# Using Relax Operators into an MDE Security Requirement Elicitation Process for Systems of Systems

Nicolas Belloir
LIUPPA - University of Pau
BP1155, 640014 Pau, France
nicolas.belloir@univ-pau.fr

Vanea Chiprianov
LIUPPA - University of Pau
BP 201, 40004
Mont-de-Marsan, France
vanea.chiprianov@univ-pau.fr

Manzoor Ahmad
LIUPPA - University of Pau
BP1155, 640014 Pau, France
manzoor.ahmad@univ-pau.fr

Manuel Munier
LIUPPA - University of Pau
BP 201,40004
Mont-de-Marsan, France
manuel.munier@univ-pau.fr

Laurent Gallon
LIUPPA - University of Pau
BP 201,40004
Mont-de-Marsan, France
laurent.gallon@univ-pau.fr

Jean-Michel Bruel
CNRS/IRIT
F-31062 Toulouse University,
France
bruel@irit.fr

## ABSTRACT

Systems of systems (SoS) are large-scale systems composed of complex systems with difficult to elicit and model emergent properties. One of the most significant challenges in the engineering of such systems is how to elicit their non-functional requirements such as security. In this proposal paper we introduce a Model Driven Engineering (MDE) security requirement process for SoS. It is based on the RELAX language to define invariant and relaxed security requirements. This enables taking into account security concerns early in the requirements phase of the SoS. We illustrate our process on a maritime safety and security case study.

## Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: System modeling languages—*Software development methods, security, Requirements analysis*; Software and its engineering [**Software notations and tools**]: System description languages—*System modeling languages*; Security and privacy [**Software and application security**]: [Software security engineering]

## Keywords

Systems of Systems, Requirement Engineering, Security, MDE

## 1. INTRODUCTION

Systems-of-Systems (SoS) are large-scale concurrent and distributed systems, comprised of complex systems [9]. Several definitions of SoS have been advanced, some of them historically reviewed in e.g. [7]. SoS are complex systems themselves, and thus are distributed and characterized by interdependence, independence, cooperation, competition, and adaptation [5]. Characteristics that have been proposed to distinguish between complex but monolithic systems and SoS are [13]: operational independence of the elements, managerial independence of the elements, evolutionary development, emergent behavior, geographic distribution.

The security of such SoS is essential, as the vulnerabilities of one composing system are cascaded into the other systems composing the SoS, resulting in possible debilitating attacks [12]. Most SoS that have been studied are in the defense and national security domain, as a systematic literature review [10] identifies for example.

The characteristics of SoS raise specific challenges to their security engineering, identified e.g. in [4]. In this work we focus on some of the security requirements challenges: identifying SoS security requirements, security requirements modeling and requirements as source of variability.

Security requirements engineering for systems is still struggling to provide a complete integration with the functional and the other non-functional requirements [14]. For SoS, there are relatively few approaches, e.g. [6].

On the other hand, we previously realized a study in the field of adaptative systems [2, 3]. In this study we defined a process allowing (i) identifying and adapting some non functional requirements, (ii) focusing on those adapted requirements into an elicitation requirement process, (iii) transforming them by Model-Driven Engineering (MDE) tools and methods into a system specification expressed in SYSML [16], and (iv) finally verifying this process using formal methods. In this process, we used RELAX [18], a requirements engineering language for Dynamic Adaptive Systems (DAS), where explicit constructs are included to handle uncertainty.

In this paper, we propose to adapt this previous work focusing on security properties of SoS. More specifically, we propose using RELAX to identify security properties that would be contextually adapted. These properties are transformed using MDE methods and tools towards both SYSML specification models and the security verification method OrBAC. We illustrate the process with a maritime case study.

This paper is structured as follows. Section 2 presents a motivating example for our proposal. Section 3 presents Relax and its application to the case study. Section 4 presents an MDE process focusing on relaxed properties allowing to generate both SYSML system specifications and OrBAC rules from user textual requirements. Finally Section 5 concludes and presents some perspectives.

