# TOWARDS AN ONTOLOGY-BASED APPROACH FOR REUSING NON-FUNCTIONAL REQUIREMENTS KNOWLEDGE

RODRIGO VAUGHAN VELEDA

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF ARTS

GRADUATE PROGRAM IN INFORMATION SYSTEMS AND TECHNOLOGIES
YORK UNIVERSITY
TORONTO, ONTARIO

OCTOBER 2016

# Abstract

Requirements Engineering play a crucial role during the software development process. Many works have pointed out that Non-Functional Requirements (NFR) are currently more important than Functional Requirements. NFRs can be very complicated to understand due to its diversity and subjective nature. The NDR Framework has been proposed to fill some of the existing gaps to facilitate NFR elicitation and modeling. In this thesis, we introduce a tool that plays a major role in the NDR Framework allowing software engineers to store and reuse NFR knowledge. The NDR Tool converts the knowledge contained in Softgoal Interdependency Graphs (SIGs) into a machine-readable format that follows the NFR and Design Rationale (NDR) Ontology. It also provides mechanisms to query the knowledge base and produces graphical representation for the results obtained.  To evaluate whether our approach aids eliciting NFRs, we conducted an experiment performing a software development scenario.

## Acknowledgements

This accomplishment would not have been possible without the support and assistance of those people who surrounded me during this endeavor.

Firstly, I would like to express gratitude to my supervisor. Professor Cysneiros was responsible for introducing me a greater view of the Requirements Engineering field still during my undergraduate studies. I feel honored to have worked with him throughout this experience. His unconditional academic support and life advices guided me to overcome every challenge associated with pursuing a Master's degree, especially in a foreign country. Without his guidance, this achievement would never have been possible.

I am also grateful for the assistance and support provided by my co-supervisor, Professor Litoiu. I feel privileged to be his student and appreciate the opportunity he granted me to be part of his research team. The learning curve that I exposed myself as a result of this experience is unmeasurable. By being such a renowned expert both in academic and industry fields, Professor Litoiu inspires me as a role-model.

Moreover, I would like to thank the committee members Professor Liaskos and Professor Lesperance for participating and providing feedback during my thesis defense.

I would also like to express gratitude to everyone I worked with at CERAS lab. Dr. Hamzeh Khazaei, Dr. Marios-Eleftherios Fokaefs, Dr. Cornel Barna, Dr. Mark

Shtern, Nasim Beigi Mohammadi, Saeed Zareian, and Brian Rampasad. Thanks for sharing your knowledge and time with myself. I have learned a lot with all of you.

Lastly, I would like to dedicate this work to my lovely family and friends who always stood beside me, even in person or remotely. This achievement would not have been possible without your support and kind thoughts. Thank you for providing me the necessary strength to thrive in this challenge.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1
# Introduction

Software engineers must address both Functional and Non-Functional Requirements (NFRs) during the software development process [1]. Functional requirements represent procedures that a given system will be capable of executing. In a different manner, NFRs are known to define quality attributes for a software system [2, 3], including characteristics such as privacy, security, usability, and other similar aspects related to software quality.

## 1.1 Problem and Motivation

Currently, most of the notations and techniques available for representing requirements focus on functional characteristics for a given system. Software engineers rarely take into account the elicitation and modelling of NFRs during the early stage of the development cycle [1]. The incident with the London Ambulance System is a comprehensive example of how neglecting or not adequately addressing NFRs can affect a software system [4]. The system had to be shut down right after its first deployment due to major problems mostly related to a set of NFRs including performance and usability.

It is noteworthy to mention that NFRs are considered complex due to its diversity and fuzziness. Different types of NFRs include constraints that may not be presentable in a formal way and also not defined as clear as they should be. For instance, some constraints such as expected response time and failure provisioning may be related to design implementations that are not acknowledged by the time NFR requirements are specified [5]. Also, interpreting NFRs is a task that relies on a subjective understanding. Possible Tasks and/or *Operationalizations* for *satisficing* a given NFR might differ according to each stakeholder's needs. Moreover, one solution to implement a single NFR may produce synergies and perhaps more important conflicts with another NFR. A necessary task such as 'Use of Cryptography' for *satisficing* an NFR of Security, for instance, might directly affect the *satisficing* of a Performance NFR in a negative manner hence, leading to a possible conflict. Therefore, this relationship behavior brings the perception that one NFR can rarely be 100% satisfied. As a result of this understanding, the term *Satisficed* was introduced in [6, 7] to represent the idea of having a given NFR satisfied within tolerable limits.

The need to deal with the frequent scenario where one NFR has many interdependencies with other NFRs has led many researchers to investigate how to improve elicitation and modelling of NFRs. Chung et al. [6] proposed the representation of NFRs through *Softgoal Interdependency Graphs* (SIGs) as part of the NFR Framework [6], [8]. SIG catalogues denote a graphical illustration of fundamental quality aspects, including conflicts and trade-offs, for *satisficing* a given NFR. By using this graphical approach, a software engineer can record the design and reasoning process of each NFR into distinct graphs. Consequently, generated graph records can be further applied as a supportive knowledge regarding NFRs in a given domain in the future. Also, an empirical work proposed by Cysneiros [9] illustrates that using SIG catalogues can contribute to avoiding omissions and missed conflicts among NFRs, despite the fact that SIG catalogues do not scale too well. Additionally, a comprehensive analysis performed by de Gramatica et al. [10] demonstrates that the use of catalogs can assist non-expert users to adequately identify satisfactory characteristics regarding a particular field even considering the difficulty of navigating through extensive models.

Although SIG catalogues provide the storage of NFR knowledge, identifying the desired reusable knowledge among a vast number of graphs can be a tedious and tricky task. As denoted by Cysneiros [9], SIG catalogues tend to grow in a complex graphical way. Therefore, even providing a graphical notation, SIG catalogues can be hard to be interpreted by humans if we are dealing with  broad contexts with a large amount of design rationale descriptions. Also, access to information embedded in multiple SIGs tends to be limited to a particular and local use only, hence preventing a broader knowledge re-utilization.

## 1.2 Research objectives and questions

The primary objective of the research work in this thesis is to assemble a strong foundation for the NDR Framework to grow. Therefore, we decompose our main objective into two major accomplishments:

1. Provide a mechanism to facilitate the reuse of NF knowledge. In order to do that we designed and implemented a tool which operates as the framework's core. The tool mainly works as information browser regarding NFR knowledge.

2. Evaluate whether using the tool helps stakeholders to better *satisfice* NFRs on real-world software systems.

To achieve these accomplishments and consequently our main objective, we address the following research questions:

1. What is the overall applicability of the tool?
2. How granular is the knowledge provided by the tool?
3. How can the tool help with identifying trade-offs and conflicts among NFRs?

## 1.3 Thesis Contributions

The work developed in this thesis is directed towards the reuse of NFR knowledge. By answering the previously mentioned research questions, our contributions are the following:

- Firstly, we designed and implemented the NDR Tool, which plays a key role in the NDR Framework. The tool is responsible for concentrating NFR knowledge into an ontology base and allowing its reuse. Its architecture follows the web service principles for cloud computing. Also, it provides a graphical user interface as a requirement to enhance user experience. The NDR Tool is the baseline of the work developed in this thesis.
- Secondly, we have conducted a study on the evaluation involving the usage of the NDR Tool by human participants. The experiment scenario was related to the development of a real-world software system, and the participants had to identify

and model possible NFRs following the NFR Framework notation. The NDR Tool supported just half of the participants. The other half could only rely on SIG catalogs previously developed according to the literature and represented by static images expressing the same NFRs offered by the NDR Tool.

- Our final contribution is the assessment of the overall applicability of the NDR Tool based on the analysis of the results obtained from the case study evaluation. As the NDR Tool represents an important role in the framework, our findings statistically demonstrate that our proposed solution can increase the support to software engineers to better *satisfice* NFRs in real-world scenarios during the software development life cycle. More than graphically express NFRs, the NDR Tool provides an amalgamation of diverse SIG catalogs associated with multiple NFRs into a single representation, defining a permanent NFR knowledge evolution.

Additionally, the work presented in this thesis is the unfolding research output first introduced in a short paper published in proceedings of the 8th International I* Workshop (iStar 2015). The full citation can be found in [11].

## 1.4 Thesis organization

The research work in this thesis is structured as follows. Chapter 2 emphasizes the related work to this research field. Chapter 3 presents our proposed approach and describes its architecture and characteristics. We describe the performed experiment for evaluation of the proposed work on Chapter 4. On Chapter 5, we discuss the obtained findings from the performed experiment. Finally, on Chapter 6 we illustrate our conclusions and present future ideas.

# Chapter 2
# Related Work

Since errors due to improperly dealing with NFR are among the most difficult and expensive to fix [12]–[14] identification and proper expression of NFRs are essential for understanding and reasoning about NFR satisficing. Some works have been trying to address this issue through different perspectives.

In this chapter, we illustrate relevant literature on NFR knowledge representation including diverse approaches and perspectives. Moreover, we aggregated and divided the findings of our literature research into two main groups: Representing NFR Knowledge and Using ontologies to deal with NFRs.

## 2.1 Representing NFR knowledge

Mylopoulos et al. [8] and Chung et al. [6] idealized and established the NFR Framework approach where SIGs represent NFR information. As the NFR Framework acts as a part of the baseline of the proposed work in this thesis, a further description regarding its features and components will be depicted in the subsequent Chapter unitedly with the characteristics of this developed study.

Chung and Nixon [15] proposed the NFR-Assistant as a proof of concept CASE tool for modeling NFRs according to the NFR Framework guidelines. OpenOME [16], RE-Tools for StarUML [17], and jUCMNav [18] are also modeling mechanisms for representing NFR Knowledge using NFR Framework. Nevertheless, none of these technologies promote the reuse of knowledge by concentrating and linking NFR information from multiple developed models. In other words, they simply provide the software engineer the ability of visual modeling NFR characteristics as part of the system they are modelling.

Cysneiros et al. [19] illustrated a framework to integrate the representation of NFRs as NFR Graphs linked and integrated to functional conceptual models expressed using UML and Object-Oriented models, commonly used in the industry. Figure 1 illustrates the designated usage of NFR Graphs within the proposed framework. The main link between the functional and non-functional views is provided by the *Language Extended Lexicon* (LEL). For validation, the authors carried three different case studies based on a software system specification. The findings of the performed evaluation suggest that their proposed approach leads to a more prolific software development process and conceptual models with greater quality regarding both functional and non-functional characteristics.

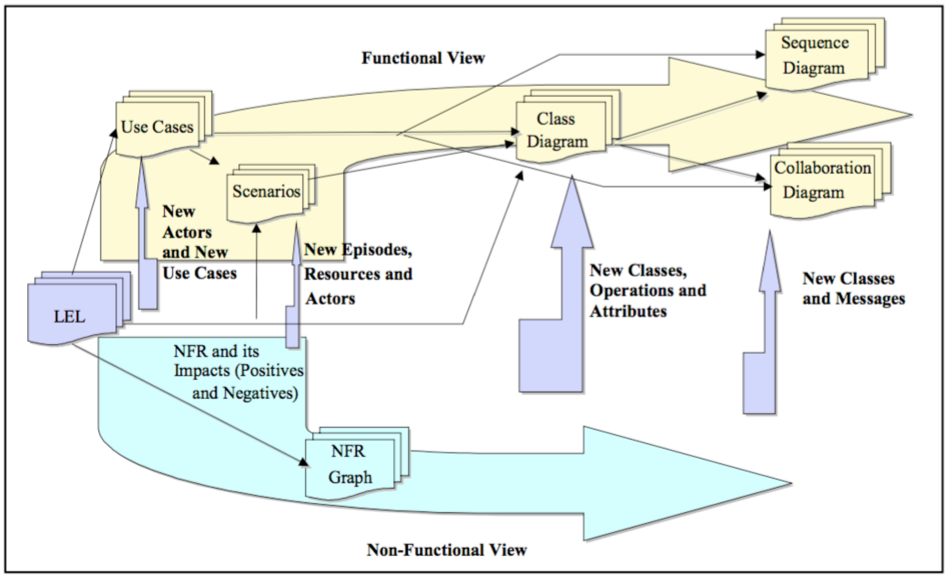
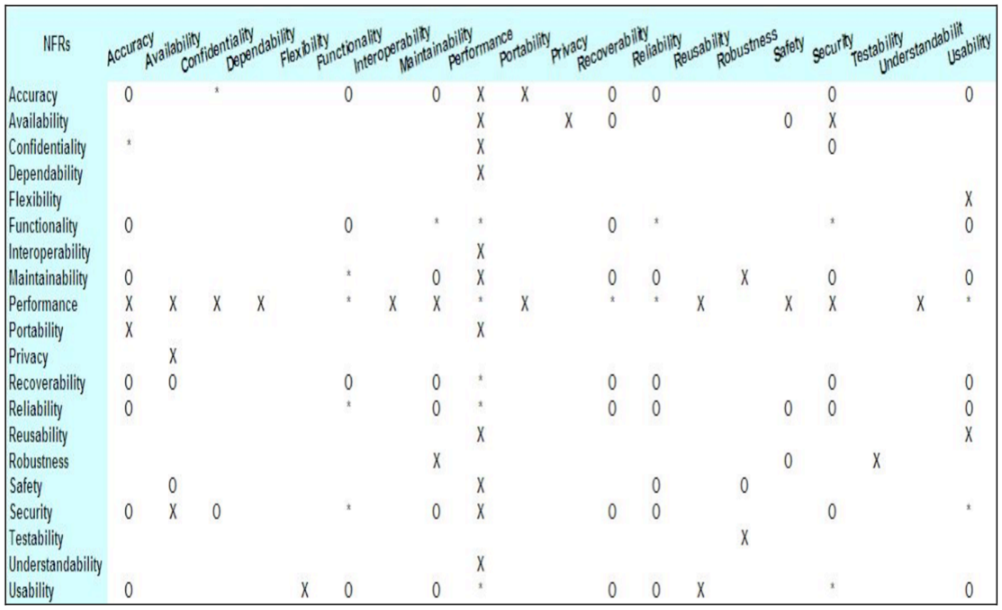

Functional View
Sequence Diagram
Use Cases
Class Diagram
Scenarios
Collaboration Diagram
New Actors and New Use Cases
New Episodes, Resources and Actors
LEL
New Classes, Operations and Attributes
New Classes and Messages
NFR and its Impacts (Positives and Negatives)
NFR Graph
Non-Functional View

**Figure 1 The usage of NFR Graphs** [19]

Mairiza and Zowghi [20] developed a catalog of conflicts amid NFRs. As a result of an extensive analysis of the understanding of NFRs throughout the relevant literature, the proposed conflict catalog is designated to assist software engineers in identifying and resolving potential conflicts among NFRs during several phases of software development. Moreover, the catalog defines three different levels for conflict categorization: absolute conflict, relative conflict, and never conflict. The following Figure 2 illustrates the catalog of conflicts amid NFRs where "X" represents absolute conflict, "*" denotes relative conflict, and "0" expresses the never conflict category.



| NFRs | Accuracy | Availability | Confidentiality | Dependability | Flexibility | Functionality | Interoperability | Maintainability | Performance | Portability | Privacy | Recoverability | Reliability | Reusability | Robustness | Safety | Security | Testability | Understandabilit | Usability |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Accuracy | 0 | | * | | | 0 | | 0 | X | X | | 0 | 0 | | | | 0 | | | 0 |
| Availability | | | | | | | | | X | | X | 0 | | | | 0 | X | | | |
| Confidentiality | * | | | | | | | | X | | | | | | | | 0 | | | |
| Dependability | | | | | | | | | X | | | | | | | | | | | |
| Flexibility | | | | | | | | | | | | | | | | | | | | X |
| Functionality | 0 | | | | | 0 | | * | * | | | 0 | * | | | | * | | | 0 |
| Interoperability | | | | | | | | | X | | | | | | | | | | | |
| Maintainability | 0 | | | | | * | | 0 | X | | | 0 | 0 | | X | | 0 | | | 0 |
| Performance | X | X | X | X | | * | X | X | * | X | | * | * | X | | X | X | | X | * |
| Portability | X | | | | | | | | X | | | | | | | | | | | |
| Privacy | | X | | | | | | | | | | | | | | | | | | |
| Recoverability | 0 | 0 | | | | 0 | | 0 | * | | | 0 | 0 | | | | 0 | | | 0 |
| Reliability | 0 | | | | | * | | 0 | * | | | 0 | 0 | | | 0 | 0 | | | 0 |
| Reusability | | | | | | | | | X | | | | | | | | | | | X |
| Robustness | | | | | | | | X | | | | | | | | 0 | | X | | |
| Safety | | 0 | | | | | | | X | | | | 0 | | 0 | | | | | |
| Security | 0 | X | 0 | | | * | | 0 | X | | | 0 | 0 | | | | 0 | | | * |
| Testability | | | | | | | | | | | | | | | X | | | | | |
| Understandability | | | | | | | | | X | | | | | | | | | | | |
| Usability | 0 | | | | X | 0 | | 0 | * | | | 0 | 0 | X | | | * | | | 0 |

**Figure 2 Catalog of conflicts amid NFRs** [20]

Doerr et al. [21] proposed an experience-based NFR Method to elicit, document, and analyze NFRs. The NFR Method is designed to define a minimum and adequate group of weighable and accountable NFRs. As an evaluation of the proposed work, the authors demonstrated three different case studies for requirements elicitation utilizing the NFR method. The experiments were performed in real industry, and each case study had different input settings. As an outcome of the study, overall, more detailed NFRs were identified and documented using the suggested NFR Method.

Considering that all these proposed approaches are appropriately associated with the representation of NFR knowledge, none of them exploits the storage and reuse of NFR information in a dynamic manner. Some techniques [6], [8], [19]–[21] indeed document and describe NFR knowledge. However, taking into the account their outputs, software engineers still rely on a manual manner for searching for specific knowledge and identifying possible associations. Others [15]–[18] provide methods for modeling NFR knowledge. Still, none of these tools offers such a mechanism to aggregate and make dynamically available the generated knowledge.

## 2.2 Using ontologies to deal with NFRs

Al Balushi et al. [22] proposed the ElicitO as an ontology-based framework and tool that meets the elicitation, prioritization, and validation of functional and non-functional requirements. Figure 3 illustrates the architecture of ElicitO. Mainly, the ontology level presents information regarding quality metrics and characteristics according to the model specified in the *ISO 9126* standard. The user interaction level provides a graphical interface for registering NFRs based on a needed scenario. Also, it communicates with the ontology level and verifies the satisfaction and possible conflicts among NFRs based on pre-defined rules originated from the quality model. In order to validate the developed work, the authors performed a case study by developing an intranet portal and were able to conclude that NFRs can be better understood, negotiated, captured, and documented with the ElicitO assistance.

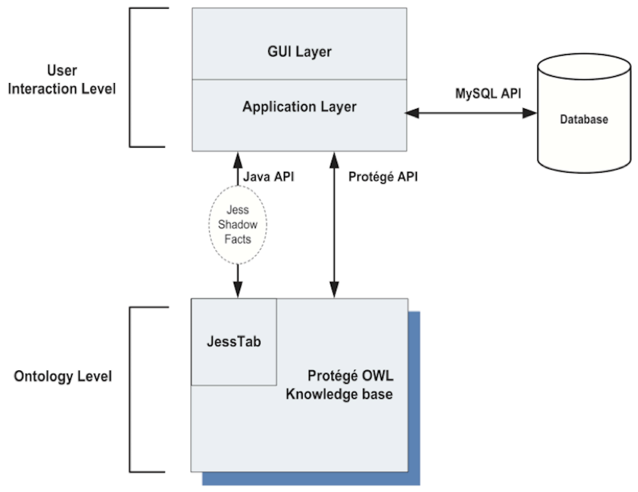


**Figure 3 The architecture of ElicitO** [22]

Guizzardi et al. [23] proposed a UFO ontology-based [24] technique for understanding NFRs as quality attributes. This approach provides guidelines for differentiating functional and non-functional requirements. Additionally, this method suggests a specification language designed to assist in the capture of NFRs.

An ontology-based for dealing with conflicts among NFRs was proposed by Liu [25]. This method uses ontologies and metadata for representing NFR knowledge and pre-defined rules for reasoning about possible conflicts. These rules may vary according to the target domain.

Dobson et al. [26] proposed a domain-independent ontology for expressing NFR information regarding *Quality of Service* (QoS) constraints. Despite the semantic model, this suggested method also includes useful rules for automatic conversion between metrics and units for a given constraint. As an evaluation process, the proposed ontology was successfully applied in a case study designed to exploit its reasoning functionalities.

Regarding a domain-specific ontology for representing NFR knowledge, Koay et al. [27] suggested an ontological model for making provisions in pervasive healthcare. Multiple ontologies were modeled as part of the core of a remote patient monitoring system. As a result of ontology integrations, several semantical rules representing a possible scenario associated with a patient were designed to assist with needed provisions.

In regards to the representation of NFR knowledge through ontologies, Sancho et al. [28] proposed an ontological database consisting of two ontologies: The NFR Type Ontology and SIG Ontology. Both ontologies combined can represent the knowledge from models generated according to the NFR Framework guidelines. However, the main shortcoming of this developed approach is the absence of a class for defining a *Correlation* between *Softgoals*. Consequently, this lack of definition implies the impossibility of reasoning involving more than one NFR. To overcome this drawback, Lopez et al. [29] introduced the NDR Ontology. As the NDR Ontology plays a significant role in our developed work, we further emphasize its usage and characteristics in the following Chapter.

Hu et al. [30] proposed an ontology model as a baseline for modeling NFRs. The approach also follows the NFR Framework guidelines for representing NFR knowledge. Additionally, the developed solution provides a rule-based method for promoting an automated reasoning of possible conflicts among NFRs. However, the proposed work does not mention any storing feature for handling NFR knowledge involving multiple contexts. Also, despite promoting automated reasoning of conflicts, the suggested approach does not seem to offer a search method to facilitate knowledge retrieval.

Despite the utilization of ontologies, none of the mentioned approaches suggest the reuse of NFR knowledge as the main feature and in a dynamic manner. ElicitO framework and tool [22] employs a knowledge base for storing NFR information through a user interface. However, it does not sufficiently exploit the stored knowledge in order to demonstrate possible synergies between NFRs. Although it can indicate conflicts among NFRs by following a static rule-based strategy, the framework does not depict adequately in which level such friction may occur and the primary reason associated with it.

Furthermore, despite other approaches [23], [25]–[30] define ontologies for representing NFR knowledge,  none of them cover details about handling and storing the generated information as a whole for future use. Since ontology models tend to be vast and complex, dynamically adding new information and handling persisted knowledge in an efficient manner may be an alternative to assure a well-maintained environment that promotes the reuse of NFR knowledge.

## 2.3 Summary

In this chapter, we introduced relevant literature on NFR knowledge representation. Firstly, we discussed important approaches simply regarding the representation of NFR information.

Lastly, we emphasized fundamental methods for representing NFR knowledge with ontologies. Regarding both sections, we stated the most important circumstances where the mentioned approaches differ from our developed work.

# Chapter 3
# Foundation

In this chapter, we introduce essential concepts adopted as a baseline for our work. Firstly, we highlight the NFR Framework [6], [8], which is an approach for representing NFRs in a graphical manner. Next, we emphasize the Ontology concept in the Semantic Web field and describe its main components and characteristics. Lastly, we explain the NDR Ontology [29], which plays a major role in our developed work.

