Model Driven Software Security Architecture of Systems-of-Systems

Jamal El Hachem, Zi Pang, Vanea Chiprianov, Ali Babar, Philippe Aniorte

Model Driven Software Security Architecture of Systems-of-Systems

Jamal EL HACHEM*, Zi Yang PANG†, Vanea CHIPRIANOV*, Ali BABAR†, Philippe ANIORTE*
*UNIV PAU & PAYS ADOUR, LIUPPA, PAU, FRANCE
{jamal.elhachem, vanea.chiprianov, philippe.aniorte}@univ-pau.fr
†University of Adelaide, Australia
a1681939@student.adelaide.edu.au, ali.babar@adelaide.edu.au

***Abstract*—Recently, there is a growing interest in Systems of Systems (SoS), their architecture, security and application domains. However, their specific characteristics such as the operational independence of SoS constituent systems (CS), the absence of central authority and their emergent behavior make the modeling of their structure, behavior and security a complex task. One of the current main security challenges in the context of SoS is the cascading attack problem. The challenge is to predict the concatenation/sequence of CS's vulnerabilities that could be triggered resulting in destructive cascading failures and take corrective actions to reduce the cost, development time and effect of later changes. In this paper, we propose a domain specific modeling language (DSML) to represent SoS security architecture. Having SoS security models will enable the discovery, analysis and resolution of cascading attacks, in the architecture phase, preventing development time and cost wastage. Following a Model Driven Engineering (MDE) approach, we generate a graphical editor for our DSML and use it to model a Smart Campus case study.**

***Keywords*—Model Driven Engineering, Model-based Software Engineering, Modeling Language, Software Architecture, Systems-of-Systems Security, Smart Cities, Smart Campus.**

I. Introduction and Background

We live in a hyper-connected world where software systems become more and more linked. There are estimates that by 2020 thirty one billion intelligent devices will be connected and world wide market for Systems of Systems (SoS) will surpass 6.5 billion euros [1].

SoS is a promising research field gaining increasing importance. Although there have been several attempts to define SoS, there is yet no standard definition [2]. However, one of the popular definitions, which we also use in this paper, is that of Jamshidi [3]: "SoS are large-scale, distributed, concurrent systems comprised of complex systems". These complex systems are systems themselves and in our work we choose to call them Constituent Systems (CS). Many other researchers identified SoS by their distinguished characteristics, particularly Mair specifies the following five essential characteristics: operational independence of the elements, managerial independence of the elements, evolutionary development, emergent behavior and geographic distribution [4]. Several other concepts could describe SoS such as: global mission, belonging, autonomy, connectivity, diversity [5].

These characteristics impact SoS non functional properties as well, specially security, on which we will focus in this paper. Security has been identified as a critical SoS research theme [6]. In fact, SoS suffers from complex systems security problems, as well as additional problems raising from their specific characteristics as shown in [7] [8].

One of the crucial SoS-specific security problems is the *cascading attacks*. An *attack*, as defined by ISO IEC 27000 [9], "is any unauthorized attempt to access, use, alter, expose, steal, disable, or destroy an asset". Considering that:

1) It is rare that an attacker exploits a single vulnerability on a single target system to achieve its objective, in most cases he uses several single attacks on several systems to accomplish his attack [10].
2) "The consequences of attacks on the SoS cannot be understood by means of the merely evaluation of the behavior of the single systems, but require an assessment of the effect of the inter-dependencies on the behavior of the whole SoS" [11].

We define a *cascading attack* as a succession of possible single attacks or a sequence of exploited vulnerabilities resulting from the CS inter-dependencies employed for the achievement of the SoS global goal.

These cascading vulnerabilities may serve, intentionally or unintentionally, to compose a cascading attack that affects the whole SoS and cause huge important damages. The stuxnet attack [12] is one of the most well-known cascading attack. It was based on a 500 kilobyte computer warm that attacked in three phases: first it targeted Microsoft Windows machines and networks, repeatedly replicating itself, finally it compromised the programmable logic controllers. Starting from a system via a USB stick, stuxnet infected 14 industrial sites in Iran, including an Uranium-enrichment plant. Recently, *FireEye* researchers have discovered a complex malware which uses some elements from stuxnet[1].

However, addressing this problem involves studying how to model SoS structure and behavior, focusing on their security aspects. Furthermore, knowing that early discovered vulnerabilities are less expensive to fix than those discovered later [13], how could security vulnerabilities be modeled within SoS software architecture to allow the prediction, at an early stage, of their concatenation/sequence which could

[1]http://www.silicon.fr/irongate-un-malware-aux-airs-de-stuxnet-cible-les-scada-149299.html; accessed on 21st June 2016

be triggered in an unknown way and result in a cascading attack?

In this paper, **we propose a Domain Specific Modeling Language (DSML) called SoSSec: Systems-of-Systems Security, with its corresponding graphical editor**. We adopt the Model Driven Engineering (MDE) approach to define our DSML which will be used to model SoS security architectures.

