Adding the non-functional Security property in an UML Use case diagram

Sana Oueslati¹, Mouez ALI¹, Faiez Gargouri¹
¹ MIRACL Laboratory, ISIMS, University of Sfax
Sfax, Tunisia
sana.oueslati.ba@gmail.com, mouez.ali@fmsf.rnu.tn, faiez.gargouri@fsegs.rnu.tn

Abstract. In Software engineering, elicitation phase consists of identifying functional and non-functional requirements. Functional requirements express an action to be performed by the system in response to a request. Non-functional requirements are implicit requirements to which the system must respond. Both kinds of properties are relevant to software development. Software systems, apart from implementing all the desired functionalities, must also cope with non-functional aspects such as: reliability, security, accuracy, performance... Despite the fact that non-functional properties are very difficult to achieve and at the same time are expensive to deal with, non-functional requirements have been, for too long, overlooked during the development of software systems. In this paper, we present an approach to extend the UML use case diagram for specifying non-functional requirements. Furthermore, we provide a categorization of the non-functional properties into Architectural, dynamic and temporal properties. Finally, we suggest a method to add non-functional properties into UML use case diagram.

Keywords: UML, functional requirement, non-functional requirement, use case diagram.

1 Introduction

The Unified Modeling Language (UML) [1] allows us to model information on the static and dynamic behavior of a system. UML has emerged as the standard notation for modeling software systems. Software systems are characterized both by functional and non-functional requirements. Both aspects are relevant to software development.

Functional requirements (FRs) express an action to be performed by the system in response to a request. It describes the behaviors (functions or services) system that supports users' goals, tasks or activities. An FR is a requirement that, when satisfied, will allow the user to perform a specific function [8].

Non-functional requirements (NFRs) are implicit requirements to which the system must respond. Such requirements can be performance, security, reusability, reliability, portability. An NFR is usually some form of constraint or restriction that must be considered when designing the solution. An NFR is a statement of how a system must behave; it is a constraint upon the systems behavior [5].
However, the aspects of non-functionality information on how the system acts with respect to some observable attributes become also relevant to software development [17].

As software systems become more complex and important for business and everyday life, the need to better address NFRs becomes increasingly more crucial [3].

Multiple proposals have been developed to analyze and model the functional requirement, and are nowadays widely used like use case diagram [14].

However, in practice, based on different proposed methodologies and languages, the designer and the developer often focus on functional properties and do not directly address the non-functional ones. The latter are generally ignored in the early phases of the software development process and managed in an ad-hoc manner without following a rigorous process [8].

In this paper, we focus on the integration of NFRs in an UML diagram. However, UML particularly the use case diagram the current standard method for static functional requirements modeling lacks equally mature modeling constructs for dealing with NFRs.

This paper is organized as follows; Section 2 describes different approach for NFRs classification. In section 3, we present our approach of the integration of NFRs into an UML diagram. We focus on the integration of the security properties into the use case UML diagram. In section 4, we give an overview of related work which focuses on the integration of NFRs into UML diagrams. In section 5, we summarize the presented work and we give an outline for future work.

## 2 Classification of NFRs

There are several classifications of NFRs: Sommerville and Kotonya consider NFRs as “restrictions and constraints among system services” [6]. This classification may be considered the most comprehensive in terms of coverage of NFRs types under three key categories: product, process, and externally related NFRs. The first category relates to the possible or desired attributes that a system may possess. Any constraints and restrictions on the development process over the system relate to the second category of this classification. Finally, externally related NFRs are concerned with organizational regulations, national or international standards, and interoperability requirements [13].

[7] Classifies NFRs as quality of service, compliance, architectural constraint and development constraint.

In [8], the author proposes two classes of NFRs. The first class called non-functional safety properties corresponds to the properties that can be monitored and verified at runtime, like security properties, time related constraints, architectural invariants, etc. The second class corresponds to the properties that represent qualities of software such as portability, reusability, maintainability, adaptively, etc. Properties of the second class are called non-functional liveness properties.

The non-functional safety properties are classified into two categories: structural and behavioral non-functional properties. By structural properties, we mean those properties related to the structure and the architecture of the application, e.g., the cardinality of the objects constituting the system, the types of relation between
objects. By behavioral properties, we mean those properties related to the behavior of
the application, e.g., temporal constraints on certain operations of the application, or
constraints on permitted operation sequences. We distinguish two kinds of behavioral
properties: qualitative and quantitative properties. In the context of qualitative
behavioral properties, Seven-pro is applied for specifying and enforcing static and
dynamic separation of duties and different types and characteristics of delegation
policies on top of role-based access control.

In the context of quantitative behavioral properties, Seven-pro is applied for
specifying and enforcing temporal properties in Web service compositions [8].

There are many classifications of NFRs. In the next section, we present our
classification inspired from the other classification.