## 3.1 NFR Framework

The NFR Framework was first idealized by Mylopoulos et al. [8] and further established by Chung et al. [6]. Essentially, the framework defines a systematic and practical method for representing and using NFRs during the software development process as quality attributes [6].

Fundamentally, the NFR Framework promotes the idea that completely satisfying a given NFR is quite unusual. Hence, the term *Satisficed* was adopted to represent this notion of fulfilling a particular NFR within reasonable boundaries. Reflecting this understanding, the NFR Framework describes NFRs as *softgoals*.

*Softgoals* detail goals that have no sharply defined criteria or rationale characterizing its satisfaction [6]. Moreover, the potential relationships among *softgoals* are denoted as interdependencies. Consequently, *softgoals* and its interdependencies are graphically represented as catalogs named as *Softgoal Interdependency Graphs* (SIGs). The following Figure 4 illustrates the potential elements of a SIG catalog:

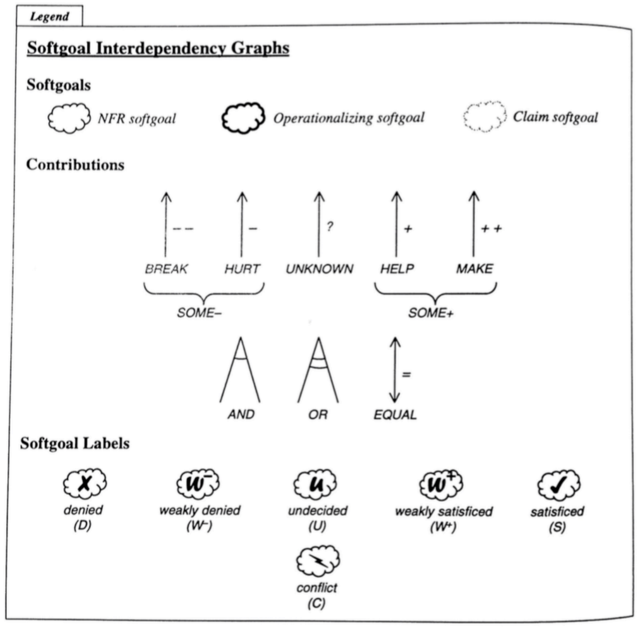


**Figure 4 The potential elements of a SIG catalog [6]**

## Softgoals

As previously mentioned, *softgoals* mainly represent NFRs. Therefore, the NFR Framework defines three different types of *softgoals* including NFR *Softgoal*, *Operationalizing Softgoal*, and Claim *Softgoal*.

NFR *Softgoal* elements typically denote a major NFR type. *Decomposition* methods can be applied to refine the *softgoal* into lower-level *softgoals* towards determining one or more solutions to implement this *softgoal*.

On the other hand, *Operationalizing Softgoals* are responsible for expressing certain positive or negative guidelines attached to another *softgoal*. Typically, an *operationalization* will reflect a characteristic the software should provide in order to satisfice a *softgoal* and therefore will frequently have a positive *contribution*. However, in many situations an *operationalization* may also present negative *contributions* not only to its direct parent *softgoal* but more important to other *softgoals*. In many cases *operationalizations* will reflect new functional requirements that should be implemented in the software.

Lastly, Claim *Softgoal* elements commonly represent the rationale associated to another *softgoal* or interdependency.

**Contributions**

A *Contribution* defines each relationship between *softgoal* elements. In other words, a *Contribution* characterizes the type of an interdependency among *softgoals*. It varies not only from negative (*BREAK*, *HURT*, and *SOME-*) to positive (*HELP*, *MAKE*, and *SOME+*) definitions, but it also incorporates a neutral (*UNKNOWN*) type and equality (*EQUAL*) and conditional (*AND* and *OR*) operators.

**Softgoal Labels**

The NFR Framework also provides a manual procedure for evaluating the decisions related to *satisficing* a particular *softgoal*. Hence, *Softgoal* Labels are applied to previous *softgoal* elements as a result of the decision assessment.

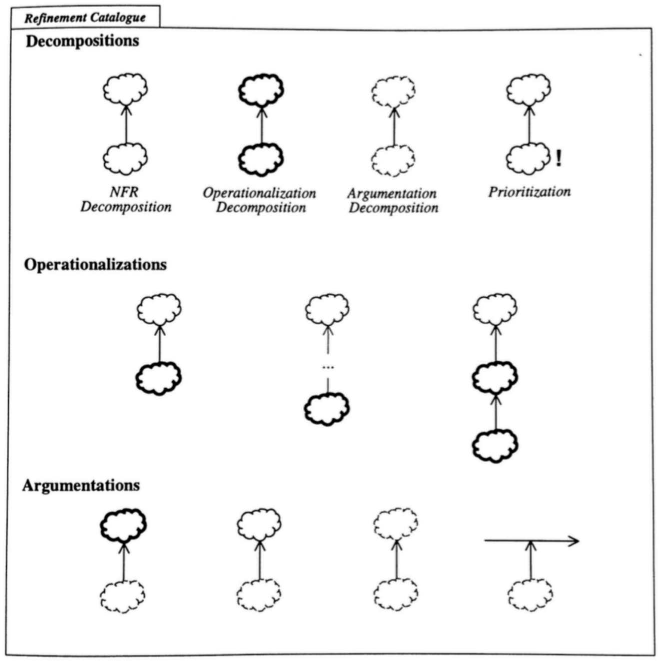


**Figure 5 The potential Explicit Interdependencies among *softgoals* [6]**

Figure 5 illustrates the potential Explicit Interdependencies among *softgoals*. Essentially, each *softgoal* can be linked by using *Decompositions*, *Operationalizations*, or *Argumentations*. *Decompositions* produce an Interdependency involving *softgoals* of the same type. *Operationalizations* create an Interdependency relating an NFR *softgoal* and an *Operationalizing softgoal*. Lastly, *Argumentations* represent the recording of design rationale through the use of Claim *softgoal*. Hence, this type of element is the only one that can be applied to every type of *softgoal* and Interdependencies.

The NFR Framework also specifies Implicit Interdependencies. Their main element is defined as *Correlations*. *Correlations* are used to link *softgoals* related to a given NFR to *softgoals* associated with another NFR, characterizing synergies and conflicts.

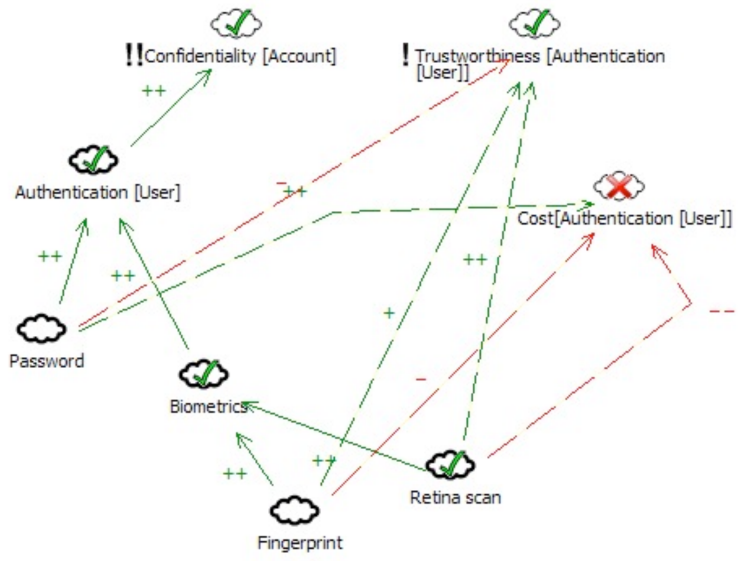

!!Confidentiality [Account]
! Trustworthiness [Authentication [User]]
Authentication [User]
Cost[Authentication [User]]
Password
Biometrics
Retina scan
Fingerprint
++
+
-
--

**Figure 6 Partial SIG developed with NFR Framework [17]**

Figure 6 demonstrates a representation of a partial SIG developed using some of the inherent elements of NFR Framework. It is noteworthy to mention that SIGs may be domain-free or related to a particular field. In this particular case, Figure 6 depicts a SIG related to *Confidentiality* regarding the access of a user account.

## 3.2 Ontology

Beners-lee et al. [31] define Ontology as a fundamental concept of the Semantic Web. Differently from the Philosophy domain where it explores the study of the human being existence, Ontology in the Semantic Web field points towards the aggregation of term definitions that can express the semantic knowledge of a particular artifact through the taxonomy and inference rules.

Within an Ontology, the taxonomy allows the definition of classes of objects and its relationships. For instance, *Address* and *GeoLocation* classes may be defined as subclasses of a parent class *LocationType*, and *CityCodes* may be established to be applied only to objects that serve as a *LocationType*. In this scenario, *CityCodes* would only be able to be applied to objects representing either an *Address* or a *GeoLocation*. Additionally, classes of objects may have subclasses that inherit parent properties. For example, if *CityCodes* must be a type of *City* and *City* objects typically have *Websites*, it is reasonable to assume that *CityCodes* also have *Websites* [31].

On the other hand, inference rules in Ontology can provide a more robust applicability. If a *CityCode* is linked to a *StateCode*, and a given *Address* has that *CityCode*, one can assume that the *Address* is associated with the *State Code* as well [31]. In other words, for example, a tax software system employing this Ontology could effortlessly identify that the address of York University in Toronto must be associated with the Ontario province, which is in Canada. Therefore, the tax software system must comply with Canadian tax regulations.

The modeling and description of an Ontology may involve different technologies. Resource Description Framework (RDF) [32] and RDF Schema (RDFS) [33], Web Ontology Language (OWL) [34], and SPARQL [35] are technologies that unitedly provides mechanisms to represent and explore a given Ontology.

## Resource Description Framework (RDF) and RDF Schema (RDFS)

RDF denotes a specification for expressing content through web identifiers commonly known as *Uniform Resource Identifiers* (URIs) [32]. Therefore, RDF provides the possibility of representing resources through a node graph, including properties and values. The following Figure 7 demonstrates a node graph that follows the RDF specification. As it is noticeable, an individual of type *Person* is instantiated according to a target ontology. The attribute *type*, which characterizes *Person*, is defined by the standard RDF specification. Therefore, its URI differs from the other ones. On the other hand, *fullName*, *mailbox*, and *personalTitle* custom attributes follow the same URI as the individual one since they are defined by the target ontology.

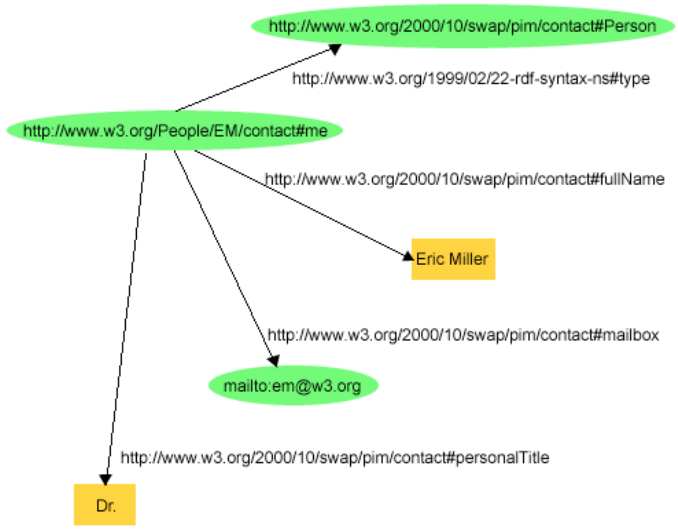

http://www.w3.org/2000/10/swap/pim/contact#Person
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.w3.org/People/EM/contact#me
http://www.w3.org/2000/10/swap/pim/contact#fullName
Eric Miller
http://www.w3.org/2000/10/swap/pim/contact#mailbox
mailto:em@w3.org
http://www.w3.org/2000/10/swap/pim/contact#personalTitle
Dr.

**Figure 7 Node graph in RDF/XML** [32]

In order to acquire the most from representing resources with RDF, a vocabulary description language identified as RDF Schema have been proposed. Mostly, RDF

Schema defines classes and properties over RDF resources in a comparable approach to object-oriented programming languages. In other words, RDF Schema proposes fundamental classes to handle common attributes such as collections and literals. Also, it provides essential properties including range and domain definition that can be applied to a specific class. Figure 8 demonstrates the applicability of RDF Schema in RDF/XML:

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<rdfs:Class rdf:about="http://www.w3.org/2000/01/rdf-schema#Resource">
  <rdfs:isDefinedBy rdf:resource="http://www.w3.org/2000/01/rdf-schema#"/>
  <rdfs:label xml:lang="en">Resource</rdfs:label>
  <rdfs:comment>The class resource, everything.</rdfs:comment>
</rdfs:Class>

<rdf:Property rdf:about="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">
  <rdfs:isDefinedBy rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
  <rdfs:label xml:lang="en">type</rdfs:label>
  <rdfs:comment>Indicates membership of a class</rdfs:comment>
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:domain rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdf:Property>

<rdfs:Class rdf:about="http://www.w3.org/2000/01/rdf-schema#Class">
  <rdfs:isDefinedBy rdf:resource="http://www.w3.org/2000/01/rdf-schema#"/>
  <rdfs:label xml:lang="en">Class</rdfs:label>
  <rdfs:comment>The concept of Class</rdfs:comment>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdfs:Class>
```

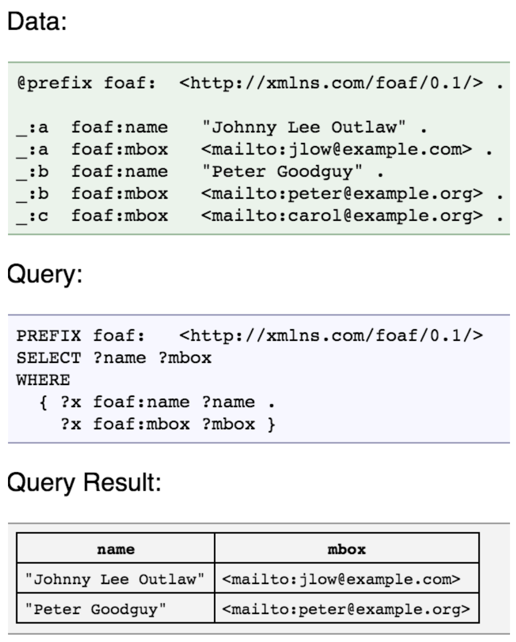
**Figure 8 RDF Schema in RDF/XML** [33]

### Web Ontology Language (OWL)

OWL proposes an additional vocabulary for classes and properties. The language specification is defined based on the RDF Schema. Consequently, it inherits all the previous mentioned features provided by RDF Schema. Additionally, OWL supplements RDF Schema with peculiarities that include Property Cardinality and Class Intersection. Also, OWL introduces the concept of Axioms. Axioms link classes and properties, creating statements or definitions that assert what is valid in a given domain [36]. For instance, two distinct classes can be defined as to be disjoint classes by using Axioms.

**SPARQL**

SPARQL is a query language for RDF. Similar to database query languages, SPARQL provides a mechanism to retrieve any element specified in a particular Ontology by writing a simple query. This query can also match data types for specific properties if necessary. The output of an SPARQL query can either be standard result sets or RDF node graphs. The following Figure 9 shows a snippet of an SPARQL query that inquiries *name* and *mbox* from a dataset based on the FOAF vocabulary specification[1]:



Data:

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .

_:a  foaf:name   "Johnny Lee Outlaw" .
_:a  foaf:mbox   <mailto:jlow@example.com> .
_:b  foaf:name   "Peter Goodguy" .
_:b  foaf:mbox   <mailto:peter@example.org> .
_:c  foaf:mbox   <mailto:carol@example.org> .
```

Query:

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE
  { ?x foaf:name ?name .
    ?x foaf:mbox ?mbox }
```

Query Result:

| name | mbox |
|---|---|
| "Johnny Lee Outlaw" | <mailto:jlow@example.com> |
| "Peter Goodguy" | <mailto:peter@example.org> |

**Figure 9 SPARQL query and its output** [35]

[1] http://xmlns.com/foaf/spec/

There are several tools to model and visualize Ontologies using OWL including Protégé[2], NeON Toolkit[3], and Vitro[4]. In this research work, we particularly preferred the use of Protégé for our Ontology needs due to our previous expertise with this tool in past studies. Protégé is an open-source platform and provides multiple plugins to design and visualize Ontologies. Also, Protégé project is maintained by Stanford University and has an active community of users. Figure 10 illustrates an Ontology being modeled in Protégé:

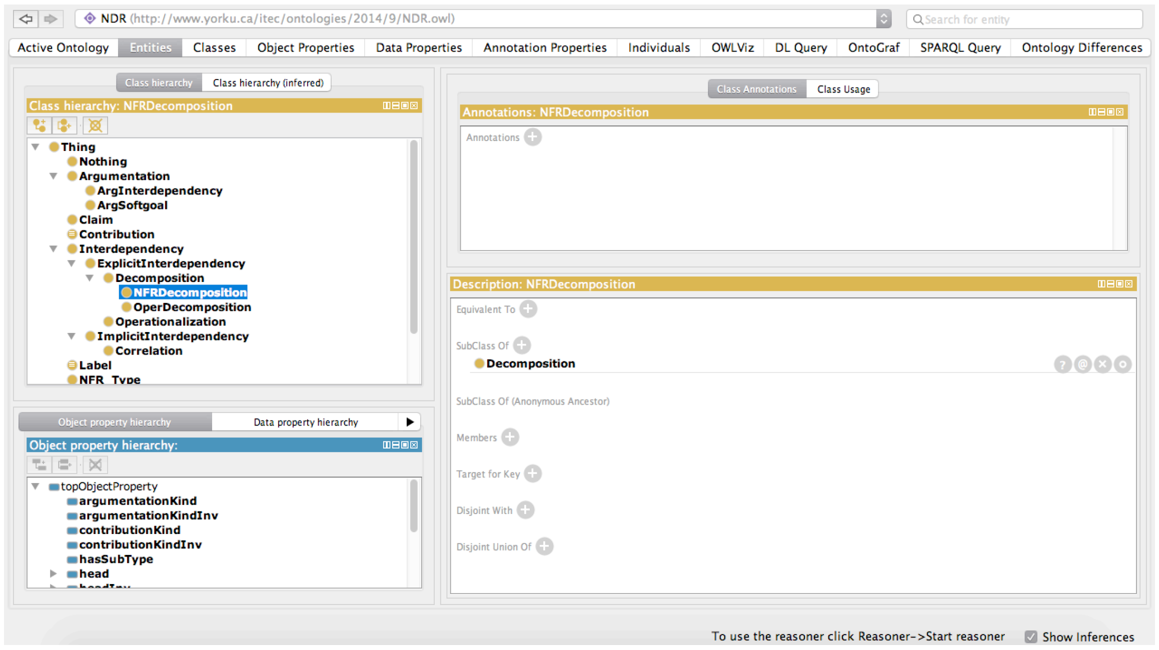


**Figure 10 Ontology being modeled in Protégé**

[2] http://protege.stanford.edu/

[3] http://neon-toolkit.org/

[4] http://vitro.mannlib.cornell.edu/

## 3.3 NDR Ontology

Lopez et al. [29] proposed the Non-Functional Requirements and Design Rationale (NDR) Ontology as a mechanism to represent NFR and design argumentative logic knowledge in a machine-readable format. The ontology was developed on the top of the approach proposed by Sancho et al. [28], which partially follows the guidelines of the NFR Framework. More important, the NDR Ontology adds the concepts of *Claims* and *Correlation* among NFRs that are not covered in Sancho et al. [28]. Also, it supplements the handling of *Decompositions* on a specialized element level. In other words, the NDR Ontology defines specialized elements as *Head* and *Tail* properties according to a *Decomposition* type. For instance, a given *NFRDecomposition* must have only *NFRSoftgoal* elements as *Head* and *Tail* properties. Consequently, the NDR Ontology describes in OWL all the necessary elements, properties, and interdependencies of the NFR Framework previously mentioned at the beginning of this section.

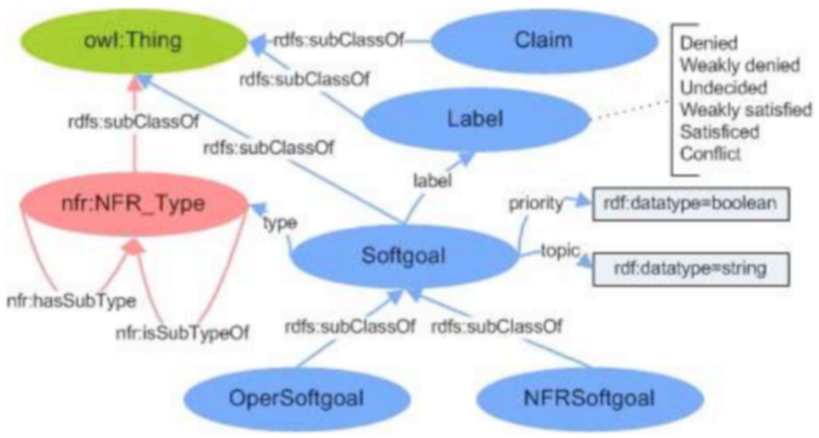


**Figure 11 Main elements of the NDR Ontology** [29]

Figure 11 shows the main elements of the NDR Ontology and its properties. As it is noticeable, the *Softgoal* class act as a parent of *OperSoftgoal* and *NFRSoftgoal* classes. This class inheritance is necessary since both *OperSoftgoal* and *NFRSoftgoal* share equivalent properties such as *Type*, *Label*, *Priority*, and *Topic*. The property *Type* links the target *Softgoal* to its particular NFR type. *Label* associates a *Softgoal* with its status after a decision assessment. *Priority* defines whether a given *Softgoal* has preference over the other ones. *Topic* connects a *Softgoal* to a specific domain. In this work, only *Type* is handled. The other properties are part of the NDR Ontology but they are intended to cover the evolutionary aspect of NFR during a project. Since this work aims at capturing as much alternative as possible to facilitate reuse in different domains/projects, these properties were left for future work. Lastly, a *Claim* class represents the *Claim Softgoal* element. As *Claim Softgoal*s are designed to record the design rationale behind certain decompositions, *Claim* is defined as a separated class.

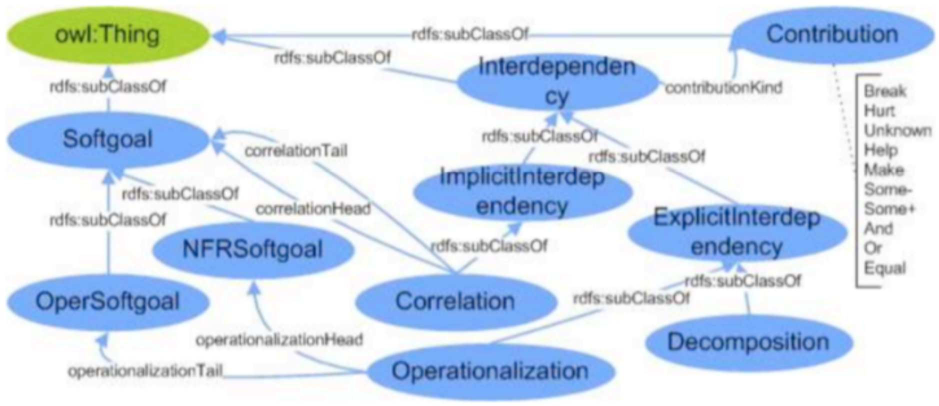


**Figure 12 Interdependency elements of the NDR Ontology** [29]

As illustrated in Figure 12, the NDR Ontology describes every *Interdependency* elements of the NFR Framework. Each possible *Interdependency* has a *Contribution* kind. Therefore, *Contribution* is represented as a separated class containing an enumeration of pre-defined labels. Moreover, *Correlation* is defined as *Implicit Interdependency*, being able to have *Softgoal* individuals of any kind as its *Tail* and *Head* properties. As *Explicit Interdependencies*, the ontology expresses *Operationalization* and *Decomposition* classes. *Operationalization* can only have *NFRSoftgoal* individuals as *Head* property and *OperSoftgoal* as *Tail* attribute. *Decomposition* is a parent class for *OperDecomposition* and *NFRDecomposition* classes. The former represents a *Decomposition* between *OperSoftgoal* individuals, and the latter describes a *Decomposition* among *NFRSoftgoal* individuals. Figure 13 illustrates the mentioned *Decomposition* definitions.

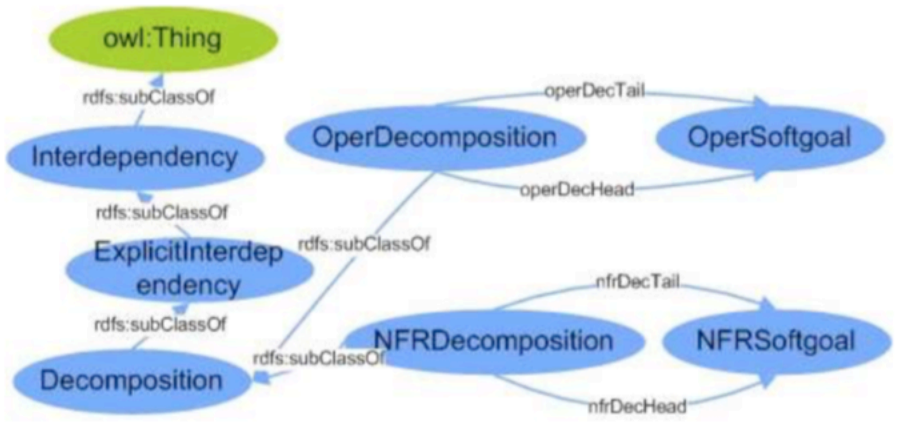


**Figure 13 Decompositions definitions of the NDR Ontology** [29]

Argumentation is also defined as a separated class in the NDR Ontology. It has a *Kind* attribute represented by the *Contribution* class as well. On the other hand, it has two specialized classes: *ArgNFRSoftgoal* and *ArgInterdependency*. Both classes define a *Claim Softgoal* individual as their *Tail* attribute. However, *ArgNFRSofgoal* class can only have a *Softgoal* individual as a *Head* property. Similarly, *ArgInterdependency* can only have an *Interdependency* individual as a *Head* attribute. These definitions express the fact that an *Argumentation* can be associated to both *Softgoal* and *Interdependency* elements. Figure 14 shows these Argumentation characteristics:

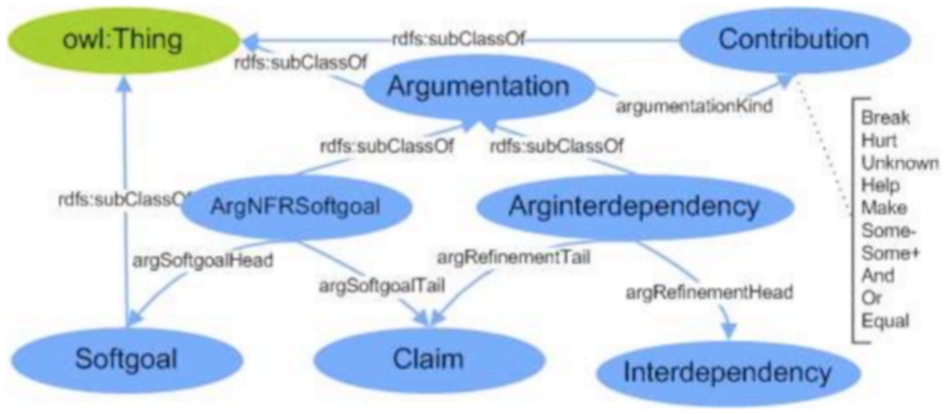


**Figure 14 Argumentation definitions of the NDR Ontology** [29]

Holding all the fundamental elements and relations of the NFR Framework, the NDR Ontology can adequately portray the knowledge existent in SIGs. The following Figure 15 shows a partial SIG depicting the NFR of Usability:

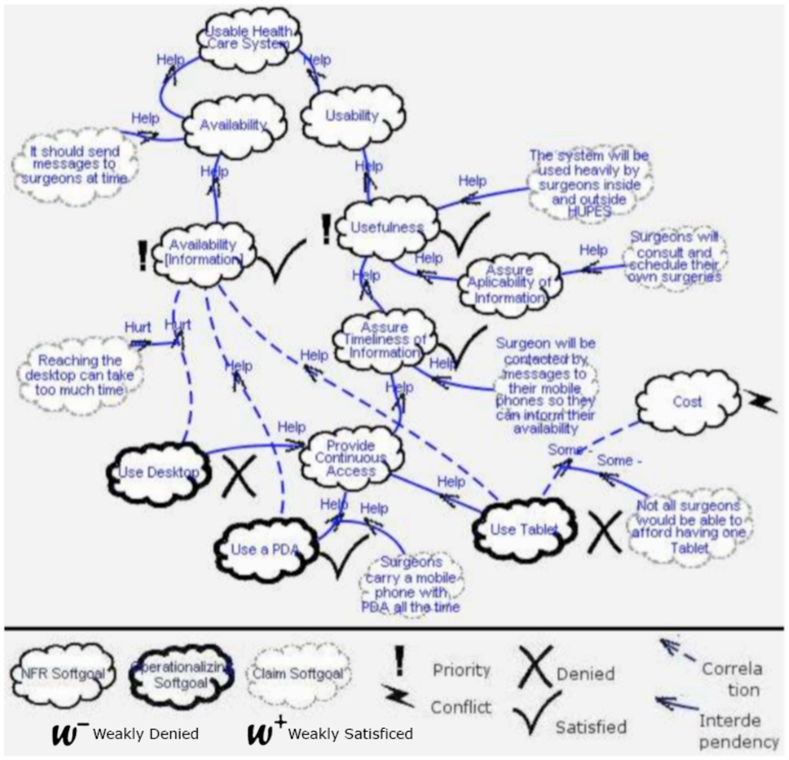

Usable Health Care System
Help
Help
Availability
Usability
Help
It should send messages to surgeons at time
Help
Help
The system will be used heavily by surgeons inside and outside HUPES
Help
Availability [Information]
Usefulness
Help
Surgeons will consult and schedule their own surgeries
Help
Assure Aplicability of Information
Help
Hurt
Hurt
Assure Timeliness of Information
Help
Reaching the desktop can take too much time
Help
Help
Surgeon will be contacted by messages to their mobile phones so they can inform their availability
Help
Cost
Help
Provide Continuous Access
Use Desktop
Some -
Some -
Help
Help
Help
Use Tablet
Not all surgeons would be able to afford having one Tablet
Use a PDA
Surgeons carry a mobile phone with PDA all the time
NFR Softgoal
Operationalizing Softgoal
Claim Softgoal
Priority
Conflict
Denied
Satisfied
Correlation
Interdependency
Weakly Denied
Weakly Satisficed

**Figure 15 Partial SIG of Usability [29]**

Accordingly, the representation of the knowledge contained in the partial SIG of Figure 15 is demonstrated in Figure 16 as OWL node graph. An *NFRDecomposition*, between *Usability* and *Usefulness Softgoals*, with the *Contribution* kind of *Help*, and *Satisficed* as the *Label* is expressed using the NDR Ontology classes. This same *Decomposition* can be envisioned in Figure 15.

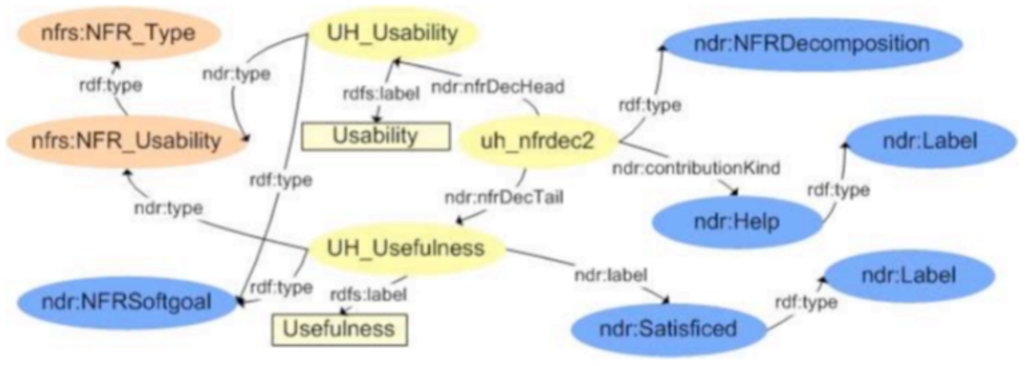


**Figure 16 Decomposition individual in NDR Ontology** [29]

Finally, Figure 17 depicts the same example scenario mentioned but expressed in a machine-readable format. The *NFRDecomposition* between *Usability* and *Usefulness* is described in OWL format following the guidelines of the NDR Ontology.