MDE is considered as an effective approach to address significant challenges in SoS architectures modeling [14]. In MDE, models can be expressed using different languages like General Purpose Modeling Languages (GPML) or DSML. The main advantage of a DSML is its ability to simplify the modeling of complex systems by offering expressive domain specific abstractions and notations focusing on a particular problem domain in a precise and concise way. The use of a DSML to model non-functional properties, like security, lead to better solutions [15]. In our case, the DSML will be dedicated for SoS cascading security attacks modeling. However, since it is very costly to define a new language from scratch, our DSML is defined as an extension of the **Sys**tem **M**odeling **L**anguage (SysML). SysML is a general-purpose graphical modeling language for specifying, analyzing, designing, and verifying complex systems. One of the main advantages of adopting an extension mechanism, is that the reference language is refined in a strictly additive manner preventing the contradiction of the standard semantics.

Generally, the definition of a DSML consists of three essential components [16]:
- Abstract syntax: set of concepts and relations that serve to express a model. It is usually described using a MetaModel;
- Concrete syntax: set of symbols to represent the MetaModel;
- Semantics: to determine the meaning of the models, usually expressed by simulation or code generation.

Once the concrete syntax of the DSML is defined, MDE allow the (semi)automatic generation of its tools, such as the graphical editor modeling tool and the simulation tool.

After defining the SoSSec MetaModel (abstract syntax) and developing its graphical editor (concrete syntax), we use it to model a Smart Campus SoS case study.

The remainder of this paper is organized as follow: Section 2 briefly introduces the MDE approach for DSML definition. In Section 3 we discuss related work. Section 4 presents our SoSSec DSML and its tools. Smart Campus SoS security models are presented in Section 5 while Section 6 concludes the paper.

II. State of the Art

Historically, many approaches were used to model and analyze software architectures of a system, like: formal approaches which incorporate mathematical analysis in the software architecture models; goal oriented approaches which focus on goals to capture the functional and non functional system requirements in the software architecture models; and Architectural Description Languages (ADLs) which are languages used to describe system software architecture, like its components, connectors, rules and guidelines.

In trying to choose between these broad types of approaches, we focused on analyzing their suitability for modeling SoS specific characteristics, as identified by Mair [4].

Some formal methods (e.g. graph based-methods: Bayesian Networks (BN), Functional Dependency Network Analysis (FDNA) seem well suited to model the geographic distribution and interactions between CS but they are still limited compared to ADLs regarding the representation of the structural architecture (like components; interfaces; contracts and ability to describe hierarchical levels of details).

Goal oriented approaches such as Multi-Agent Systems (MAS) could be used to model SoS global goal and CSs individual goals, as well as behavioral aspects like the emergence and the interaction between different CSs. However, like in the case of formal methods, goal oriented approaches are not the best suited for describing SoS structural architecture. Besides, seeing that another crucial aspect in the context of our work is the ability to model security mechanisms, properties and concepts like vulnerabilities, pre/post conditions and attacks, ADLs security extensions presents more useful modeling concepts than goal oriented approaches.

For these reasons, we choose to build our DSML upon an existing ADL to model SoS taking into account their characteristics and security properties. However additional concepts inspired from goal oriented approaches will be used to describe some SoS concepts like global and individual goals. Furthermore, once the SoS security models are designed, formal methods-inspired approaches could be used to discover possible paths for cascading attacks and analyzing and proposing solutions to tackle them.

A. System of Systems Modeling Approaches

After restraining our state of the art to ADL for modeling SoS structure, behavior and security, we selected the latest and most used approaches adopted in recent International and European SoS projects like COMPASS [2], DANSE [3], Road2SoS [4]. Following we present the analyzed ADLs:

An extension of the Architecture Analysis and Design Language (AADL) was proposed in a recent work [17] to specify and model cyber-physical systems based on SoS approach.

In [18] the authors propose an extension of Acme to model the architecture of cloud applications by introducing some concepts like relationships between elements, contracts, etc. The proposed ADL remains a simple specific language for cloud SoS service providers and applications. Previously mentioned ADLs lack concepts to model SoS taking into account their special characteristics, and remain specific to particular domains.

Lately, in the COMPASS European project, SysML was used to model SoS CSs, their interfaces and contracts that oversee operations between them. Along with SysML, Compass Modeling Language (CML) was proposed for contract

[2]http://www.compass-research.eu/; accessed on 21st June 2016
[3]http://danse-ip.eu/home/; accessed on 21st June 2016
[4]http://road2sos-project.eu/; accessed on 21st June 2016

TABLE I
ADL COMPARISON FOR SYSTEMS SECURITY MODELING

	UMLSec	**Secure UML**	**Secure i***	**MODELO**	**SysML -Sec**
Security concerns	**Integrity Authenticity Access-Control Confidentiality Non-Repudiation Non-Interference**	Access Control	Access Control	Access Control	Authenticity Confidentiality Integrity Autonomy
Modeling Language	UML	UML	UML	UML	**SysML**
Model Transformations	Model to Text	Model to Text	?	**Model to Model**	Model to Text
Threat-Vulnerability concepts	-	-	**+**	-	-
Goal concept	-	+	+	+- (task)	-
Context concept	-	-	-	**+**	-
Attack concept	-	-	-	-	**+**
Security prediction	-	-	**+- (goal satisfaction)**	-	-

verification at interfaces [19]. The proposed approach does not take into account yet security features in the contracts.

Some recent studies on ADL for software intensive SoS compare UML, SysML, SysML+CML and X-UNITY regarding a number of features to exhibit their ability to model SoS [20] and propose a conceptual model for missions of SoS [21]. Results show that SysML seems to be the best suited approach for SoS architecture modeling due to its ability to model CS structure and interactions in terms of interfaces and constraints. This study shows that the existing ADLs that have been used recently for describing SoS architectures miss some of the SoS features like: The representation of CS missions (goals) and evolution, architectural elements that manages the interactions (mediators) among CS, the architecture description of SoS emergent behavior resulting from CS interactions related to SoS mission accomplishment.