## 2. CASE STUDY

We present here an SoS case study in the domain of Maritime Safety and Security, inspired by the one described in [17]. SoS in this domain have the task of monitoring a given maritime area and taking appropriate actions in case unwanted events take place. The SoS is constituted from five organisations: three military navies of Danemark, Netherlands and Italy respectively, DK_Navy, NL_Navy and IT_Navy, the Netherlands coastguard NL_CG and the European navy EU_NAVFOR which regroups ships from the three national navies, which are its members. Each national navy has three types of ships: Frigate, Patrol and Surveillance And Reconnaissance (SAR) and a Command and Control System (C2S). The Netherlands coastguard has only one type of ships - SAR, and a C2S. The European navy has a general type of ships - EU_Vessels, which contains four types of vessels: EU_Frigate, EU_Patrol and EU_SAR which correspond to the ship types from the three national navies, and the type EU_Law_enforcement, which is used to model the ships which, at a certain moment, have the task of preventing/figthing against crime. The European C2S is the Maritime Security Center (MSC), which will verify if the different ships have the rights to access information. Information is of two types: public and private.

The textual security requirements of this case study, focused on the MSC, are reprised here as described in [17]:

Msc1: Operators on vessels of the EU_NAVFOR can access public information about the ships transiting in the operation area.

Msc2: Operators on vessels of the EU_NAVFOR which are assigned to the prevention of criminal activities (or similar tasks) can access additional "off the record" information about ships which has been gathered during the operation.

Msc3: Operators on SAR vessels certified by EU_NAVFOR members can access all the information about a ship in case of emergency.

These security requirements could be verified with a formal method like OrBAC [15]. OrBAC enables writing security rules as first order logic predicates to describe rights like permissions or prohibitions. For example, a permission predicate has as parameters: the organization for which the rule is written, the role in that organization to which the role applies, the type of access the role has, the resource to which the role has access and optionally the context.

If we verified the three requirements of our Maritime SoS with OrBAC, we would model them for example like this :

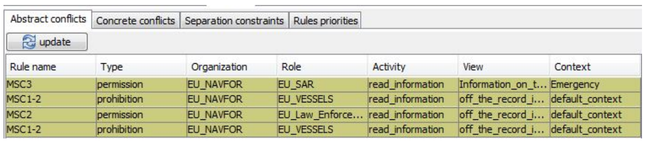
Msc1 : permission(EU_NAVFOR, EU_Vessels, read_info, public_info, default_context);

Msc1-2 : prohibition(EU_NAVFOR, EU_Vessels, read_info, private_info, default_context);

Msc2 : permission(EU_NAVFOR, EU_Law_enforcement, read_info, private_info, default_context);

Msc3 : permission(EU_NAVFOR, EU_SAR, read_info, all_info, emergency);

Please note that, consistently with the "military spirit", we describe not only the permissions, but also the prohibitions. This results in the rule Msc1-2. On these rules, OrBAC detects two abstract conflicts, cf. Figure 1, between: (i) Msc1-2 and Msc3, due to the fact that we give both a permission and a prohibition on the private information, in two different contexts; (ii) Msc1-2 and Msc2, due to the fact that the role EU_Law_enforcement is modelled in OrBAC as a subrole of the EU_Vessels role, and thus it has both a permission and a prohibition to read private information.



| Rule name | Type | Organization | Role | Activity | View | Context |
|---|---|---|---|---|---|---|
| MSC3 | permission | EU_NAVFOR | EU_SAR | read_information | Information_on_t... | Emergency |
| MSC1-2 | prohibition | EU_NAVFOR | EU_VESSELS | read_information | off_the_record_i... | default_context |
| MSC2 | permission | EU_NAVFOR | EU_Law_Enforce... | read_information | off_the_record_i... | default_context |
| MSC1-2 | prohibition | EU_NAVFOR | EU_VESSELS | read_information | off_the_record_i... | default_context |

**Figure 1: Conflicts detected by OrBAC**

In the rest of the paper, we propose to reuse adaptation methods in order to specify which security requirement would be adapted in order to limit the conflicts between security requirements. Indeed, in a precise context, some security requirement would be lower without presenting a danger for the complete system. But, we must be able to specify it early to take it into account both in the system specification and the security management environment.