```
<nfrs:NFR_Type rdf:ID="NFR_Usability">
  <rdfs:label>Usability</rdfs:label>
</nfrs:NFR_Type>
<ndr:NFRSoftgoal rdf:ID="UH_Usability">
  <ndr:type rdf:resource="#NFR_Usabili
  ty"/>
  <rdfs:label>Usability</rdfs:label>
</ndr:NFRSoftgoal>
<ndr:NFRSoftgoal rdf:ID="UH_Usefulness">
  <rdfs:label>Usefulness</rdfs:label>
  <ndr:label rdf:resource="../ndr/
  ndr.owl#Satisficed"/>
  <ndr:type rdf:resource=
  "#NFR_Usability"/>
</ndr:NFRSoftgoal>
<ndr:NFRDecomposition rdf:ID=
"uh_nfrdec2">
  <ndr:nfrDecHead rdf:resource=
  "#UH_Usability"/>
  <ndr:nfrDecTail rdf:resource=
  "#UH_Usefulness"/>
  <ndr:contributionKind rdf:resource=
  "../ndr/ndr.owl#Help"/>
</ndr:NFRDecomposition>
```

**Figure 17 Decomposition individual in OWL format following NDR Ontology** [29]

# Chapter 4
# NDR Tool: The core of NDR Framework

In this chapter, we describe our proposal for concentrating NFR knowledge into an ontology base and allow its reuse. The NDR Tool was designed from the scratch to help software engineers to elicit more and better NFRs. Therefore, the tool promotes the reuse of existent NFR knowledge expressed using *Softgoal Interdependency Graphs* (SIGs) in accordance with the work proposed on [6], [29]. We have investigated and consolidated into the NDR Tool mechanisms to add knowledge to the NDR ontology as well as mechanisms to facilitate knowledge retrieval.

## 4.1 Definition

The NDR Tool plays a significant role in the NDR Framework as a knowledge handler. Hence, as a first implementation version, we designed the tool from the ground up aiming to fulfill the following major requirements:

- **FRQ1:** The tool must be able to extract information from SIGs generated by third-party applications. An administrator user must be able to provide an XML version of a SIG as an input file, and the tool needs to parse its existent information.
- **FRQ2:** The proposed tool needs to transform obtained information from SIGs to NFR knowledge according to the NDR Ontology specification.

- **FRQ3:** The tool must be able to query NFR information from a Knowledge Repository based on the search for a particular term. The provided NFR information must be as much complete as possible. It must include all the interdependencies related to a given element and its occurrence among parent NFRs whether applicable.
- **NFR1:** The tool must provide a graphic visualization that illustrates the result of querying a Knowledge Repository for NFR knowledge. The graphic visualization must illustrate all the information related to a query result, including every element and its relationships. Also, the graphic display should let the end-user navigate throughout the generated illustration.

Within the NDR Framework, the tool is intended to be used as a supporting system for software engineers during the elicitation and modeling of NFRs. We believe that the knowledge provided by the tool can be queried at any time by a software engineer working on the development of a new software system for a particular scenario. For instance, a software engineer may utilize the NDR Tool to identify potential alternatives for *satisficing* NFRs that apply for a needed scenario. Since the tool demonstrates all the possible options for *satisficing* a particular NFR, a software engineer may be able to choose the ones that apply to the required scenario, meeting stakeholders' expectations. Additionally, a software engineer may be able to identify new needed NFRs based on the trade-offs demonstrated by the NDR Tool. As the tool illustrates the conflicts and synergies among NFRs, it may help software engineers to obtain a solid idea of the consequences involved in *satisficing* a particular NFR.

Consequently, our approach can be employed as a reference guide regarding NFRs. We aim to keep a solid knowledge base regarding domain-free NFR information as a short-term goal. However, we envision to expand our knowledge base to domain-specific NFR information in the near future.

The following sections illustrate the architecture of our tool and the utilized technologies. Also, we demonstrate how our solution addresses the mentioned major requirements by emphasizing its main features. Lastly, we discuss the current limitations of the NDR Tool.

## 4.2 Architecture

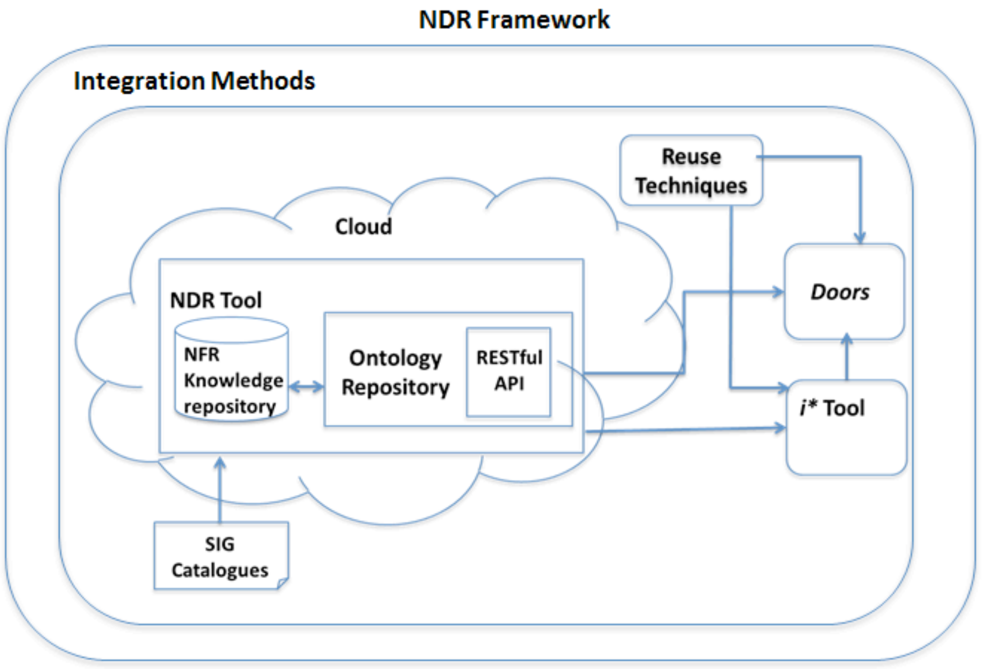


**Figure 18 NDR Framework's architecture overview**

Figure 18 demonstrates a high-level overview of the NDR Framework's architecture, which is the primary context of our proposed approach. The NDR Tool represents a fundamental part of the framework.

The tool is designed to centralize all the features involving NFR information, including knowledge extraction and creation, querying, and visualization. Therefore, approaches proposed in the NDR Framework such as Integration Methods and Reuse Techniques are dependent on the functionality of the NDR Tool.

We designed the NDR Tool to be deployed in a Cloud environment. The system has a full implemented RESTful API, characterizing the behavior of a *Web Service*. The motivation behind this design choice is associated to the application's extensibility. Consequently, the tool can provide a greater integration with third-party modeling applications. Also, smart applications can communicate with the NDR Tool in order to retrieve NFR knowledge information on demand. We envision that in the near future, the NDR Tool can be utilized by self-adaptive systems as a resource for supporting real-time decisions regarding NFRs.

Zareian et al. [37] proposed a data-oriented approach for analyzing and provisioning application performance in Cloud environments. We believe that the NDR Tool can be integrated into this framework to support *satisficing* Performance as a type of NFR. Hence, working as a *Web Service*, the tool can provide real-time needed knowledge for *satisficing* Performance in Cloud for a particular application based on the produced analysis of the K-Feed framework.

We also believe that the NDR Tool can be utilized towards real-time big data analysis. Khazaei et al. [38] demonstrates an approach for big data analysis regarding smart transportation. Since big data studies usually involve clustering techniques in a shared Cloud environment, the NDR Tool can provide NFR knowledge regarding possible scaling decisions on demand.

Additionally, we designed the tool based on two main repositories: Knowledge and Ontology. The Knowledge Repository aggregates information regarding the knowledge evolution associated to a particular ontology. On the other hand, the Ontology Repository is responsible for storing different ontology definitions. Our intention behind this design is to keep the NDR Tool modular and able to handle multiple ontologies in the future.

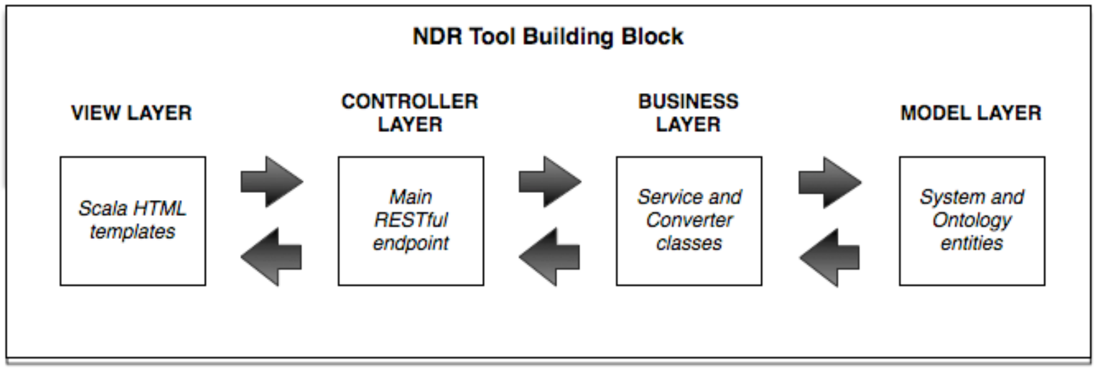


**Figure 19 NDR Tool's architecture overview**

Figure 19 illustrates a more specific detailed level of the NDR Tool's architecture, demonstrating its layers. In this work, Java[5] is the primarily utilized language, and the implementation mainly follows the standard *Model, View*, and *Controller* (MVC) pattern [39] with some customization in order to fulfill our requirements.

As *Models*, we define every essential entity in the system. They vary from considerably simple classes such as *User* and *NFRCatalog* to more enhanced implementations including NDR Ontology-related classes such as *NFRSoftgoal* and *OperDecomposition*. Essentially, most of the *Models* are used as object instances throughout our application's lifecycle.

---

[5] https://www.oracle.com/java/index.html

Our *Controller* layer is mainly defined by a *Web Request* handler class. It follows the standards of a *RESTful*[6] endpoint. As the NDR Tool is designed to be deployed in a Cloud environment, the Controller layer works as a *Web Service*. Additionally, the *Controller* layer is mostly responsible for linking the *View* layer with the Business layer.

The logic performed by the application defines the *Business* layer. *Services* and *Converters* are the most common classes implemented in this layer. Basically, *Services* are responsible for handling the necessary logic regarding a specific *Web Request*. It receives the required information from the *Controller* layer and performs the appropriate method calls. Similarly, *Converters* also control the logic associated with a particular need. However, the use of *Converters* is aimed towards the first step of knowledge creation only, which is associated with the extraction of information. Since creating knowledge expresses a complex task, we decided to separate its logic over *Service* and *Converter* classes. As a result, *Service* classes have more responsibilities in an overall context, although both *Service* and *Converter* classes are necessary for handling knowledge creation.

The *View* layer represents every *Web Page* of our proposed solution. They are the entry point of the interaction between the end-user and the system. Also, they are responsible for illustrating every output provided by the tool, including the graphic representation of NFR knowledge once requested. These *Web Pages* were developed

[6] https://www.w3.org/2001/sw/wiki/REST

using Scala[7] templates. Scala facilitates the reuse of object-oriented standards at a coding level, generating pure HTML as an output on the client side.

## 4.3 Main utilized technologies

The NDR Tool applies different technologies to fulfill its operation. To keep our proposed solution suitable to an academic environment and attract as many collaborators as possible, we adopted technologies that match certain criteria such as open-source implementation and active community support. Therefore, we identified the following frameworks as our primary utilized technologies:

**PlayFramework**

PlayFramework [40] is a Web framework that can be utilized in Java and Scala projects. It provides a range of techniques that facilitate the coding, compiling, and building of a project.

We applied the PlayFramework as the core of our system. The entire MVC architecture previously described is handled by this technology. PlayFramework guarantees the flow among the system layers, manipulating every web request and response.

[7] http://www.scala-lang.org/

### Apache Jena

Apache Jena [41] is a framework that provides functionalities for dealing with Semantic Web needs at a Java programming level. The following Figure 20 illustrates the architecture of Apache Jena:

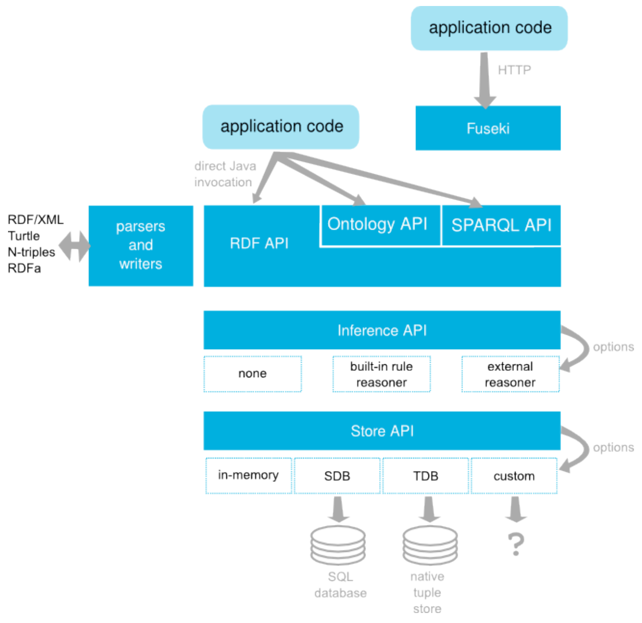


**Figure 20 The architecture of Apache Jena** [41]

Mainly, we exploit the *Application Programming Interfaces* (APIs) from Apache Jena to support our Ontology Repository needs. These APIs provide an abstraction that facilitates the information manipulation within an ontology. Instead of dealing directly with pure OWL and RDF/XML, a software developer can treat ontology elements as Java Objects, facilitating operations such as addition and removal of individuals. Also, these APIs allow the execution of SPARQL queries directly from source-code. Hence, every task performed by the NDR Tool regarding knowledge importation and search is assisted by Apache Jena at an object-oriented level.

### Apache Fuseki

Similarly to Apache Jena, Apache Fuseki [42] also operates towards the ontology assistance. However, instead of defining a framework, Apache Fuseki characterizes a SPARQL server.

As also demonstrated in Figure 20, Apache Fuseki can be combined with Apache Jena. Therefore, we decided to employ Apache Fuseki as the primary component of our Knowledge and Ontology Repositories. The server can handle multiple ontologies at the same time and execute SPARQL queries over the resources and individuals of each ontology. Apache Fuseki also follows a *RESTful* implementation. Consequently, it exposes its main operations as *Web services* endpoints. The NDR Tool invokes these endpoints to manipulate the existent information regarding a particular ontology.

### GraphViz

Graphviz [43] is a framework for graphical visualization. It allows the drawing of diagrams by interpreting elements defined using the DOT language. The following Figure 21 illustrates an example of DOT language syntax on the left side and its graphical output on the right side:

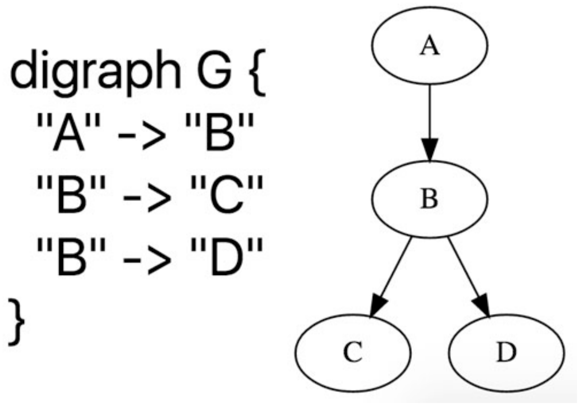


**Figure 21 DOT Language syntax and output**

The NDR Tool employs the Graphviz as the primary component for representing knowledge graphically. After every search on the Knowledge Repository, the tool converts the query result into DOT language syntax and produces the graphical output.

## 4.4 Importing NFR Knowledge

Importing Knowledge into the NDR Tool consists of a three-step process: Information Extraction, Knowledge Conversion, Ontology Update. Figure 22 illustrates the sequence of these main phases:

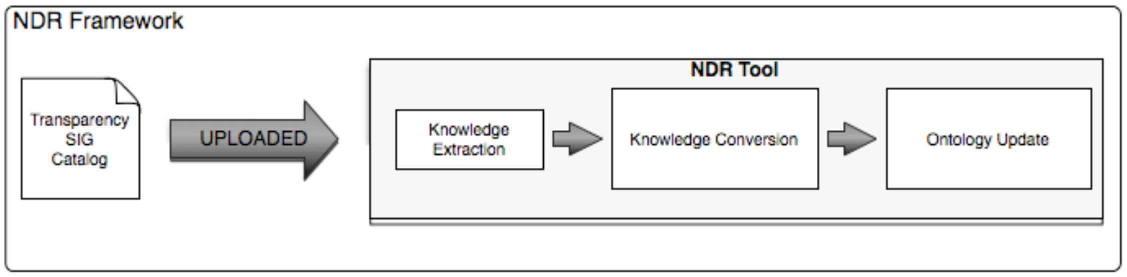


**Figure 22 NFR Knowledge importation phases**

Furthermore, to accurately represent the behavior of this process, we demonstrate these three primary stages decomposed into multiple activities triggered by a system-user interaction flow. Figure 23 expresses this representation:

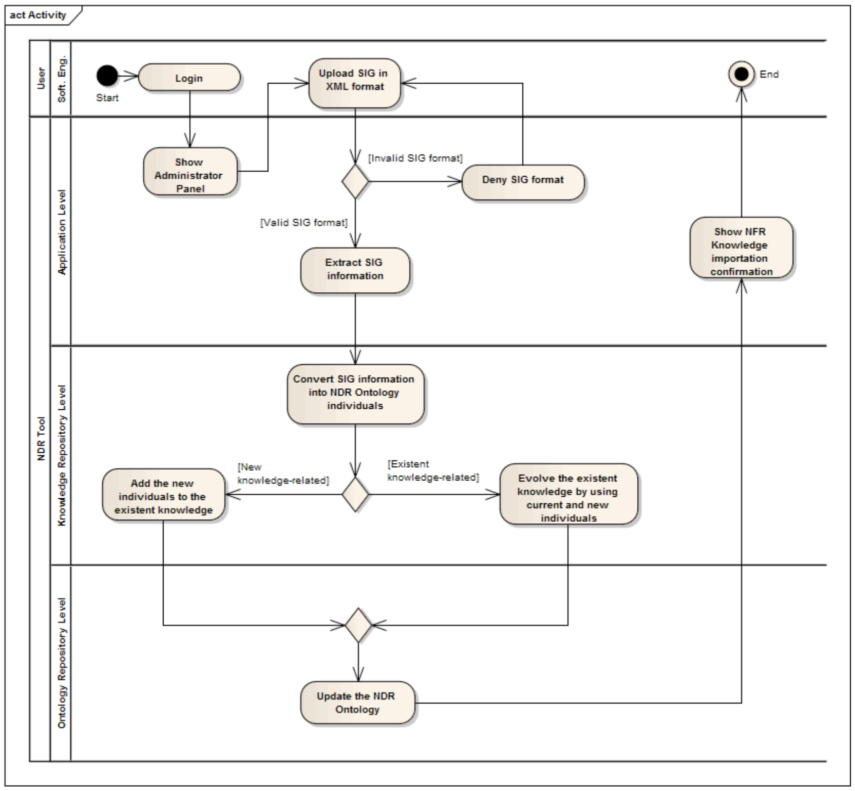

act Activity
User
Soft. Eng.
NDR Tool
Application Level
Knowledge Repository Level
Ontology Repository Level
Start
Login
Upload SIG in XML format
End
Show Administrator Panel
[Invalid SIG format]
Deny SIG format
[Valid SIG format]
Extract SIG information
Show NFR Knowledge importation confirmation
Convert SIG information into NDR Ontology individuals
[New knowledge-related]
[Existent knowledge-related]
Add the new individuals to the existent knowledge
Evolve the existent knowledge by using current and new individuals
Update the NDR Ontology

**Figure 23 NFR Knowledge importation Activity Diagram**

To fully explain every activity involved in this process, we propose the given scenario: A software engineer as an administrator user needs to import a SIG containing knowledge related to the NFR of Transparency. After performing the login into the system, the user uploads the artifact in XML format. For this particular example, both the SIG and its XML representation were generated using RE-Tools (StarUML) due to our expertise with this modeling tool. However, the central idea is to keep the NDR Tool capable of handling XML representations from multiple modeling tools. Figure 24 and Figure 25 demonstrate both the graphical and a partial XML representation of a SIG containing Transparency knowledge:

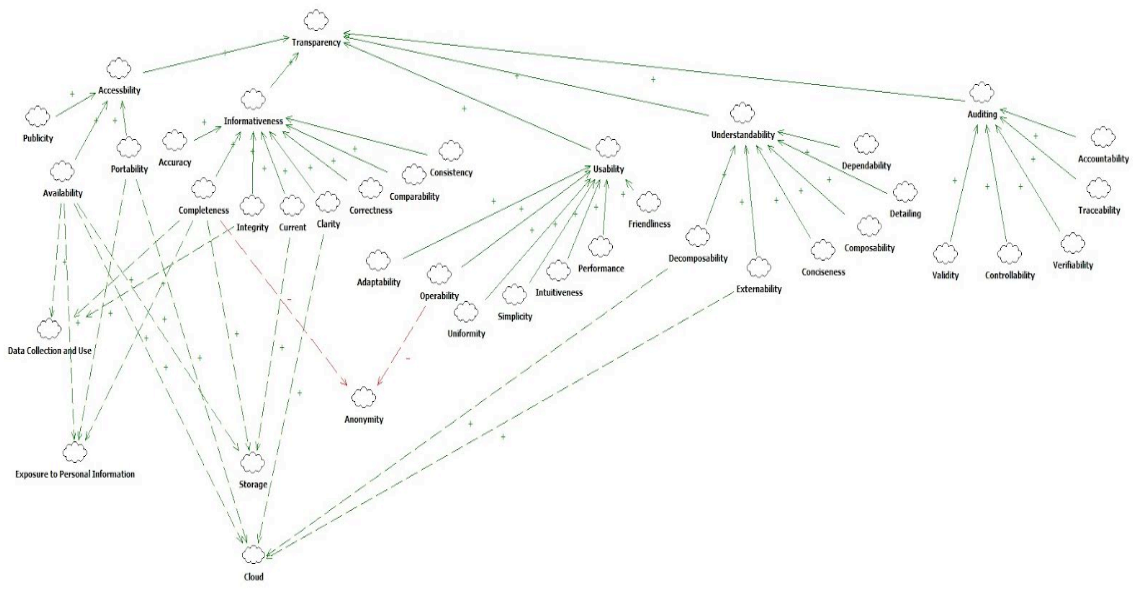


**Figure 24 SIG representing the NFR of Transparency**

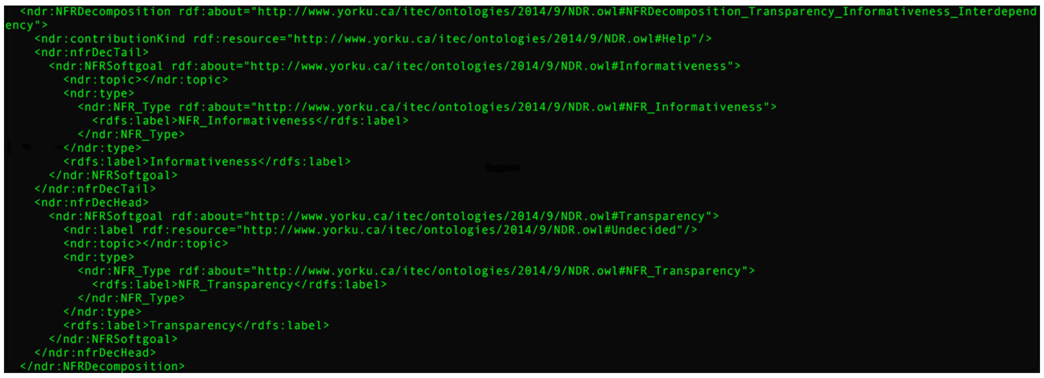
```
<UML:Model xmi.id="UMLProject.1">¬
  <UML:Namespace.ownedElement>¬
    <UML:Package xmi.id="UMLPackage.2" name="Transparency package" visibility="public"
isSpecification="false" namespace="UMLProject.1" isRoot="false" isLeaf="false" isAbstract="false">¬
      <UML:Namespace.ownedElement>¬
        <UML:Class xmi.id="UMLClass.3" name="Transparency" visibility="public" isSpecification="false"
namespace="UMLPackage.2" supplierDependency="UMLDependency.9 UMLDependency.10 UMLDependency.11
UMLDependency.30 UMLDependency.52" isRoot="false" isLeaf="false" isAbstract="false" isActive="false"/>¬
        <UML:Class xmi.id="UMLClass.4" name="Availability" visibility="public" isSpecification="false"
namespace="UMLPackage.2" clientDependency="UMLDependency.20 UMLDependency.35 UMLDependency.36
UMLDependency.40 UMLDependency.46" isRoot="false" isLeaf="false" isAbstract="false" isActive="false"/>¬
        <UML:Class xmi.id="UMLClass.5" name="Accessbility" visibility="public" isSpecification="false"
namespace="UMLPackage.2" clientDependency="UMLDependency.9" supplierDependency="UMLDependency.20
UMLDependency.29" isRoot="false" isLeaf="false" isAbstract="false" isActive="false"/>¬
        <UML:Class xmi.id="UMLClass.6" name="Informativeness" visibility="public" isSpecification="false"
namespace="UMLPackage.2" clientDependency="UMLDependency.10" supplierDependency="UMLDependency.21
UMLDependency.22 UMLDependency.31 UMLDependency.32" isRoot="false" isLeaf="false" isAbstract="false"
isActive="false"/>¬
        <UML:Class xmi.id="UMLClass.7" name="Usability" visibility="public" isSpecification="false"
namespace="UMLPackage.2" clientDependency="UMLDependency.30" supplierDependency="UMLDependency.33"
isRoot="false" isLeaf="false" isAbstract="false" isActive="false"/>¬
```

**Figure 25 Partial XML format of the SIG representing the NFR of Transparency**

Once the SIG catalog is uploaded into the system and has a valid format, the NDR Tool performs a series of actions in order to extract the existent information and convert it into NDR Ontology individuals. Firstly, after receiving the request from the *Controller* layer, methods within a *Converter* class parse every XML element and store them in memory as new *Model* instances, which characterizes new NDR ontology individuals.

Subsequently, the NDR Tool verifies if the recently created elements already exist in the Knowledge Repository. If they previously exist as NFR Knowledge, instead of keeping them as new NDR ontology instances in memory, the system disregards these elements and utilizes the already existent ones for the knowledge addition process. This behavior expresses how the NDR Tool handles the knowledge evolution by reutilizing existent NFR Knowledge and associating them with the identified new ontology individuals when adding new information into the Knowledge Repository. As a consequence, this activity may lead to new interdependencies among existent and new knowledge. By following our mentioned scenario, *Traceability* is listed as a result of a *Decomposition* in the SIG of *Transparency*. If the Knowledge Repository contains information about *Traceability*, the NDR Tool will automatically associate *Transparency* with all the knowledge related to *Traceability* after successfully importing the SIG.

Finally, after handling the knowledge addition process, the NDR Tool updates the ontology baseline on the Ontology Repository. In this particular scenario, the system will update the NDR Ontology by adding all the provided knowledge associated to Transparency. This process occurs on the SPARQL server, and it is performed in a OWL level. Figure 26 demonstrates the result of the ontology update process. A *Decomposition* between *Transparency* and *Informativeness* is described in OWL and now part of the NDR Ontology as a new individual. This same *Interdependency* is illustrated in Figure 24, which expresses a SIG representing *Transparency* as NFR. Lastly, when the NDR Ontology is successfully updated, and the process is eventually completed, the tool shows a confirmation to the end-user.