We can conclude that none of the existing languages can be used as a standalone approach to model SoS taking into account their specific characteristics as well as security properties and specially the cascading attack problem. Despite that, SysML extended with some concepts inspired from [20][21] like *mission, global mission and individual mission* seems like a strong basis for SoSSec.

B. System Security Modeling Approaches

Generally, security deals with confidentiality, integrity and availability of data [22]. These essential properties, as well as security mechanisms like access control, authorization, encryption need to be described and verified as soon as possible in the development life cycle of any system. This is the more important in the context of SoS as additional security problems may raise. In this work we will focus on the cascading attacks problem. In [7], there have been identified that approaches which handle SoS, architecture and security aspects, all three together are not numerous: [23] discuss a design for evolution to maintain operation regardless of SoS state and [24] describe the importance of designing solutions that consider security without detailing these solutions.

As no ADL have been developed to model SoS security, we studied existing techniques that address system security. To determine their usefulness for SoS, we analyzed them regarding the following criteria which seem essential to meet our objectives: model and discover security cascading attacks in the context of the SoS (table I). Indeed, to model security, we prefer to remain general to cover many *security concerns* (like integrity, confidentiality and access control), so we choose this aspect as a criteria; while we select some additional criteria, like *attack, threat, vulnerability* and *context*, specific to the SoS cascading attack security problem that we treat in our work. Another important criteria is the *goal concept* because it is necessary to model SoS aspects like global goal, individual goals and security goals. Furthermore, since we are following the MDE approach, the *modeling language* upon which they are based and their ability to support *model transformations* are important criteria. An extra criteria, which is not in the thick of the current work, is *security prediction*. This will be useful for future work in order to predict cascading attack paths.

UMLSec is a UML extension proposed by Jurjens [25] to express security requirements like data secrecy and integrity, information flows restrictions and role based access control.

SecureUML is an approach based on role based access control with additional support for specifying authorization constraints. It was proposed to increase the productivity and quality of secure distributed systems [26].

Secure i* is a goal oriented approach used to model and analyze security trade-offs among competing goals of multiple actor systems and quantitatively assess security mechanism's impact on actor's goals and threats [27].

MoDELO is a MOdel-Driven sEcurity poLicy approach that extends UMLSec with OrBAC elements for early modeling and evaluation of access control requirements [28].

SysML-Sec [29] is a SysML based MDE environment inspired by KAOS. SysML-Sec process includes three stages: system analysis (identification of security requirements and threats by system's partitioning), system design (implementation of software security mechanisms) and system validation (formal verification, simulation and test of the models).

The analysis of the listed approaches is presented in table I. Plus and minus signs means that the evoked criteria are,

respectively, taken or not into consideration by the analyzed security modeling languages. This analysis shows that none of the studied ADL satisfy all the investigated criteria. Hence concepts from each approach should be used by any future ADL to be proposed (like the concepts marked in red in table I. In addition, other concepts need to be proposed to represent the cascading attack, and to explicitly model threats and vulnerabilities.

III. Proposal: SoS Security DSML

After identifying the limitations of the existing languages and security extensions in modeling SoS when taking into account their characteristics and security, specially attacks, we now detail the abstract and concrete syntax of our SoSSec language.

As mentioned before, to represent SoS structure and behavior, we extend the SysML MetaModel by modifying the Block Definition Diagram and proposing two new diagrams: security and goal diagrams. For the security diagram, we p artially leverage existing security modeling language semantics such as SysML-Sec, Secure i*, MoDELO and UMLSec by adding new concepts and relations to represent vulnerabilities and cascading attacks. As for the goal diagram, we propose SoS specific concepts, with additional concepts inspired from [20][21] like *global goal and individual goal*.

Extending SysML and introducing new diagrams means that we have one unifying MetaModel and several views on it (defined as diagrams). Having a unified MetaModel has the advantage of not having to deal with composition of models. If we defined a family of DSMLs, each with their own MetaModel, each corresponding to a view or a purpose, the models created with these DSMLs would need to be composed/combined in order to obtain a complete overview. This would result in important composition challenges [30] to address, while from the user/modeler point of view, both approaches result in similar functionalities of several views.

A. SoSSec Abstract and Concrete Syntaxes

SoSSec MetaModel is an extension of SysML MetaModel, indeed, we modified Block definition diagram (Fig. 1) to model SoS, their interfaces and operations on these interfaces. Additionally, we propose two new diagrams: Goal Diagram (Fig. 2) to model SoS global goal and CS's individual goals as well as necessary refinement relationships between these goals; and Security Diagram (Fig. 3) to illustrate the cascading attack security problem.