## 3. RELAXING SECURITY PROPERTIES

### 3.1 Relax presentation

Relax is a requirements engineering language for Dynamic Adaptive Systems (DAS), where explicit constructs are included to handle uncertainty. Typically textual requirements prescribe behavior using a modal verb such as SHALL that defines functionality that a software system must always provide. For DAS however, it is not always possible to achieve all of those SHALL statements; or we may allow for trade-offs between SHALL statements to relax noncritical statements in favor of other, more critical ones. Therefore Relax identifies two types of requirements : one that can be relaxed in favor of other ones called variant or relaxed and other that should never change called invariant. This is done through the inclusion of an alternative, temporal or ordinal Relaxation modifier that will define the requirement as relaxable.

The Relax operators are designed to enable requirements engineers to explicitly identify requirements that should never change (invariants) as well as requirements that a system could temporarily relax under certain conditions. Each of the relaxation operator defines constraints on how a requirement may be relaxed at run-time. In addition, it is important to indicate what uncertainty factors warrant a relaxation of these requirements, thereby requiring adaptive behavior. This information is specified using the *MON* (monitor), *ENV* (environment), *REL* (relationship) and *DEP* (dependency) keywords. The environment properties capture the state of the world i.e., they are characteristics of the operating context of the system. Often, however, environmental properties cannot be monitored directly because they are not observable. The *MON* keyword is used to define those properties which are directly observable and which may contribute information towards determining the state of the environment. The REL keyword is used to specify in what way the observable (given by *MON*) can be used to derive information about the environment (given by *ENV*). Finally requirements dependencies are delimited by *DEP*, as it is important to assess the impact on dependent requirements after relaxing a given requirement.

Relax has been used for the requirements definition of different DAS. In [2], we have applied Relax on the requirements of an Ambient Assisted Living (AAL) system. We start by applying the Relax process on Functional Requirements and Non Functional Requirements of the AAL. The Relax process results in the distinction of relaxed requirements which are adaptable and invariant requirements which are fixed. We then model these requirements using goal oriented approach. To go one step further, we applied Relax on another Barbados Car Crash Crisis Management System (bCMS) case study. In [3] we have compared the Relax based requirements definition approach with other Relax approaches and we concluded that the approach based on Relax better helps in identifying the uncertainty factors associated with DAS.

## 3.2 Application of Relax to the case study

We think that Relax can be used in defining the security requirements of an SoS. For illustration, we model the textual requirements from the maritime case study.

As detected previously by OrBAC, Msc1-2 is in conflict with Msc2 and respectively Msc3. In order to avoid this, we need to relax Msc2 and Msc3. The relaxed version is given below with its associated uncertainty factors : we have applied the **MAY** and **OR** Relax operators. The *ENV* operator specifies the context in which the requirement is concerning. The *MON* operator specifies some information monitored by the requirement. The *REL* operator specifies relationships such as those illustrated. The *DEP* operator is issued from *positive* and *negative* impacts in Kaos (see next section).

Private information **MAY** be read by ships that are executing a task of fighting against crime **OR** by SAR ships in case of emergency.

- **ENV** : fight against crime (FAC), access to private information (API)
- **MON** : Aggression level (AL), Access rules (AR)
- **REL** : $FAC = (AL > 10?true; false); API = select * from\ AR\ where\ \ldots$
- **DEP** : it has a positive dependency on Msc1-2.

Detecting conflicts between security rules is not obvious. We argue that this would be realized with systems stakeholders and requirement engineers using existing security methods. Using Relax is useful for specifying within a dedicated language the adaptation and its impact.

## 4. A COMPLETE MDE BASED PROCESS

### 4.1 General Process Overview

Considering security requirements through the entire requirement engineering process needs dealing with them from textual requirements to system specification. In our previous work, we have showed that it is important to use some well established engineering methods like GORE (Goal Oriented Requirement Engineering). In this context, after treating security requirements with Relax, we propose to use Kaos [11], a GORE method, to help the future system stakeholders to better define the security goals of the system. Once the system goals have been established, we argue it is important