```
  <ndr:NFRDecomposition rdf:about="http://www.yorku.ca/itec/ontologies/2014/9/NDR.owl#NFRDecomposition_Transparency_Informativeness_Interdepend
ency">
    <ndr:contributionKind rdf:resource="http://www.yorku.ca/itec/ontologies/2014/9/NDR.owl#Help"/>
    <ndr:nfrDecTail>
      <ndr:NFRSoftgoal rdf:about="http://www.yorku.ca/itec/ontologies/2014/9/NDR.owl#Informativeness">
        <ndr:topic></ndr:topic>
        <ndr:type>
          <ndr:NFR_Type rdf:about="http://www.yorku.ca/itec/ontologies/2014/9/NDR.owl#NFR_Informativeness">
            <rdfs:label>NFR_Informativeness</rdfs:label>
          </ndr:NFR_Type>
        </ndr:type>
        <rdfs:label>Informativeness</rdfs:label>
      </ndr:NFRSoftgoal>
    </ndr:nfrDecTail>
    <ndr:nfrDecHead>
      <ndr:NFRSoftgoal rdf:about="http://www.yorku.ca/itec/ontologies/2014/9/NDR.owl#Transparency">
        <ndr:label rdf:resource="http://www.yorku.ca/itec/ontologies/2014/9/NDR.owl#Undecided"/>
        <ndr:topic></ndr:topic>
        <ndr:type>
          <ndr:NFR_Type rdf:about="http://www.yorku.ca/itec/ontologies/2014/9/NDR.owl#NFR_Transparency">
            <rdfs:label>NFR_Transparency</rdfs:label>
          </ndr:NFR_Type>
        </ndr:type>
        <rdfs:label>Transparency</rdfs:label>
      </ndr:NFRSoftgoal>
    </ndr:nfrDecHead>
  </ndr:NFRDecomposition>
```

**Figure 26 A Decomposition between Transparency and Informativeness in OWL**

Currently, the NDR Tool uses inference capabilities provided by the use of ontologies during the association of information that is being imported with already existent data in the knowledge base. If the information that is being imported already exists in the knowledge base, the tool verifies and performs inferences on each provided element. For instance, if *Usability* already exists in the knowledge base as an *Operationalizing Softgoal* related to an *Operationalization* of *Transparency*, and a new SIG representing *Usability* as a NFR Softgoal is imported, the tool will automatically redefine the already existent version of *Usability* to the one that was just imported. In other words, the *Operationalizing Softgoal* of *Usability* will become a *NFR Softgoal* of *Usability*. This behavior is a result of an inference rule that defines that an *Operationalizing Softgoal* may become a *NFR Softgoal* during the importation of new information. This rule is based on the constraints of the NFR Framework. Additionally, this scenario would also produce a new inference rule stating an association between *Transparency* and *Usability*. Consequently, the tool can assert the occurrence of *Usability* among the alternatives to *satisfice Transparency*.

We aim to expand the use of a reasoner for automatically inference identification in future versions of the tool to deal with the support of domain-specific knowledge. Such a mechanism will provide the ability to automatically identify relationships between different individuals representing knowledge regarding the same NFR according to distinct domains. In an ontology level, we aim to keep every individual representing a

NFR related to a particular domain as a new subclass of the individual that represents the same domain-free NFR.

## 4.5 Searching NFR Knowledge

After logging into the system, a user may search for NFR knowledge by following one of the two possibilities: (i) choose one of the NFRs from the drop-down menu, or (ii) search for any element by providing a specific term in the search box. Figure 27 demonstrates the interface where a user may choose one of the two possibilities:

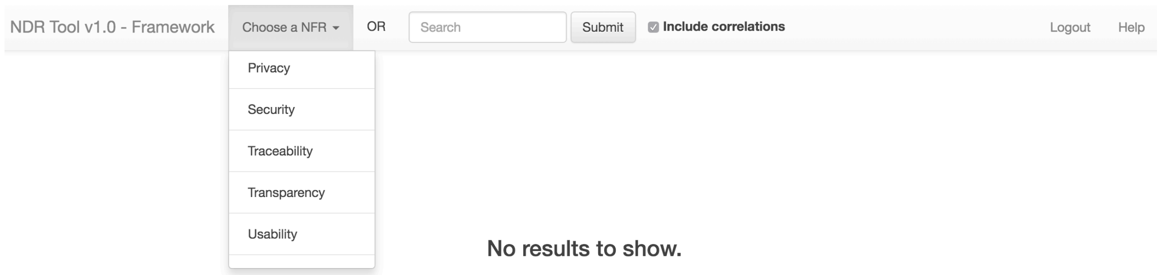
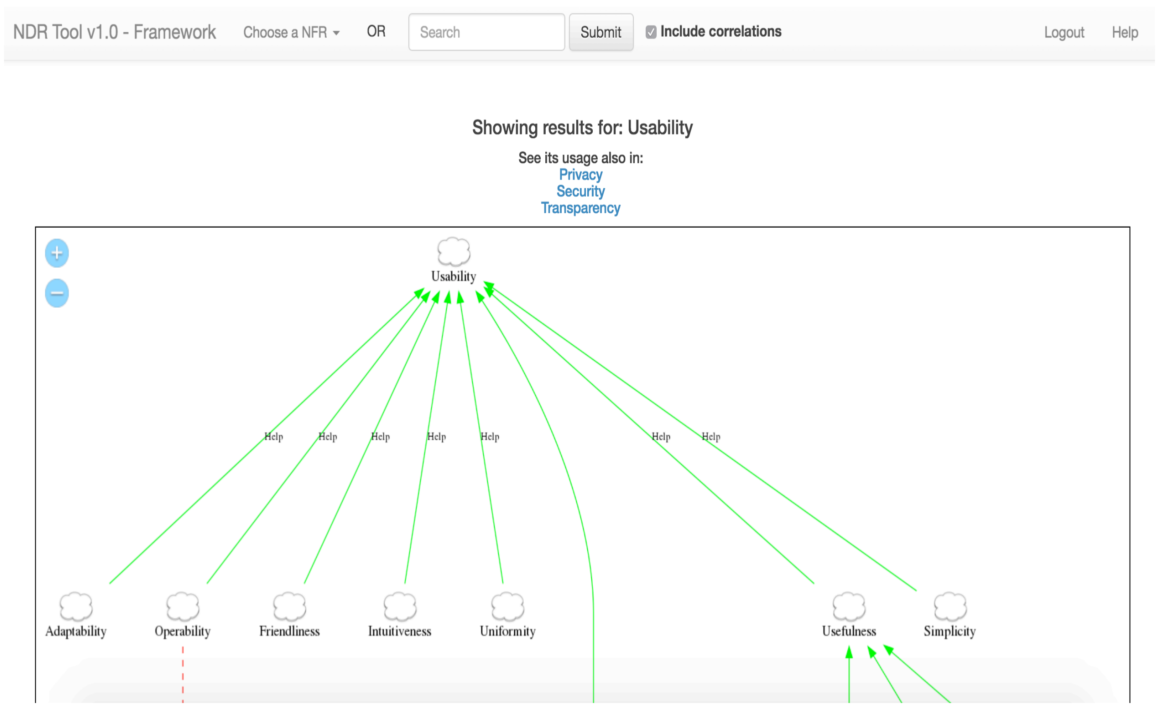


**Figure 27 NDR Tool: Search for NFR Knowledge**

The available NFRs in the drop-down menu represents the major NFRs available in the Knowledge Repository. By selecting one of them, the system will search and show all the relevant information associated with the chosen NFR. This process occurs as the following:

After gathering all the necessary knowledge related to a provided input, the tool prepares a graphical representation for the end-user. By using GraphViz as a drawing framework, the system transforms the collected knowledge into a SIG catalog on demand. Moreover, the NDR Tool follows the same notation proposed by the NFR

Framework in order to guarantee a consistent understanding. Also, the system provides the occurrences for the queried element. Figure 28 expresses the graphical output generated in real-time by the NDR Tool based on a provided input. Above the graphical representation, it is possible to visualize and understand that the queried element, *Usability* in this particular case, is also associated with *Privacy*, *Security*, and *Transparency*. If the user selects one of these associated NFRs, the system will once again generate in real-time another graphical output for the chosen item.

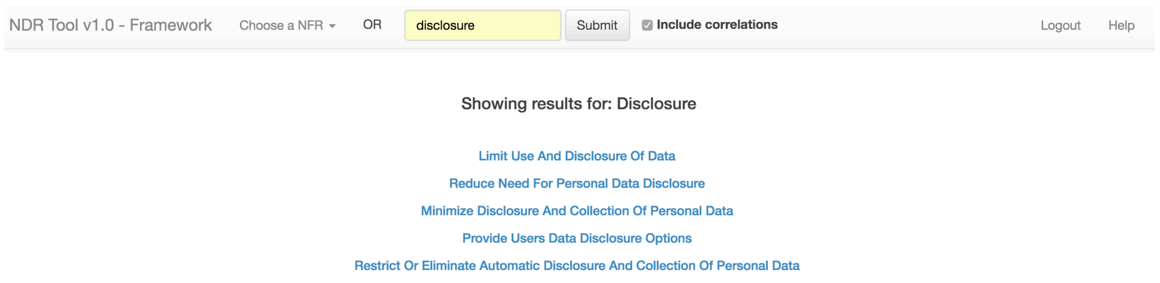


Showing results for: Usability

See its usage also in:
Privacy
Security
Transparency

**Figure 28 NDR Tool: Graphical Output**

On the other hand, by entering a search term, the NDR Tool will perform an extensive search for the provided term throughout the entire Knowledge Repository. It is noteworthy to mention that this broad search is not restrictive. In other words, the system will search for every element that contains the provided term, and not only for those elements that match the provided term. The following Figure 29 illustrates this mentioned behavior:

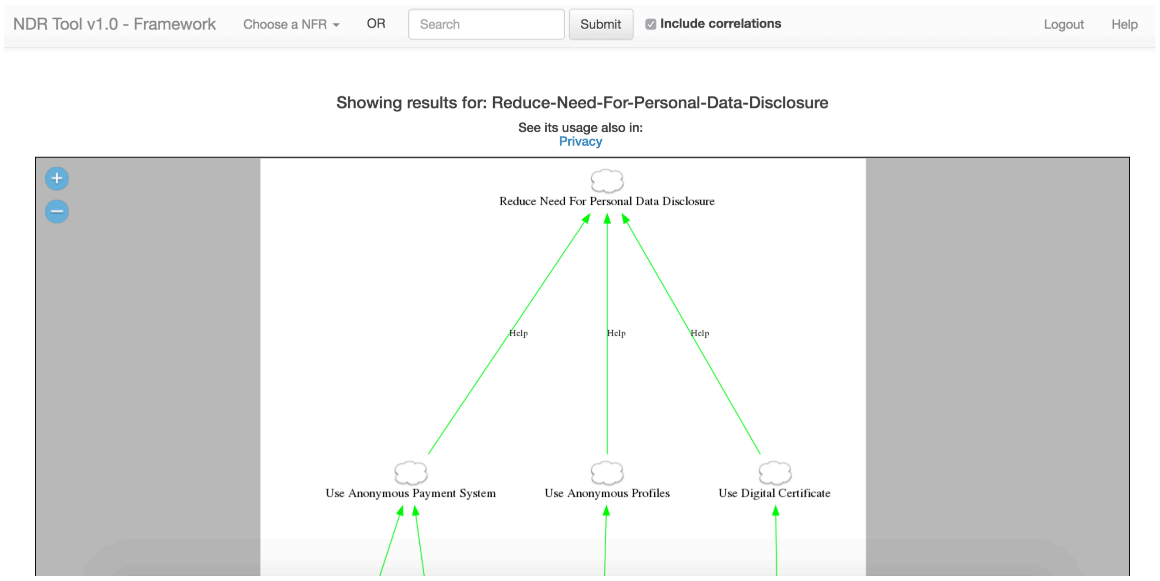


Showing results for: Disclosure

Limit Use And Disclosure Of Data
Reduce Need For Personal Data Disclosure
Minimize Disclosure And Collection Of Personal Data
Provide Users Data Disclosure Options
Restrict Or Eliminate Automatic Disclosure And Collection Of Personal Data

**Figure 29 NDR Tool: Non-restrictive Search**

Once a user selects one of the search possibilities, the system identifies the search element and produces a series of SPARQL queries. These queries are then executed on the SPARQL server side in a recursive manner. Hence, the system can retrieve every interdependency associated with each element related to the search one. By the moment the tool finishes performing the necessary queries, the drawing process begins. As a result, the system outputs a partial graphic visualization starting from the chosen search possibility. If necessary, the user can still select among the NFRs associated with the target possibility. Figure 30 illustrates the partial graphic output originated from a search possibility selection. In this particular case, *Reduce Need For Personal Data Disclosure* was the selected option among the listed results for *Disclosure*.



Showing results for: Reduce-Need-For-Personal-Data-Disclosure

See its usage also in:
Privacy

**Figure 30 NDR Tool: Partial graphic output resulted from search**

It is important to mention that the search ability provided by the NDR Tool works on different levels of granularity. In other words, a software engineer can search for any NFR related capabilities, ranging from early refinements to a very specific refinement level. This characteristic provides a versatile mechanism for scenarios where the granular level of a needed NFR related solution is unknown. The previous Figure 30 demonstrates a search result at a specific granularity level.

Also, the search capability can provide inferences among available NFRs in the Knowledge Repository. As a consequence of demonstrating the occurrences of a queried element, the NDR Tool may infer existent correlations when a parent NFR is utilized as a query term. Figure 31 illustrates this scenario. After searching for *Security*, the tool graphically outputs the whole existent knowledge associated with *Security* and also denotes its occurrences in other NFRs. At this point, in this particular scenario, a software engineer can assume that *Security* correlates with *Privacy*, *Traceability*, and *Usability*.

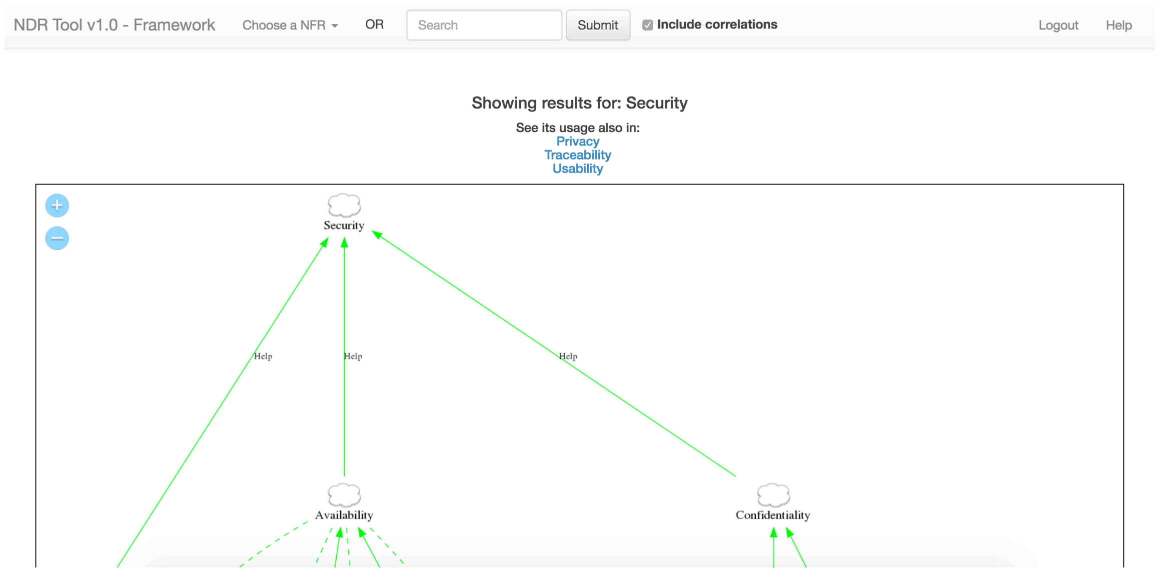


**Figure 31 NDR Tool: Correlation inferences**

To find out whether the correlation produces a synergy or a conflict, a software engineer can simply click on one of the listed NFRs and visually verify the nature of the correlation. Figure 32 demonstrates a partial graphical output generated after selecting *Privacy* from the list of occurrences. It is possible to notice the details of the correlation between *Security* and *Privacy*. In this case, there are multiple correlations with a negative nature, characterizing a potential conflict among *Security* and *Privacy*.

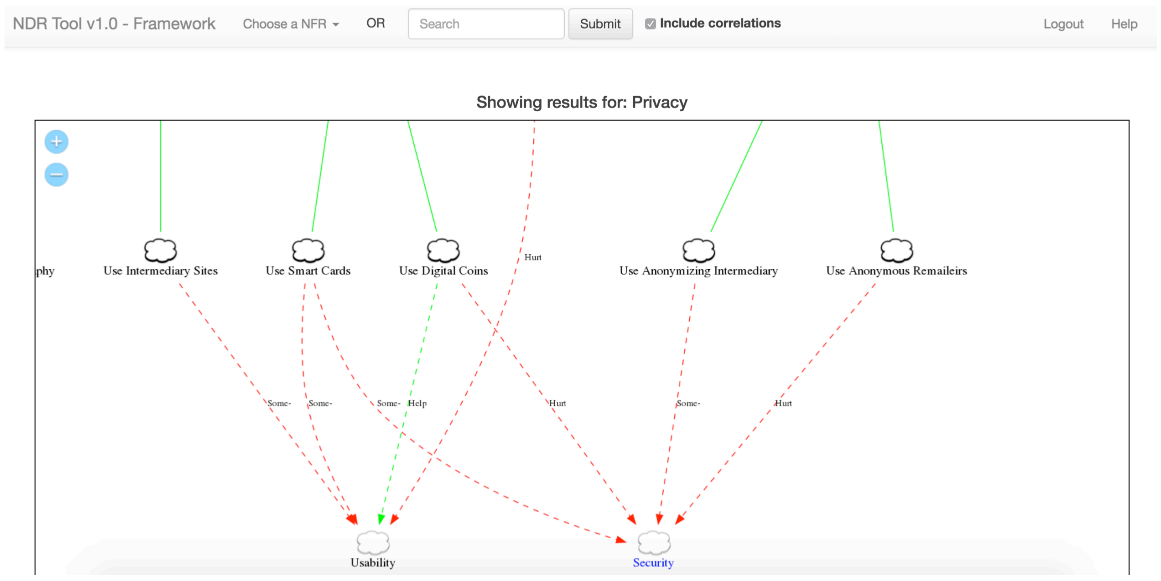


**Figure 32 NDR Tool: Conflict between Security and Privacy**

To adequately emphasize the search for NFR Knowledge among the levels of the NDR Tool, we demonstrate the following activity diagram. Figure 33 illustrates the previously specified behavior of the NDR Tool on handling the search for knowledge user interaction:

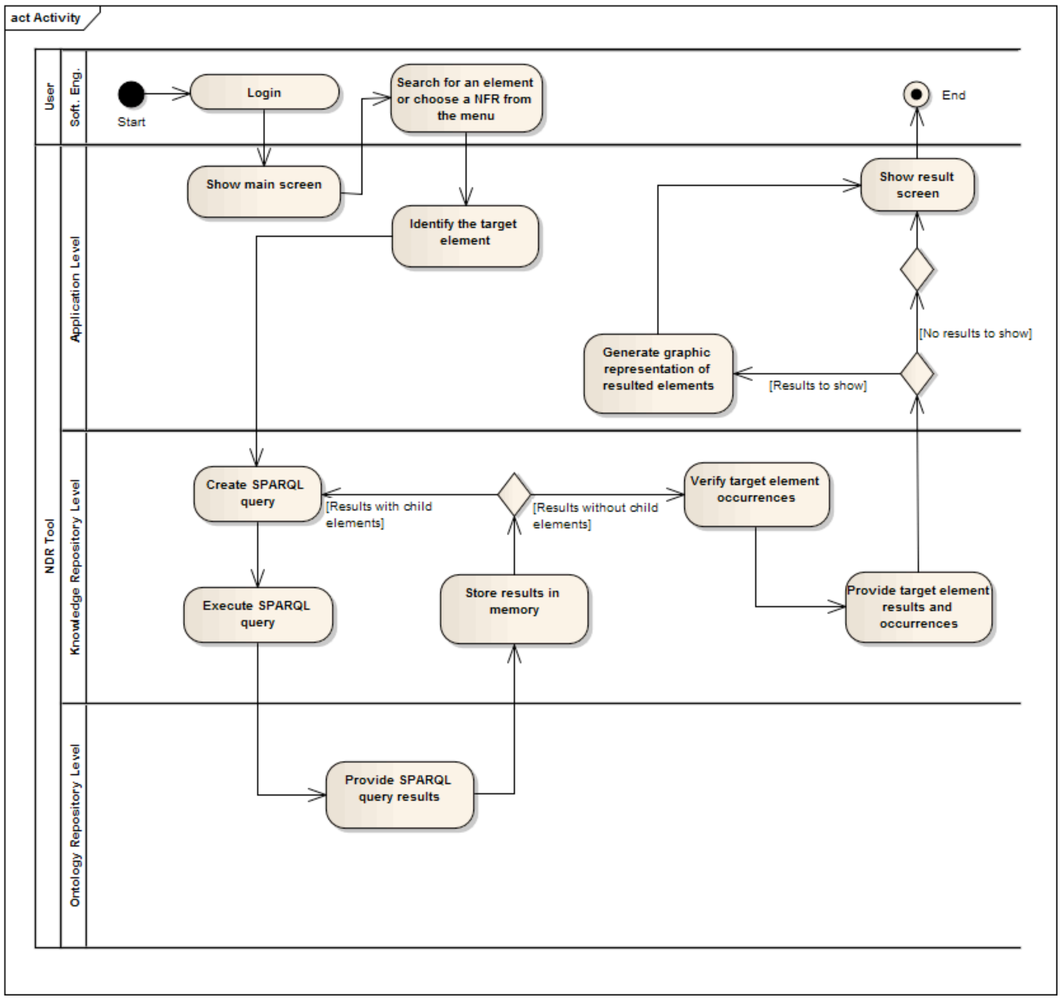

act Activity
User
Soft. Eng.
NDR Tool
Application Level
Knowledge Repository Level
Ontology Repository Level
Start
Login
Search for an element or choose a NFR from the menu
End
Show main screen
Identify the target element
Show result screen
[No results to show]
Generate graphic representation of resulted elements
[Results to show]
Create SPARQL query
[Results with child elements]
[Results without child elements]
Verify target element occurrences
Execute SPARQL query
Store results in memory
Provide target element results and occurrences
Provide SPARQL query results

**Figure 33 Search for Knowledge Activity Diagram**

## 4.6 Current Limitations

As a preliminary implementation, the present version of the NDR Tool includes a few limitations. Currently, the system does not allow any exportation of information. We first focused our efforts on having a stable knowledge explorer version of the tool. However, we envision to implement a generic exportation mechanism in a evolved version of the system.

Also, the existing NFR knowledge importation process requires an automated mechanism for knowledge recognition. Currently, the system assumes that every SIG provided as an input to the import process contains a valid NFR information. Hence, the tool still relies on a manual knowledge validation as a pre-step before the importation. We aim to fulfill this requirement by exploring the *Axioms* and *Inferences* features provided within the OWL specification for a particular ontology. *Axioms* and *Inferences* provide implicit information about ontology elements generated by a reasoner. Therefore, to overcome this challenge, we believe in the possibility of implementing an automated reasoner that can exploit the implicit knowledge within an ontology and automatically validate new individuals before its importation.

Despite the fact that the first version of the NDR Tool is already deployed and running in a cloud environment, its availability is yet limited for academic purposes only. We envision to have the tool available for both academic and professional areas in a near future. However, at this moment, we are still aiming at a robust implementation version and using the educational field as our testable environment.

Lastly, the present implementation of the NDR Tool is designed to work with domain-free NFR knowledge only. We believe that the system is already capable of working with multiple domains. However, this feature still has to be extensively validated by our test cases before being implemented in a production environment. Hence, we assume that a future version of the tool will certainly include this feature.

## 4.7 Summary

In this chapter, we presented our proposed approach for reusing NFR knowledge. We introduced the NDR Tool as our major developed work. Subsequently, we emphasized its architecture by detailing the information flow through the system layers and describing every associated component. We also reported the key technologies employed in the processes within the tool.

Furthermore, we demonstrated two usage scenarios that fully covers the previously stated requirements. We described different situations for both scenarios to facilitate their understanding. We also included Activity Diagrams in order to express the decisions associated with every process that occurs on the system side.

Lastly, we emphasized the current limitations of our proposed approach. As a preliminary version, the NDR Tool has a few restrictions regarding its availability, features, and domain coverage.

# Chapter 5
# Evaluating the Tool

This chapter emphasizes the methodology for evaluating the applicability of our proposed approach. We denote our main hypothesis and the associated variables followed by the strategy and design of our research experiment. We also describe the characteristics related to the participants in this study, the expected outcomes, and the proposed scenario.

Easterbrook et al. [44] suggests a controlled experiment as an empirical evaluation method for software engineering research. It is intended to verify the cause and effect relationship among independent and dependent variables in a testable hypothesis. Consequently, to assess whether our approach can facilitate software engineers to elicit more and better NFRs, we conducted a controlled experiment with human participants.

In this proposed experiment, the participants performed the role of a software engineer and were asked to elicit and model NFRs under a particular scenario that involved the development of a software system. The NFRs had to be modeled according to the notation specified by the NFR Framework. Hence, we provided multiple workshops as training sessions for every participant of this study before the experiment execution.

As an outcome, multiple SIGs were modeled by each participant. Participants were asked to produce SIGs for every NFR they deemed necessary and evaluate all possible *Operationalizations* as well as *Correlations* they could identify. They were not requested to choose one alternative over another or to produce a single solution. Instead, they were carefully instructed that the more *correct* alternatives regarding *Operationalizations* and *Correlations* they could identify the *better* their achieved performance would be.

After gathering the results, we compared each developed SIG to a positive control. This positive control was produced based on the literature on SIG catalogs and represented the elicited NFRs we were expecting for that particular scenario. The author of this thesis elicited and modeled every expected NFR, including its *Operationalizations* and *Correlations* and later submitted to Prof. Cysneiros to further check for missing or invalid options. Mainly, we focused the comparison on the number of *Operationalizations* and *Correlations* obtained for a specific NFR. We followed the notion that the higher the number of *Operationalizations*, the greater the chance of better *satisficing* a NFR. The same theory applies for *Correlations*, notably because it expresses possible synergies and conflicts among NFRs. Furthermore, following the idea that each project may demand different solutions for the same NFR, we wanted to evaluate if a software engineer using the NDR tool would be able to recognize a more comprehensive set of alternatives for *satisficing* a NFR than a software engineer not using it. Evaluating if this software engineer would still be able to make to correct choice is a much more complex task and is outside the scope of this work.