1) Block Definition Diagram: In SoSSec MetaModel, we use *Block* to represent SoS structure (Fig. 1), since *Block* could be used to describe a system or other elements of interest (hardware, software, data, procedure, facility, person). Theoretically, an SoS could be refined into one or many other SoS, e.g. a smart city SoS may encompass a smart grid/building/transportation SoS. The latter idea could be modeled using the composition relationship *block*. Another aspect that we introduce to SysML MetaModel is *GlobalGoal*

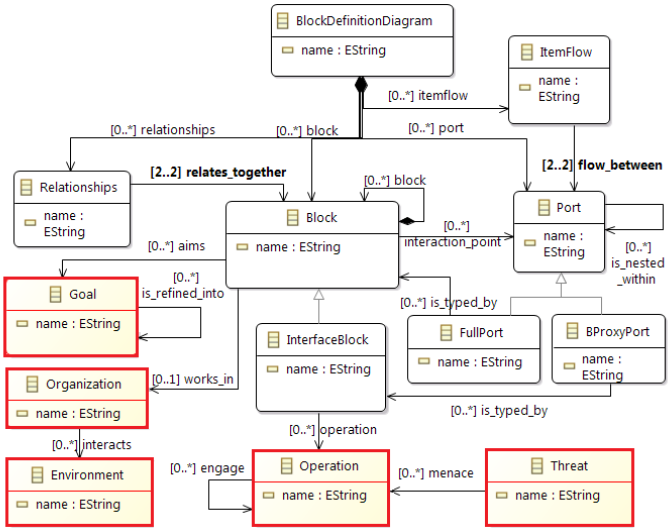

Fig. 1. SoSSec MetaModel Block Diagram.

to explicitly represent SoS aim(s). *GlobalGoal*(s) could be *refined_into* one or many individual goal achieved by its CSs.

Further, each SoS is composed of many CSs which are usually unstable because of the evolution characteristic. Collaborations between these CS are unknown due to the emergent behavior (e.g. the operations related to the achievement of global goals). During CS's modeling, it is important to well describe the interfaces because CS regular and emergent functionalities are represented at the interfaces, as well as the interactions between CSs and security aspects related to these functionalities and interactions. Mair evoked that "The greatest leverage in system architecture is at the interfaces. The greatest dangers are also at the interfaces. There is nothing else to architect" [4].

For these reasons, we model CS as "black-boxes" using *InterfaceBlock*. These CSs participate in the realization of SoS objective(s) through individual goals and *ItemFlow* (data, energy, material) exchange. Every CS has interaction points represented by *Port* which serve to establish connections with other CS to accomplish one or more *Operation*(s). To better model *Operation*(s) on interfaces due to their importance in the description of CS functionalities, we modify the latter SysML concept by making it a <graphical node>. As for the SysML *Port* concept, we adopt it without modifications.

Even if CSs collaborate within the SoS to accomplish its *GlobalGoal*(s), they remain managerially and operationally independent. To model these aspects, we introduced *Organization* concept to SysML block diagram. It serves mainly to tie a CS with the organization in which it works. The *Organization* will be responsible of the functional as well as non functional goals of the CS under its governance, specially security goals. Moreover, *Threat* concept was also introduced to SysML *BlockDefinitionDiagram* to make the link between with *SecurityDiagram*.

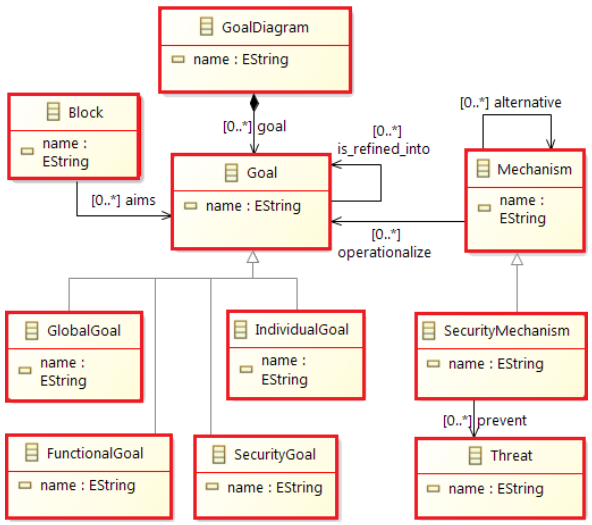

Fig. 2. SoSSec MetaModel Goal Diagram.

2) Goal Diagram: As mentioned earlier, the CSs collaborate in order to accomplish a *GlobalGoal* assigned to the *SoS* [4]. This *GlobalGoal is_refined_into IndividualGoal*(s) *accomplished* by different CS [21]. The *GoalDiagram* (Fig. 2) is a new diagram we introduce to model the *Goal* aspects of *SoS*. *Goal* and *Block* are duplicated in both *BlockDefinition* and *Goal* diagrams to make the link between them.

To clearly model SoS non-functional properties, precisely security, we refined *Goal* into two sub-categories: *FunctionalGoal* and non functional goals precisely *SecurityGoal* which describe respectively usability SoS and/or CS goals (what a system is supposed to accomplish) and quality goals which influence the operations of the SoS e.g. security (how a system is supposed to be).

Another aspect that seems essential to model the latter quality goals is the concept of *Mechanism* used to *operationalize* a *Goal*. To benefit from this concept, in the context of SoS security architecture modeling, we instantiate it into *SecurityMechanism*. In order to *operationalize* (satisfy) security *NunFunctionalGoal*, most approaches use one or more security mechanisms, e.g. access control and encryption. These *SecurityMechanism* help to detect, *prevent* and/or recover from an attack/*Threat*. The *SecurityMechanism* and *Threat* concepts are duplicated in *Goal* and *Security Diagrams* to make the link between them.