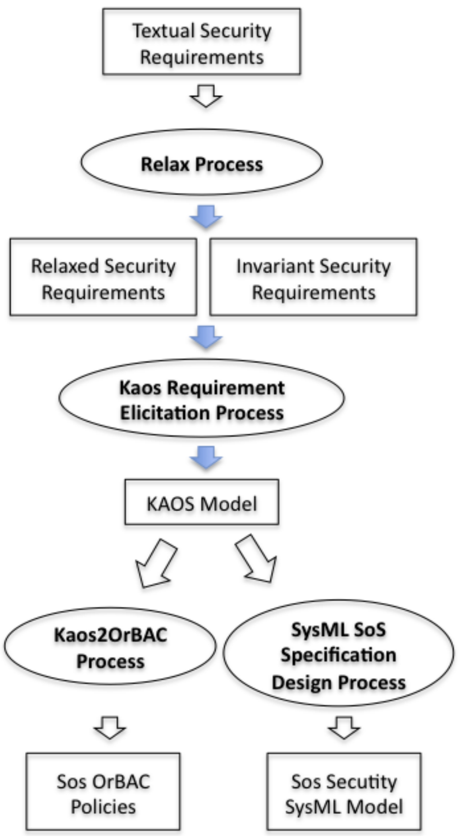

Textual Security Requirements
Relax Process
Relaxed Security Requirements
Invariant Security Requirements
Kaos Requirement Elicitation Process
KAOS Model
Kaos2OrBAC Process
SysML SoS Specification Design Process
Sos OrBAC Policies
Sos Secutity SysML Model

**Figure 2: General Requirement Process Overview**

to translate them into the system specification described in SysML. It is also important to translate them towards OrBAC, to detect if other conflicts may exist. We propose to do this translations using MDE methods. The general process overview is presented in Figure 2.

### 4.2 MDE Concerns

Model Driven Engineering allows transforming a model *m1* conforming to a metamodel *MM1* into another model *m2* conforming to another metamodel *MM2*, using transformation rules. Transformation rules specify how each model element from *m1* would be traduced into a corresponding other model element of *m2*. These transformations would be realized automatically or manually depending of the technology maturity, the need of a human decision, ...

Figure 3 shows the different metamodels used in our approach and the set of transformation rules (represented by diamonds). Metamodels exist for most of these languages: for Relax in [18], for Kaos in [11] and SysML in [16]. We explored transformation rules between these 3 metamodels in [1]. Finally, transformation rules between Kaos and OrBAC metamodels were explored in [8].

Despite the fact that all metamodels and transformations exist, it is important to show that the complete process is occurring correctly. We have not yet studied the complete transformation process but we are currently working on this.

### 4.3 Discussion

Our MDE process pushes the verification of security early to the requirements phase of the development life-cycle. Thus,

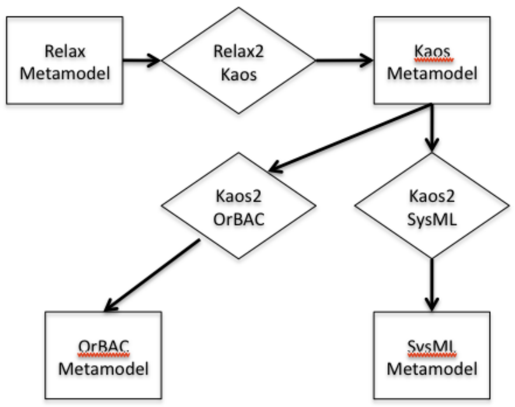


**Figure 3: Metamodels and Transformation Chain**

issues like conflicts between security policies can be caught early and costly re-design and re-implementation can be avoided. The proposal is especially beneficial to SoS, as it can capture early conflicts between security policies which may be fine at the composing system level, but when they interact at the SoS level, conflicts may emerge.

## 5. CONCLUSIONS AND PERSPECTIVES

In this paper, we proposed a process for security requirements for SoS in order to better consider their security during the requirement elicitation process. The main benefit consists in identifying some conflicting rules as early as possible and adapting them when it is possible. This adaptation is realized using RELAX, and the adapted properties are transformed both in a specification model and in OrBAC rules. Additionally, we pass via KAOS models to help system stakeholders and requirement engineers to identify system security goals.

It is important to note that this is ongoing work. Despite metamodels and transformation rules existing, we are studying more precisely their applicability to security concerns. Moreover, we are studying how RELAX and security methods like OrBAC would mutual enrich themsleves. More precisely, we expect that RELAX operators are not sufficient in the security context. We think about adding specific operators that make the difference between the *context* and the *role* concepts. In OrBAC, concepts to account for RELAX operators SHALL, MAY, OR, AND would probably need to be added. Lastly, our process provides two models. We need to be able to verify that both models are consistent. We are working on a verification process allowing to do this, using OMEGA2IFx as described in [3].