## 5.1 Research hypotheses and variables

The central premise of the assessment performed in this thesis evaluates whether using the NDR Tool can help software engineers to elicit and model better NFRs. We designed a research experiment intended to assess the resulted NFR models of each participant. The observed measures involved in the assessment were the number of identified *Operationalizations* and *Correlations*. Therefore, in order to adequately represent our appraisal needs, we state the formal hypotheses:

- **H1:** There is a significant difference between the NDR Tool and NFR Catalogs regarding the identification of *Operationalizations.*
- **H0:** There is no difference between the NDR Tool and NFR Catalogs regarding the identification of *Operationalizations*.
- **H2:** There is a significant difference between the NDR Tool and NFR Catalogs regarding the identification of *Correlations*.
- **H0:** There is no difference between the NDR Tool and NFR Catalogs regarding the identification of *Correlations*.

Additionally, as a requirement for conducting a controlled research experiment within this thesis work, we determined independent and dependent variables to allow the analysis of a possible cause and effect relationship between them. We state it as the following:

- **Independent Variable:** Knowledge-assistance technique (NDR Tool or Pure NFR Catalogs).
- **Dependent Variable:** Number of identified *Operationalizations*; Number of identified *Correlations*.

## 5.2 Experiment Design and Strategy

To appropriately test our mentioned hypotheses, we designed a research experiment that follows a between-subject design. Mostly, we divided and randomly assigned the participants into two groups: experimental group and control group.

Participants in the experimental group were in charge of eliciting and modeling NFRs with the NDR Tool as a knowledge-assistance technique. On the other hand, subjects in the control group could only rely on pure NFR catalogs as a knowledge-assistance method for identifying and modeling NFRs. Pure NFR catalogs are represented by static image files following the SIG format. Both knowledge-assistance techniques held the same amount and type of knowledge regarding NFRs. Also, both groups strived to elicit and model NFRs under an identical software system development scenario.

The participants were students enrolled in the *Requirements Management* course, a 4th year course in the *School of Information Technology* at *York University, Toronto, Canada*. Participation was on a voluntary basis, and it would count as an assignment that would add up to 10 marks to the final exam grade depending on the achieved performance in finding valid *Operationalizations* and *Correlations*.

In regards to the expected expertise associated with the NFR Framework notation necessary for producing the output models, subjects of this experiment attended training sessions before participating in the study. We provided these practice sessions in a workshop manner, totalizing twelve non-consecutive hours of training for each participant. It is important to mention that participants were at the final two weeks of a *Requirements Management* course. Hence, they had already been exposed and trained in aspects related to eliciting and modeling requirements as well as the concepts of NFRs, methods to elicit and models them and techniques to link them to functional requirements.

Regarding the reliability of our proposed assessment, we based our experiment design in previous research studies [19], [22]. Cysneiros et al. [19] conducted a similar research experiment for evaluation of an approach for representing NFR knowledge within conceptual models. Likewise, Al Balushi et al. [22] carried a comparison between groups of participants in order to assess the applicability of an ontology-based tool for assisting the elicitation and prioritization of NFRs.

Finally, we ethically treated the participants in this study by assuring confidentiality. Unfortunately, we cannot assure total anonymity regarding the participants in this study since each subject received academic credit such as a bonus mark on the final exam regarding their participation in this experiment. However, we can guarantee that the demonstration of the findings of this study will not be traced and associated back to any participant.

## 5.3 Threats to Validity

In this section, we explain the possible factors that could affect the validity of our performed experiment and how we tried to address it as a concern.

**Threats to Internal Validity**

- **Maturation:** Not considered a threat for our study since each participant had to perform the tasks for the experiment only once.
- **Instrumentation:** We controlled this threat by providing identical conditions for every subject regarding tools and execution time. Also, we designed the experiment in a similar manner to an optional academic course assignment.
- **Biased subject selection:** This threat prevailed partially uncontrolled in our study since each target subject consisted of an undergraduate student attending a Requirements Management course. However, among this population, we randomly choose the eligible ones since the participation in the experiment was not mandatory.
- **Experimental mortality:** To avoid and control this threat, after providing the entire essential material through a web-portal, we let each subject perform the experiment in their timely manner until a specific due date.

- **Statistical Regression, Testing, and History:** Not applicable in this study.

**Threats to External Validity**

- **Interactions between selection biases and the independent variable:** This threat remained uncontrolled due to our target population in this study. Our findings may only represent a group of subjects consisted of undergraduate students of Information Technology.
- **Multiple treatment interference:** Not applicable in this study since there was no exposure to early treatments that could affect the responses of later treatments.
- **Reactive Testing:** Not applicable in this study.

## 5.4 Sampling

The target population for this experiment consists of undergraduate students enrolled in a *Requirements Management* course taught by the research supervisor of this thesis at *York University, Toronto, Canada*. However, not all the students enrolled participated in the study. The experiment was considered an optional assignment for the course, hence students could opt to not to do it. For the purpose of this research work, the sample size is consisted of 12 individuals, randomly divided into two groups of 6 participants each regarding the knowledge-assistance technique.

We expected that the age of the research subjects variated from 18 to 30 years old. However, as the age factor does not impact on our hypotheses analysis, we decided not to collect this information from the participants. Also, none of the participants were expected to be previously known to the researcher of this thesis.

## 5.5 Data Measurement and Collection

The data collection regarding this experiment was based on the outcomes of each participant. These were graphical representations of NFRs in the form of SIGs. The dependent variables based on which the two groups were compared include technical aspects expressed in the resultant SIGs. These were the number of correctly identified particular interdependencies. For this study, we considered the number of correctly identified *Correlations* and *Operationalizations* as these elements represent specialized refinements that are directly associated with a better *satisficing* of NFRs.

As a deliverable of this experiment, we asked the participants to submit their SIGs with a signed informed consent by email. In this manner, we could officialize the participation of every individual and gather the outcome models for comparison with the authoritative control and further analysis.

To measure response correctness, we compared each SIG produced by each participant with an authoritative one that we have developed based on previous studies performed by Prof. Cysneiros, representing our control sample. To gauge the similarity between the participant's model and the authoritative model, we counted the number of correctly identified *Correlations* and *Operationalizations* in the participant's response. A

*Correlation* or *Operationalization* had to be expressed in the authoritative model to be considered correct.

During the comparison of each participant's resultant model with our control sample, we also have taken into the account taxonomy variations regarding the name of elements used for *Correlations* and *Operationalizations*. For instance, an element described as "Use Voice" involved in a *Operationalization* in an outcome SIG, was considered as a correct answer although the same *Operationalization* was outlined in our control sample with a "Voice Recognition" labeled element. The same principle was applied for all the elements involved in *Correlations*. We decided to follow this method based on the fact that NFRs are usually defined in a subjective manner. Therefore, different terms can be used to emphasize the same solution for *satisficing* a particular NFR.

Additionally, it is important to mention that we conducted a blind evaluation regarding the outcomes of each participant. In other words, after gathering all the results provided by the subjects, we performed the comparison against our control sample disregarding the group of the target participant. To achieve this purpose, we arranged all the obtained results unitedly and conducted our analysis in a random order. The name of the participant and consequently the group which he/she belonged was kept hidden until all the evaluations were finished.

Despite the measures that we have taken to mitigate threats regarding the internal and external validity of our experiment, we have also performed post-experiment actions.

We have assured that every outcome provided by the participants was modeled according to the NFR Framework notation in order to make the comparison with the control sample possible. Also, among our initial population for this study, we have eliminated two participants due to the detection of mutual collaboration. The outcome of both participants included the same amount of elicited NFRs, the equivalent number of found *Operationalizations* and *Correlations*, and the identical taxonomy regarding the name of elements. As a result of this action, our sample size for this study was reduced to a total of 12 participants.

## 5.6 Proposed Scenario

In order to bring this research experiment as close to reality as possible, we suggested the participants a practical scenario for developing a software system. We wanted to create a scenario that would appeal to students as an interesting one to participate, yet at the same time be complex enough to demand a significant number of NFRs to be elicited and therefore could serve as a reliable scenario to test the effectiveness of our approach. Hence, we developed a hypothetical situation involving the development of a software system for an *Autonomous Taxi Service* company as the following:

*ATS* is a company that provides a riding service with driverless taxi vehicles. Its market share is worldwide, and their number of active users is exponentially growing. Therefore, the company aims to invest in a modern and innovative software system to adequate their business properly. The following TABLE 1 emphasizes the main expected functionalities of the system for the target business.

| # | Description | Actor |
|---|---|---|
| **FRQ1** | A customer must be able to order a taxi from any device with internet connection, including smartphones, tablets, and computers. | Customer |
| **FRQ2** | The system must be able to detect the nearest available vehicle and assign it to the current customer. | Geolocation Service |
| **FRQ3** | A customer must be able to create an account with personal information such as full name, email, telephone, and desired payment method and details. | Customer |
| **FRQ4** | A customer must be able to delete its account and unsubscribe the service at any time. | Customer |
| **FRQ5** | The system must handle credit card payments. Credit card information should be linked to a customer account. | Credit Payment Service |
| **FRQ6** | The system must keep track of customer destinations in order to identify and suggest alternative routes. | Geolocation Service |
| **FRQ7** | A customer must agree to terms and conditions of the service provided by the time the account is created. | Customer |
| **FRQ8** | The system must calculate the price for a specific trip by the request time. | Vehicle |
| **FRQ9** | A customer must be able to cancel its trip at any time. In case of trip cancellation, the system should direct the driverless car to the safest drop-off point. | Customer |
| **FRQ10** | The system must charge the customer by the end of the trip. | Vehicle |
| **FRQ11** | The system must track the live location of driverless vehicles for management purposes. | Geolocation Service |

| | | |
|---|---|---|
| **FRQ12** | The system must provide reports for management purposes. | Service Administrator |

**Table 1 Expected functionalities in the experiment scenario**

To appropriately represent the hypothetical client needs, we also provided a supporting software documentation. Figures Figure 34, Figure 35, and Figure 36 illustrate samples of the produced documentation. The full documentation can be visualized in Appendix A.

Additionally, it is important to mention that the identification of NFRs for this proposed scenario was part of the experiment. We expected this outcome from each subject. Therefore, our documentation does not provide information about NFRs as we left it for the participant's interpretation.

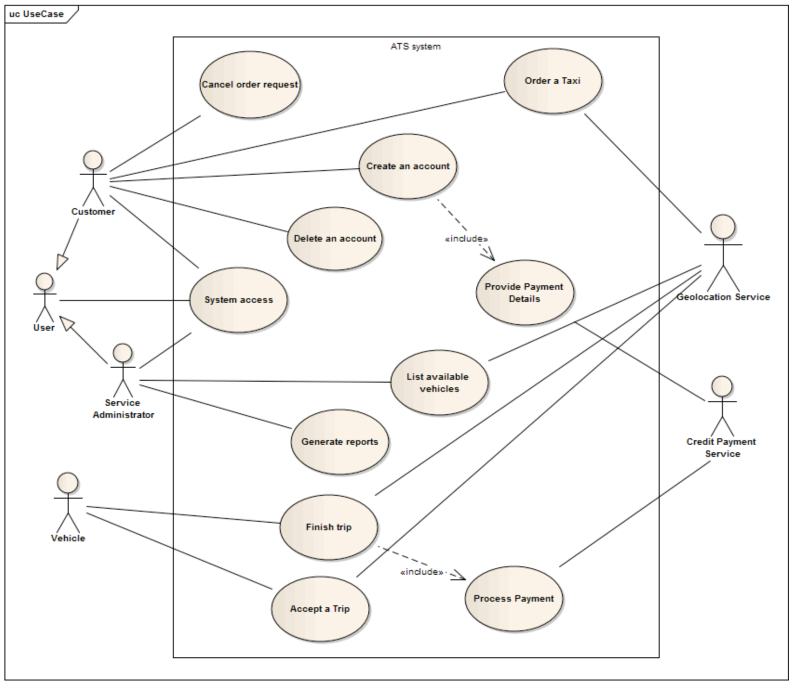


**Figure 34 Use Case for the proposed scenario**

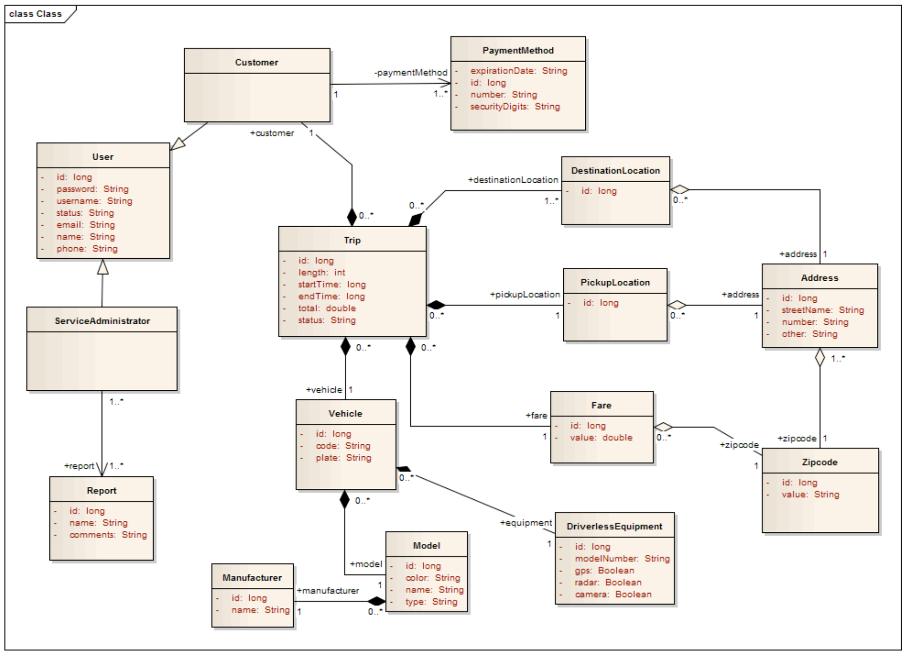


**Figure 35 Class Diagram for the proposed scenario**

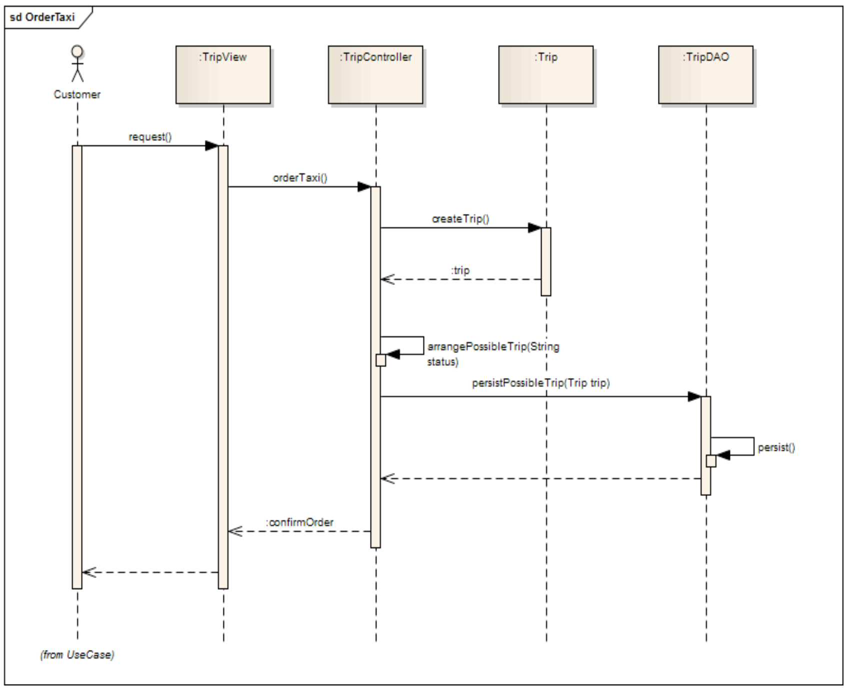


**Figure 36 Sequence Diagram for the proposed scenario**

Our positive control for the proposed scenario covered a set of expected NFRs, including *Privacy*, *Security*, *Traceability*, *Transparency*, *Usability*, and *Performance*. Figures Figure 37 and Figure 38 illustrate two SIGs that were part of this authoritative control. The full documentation regarding our control sample can be visualized in Appendix B of this thesis. Once more, it is important to emphasize that these SIGs do not aim at choosing one *Operationalization* over another. The goal is to represent the largest set of alternatives possible that could be used by a software engineer during a project.