3) Security Diagram: In our work we choose to describe the *cascading attack* problem which is a crucial security problem with destructive consequences in the context of SoS as detailed in section I. Indeed, the collaboration between independent CS for the achievement of SoS global goal(s) results in emergent behaviors. In this context, small, known, unresolved and insignificant *Vulnerabilities* in each CS could be exploited and connected in an unknown way, to form a *cascading attack*. It is important to mention that in this work we address the *unknown known vulnerabilities* category. Indeed, Rumsfeld [31] classify the knowledge in three categories: *known knowns, known unknowns, unknown unknowns*. Sawyer [32] added a complementary category, on which we will focus: the *unknown knowns*. He defined it as "knowledge that a person holds, but which they withhold". These categories could be projected onto vulnerability categories. In the context of SoS, each CS may define a list of its own vulnerabilities (*known*). Since there is no central authority in the SoS, this list of vulnerabilities will remain *unknown* by other CS, hence the nomination *unknown known vulnerabilities*.

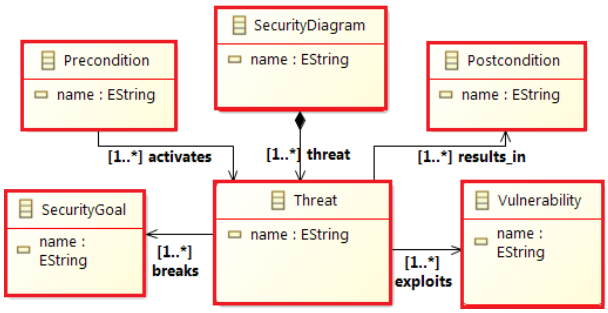

Fig. 3. SoSSec MetaModel Security Diagram.

To be able to model this *cascading attack* security problem, we propose the *SecurityDiagram* given in (Fig. 3), to extend SysML with security concepts. These concepts, will serve to discover, analyze and evaluate *cascading attacks* early at modeling level of the SoS development life cycle.

Each CS may have one or more *Threat*(s) or malicious goal(s). It represents a possible risk that might be *exploited* by one or many *vulnerabilities* to *break* a *SecurityGoal*. A security *vulnerability* is any weakness in or back door to a system [9]. Examples of most common vulnerabilities in many computer systems are "buffer overflow" and "password cracking" [33]. One or several *vulnerabilities exploit* one or several *Threat*(s). A *Threat is activated* by one or many *Precondition*(s) and *activates* one or many *Postcondition*(s) causing damages and loss in an SoS and could lead to an important cascading attack. *Precondition*(s) [21] are condition(s) which activates a threat. For example, a pre-condition of the "buffer overflow" vulnerability could be "execution stack by overwriting the stored return address, the stack pointer or the frame pointer". Whereas, *Postcondition*(s) [10] represent the results of an activated threat. For example, the results of a "buffer overflow: could be an "unauthorized access to system". The effect could directly affect the concerned CS or could lead to a cascading attack that influence the whole SoS.

After defining SoSSec's abstract syntax, we adopt existing visual notations to represent it. Indeed, defining a DSML undertakes corresponding one or many concrete syntax(es) to its abstract syntax. The concrete syntax is a set of textual or visual notations that facilitates the language presentation and construction. As SoSSec MetaModel is a SysML extension, we adopt existing related visual notations (e.g. nodes and edges) to represent model elements.

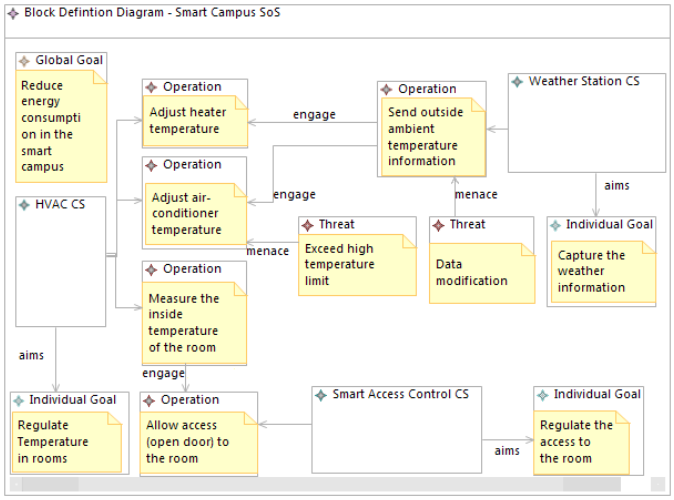

Fig. 4. Smart Campus Model - Block Diagram

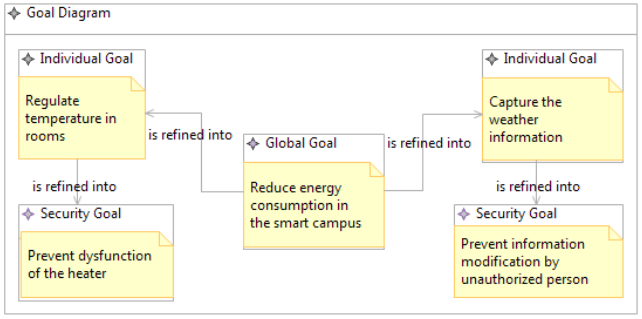

Fig. 5. Smart Campus Model - Goal Diagram

B. SoSSec Graphical Editor Development

SoSSec abstract syntax (MetaModel) is specified using the Eclipse Modeling Framework (EMF) and its concrete syntax (graphical editor) is (semi)automatically generated using model transformations through the Graphical Modeling Framework (GMF). We choose EMF/GMF tooling because they are open source, recently used by the community, well documented relatively easy to use and support MDE. First, we specified our abstract syntax using Ecore, which is the MetaModel of the EMF [5]. Then, the Ecore model was used within the GMF to generate (parts of) the graphical editor. In fact, GMF provides the fundamental infrastructure and components for developing visual models and modeling surfaces in Eclipse. Having SoSSec MetaModel (Ecore model), GMF allows the definition of diagrams based on a mapping between the abstract and the concrete syntaxes and the (semi)automatic generation of the corresponding graphical editor (visual tool). This graphical editor corresponding to our SoSSec DSML (Fig. 6) could now be used by SoS security architects to model their SoS taking into account its specific characteristics and security aspect related to the cascading attack security problem.