## 6. REFERENCES

[1] M. Ahmad. *Modeling and Verification of Functional and Non Functional Requirements of Ambient, Self Adaptive Systems.* PhD thesis, University of Toulouse Mirail, France, 2013.

[2] M. Ahmad, J. Araújo, N. Belloir, R. L. Jean M. Bruel, Christophe Gnaho, and F. Semmak. Self-Adaptive Systems Requirements Modelling: four Related Approaches Comparison. In *Comparing *Requirements* Modeling Approaches Workshop (CMA@RE), RE 2013*, Rio de Janeiro Brazil, 2013.

[3] M. Ahmad, I. Dragomir, J. M. Bruel, I. Ober, and N. Belloir. Early Analysis of Ambient Systems SysML Properties using OMEGA2-IFx. In *3rd International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, 2013.

[4] V. Chiprianov, L. Gallon, M. Munier, P. Aniorte, and V. Lalanne. Challenges in Security Engineering of Systems-of-Systems. In *Conference de l'Ingenierie Logiciel (CIEL)*, Paris, France, 2014.

[5] C. H. Dagli and N. Kilicay-Ergin. *System of Systems Architecting*, pages 77–100. John Wiley & Sons, 2008.

[6] A. Fuchs and R. Rieke. Identification of security requirements in systems of systems by functional security analysis. In A. Casimiro, R. de Lemos, and C. Gacek, editors, *Architecting Dependable Systems VII*, volume 6420 of *LNCS*, pages 74–96. 2010.

[7] A. Gorod, R. Gove, B. Sauser, and J. Boardman. System of systems management: A network management approach. In *System of Systems Engineering. IEEE Intl Conf. on*, pages 1–5, 2007.

[8] M. Graa, N. Cuppens-Boulahia, F. Autrel, H. Azkia, F. Cuppens, G. Coatrieux, A. Cavalli, and A. Mammar. Using requirements engineering in an automatic security policy derivation process. *4th SETOP International Workshop on Autonomous and Spontaneous Security*, 2011.

[9] M. Jamshidi. System of Systems - Innovations for 21st Century. In *Industrial and Information Systems, 2008. ICIIS 2008. IEEE Region 10 and the Third intl Conf on*, pages 6–7, Dec 2008.

[10] J. Klein and H. van Vliet. A systematic review of system-of-systems architecture research. In *Proc. of the 9th Intl ACM Sigsoft Conference on Quality of Software Architectures*, QoSA '13, pages 13–22, 2013.

[11] A. V. Lamsweerde. *Requirements Engineering: From System Goals to UML Models to Software Specifications.* Wiley, 1st edition edition, 2009.

[12] S. Lukasik. Vulnerabilities and failures of complex systems. *Int. J. Eng. Educ.*, 19(1):206–212, 2003.

[13] M. W. Maier. Architecting principles for systems-of-systems. *Systems Engineering*, 1(4):267–284, 1998.

[14] D. Mellado, C. Blanco, L. E. Sánchez, and E. Fernández-Medina. A systematic review of security requirements engineering. *Comput. Stand. Interfaces*, 32(4):153–165, June 2010.

[15] A. Miége. Definition of a formal framework for specifying security policies. The Or-BAC model and extensions. *Ecole Nationale Supérieure des Télécommunications*, 2005.

[16] OMG. Systems Modeling Language Specification V1.3. Technical Report formal/2012-06-01, Object Management Group, 2012.

[17] D. Trivellato, N. Zannone, M. Glaundrup, J. Skowronek, and S. Etalle. A Semantic Security Framework for Systems of Systems. *Int. journal of cooperative information systems*, 22(1):1–35, 2013.

[18] J. Whittle, P. Sawyer, N. Bencomo, B. H. C. Cheng, and J. M. Bruel. RELAX: Incorporating Uncertainty into the Specification of Self-Adaptive Systems. In *Proc. of the IEEE Intl Requirements Engineering Conference*, pages 79–88, 2009.