The whole set of expected NFRs represented a total number of 52 *Operationalizations* and 28 *Correlations*. Every participant was able to elicit and model at least one of the expected NFRs expressed in the positive control. We demonstrate a comprehensive analysis of the findings of this study in the following chapter.

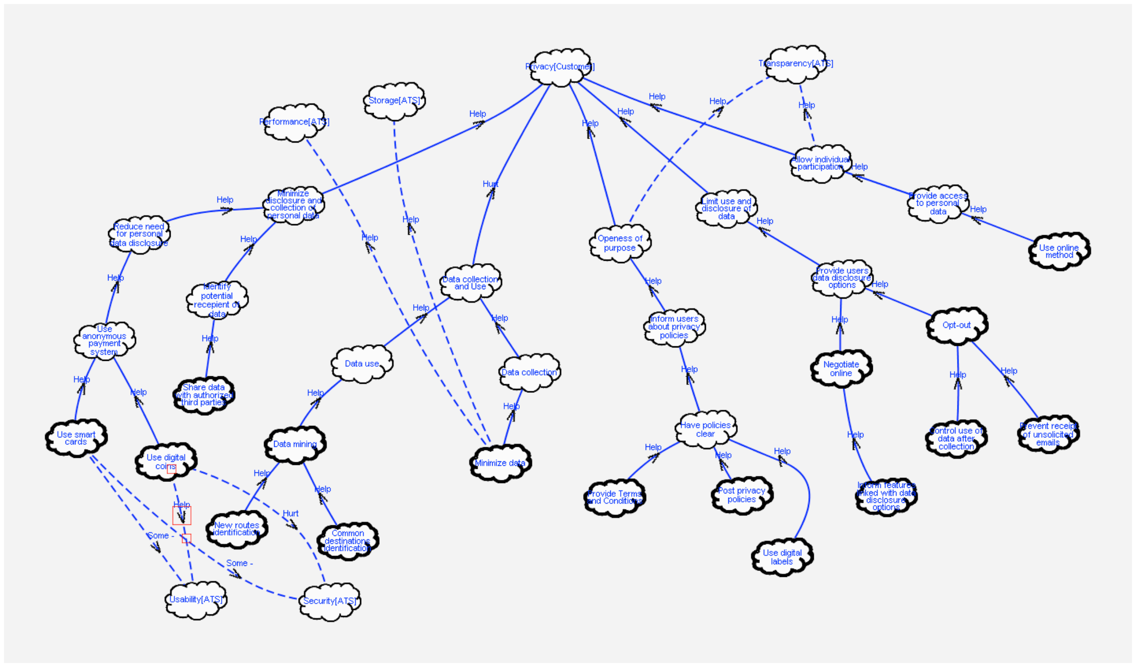

Privacy[Customer]
Transparency[ATS]
Storage[ATS]
Performance[ATS]
Minimize disclosure and collection of personal data
Reduce need for personal data disclosure
Identify potential recepient of data
Share data with authorized third parties
Use anonymous payment system
Use smart cards
Use digital coins
Usability[ATS]
Security[ATS]
Data collection and Use
Data use
Data mining
New routes identification
Common destinations identification
Data collection
Minimize data
Openess of purpose
Inform users about privacy policies
Have policies clear
Provide Terms and Conditions
Post privacy policies
Use digital labels
Limit use and disclosure of data
Provide users data disclosure options
Negotiate online
Inform features linked with data disclosure options
Opt-out
Control use of data after collection
Prevent receipt of unsolicited emails
Allow individual participation
Provide access to personal data
Use online method
Help
Hurt
Some -

**Figure 37 Positive Control: Privacy SIG**

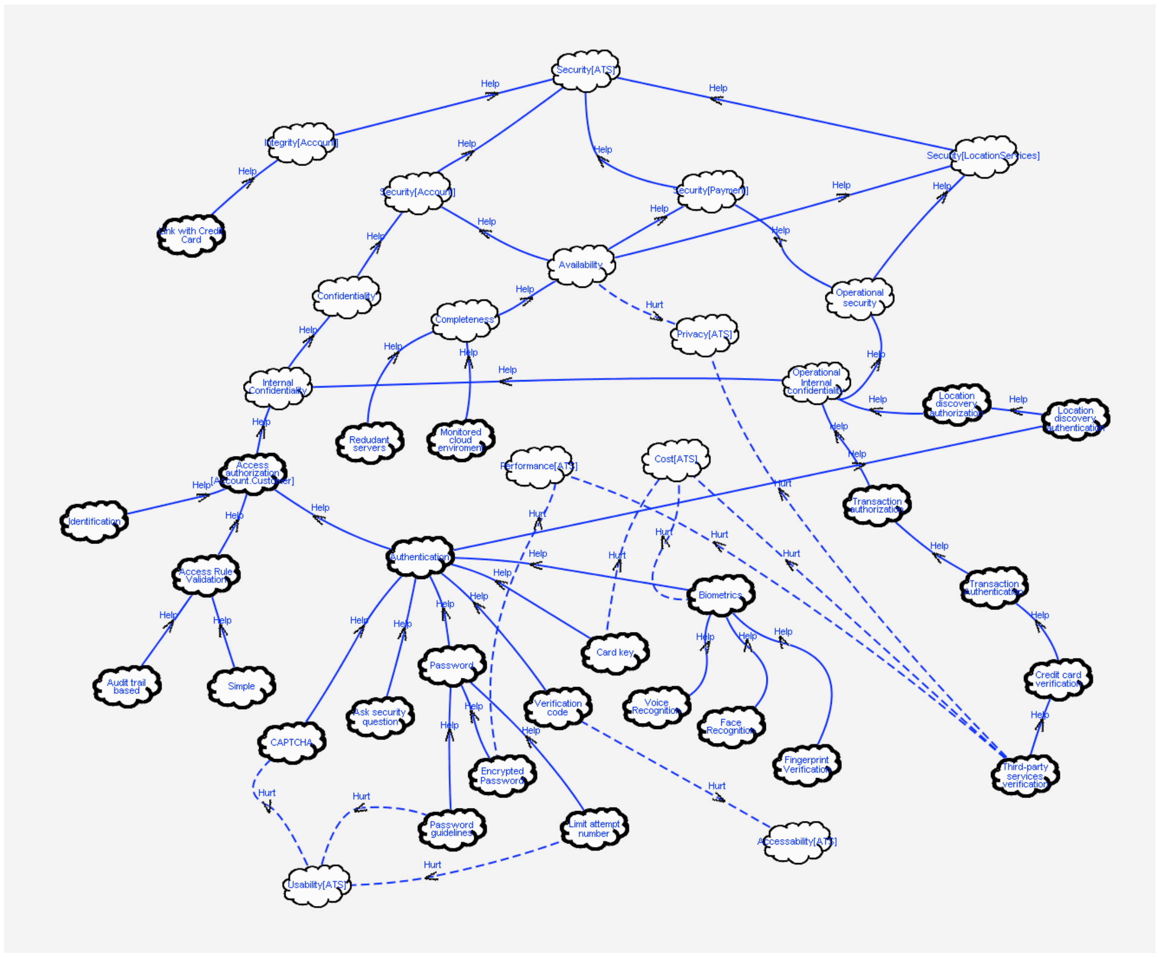


**Figure 38 Positive Control: Security SIG**

## 5.7 Summary

In this chapter, we emphasized the methodology applied in our research experiment. We discussed the hypothesis and variables of this study. Also, we described our research design and its characteristics including the utilized methods and techniques for collecting data, addressing threats to validity, and dividing participants into designated groups.

Furthermore, we described the proposed scenario for our experiment. We suggested a realistic scenario for a software system development, with the necessary fundamental documentation. Consequently, we believe that participants in this research study were able to identify and elicit NFRs in a reasonable industrial manner.

# Chapter 6
# Findings and Discussion

In this chapter, we discuss our findings based on the performed evaluation experiment. Unfortunately, we are not able to expose detailed information about the collected data due to a non-disclosure agreement. Therefore, we only demonstrate a resultant analysis represented by average and percentage values.

Also, we present our interpretation based on the demonstrated analytical results. We state our beliefs and critical findings associated with the outcomes of this research experiment.

## 6.1 Overview

As a result of conducting our evaluation, we were able to collect a sample composed of 12 participants. Half of the participants performed the experiment using the NDR Tool as a knowledge-assistance technique. The other half of subjects completed the study using only pure NFR catalogs as a knowledge-assistance technique.

Along this chapter, we will refer to this division among participants as *Group with NDR Tool* for those individuals that performed the study assisted by the NDR Tool, and *Group with NFR Catalogs* for those subjects that were aided by simple NFR catalogs during the experiment.

Both groups could rely on the same amount of knowledge. In other words, either knowledge-assistance techniques offered identical information about the following NFRs: *Privacy*, *Security*, *Traceability*, *Transparency*, and *Usability*.

The main difference between both knowledge-assistance techniques is associated with the representation of information. The NDR Tool provides an aggregated approach for visualizing NFR knowledge. Additionally, it allows users to search for a particular element by simply providing a search term. It also allows one to navigate from one NFR to another when correlations are in place. On the other hand, pure NFR catalogs demonstrate NFR knowledge simply through static pre-generated images. Even illustrating the same amount of information as the NDR Tool, pure NFR catalogs do not offer any dynamic functionality that may be used to search a specific element or demonstrate trade-offs among different NFRs.

Ultimately, our authoritative control emphasized the main expected NFRs for the provided scenario, including *Privacy*, *Security*, *Traceability*, *Transparency*, *Usability*, and *Performance*. Each participant in this study was able to elicit and model more than one of the required NFRs. The following sections describe a comprehensive analysis regarding the outcomes of this research experiment.

## 6.2 Analysis

Before demonstrating our comprehensive analysis, we first introduce fundamental statistical concepts applied in our investigation described in this section. We denote them as the following:

- **Min:** The minimum existent numerical value in a given dataset.
- **Max:** The maximum existent numerical value in a given dataset.
- **Median:** Represents the central tendency value in a population.
- **Mean:** Describes an average value in a dataset.
- **Standard Deviation:** Describes a statistical value for indicating the general variability in a numerical dataset.

After gathering the models from each participant, we manually measured the number of correct *Operationalizations* and *Correlations* according to our positive control. Also, we identified each NFR elicited by every participant. Then, we registered the totals in a CSV file and imported it into the *IBM Statistical Analysis Software Package* (SPSS)[8] in order to conduct our statistical analysis.

[8] http://www.ibm.com/analytics/us/en/technology/spss/spss.html

## Operationalizations

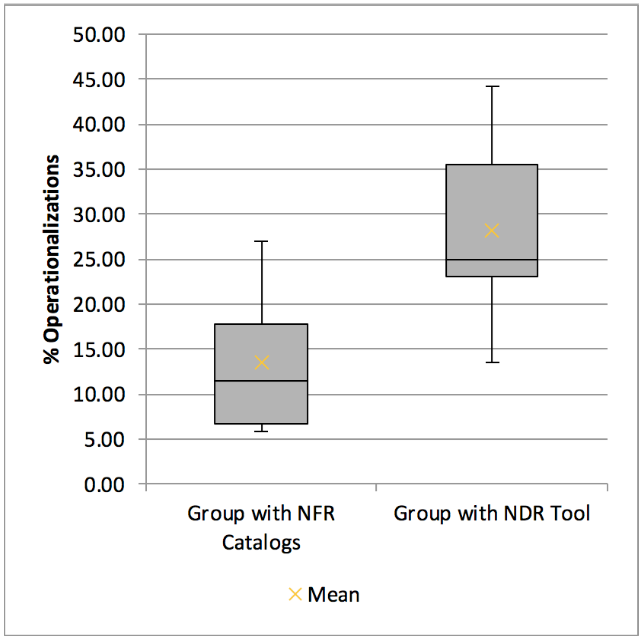


**Figure 39 Box plot of Descriptive findings: Percentage of found Operationalizations**

Figure 39 illustrates a box plot as a graphical visualization for our detailed findings regarding the percentage of the identified *Operationalizations* in each group. Both groups define a population positively skewed. The central tendency measure (*Median*) indicates that the average within *Group with NDR Tool* (*Median*=25%) is significantly greater than the one within *Group with NFR Catalogs* (*Median*=11.54%). Additionally, both groups demonstrate related variability since their spread of data points are similar regarding the graphical size and value of the *Interquartile Range* (IQR). Furthermore, *Group with NFR Catalogs* presents a distribution of results more concentrated towards to the minimum possible value than the produced distribution of *Group with NDR Tool*. This observation means that most of the participants in *Group* *with NFR Catalogs* identified a low percentage of *Operationalizations* in comparison to participants in *Group with NDR Tool*.

It is important to mention that both datasets representing each group were free of outliers. Every evaluated data point regarding the identified percentage of *Operationalizations* by each participant was part of the upper and lower limit range of this evaluation.

### Correlations

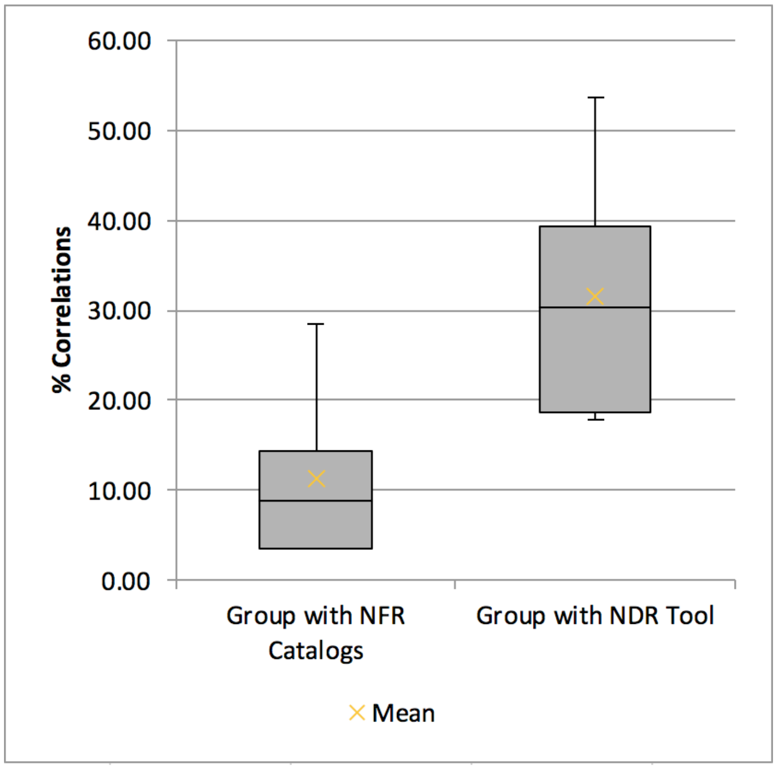


**Figure 40 Box plot of Descriptive findings: Percentage of *Correlations***

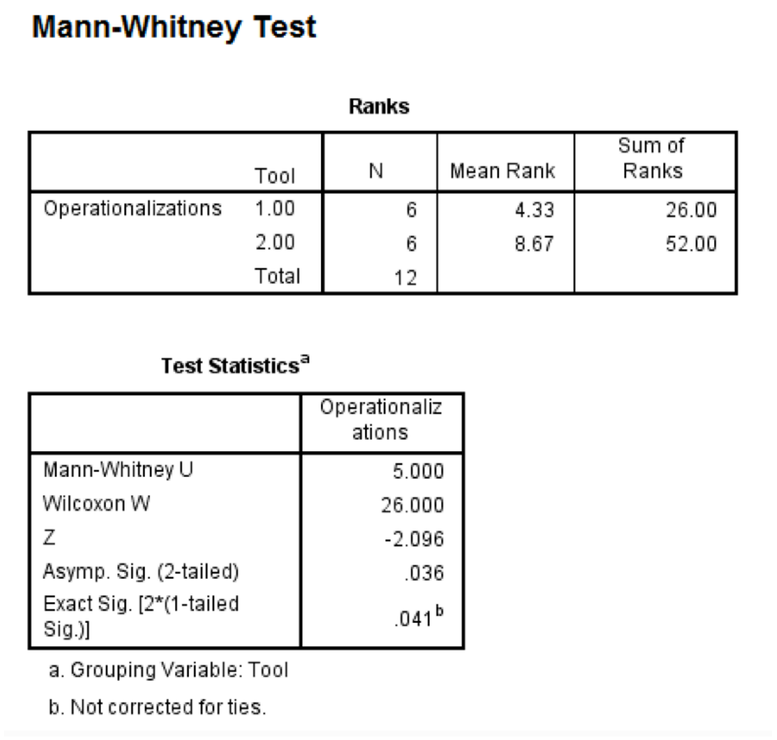
Figure 40 expresses a graphical representation for our findings regarding the percentage of found *Correlations*. The box plot graph demonstrates a positively skewed dataset for both groups. The central tendency measure (*Median*) shows that the average of *Group with NDR Tool* (*Median*=30.36%) is significantly higher than the one from *Group with NFR Catalogs* (*Median*=8.93%). Moreover, the *Group with NDR Tool* indicates a greater variability regarding the spread of data. Both the graphical and numerical value of the *Interquartile Range* (IQR) are greater than the *Group with NFR Catalogs* one. Furthermore, in this case, both groups present a distribution more concentrated towards the minimum value of each dataset. However, it is noticeable that participants within the *Group with NDR Tool* were responsible for eliciting a high percentage of *Correlations* in comparison to the other group because their minimum possible percentage value is still considered a medium-high percentage amount overall.

Lastly, as also observed in the *Operationalizations* evaluation, both of the datasets employed in this *Correlation* analysis were free of outliers. The calculated upper and lower range of the datasets in this investigation embraced all the available data points.

## 6.3 Hypotheses testing

In regards to our inferential analysis, we decided to evaluate the previously stated hypotheses. As our dataset does not characterize a normal distribution, the inferential evaluation of our hypotheses must be performed by a non-parametric test. Additionally, as our investigation only observes two categories regarding the independent variable of this study, *Mann-Whitney* turned to be the appropriated statistical test for adequately analyze our hypotheses.

The following Figure 41 demonstrates the evaluation of our H1:



**Mann-Whitney Test**

**Ranks**

| | Tool | N | Mean Rank | Sum of Ranks |
|---|---|---|---|---|
| Operationalizations | 1.00 | 6 | 4.33 | 26.00 |
| | 2.00 | 6 | 8.67 | 52.00 |
| | Total | 12 | | |

**Test Statistics[a]**

| | Operationalizations |
|---|---|
| Mann-Whitney U | 5.000 |
| Wilcoxon W | 26.000 |
| Z | -2.096 |
| Asymp. Sig. (2-tailed) | .036 |
| Exact Sig. [2*(1-tailed Sig.)] | .041[b] |

a. Grouping Variable: Tool

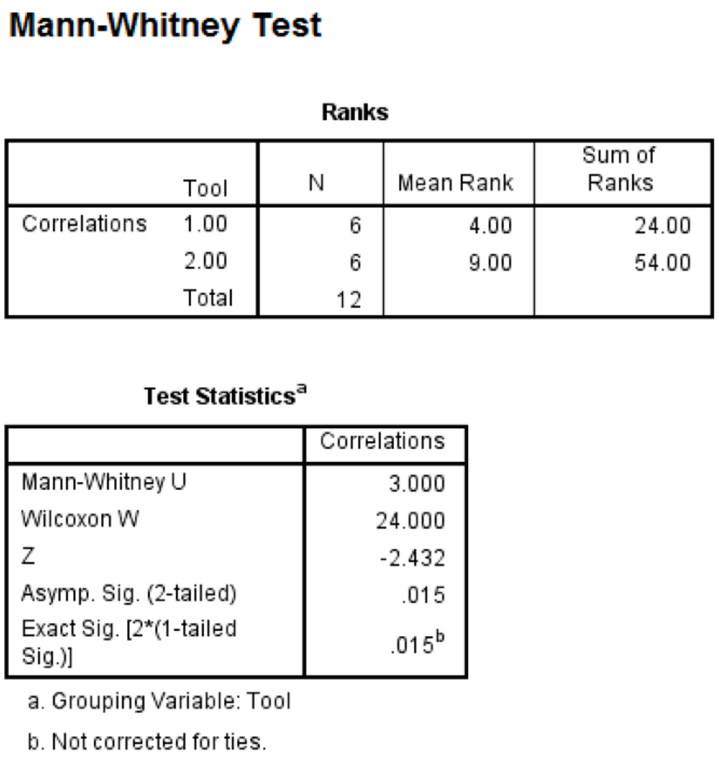
b. Not corrected for ties.

**Figure 41 Mann-Whitney Test results for H1**

The results indicate that the null hypothesis should be rejected. The *Mann-Whitney* test calculated a 2-tailed sigma of 0.036, which satisfies the condition for accepting or rejecting a given hypothesis ($p < 0.05$). Additionally, through the *Mann-Whitney* test, it is possible to note mean rank among both groups. *Group with NDR Tool* (*label* 2.00) demonstrated a significantly higher mean rank (8.67) in comparison to the value (4.33) denoted by *Group with NFR Catalogs* (*label* 1.00).

As a consequence, we can confirm our H1 hypothesis: The NDR Tool helps software engineers to identify more *Operationalizations*.

The following Figure 42 Mann-Whitney Test results for H2 illustrates the evaluation of H2:



**Mann-Whitney Test**

**Ranks**

| | Tool | N | Mean Rank | Sum of Ranks |
|---|---|---|---|---|
| Correlations | 1.00 | 6 | 4.00 | 24.00 |
| | 2.00 | 6 | 9.00 | 54.00 |
| | Total | 12 | | |

**Test Statistics**[a]

| | Correlations |
|---|---|
| Mann-Whitney U | 3.000 |
| Wilcoxon W | 24.000 |
| Z | -2.432 |
| Asymp. Sig. (2-tailed) | .015 |
| Exact Sig. [2*(1-tailed Sig.)] | .015[b] |

a. Grouping Variable: Tool

b. Not corrected for ties.

**Figure 42 Mann-Whitney Test results for H2**

The findings also indicate that the null hypothesis should be rejected. The test measured a 2-tailed sigma of 0.015, meeting the necessary condition for accepting or denying a particular hypothesis ($p < 0.05$). The mean rank for each group is also demonstrated in this analysis. *Group with NDR Tool* (*label* 2.00) showed a greater mean rank value (9.00) in comparison to the other group (4.00).

As a result of this analysis, we can also confirm our H2 hypothesis: The NDR Tool helps software engineers to identify more *Correlations*.

## 6.4 Discussion

This research experimentation was conducted to evaluate whether the use of a customized knowledge-assistance technique can support software engineers to better elicit NFRs. Our results suggest that the NDR Tool can facilitate the reuse of NFR knowledge and therefore contribute to elicit more and better NFRs. Aside from the statistical conclusions, we also raise the following assumptions that could have influenced the outcomes of this evaluation.

In both descriptive and inferential analysis, we were able to notice the significant results from *Group with NDR Tool* over *Group with NFR Catalogs*. We believe that the features provided by the NDR Tool played an influential role in the overall outcome of the experiment analysis. Among these features we should mention:

- The capability to navigate from one NFR to another with a simple click.
- The ability to query one particular NFR and get back its *Operationalizations* unitedly with possible *Correlations* that could be triggered from these *Operationalizations*.
- The non-restrictive search ability that allows the software engineer to exercise different levels of granularity to search for NFR related

capabilities. For instance, *Search for "fingerprint", Search for "Data Disclosure"*, *Search for "Speech Recognition".*

- The inferences provided by the NDR Tool regarding the occurrences of a particular element across the available NFRs in the Knowledge Repository (*Correlations*).

We believe that this set of features offered convenience and helped participants within the *Group with NDR Tool* to easily navigate through the available NFR knowledge and identify a target characteristic required by the proposed scenario. For instance, the proposed scenario demanded that the needed system had to be accessible from any device with internet connection. A participant could simply use the NDR Tool to search for *"tablet"* and understand that there is a *"Use Tablets"* capability in a certain refinement level that helps *satisficing Usability*.

We also consider that this collection of functionalities assisted *Group with NDR Tool* participants to better identify possible synergies and conflicts regarding an element that occurs in the satisficing of multiple NFRs. For example, a user wondering whether *Performance* correlates with another NFR could quickly use the NDR Tool to search for *"Performance"* and see its occurrence across the Knowledge Repository. As a result, after identifying the NFRs that correlates with *Performance*, the user could still verify specific details of the correlation by selecting a NFR from the list of occurrences.

Regarding the sample size of the experiment, we understand that a larger dataset may produce more accurate outcomes regarding both groups. Due to a class size limitation, we could only gather valid results from 12 participants. We aim to reproduce this study in a near future with a bigger population to try to generate more precise statistical results and identify more relationships regarding our variables.

In an overall manner, we consider that it was possible to highlight our proposed approach as an alternative for dealing and facilitating the reuse of NFR Knowledge through the performed research experiment. In an era where NFRs are still incorrectly underestimated on software projects, we believe that our developed work arises as an important contribution to the alternatives of relatively new resources for dealing with the reuse of NFR knowledge during the early phases of software development lifecycle.