C. Discussion

Having SoSSec DSML will allow to an SoS (security) architect to discover *unknown known* sequence of CS's *vulnerabilities* that could be triggered resulting in destructive *cascading attacks*. An architect will be able to model, using SoSSec graphical editor, the structure of an SoS taking into account its specific characteristics (like CSs, Interfaces, global and individual goals) as well as SoS behavioral aspects (like interactions, operations and exchange of data.). In addition, an architect will be able to describe a list of *vulnerabilities* for each CS, including the *pre-condition*(s) that trigger each vulnerability and the resulting damages (*post-condition*(s)).

[5]http://www.eclipse.org/modeling/emf/

Compared to existing ADLs, SoSSec covers some missing SoS features such as the representation of SoS global goal and CS sub-goals, the interactions/operations among CS, as well as non functional properties, specially the description of the cascading attack specific SoS security problem resulting from CS interactions related to SoS goals accomplishment.

Once the SoS Security model is designed, the architect could simulate it through a simulation approach like Multi-Agent Systems (MAS) (we are currently working on this semantic part of SoSSec DSML). The simulation will allow the discovery, at the architecture level of the SoS development life cycle, of possible *unknown* sequences of *known vulnerabilities*. Thus, *cascading attacks* could be discovered and analyzed at an early stage and corrective actions could be applied to prevent the cost, development time and effect of later changes.

IV. Smart Campus Case Study

SoS have many application domains, such as defense, cyberspace, health-care and electrical power grids. A more recent SoS domain with growing popularity is Smart Cities. According to [34], 80% of the world's population will be living in cities by 2020. There are estimates that smart cities market will attain 14 Bn EUR addressable revenue by 2020 and 400 Bn Dollars per year will be dedicated for smart urban systems [35].

Design and implementation of smart cities initiatives are complex and challenging tasks. Thus, it is more reasonable to design and implement solutions on a smaller but realistic scale [34]. Hence, to demonstrate our proposition, we chose to model a small-scale view of a smart city SoS: the smart campus. Indeed, smart campus have similar operationnaly and managerially independent CS such as smart lighting system and similar global goals that implies the intervention of many CSs to be achieved such as an efficient energy management and security.

Therefore, a smart campus could be analyzed as an SoS with the following CSs: smart lighting system; air-conditioner and heaters; temperature and humidity sensors and control systems; smart energy system; spatial awareness sensors and data analysis system; monitoring cameras, alarm, video surveillance and security control; internet network; smart vehicles, bicycles, parking These CS need to collaborate in order to achieve some global goals, such as: monitor the building use (energy

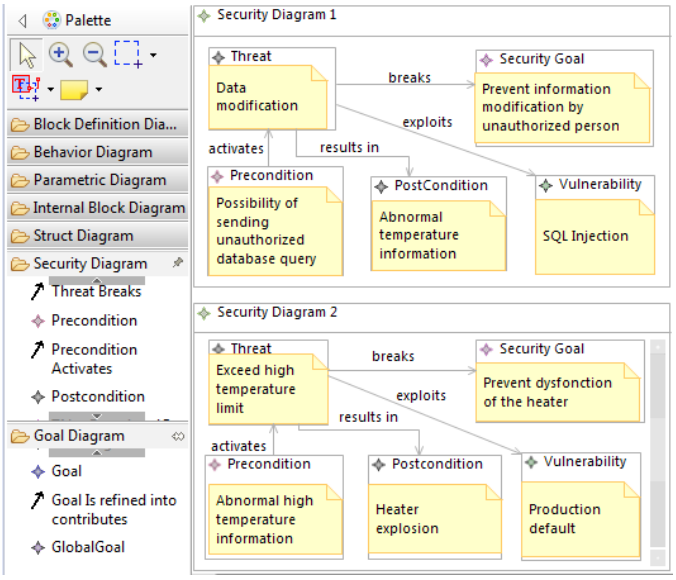

Fig. 6. Smart Campus Model - Weather Station and HVAC CSs Security Diagrams

management); evolution (new components, materials and design principles); security engagement (work place accessibility control).

Using SoSSec and its graphical editor, we have modeled a simple smart campus SoS. As we can see in the Block Definition Diagram of Fig. 4, the Smart Campus SoS is represented by a block. Three CSs were assigned to this SoS and represented as interface blocks: *Heating, Ventilation and Air-conditioning (HVAC) CS*, a *Weather Station* with sensors for weather-related information and *Smart Access Control CS*. These CSs are geographically dispersed, belong to different organizations (managerial interdependence) and each CS has its own goals represented by individual goals and operations to achieve these goals (operational independence). HVAC CS *engages* the following operations: *Adjust heater temperature*, *Adjust air-conditioner temperature* and *measure the inside temperature of a room*; The weather station CS *sends outside ambient temperature information* and smart access control CS prevent or *allow the access to the room*. The global goal of this SoS is to *reduce the energy consumption in the smart campus*.

