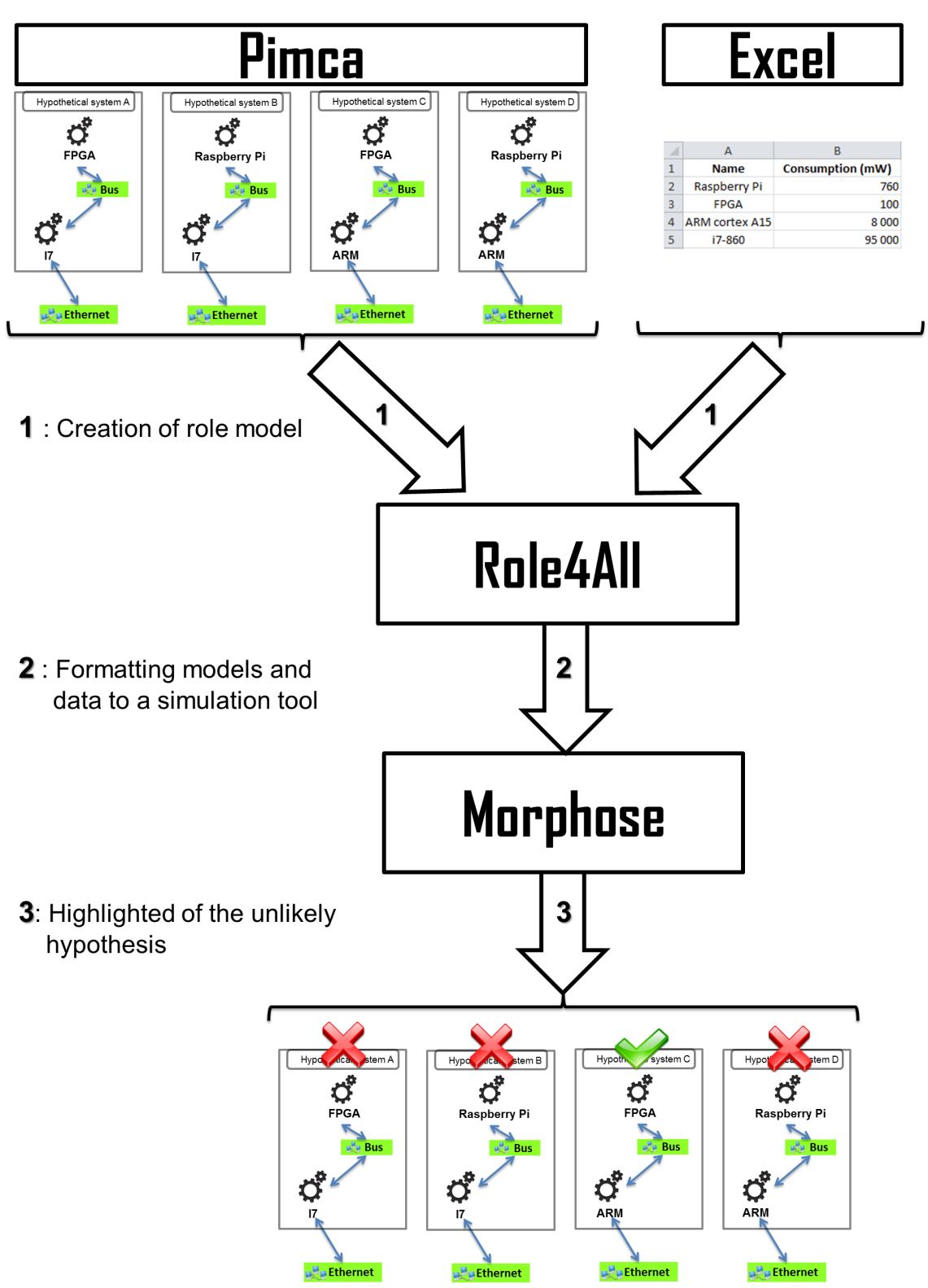


Figure 8 : Simulation in Role4All and Morphose

The figure 7 describes how the data collected and formatted through various tools are simulated on Morphose according to an example. In our example the cyberterrorist creates 4 hypothetical systems in Pimca composed of two elements: a platform (FPGA or Raspberry Pi) and a processor (I7 or ARM). Furthermore the terrorist has heterogeneous data like the consumption of the real system or an Excel file with the consumption of each platforms and processors. He used Role4All to create a point of view on his hypothetical models and data allowing implementing a simulation, so the 4 hypothetical systems (Pimca model + Excel data) are simulated. Morphose returns information about the simulation like the execution time or the consumption of the system. In our example the terrorist can compare the consumption of the hypothetical systems and the consumption of the real system to highlight the unlikely hypotheses. In our example the terrorist can select the hypothesis C (FPGA +ARM) like the only plausible system because Morphose rejects the other ones.

Finally the simulations allow testing many hypotheses according to heterogeneous data and models from various tools. Furthermore the dynamism due to Role4All allows to easily interpret the data from Morphose and to generate new hypothesis in order to better uncover the real system.

# Conclusion

Point de vue

Synchro

Symulation, detection d’information sensible, découveerte de system, prédiction