3 Our approach
In this section, we present our approach. First we present our classification. Second,
we explain the integration of NFRs.

3.1 Our classification
We can distinguish from the existing work [6], [7], [8], [13] the following classes of
NFRs:

Architectural properties concern the specification and the execution of
architectural invariants and coordination protocols dynamic software architectures.
They specify the architectural style describes the types of objects in the system, the
types of connections between these objects and architectural constraints (eg
invariants).

Dynamic properties describe the conditions without which the system cannot
function normally. These properties describe what should happen when the system
properly works. They often take the form of conditions that indicate if the system is in
a certain state, then inevitably it will reach another expected state.

Temporal properties express the probability that a system works correctly
(without failure) for a specified time under conditions (which we call basic
conditions) set accurately.

In the next section we focus on the integration of the dynamic properties
especially the security property.

3.2 Integration of NFRs
Software systems, aside from implementing all the desired functionalities, must also
cope with non-functional aspects such as: security, performance, availability,
capacity, reliability, integrity, usability, confidentiality, portability, efficiency,
recovery and reusability. We focus in this paper on the integration of security non-
functional properties into the security UML diagrams.
Figure 1 shows the simplified meta-model for UML 2 use case diagram [15]. An actor specifies a role played by a user or any other system that interacts with the subject. A use case is the specification of a set of actions performed by a system, which yields an observable result that, is, typically, of value for one or more actors or other stakeholders of the system. An extend relationship from an extending use case to an extended use case that specifies how and when the behavior defined in the extending use case can be inserted into the behavior defined in the extended use case. An include relationship defines that a use case contains the behavior defined in another use case. An extension point identifies a point in the behavior of a use case where that behavior can be extended by the behavior of some other (extending) use case, as specified by an extend relationship.

![Figure 1: A simplified Meta model for UML 2 use case diagram [15]](image)

3.2.1 Extension of meta-model

In order to integrate security non-functional properties in the use case diagram, we need first to extend the meta-model of the UML use case diagram. We extend the meta-model of a use case diagram by adding more elements such as NFRActor, NFRUsecase, SecureRelationship, Permit, Protect.

Figure 2 shows the extension of the metamodel for UML 2 use case diagrams.
Figure 2: Extended meta-model of UML use case diagram

The next table illustrates the different elements integrated into the UML2 meta-model.

<table>
<thead>
<tr>
<th>UML Element</th>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>NFRActor</td>
<td>An NFRActor models a type of role (NFR) played by an entity that interacts with the subject.</td>
</tr>
<tr>
<td>Use case</td>
<td>NFRUse case</td>
<td>A NFRUse case is a list of steps, typically defining interactions between a NFRActor and a system, to achieve a goal.</td>
</tr>
<tr>
<td>Relationship</td>
<td>SecureRelationship</td>
<td>A SecureRelationship is a connection between model elements. It is a type of model element that adds semantics of security to a model by defining the structure and behavior between the model elements.</td>
</tr>
<tr>
<td>Relationship</td>
<td>Permit</td>
<td>This relationship specifies that the behaviour of a use case may be permitted by the behaviour of a security use case.</td>
</tr>
<tr>
<td>Relationship</td>
<td>Protect</td>
<td>This relationship specifies that the behaviour of a use case may be protected by the behaviour of a security use case.</td>
</tr>
</tbody>
</table>
In this paper, we tried to integrate the security properties in the use case diagram. In the future works, we are hopping to integrate the different non-functional properties in the different UML diagrams. In the next section, we present a case study Automated Teller Machin (ATM) to illustrate the the integration of the security properties in the use case diagram.

3.2.2 Example

The purpose of this section is to illustrate, through an example, how identifying and modeling NFRs impacts on the design models. We apply our proposal to a simplified Automated Teller Machin. We outline how to integetre NFRs into the original UML models.

The functionalities of a Automated Teller Machin include allowing customer to enter her login to authorize card and cancel card, allow customer to consult balance, withdraw money, request a checkbook, charging mobile phone and print receipt.

Figure 3 shows the use case diagram. This diagram is composed of an acor “Customer” and a the use cases: Login, Request a checkbook, Withdraw money, Consult balance, Charging mobile phone, print receipt.

In this exemple we will try to integrate the security properties (dynamic properties). The integrated use case diagram shown in Figure 4 presents the integrated non-functional requirement (security properties). The integration of this requirement
is represented by adding three *NFRActor* such as *AuthenticationServer, AuthorizationServer, UnauthorizedServer* and three *NFRUseCase* such as *Authenticate, Authorize Access, Unauthorized Access*.

![Extended use case diagram](image)

**Figure 4: Extended use case diagram**

In this paper, we have integrated the non-functional UML elements that have already added to the meta-model in order to introduce the security properties in the use case diagram.