The modeling of the latter global goal are presented in Fig. 5: Smart Campus global goal is refined into individual goals to be realized by CS: HVAC CS individual goal is to *regulate the temperature in rooms*, Weather Station CS has the individual goal is *to capture weather information* and the individual goal of Smart Access Control CS is to *regulate the access to the room*. An operation could be *menaced* by a threat and an individual goal could be *refined into* a security goal to make the link with the security diagram (Fig. 6).

In order to give a small example of a cascading attack problem within the described smart campus, we associate two threats: "exceed high temperature limit" and "data modification" respectively to the HVAC CS and Weather Station CS. Each threat is described with a security diagram (Fig. 5) showing:

1) The corresponding vulnerability: "data modification" threat could be exploited by "SQL injection" or "weak authentication" vulnerability" and "exceed high temperature threat" could be exploited by HVAC CS "production default" vulnerability (Fig. 6);
2) Security goal(s) broken by the threat are also represented in the security diagram:
"data modification threat" break "Prevent Information modification by unauthorized person" individual goal and "exceed high temperature limit" threat break "prevent dysfunction of the heater" individual goal (Fig. 6);
3) The pre-condition which activates the threat and the post-condition which results from the activated threat: "data modification" threat is activated by the pre-condition "possibility of sending unauthorized database query" and results-in "abnormal temperature information" post-condition (Fig. 6) and "exceed high temperature limit" threat is activated by the pre-condition "abnormal high temperature information" and results-in "heater explosion" post-condition causing fire in the HVAC CS.

Having these models, we can imagine the following emerging cascading security problem resulting from the collaboration between CS to achieve the smart campus global goal. Normal Scenario:

1) Weather station CS send ambient temperature information to HVAC CS
2) According to the received information, HVAC CS adjusts its heater, air-conditioner temperatures
3) HVAC CS measure inside temperature room and send it to the Smart Access Control CS
4) The latter analyze the information in order to allow/prevent access to the room

Cascading attack scenario:

1) Ambient temperature information is decreased by an attacker taking advantage of the temperature monitoring CS's vulnerability: SQL injection or weak authentication
2) HVAC CS acts upon the received erroneous information by increasing heater's temperature. This action exploits another small vulnerability related to HVAC CS which is "exceed high temperature limit"
3) The latter vulnerability lead to heater explosion causing a fire in the room
4) Thus the Smart Access Control System will permit doors opening so staff could leave the room (and maybe any non-allowed person will enter the room at this moment.)
We illustrated a small example of a cascading attack security problem in smart campus case study. In the same way, more complex scenarios could be discovered having the security models of our SoS.

V. Conclusion

System of Systems (SoS) architecture is a promising research field gaining increasing importance. SoS are comprised

of complex distributed constituent systems (CS) which need to collaborate to achieve their own local goals as well as global SoS goal(s). These interactions may cumulate or trigger a sequence of small security vulnerabilities coming from different CS. These cascading vulnerabilities may serve, intentionally or unintentionally, to compose a cascading attack that affects the whole SoS and causes huge important damages.

In this work, we focused on this security problem. We introduced a DSML, a SysML extension, to model SoS security early at the architecture level of the development life cycle to reduce the cost, time and effect of later modifications. We followed a Model Driven Engineering approach to define the SoSSec parts (concrete syntax, abstract syntax). This paper describes the details of the abstract syntax of our language - the MetaModel, and its concrete syntax with the corresponding graphical editor. Our SoSSec DSML has been used to model a motivating smart campus case study and has illustrated a possible cascading attack scenario. In the future, we plan to develop the semantics of our language in order to simulate, analyze and quantitatively predict security of an SoS architecture. We also aim to extend a well known graphical editor in the enterprise and research communities such as Papyrus to have a complete and easy extensible tool compatible with existing SysML models.

REFERENCES

[1] A. Pétrissans, S. Krawczyk, G. Cattaneo, N. Feeney, L. Veronesi, and C. Meunier, "Towards SoS trends and challenges," IDC, Tech. Rep., 2012.

[2] R. Jaradat and C. Keating, "A histogram analysis for SoS," *Inderscience*, 2014.

[3] M. Jamshidi, "SoS innovations for 21st century," in *ICIIS, IEEE international Conference on Industrial and Information Systems*, 2008.

[4] M. Maier, "Architecting principles for SoS," *Systems Engineering*, 1998.

[5] M. Gonçalves, E. Cavalcante, T. Batista, F. Oquendo, and E. Nakagawa, "Towards a conceptual model for software-intensive SoS," *IEEE International Conference on Systems, San Diego, CA*, 2014.

[6] H. Dogan, C. Ncube, S. Lim, M. Henshaw, C. Siemieniuch, M. Sinclair, V. Barot, S. Henson, M. Jamshidi, and D. DeLaurentis, "Economic and societal significance of the sos research agenda," *In IEEE International Conference on Systems*, 2013.

[7] J. E. Hachem, "Towards Model Driven Architecture and analysis of SoS access control," *International conference on software Engineering Doctoral Symposium, Florance, Italy*, 2015.