In the next section, we present the related research that suggested methods to illustrate the non-functional requirements in UML conceptual representations.

### 4 Related work

The way requirements should drive the rest of the software development process has been a subject of many research projects in the past. Unfortunately, all of them focus primarily, when not exclusively, on the functional requirements regardless of the fact that NFRs are among the most expensive and difficult to deal with [13].
Despite the fact that NFRs are very difficult to achieve and at the same time are expensive to deal with, a few research works have focused on them as first class requirements in a development process.

Several approaches have been proposed for integrating non-functional properties into UML diagrams. Some of them are generic (i.e., can be applied to any type of property) and others are applied only to a specific property and/or to a specific application domain [8]. We classify these approaches into three groups: (i) approaches of integrating NFRs into conception UML, (ii) Approaches that propose a UML profile for representing the NFRs (Extension of UML) (iii) approaches that suggest a double use (marriage) of UML with a language of requirement specification.

The first category is classified into two possibilities. The first possibility is the integration of NFRs into one diagram like use case diagram [3] or class diagram [18]. In [3], the author proposes a framework for representing and integrating NFRs with FRs in the use case model at four association points: subject (system boundary), actor, use case, and communicate association. In this framework, NFRs are represented as softgoals to be satisfied. The contributions of this work include: 1) an intuitive approach for using use case model elements to provide context for NFRs. 2) NFR propagation rules to eliminate redundant specification for common NFRs; and 3) a process for representing and integrating NFRs and FRs.

[18] Proposes a framework to incorporate NFRs, as reusable components, into standard UML notations. Such a framework can also integrate those reusable NFRs with the extracted UML representations of legacy systems during the reverse engineering process. This research work uses standard XMI representation of UML models without proposing any extension to it.

The second possibilities are the integration of NFRs into several diagrams like use case diagram, class diagram and sequence diagram.

In [11], the author proposes an approach to modeling FRs, analyzing their impacts to the design and making a complement to the original design models for the Model Driven Development (MDD) methods [10]. First, taking the existing functional models represented by the UML [14] as an input, especially the use case diagram, class diagram and sequence diagram, the author’s identify NFRs, refine them, and identify alternative development technique for achieving them. This process is inspired by the use-case driven development method and the NFR framework proposed by [2] [12]. Furthermore, this work propose a meta model to model the information needed when integrating them to the original UML models and then, a process is proposed to model the analysis result and integrate these elements into the functional model from both the structural and behavioral view.

Concerning the second class of the approaches that propose a UML profile for integrate NFRs into UML diagrams, [4] build use case diagrams for a real mobile Grid application by using a UML-extension, called GridUCSec-Profile, through which it is possible to represent specific mobile Grid features and security aspects for use case diagrams, thus obtaining diagrams for secure mobile Grid environments.

Touching the third class, [17] introduces a proposal for representing non-functional aspects by using UML. This purpose is to apply UML-concepts in combination with OCL (Object Constraint Language) [16] in order to describe the non-functional aspects of software systems by resorting to the NoFun notation formulated by [5].
This approach uses the UML concepts for defining stereotypes, class compartments and stereotyped dependencies to represent the non-functionality. And also OCL language is used for representing the constraints imposed on the implementations of software components.

In [13], the authors propose a systematic approach to assure that conceptual models will reflect the NFRs elicited. We propose to build both the functional and the non-functional views anchored in the Language Extended Lexicon (LEL) [9]. Building the non-functional view departs from the use of an existing LEL. This work has extended the LEL to help NFR elicitation. The LEL is now structured to express that one or more NFRs are needed by a symbol. It is also structured to handle dependency links among one NFR and all the notions and behavioral responses that are necessary to satisfy this NFR.

There are not completed approaches that integrate all properties into UML diagrams. Many properties are neglected. Our work tries to integrate in the first step the security properties into the use case diagram that presented in this paper. Next, we tries to integrate the different properties in the diagrams UML. In the next section, we summarizes the presented work and we gives outlines of future works.

5 Conclusion and future works

NFRs are implicit requirements which the system must respond such as performance, security, reusability, reliability, portability, etc. we are classified this requirement into three classes such as architectural properties, dynamic properties and temporal properties. First, architectural properties concern the specification and the execution of architectural invariants. The dynamic properties describe the conditions without which the system cannot function normally. Finally, temporal properties express the probability that a system works correctly (without failure) for a specified time.

This paper focused on the integration the security properties into the use case diagram. As perspectives, it remains to refine our approach by adding other element to represent the security proprieties in the use case diagram. Indeed, we will look to integrate the different classes of non functional properties (architectural, dynamic and temporal properties) into the different diagrams UML (class, activity, sequence…). Moreover, we look to develop our tool in order to integrate the diverse classes on NFRs in the different diagrams UML.

6 References