[8] V. Chiprianov, L. Gallon, M. Munier, P. Aniorte, and V. Lalanne, "Challenges in security engineering of SoS," *Conférence de l'Ingénierie Logiciel, Paris, France*, 2014.

[9] *Information technology – Security techniques – Information security management systems – Overview and vocabulary*, ISO IEC 27000, section 2, Terms and definition; 2014 Std.

[10] L. Gallon and J.-J. Bascou, "Cvss attack graphs," *7th International Conference on Signal-Image Technology and Internet-Based Systems (SITIS), Dijon, France*, 2011.

[11] C. Guariniello and D. DeLaurentis, "Communications, information, and cyber security in SoS: Assessing the impact of attacks through interdependency analysis," *Conference on Systems Engineering Research, California, USA*, 2014.

[12] NERC, "Sql slammer worm lessons learned for consideration by the electricity sector," North American Electric Reliability Council, Tech. Rep., 2003.

[13] A. Austin, C. Holmgreen, and L. Williams, "A comparison of the efficiency and effectiveness of vulnerability discovery techniques," *Information and Software Technology, vol 55, pp. 1279-1288*, 2013.

[14] E. Nakagawa and O. Oquendo, "The state of the art and future perspectives in SoS software architecture," *Proceedings of the International Workshop on Software Engineering for SoS, NY, USA*, 2013.

[15] P. Baker, S. Loh, and F. Weil, "Model-Driven Engineering in a large industrial context - motorola case study," *MDE Languages and Systems, LNCS, vol. 3713, pp. 476?491. Springer*, 2005.

[16] I. Kurtev, J. Bézivin, F. Jouault, and P. Valduriez, "Model-based DSL frameworks," *In Companion to the ACM symposium on Object-oriented programming systems, languages, and applications, NY, USA*, 2006.

[17] L. Zhang, "Applying system of systems engineering approach to build complex cyber physical systems," *Advances in Intelligent Systems and Computing*, 2015.

[18] E. Cavalcante, A. Medeiros, and T. Batista, "Describing cloud applications architectures," *In Proceedings of the 7th European conference on Software Architecture, Montpellier, France*, 2013.

[19] J. Bryans, J. Fitzgerald, R. Payne, A. Myazawa, and K. Kristensen, "SysML contracts for SoS," International SoS Engineering Conference, Stamford Grand, Adelaide, Tech. Rep., 2014.

[20] M. Guessi, E. Cavalcante, and L. Oliveira, "Characterizing architecture description languages for Software-Intensive SoS," *International Workshop on Software Engineering for Systems-of-Systems Conference, Florence, Italy*, 2015.

[21] E. Silva, E. Cavalcante, T. Batista, F. Oquendo, F. Delicato, and P. Pires, "On the characterization of missions of SoS," *European Conference on Software Architecture Workshops, Vienna, Austria*, 2014.

[22] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," in *IEEE Trans. on Dependable and Secure Computing, vol.1, no.1, pp.11-33*, 2004.

[23] M. Duren, H. Aldridge, R. Abercrombie, Sheldon, and T. Frederick, "Designing and operating through compromise: Architectural analysis of CKMS for the advanced metering infrastructure," in *8th Cyber Security and Information Intelligence Research Workshop, Tennessee, USA*, 2013.

[24] M. Merabti, M. Kennedy, and W. Hurst, "Critical infrastructure protection: A 21st century challenge," in *International Conference on Communications and Information Technology , Aqaba, Jordan*, 2011.

[25] J. Jurjens, "UMLsec : extending UML for secure systems development," *Fifth International Conference on The Unified Modeling Language, Model Engineering, Languages Concepts and Tools, London, UK*, 2002.

[26] T. Lodderstedt, D. Basin, and J. Doser, "SecureUML : A UML-Based Modeling Language for Model-Driven Security," *International Conference Unified Modeling Language, Model Engineering, Languages Concepts and Tools, Dresden, Germany*, 2002.

[27] G. Elahi and E. Yu, "A goal oriented approach for modeling and analyzing security trade-offs," *International Conference on Conceptual Modeling, ER, Auckland, New Zealand*, 2007.

[28] D. Arzapalo, L. Gallon, and P. Aniorte, "MoDELO: a MOdel-Driven sEcurity poLicy approach based on Orbac," *8ème conférence sur la Sécurité des Architectures Réseaux et des Systèmes d'Information, Mont de Marsan, France*, 2013.

[29] L. Apvrille and Y. Roudier, "Towards the model-driven engineering of secure yet safe embedded systems," *1st International Workshop on Graphical Models for Security, Grenoble, France*, 2014.

[30] A. Vallecillo, *On the Combination of Domain Specific Modeling Languages*. Springer Berlin Heidelberg, 2010, pp. 305–320.

[31] D. Rumsfeld, *Known and Unknown: a memoir*. Penguin group, 2011.

[32] A. Rashid, S. Nakvi, R. Ramdhany, M. Edwards, R. Chitchyan, and M. Babar, "Discovering unknown known security requirements," *In the 38th International Conference on Software Engineering, New York, USA*, 2016.

[33] R. Anderson, "Security engineering: a guide to building dependable distributed systems," *Wiley*, 2001.

[34] A. Babar, "Smart cities: Socio-technical innovation for empowering citizens," *Australian Quarterly*, 2016.

[35] G. S. M. Association, "Smart cities, developing collaborative mobile-based city solutions for smart cities," Shanghai, Tech. Rep., 2013.