## 6.5 Summary

In this chapter, we demonstrated our findings regarding the performed controlled experiment. Firstly, we emphasized an overview associated with the circumstances of the experiment and its outcomes.

Then, we depicted a descriptive and inferential analysis conducted over the results of this study. We compared the statistical outputs generated by the analysis over both observed groups: *Group with NDR Tool* and *Group with NFR Catalogs*. In a statistical comparison, *Group with NDR Tool* performance was satisfactorily greater than *Group with NFR Catalogs*. We also tested our proposed hypotheses and statistically accepted them.

Finally, we discussed our findings regarding the performed experiment. We denoted our central beliefs about the obtained results and which factors could have affected these outcomes. Mainly, we believe that the main features of our proposed approach played a significant role in the experiment output. Additionally, we understand that a larger sample size may produce more accurate results.

# Chapter 7
# Conclusions

Dealing with NFRs within software projects have always been a challenge since most of the notations and techniques are currently designed primarily focusing on the representation and understanding of Functional Requirements. Moreover, several recent studies have demonstrated that NFRs are still not being addressed adequately since software engineers rarely take into the account its elicitation and modeling during the early stage of the development cycle.

In this study, we proposed an approach for assisting software engineers with the reuse of NFR Knowledge for a better elicitation and modeling. Our developed approach uses the NDR Ontology and the NFR Framework as a baseline. The NDR Tool uses SIGs developed under the NFR Framework notation as input for adding NFR information into a Knowledge Repository. On the other hand, the NDR Ontology is employed as a model on a knowledge-representation level. The system follows the ontology guidelines to transform the information obtained from SIGs into a machine-readable format data. By having NFR knowledge in a machine-readable composition, the NDR Tool allows software engineers to search for particular elements associated with multiple NFRs by abstractly querying the Knowledge Repository. Then, when applicable, it displays a real-time generated graphical model that also follows the NFR Framework notation. As an additional feature for identifying possible synergies and conflicts, the tool also demonstrates the occurrences of a particular element among the existent NFRs in the Knowledge Repository.

Our major goal in this thesis is to assemble a strong foundation for the NDR Framework to grow. Therefore, as part of our contributions, we developed and implemented the NDR Tool which will be the NDR Framework's core.

As a second contribution, we conducted a comprehensive analysis of the applicability of the NDR Tool in a real-world scenario. We carried out a controlled experiment with human participants. The individuals were divided into two groups, one required to perform the experiment with the NDR Tool as a knowledge-assistance method, and the other needed to complete the study using pure NFR Catalogs as a supporting knowledge technique. Both groups needed to elicit and model as many as possible NFRs regarding a provided real-world scenario. We evaluated their outcomes on the number of identified *Operationalizations* and *Correlations* level against an authoritative control.

As a final contribution, we interpreted the results of our extensive analysis over the NDR Tool's applicability. Our findings statistically demonstrated that the NDR Tool could help software engineers to elicit better NFRs by providing the reuse of its knowledge. More than providing a graphical visualization, the NDR Tool provides an integration of different SIGs regarding one or multiple NFRs into a sole representation, characterizing a constant NFR knowledge evolution.

## 7.1 Future work

As a future work of this study, we aim to reproduce our controlled experiment with a larger population. We believe that having a greater sample size will increase the chances of producing more accurate results on the analysis of the applicability of the NDR Tool. As well it will provide us important feedback on weaknesses and strengths of the current approach.

Additionally, we aim to add more features to the NDR Tool. As the graphical representations tend to scale in a broad manner, we intend to implement alternative methods to visualize the information. We consider developing filters regarding refinements and interdependencies in order to demonstrate the knowledge with different granularity levels. Also, we aim to implement an exportation feature to promote the integration of existent systems with our proposed approach.

We also envision to exploit the possible extensibility of our proposed approach. As a cloud-designed system, we believe that the NDR Tool can be employed as a service for self-adaptive systems. In other words, we envision to provide NFR knowledge in a real-time manner for self-adaptive systems through the NDR Tool. Several of these systems rely on NFR characteristics such as *Performance* and *Availability*. Therefore, we envision our approach as a decision-making support regarding NFR knowledge in adaptive scenarios. A suitable environment for our developed approach would be the MAPE-K Loop [45]. We understand that the NDR Tool can fit as a resource on the knowledge layer and provide information when requested during the *Analyze* and *Plan* stages of the MAPE-K Loop.

# Bibliography

[1] L. Chung and J. do Prado Leite, “On Non-Functional Requirements in Software Engineering,” in *Conceptual Modeling: Foundations and Applications*, vol. 5600, A. Borgida, V. Chaudhri, P. Giorgini, and E. Yu, Eds. Springer Berlin Heidelberg, 2009, pp. 363–379.

[2] B. W. Boehm, J. R. Brown, H. Kaspar, and M. Lipow, *Characteristics of software quality*. Amsterdam: North-Holland, 1978.

[3] S. E. Keller, L. G. Kahn, and R. B. Panara, “Specifying software quality requirements with metrics,” *Syst. Softw. Requir. Eng.*, pp. 145–163, 1990.

[4] A. Finkelstein and J. Dowell, “A comedy of errors: the London Ambulance Service case study,” in *Software Specification and Design, 1996., Proceedings of the 8th International Workshop on*, 1996, pp. 2–4.

[5] G. Roman, “A taxonomy of current issues in requirements engineering,” *Computer (Long. Beach. Calif).*, vol. 18, no. 4, pp. 14–23, Apr. 1985.

[6] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*. Springer US, 1999.

[7] H. A. Simon, *The sciences of the artificial*, vol. 136. 1996.

[8] J. Mylopoulos, L. Chung, and B. Nixon, “Representing and using nonfunctional requirements: a process-oriented approach,” *Softw. Eng. IEEE Trans.*, vol. 18, no. 6, pp. 483–497, Jun. 1992.

[9] L. M. Cysneiros, “Evaluating the Effectiveness of Using Catalogues to Elicit Non-Functional Requirements,” in *WER*, 2007, pp. 107–115.

[10] M. de Gramatica, K. Labunets, F. Massacci, F. Paci, and A. Tedeschi, “The Role of Catalogues of Threats and Security Controls in Security Risk Assessment: An Empirical Study with ATM Professionals,” in *Requirements Engineering: Foundation for Software Quality: 21st International Working Conference, REFSQ 2015, Essen, Germany, March 23-26, 2015. Proceedings*, A. S. Fricker and K. Schneider, Eds. Cham: Springer International Publishing, 2015, pp. 98–114.

[11] R. Veleda and L. M. Cysneiros, “An Initial Approach to Reuse Non-Functional Requirements Knowledge,” in *Proceedings of the Eighth International i* Workshop (istar 2015)*, 2015, pp. 25–30.

[12] N. S. Bullet, “Essence and Accidents of Software Engineering, FP Brooks,” *IEEE Comput.*, vol. 20, no. 4, pp. 10–19, 1987.

[13] A. M. Davis, *Software Requirements: Objects, Functions, and States*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[14] L. M. Cysneiros and J. C. S. do Prado Leite, “Nonfunctional requirements: from elicitation to conceptual models,” *IEEE Trans. Softw. Eng.*, vol. 30, no. 5, pp. 328–350, May 2004.

[15] L. Chung and B. A. Nixon, “Tool Support for Systematic Treatment of Non-

Functional Requirements.",” *Manuscript, December*, 1994.

[16] J. Horkoff, Y. Yu, and S. K. Eric, “OpenOME: An Open-source Goal and Agent-Oriented Model Drawing and Analysis Tool.,” in *iStar*, 2011, pp. 154–156.

[17] S. Supakkul and L. Chung, “The RE-Tools: A multi-notational requirements modeling toolkit,” in *Requirements Engineering Conference (RE), 2012 20th IEEE International*, 2012, pp. 333–334.

[18] G. Mussbacher and D. Amyot, “Goal and scenario modeling, analysis, and transformation with jUCMNav,” in *Software Engineering - Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on*, 2009, pp. 431–432.

[19] L. M. Cysneiros, J. C. S. do Prado Leite, and J. de Melo Sabat Neto, “A Framework for Integrating Non-Functional Requirements into Conceptual Models,” *Requir. Eng.*, vol. 6, no. 2, pp. 97–115, 2001.

[20] D. Mairiza and D. Zowghi, “Constructing a Catalogue of Conflicts among Non-functional Requirements,” in *Evaluation of Novel Approaches to Software Engineering: 5th International Conference, ENASE 2010, Athens, Greece, July 22-24, 2010, Revised Selected Papers*, L. A. Maciaszek and P. Loucopoulos, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 31–44.

[21] J. Doerr, D. Kerkow, T. Koenig, T. Olsson, and T. Suzuki, “Non-functional requirements in industry - three case studies adopting an experience-based NFR method,” in *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on*, 2005, pp. 373–382.

[22] T. H. Al Balushi, P. R. F. Sampaio, and P. Loucopoulos, “Eliciting and prioritizing quality requirements supported by ontologies: a case study using the ElicitO framework and tool,” *Expert Syst.*, vol. 30, no. 2, pp. 129–151, 2013.

[23] R. Guizzardi, F.-L. Li, A. Borgida, G. Guizzardi, J. Horkoff, and J. Mylopoulos, “An Ontological Interpretation of Non-Functional Requirements,” 2014.

[24] G. Guizzardi and G. Wagner, “A Unified Foundational Ontology and some Applications of it in Business Modeling.,” in *CAiSE Workshops (3)*, 2004, pp. 129–143.

[25] C. L. Liu, “Ontology-Based Conflict Analysis Method in Non-functional Requirements,” in *Computer and Information Science (ICIS), 2010 IEEE/ACIS 9th International Conference on*, 2010, pp. 491–496.

[26] G. Dobson, S. Hall, and G. Kotonya, “A Domain-Independent Ontology for Non-Functional Requirements,” in *e-Business Engineering, 2007. ICEBE 2007. IEEE International Conference on*, 2007, pp. 563–566.

[27] N. Koay, P. Kataria, R. Juric, P. Oberndorf, and G. Terstyanszky, “Ontological support for managing non-functional requirements in pervasive healthcare,” in *System Sciences, 2009. HICSS’09. 42nd Hawaii International Conference on*, 2009, pp. 1–10.

[28] P. P. Sancho, C. Juiz, R. Puigjaner, L. Chung, and N. Subramanian, “An Approach to Ontology-aided Performance Engineering Through NFR Framework,” in

*Proceedings of the 6th International Workshop on Software and Performance*, 2007, pp. 125–128.

[29] C. Lopez, L. M. Cysneiros, and H. Astudillo, “NDR Ontology: Sharing and Reusing NFR and Design Rationale Knowledge,” in *Managing Requirements Knowledge, 2008. MARK ’08. First International Workshop on*, 2008, pp. 1–10.

[30] H. Hu, Q. Ma, T. Zhang, Y. Tan, H. Xiang, C. Fu, and Y. Feng, “Semantic modelling and automated reasoning of non-functional requirement conflicts in the context of softgoal interdependencies,” *IET Softw.*, vol. 9, no. 6, pp. 145–156, 2015.

[31] T. Berners-Lee, J. Hendler, O. Lassila, and others, “The semantic web,” *Sci. Am.*, vol. 284, no. 5, pp. 28–37, 2001.

[32] F. Manola and E. Miller, “RDF Primer,” 2004. [Online]. Available: http://www.w3.org/TR/2004/REC-rdf-primer-20040210/.

[33] D. Brickley and R. V. Guha, “RDF Vocabulary Description Language 1.0: RDF Schema,” 2002. [Online]. Available: https://www.w3.org/TR/2002/WD-rdf-schema-20021112/.

[34] D. L. McGuinness, F. Van Harmelen, and others, “OWL web ontology language overview,” 2004.

[35] E. Prud’hommeaux and A. Seaborne, “SPARQL Query Language for RDF,” techreport, Jan. 2008.

[36] P. F. Patel-Schneider, P. Hayes, I. Horrocks, and F. Van Harmelen, “Web Ontology Language (OWL) Abstract Syntax and Semantics,” 2002.

[37] S. Zareian, R. Veleda, M. Litoiu, M. Shtern, H. Ghanbari, and M. Garg, “K-Feed - A Data-Oriented Approach to Application Performance Management in Cloud,” in *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*, 2015, pp. 1045–1048.

[38] H. Khazaei, S. Zareian, R. Veleda, and M. Litoiu, “Sipresk: A Big Data Analytic Platform for Smart Transportation,” *EAI Int. Conf. Big Data Anal. Smart Cities*, 2015.

[39] A. Leff and J. T. Rayfield, “Web-application development using the Model/View/Controller design pattern,” in *Enterprise Distributed Object Computing Conference, 2001. EDOC ’01. Proceedings. Fifth IEEE International*, 2001, pp. 118–127.

[40] PlayFramework, “PlayFramework,” 2016. [Online]. Available: https://www.playframework.com/.

[41] Apache Foundation, “Apache Jena,” 2016. [Online]. Available: https://jena.apache.org/.

[42] Apache Foundation, “Apache Fuseki,” 2016. [Online]. Available: https://jena.apache.org/documentation/serving_data/.

[43] Graphviz, “Graphviz - Graph Visualization Software,” 2016. [Online]. Available: http://www.graphviz.org/.

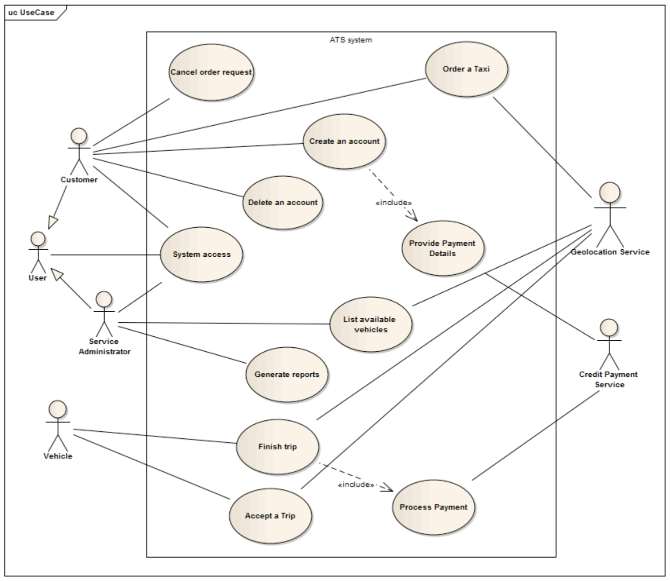
[44] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, “Selecting Empirical

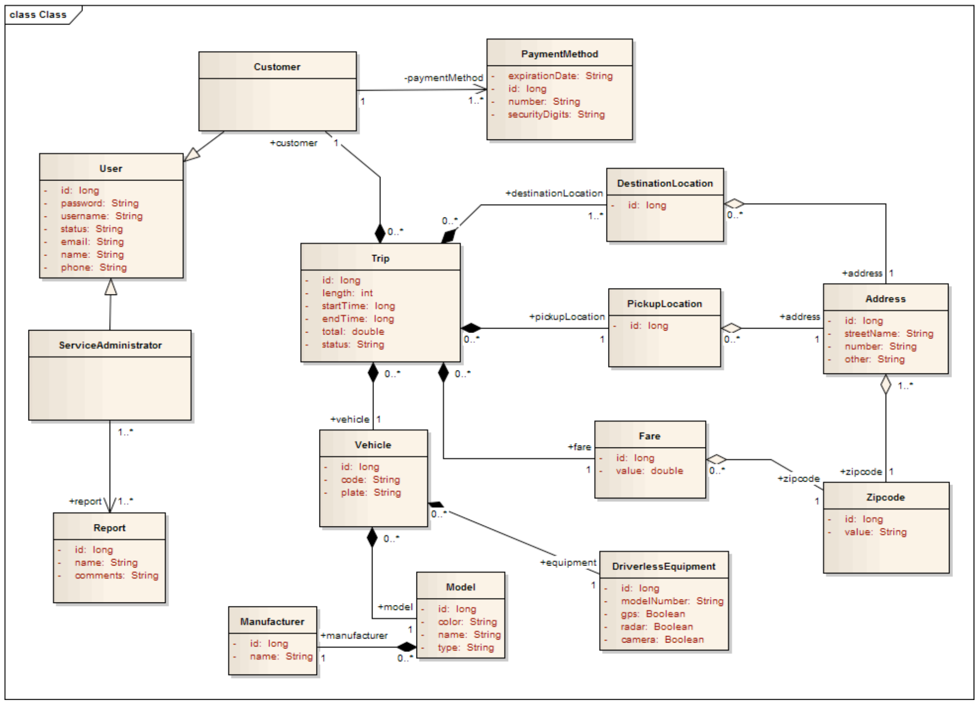
Methods for Software Engineering Research," in *Guide to Advanced Empirical Software Engineering*, F. Shull, J. Singer, and D. I. K. Sjøberg, Eds. London: Springer London, 2008, pp. 285–311.

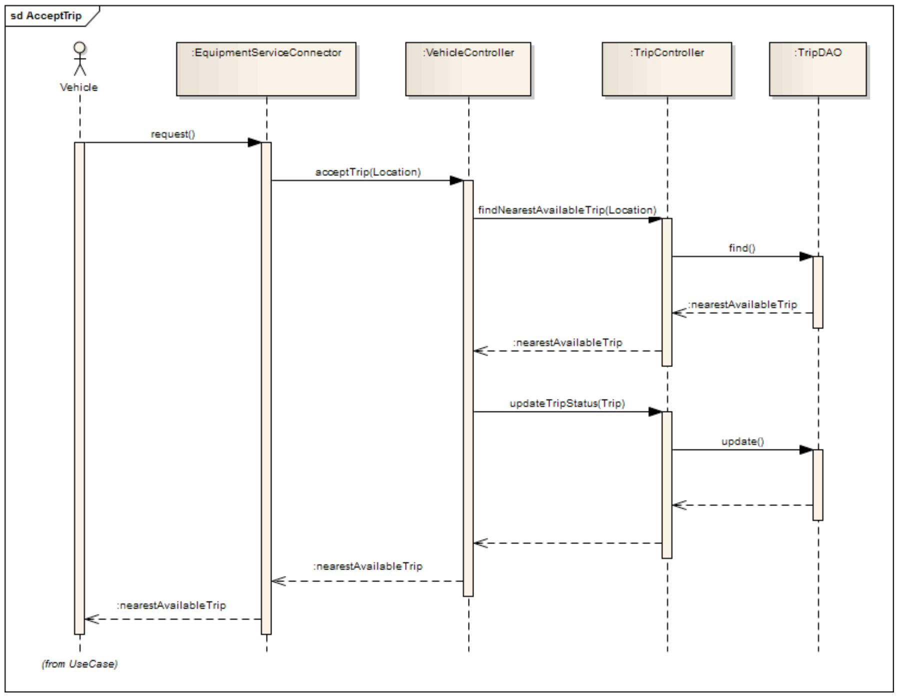
[45] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer (Long. Beach. Calif).*, vol. 36, no. 1, pp. 41–50, Jan. 2003.

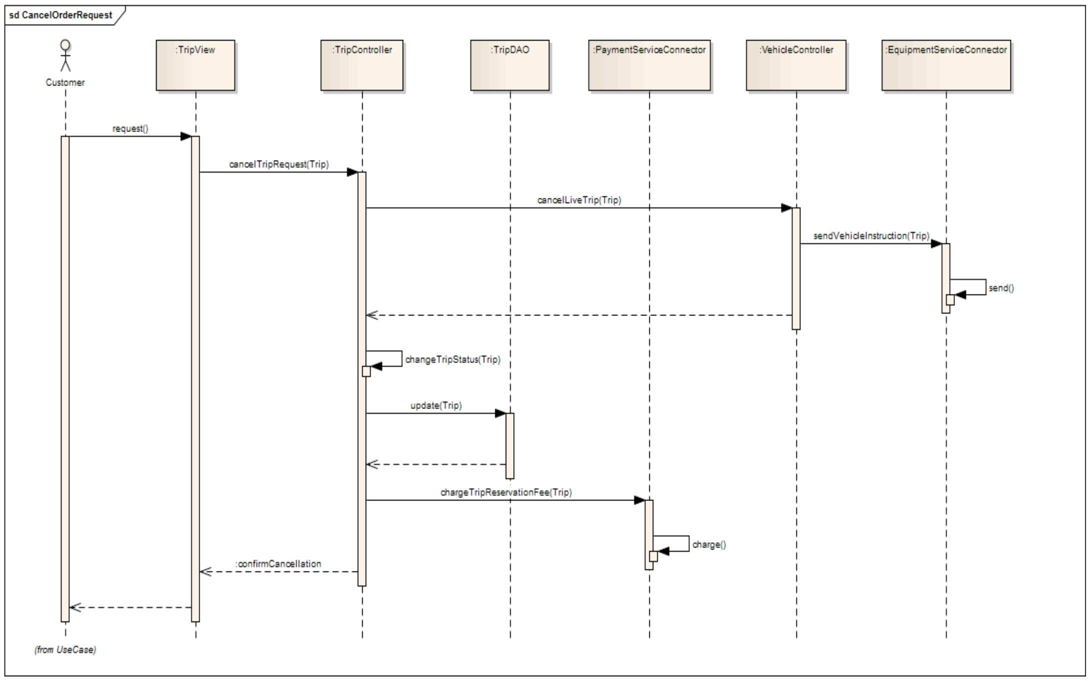
# Appendices

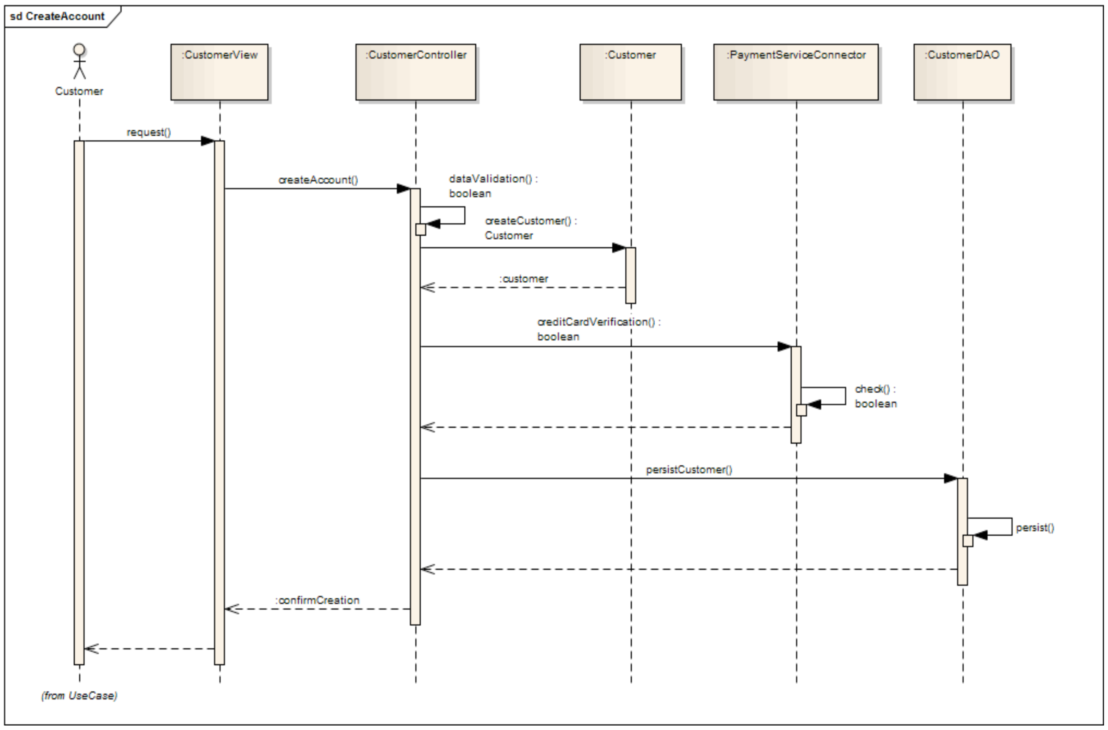
## Appendix A – Proposed Scenario Documentation

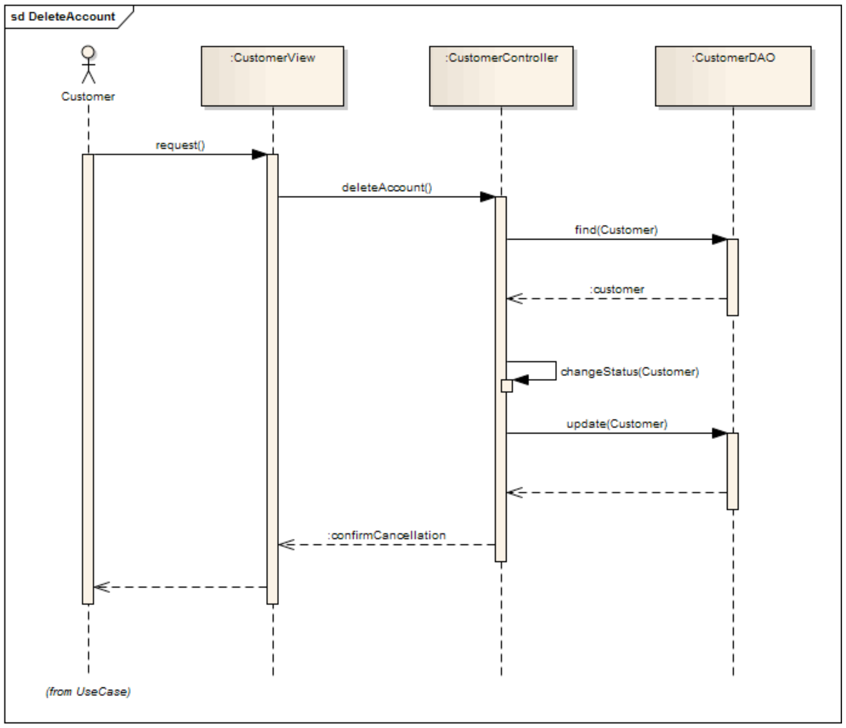


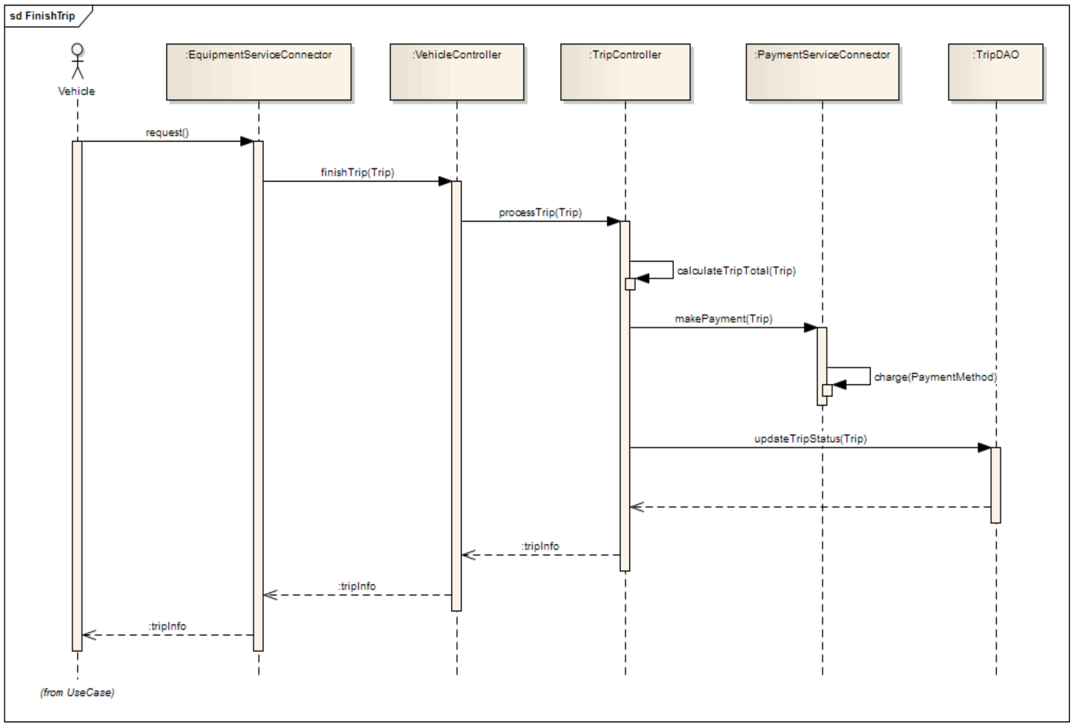


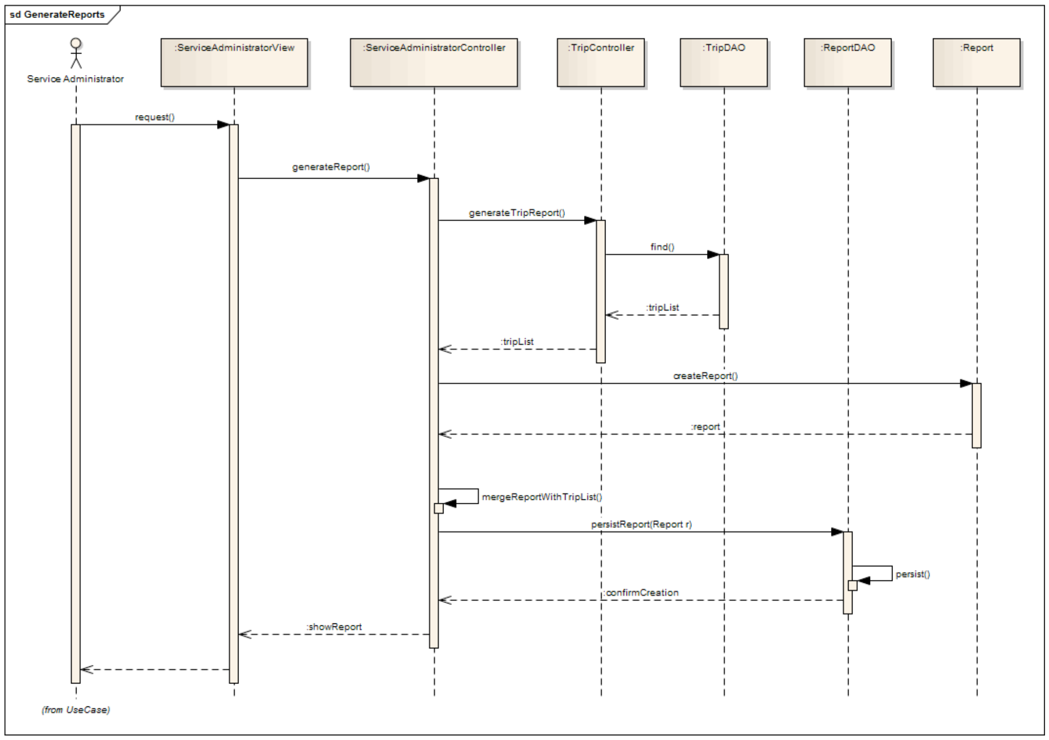

sd AcceptTrip
Vehicle
:EquipmentServiceConnector
:VehicleController
:TripController
:TripDAO
request()
acceptTrip(Location)
findNearestAvailableTrip(Location)
find()
:nearestAvailableTrip
:nearestAvailableTrip
updateTripStatus(Trip)
update()
:nearestAvailableTrip
:nearestAvailableTrip
(from UseCase)

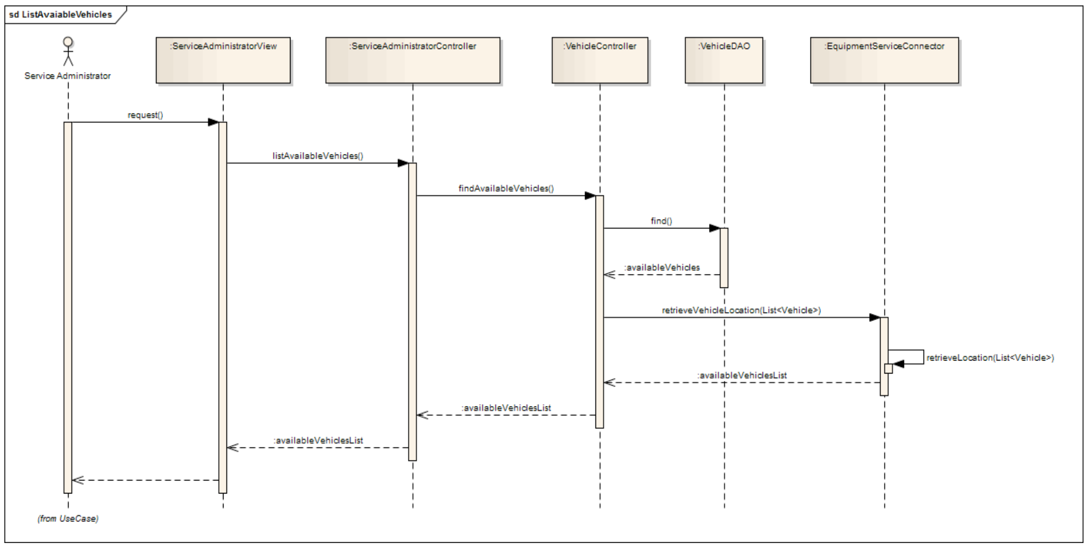

sd CancelOrderRequest
Customer
:TripView
:TripController
:TripDAO
:PaymentServiceConnector
:VehicleController
:EquipmentServiceConnector
request()
cancelTripRequest(Trip)
cancelLiveTrip(Trip)
sendVehicleInstruction(Trip)
send()
changeTripStatus(Trip)
update(Trip)
chargeTripReservationFee(Trip)
charge()
:confirmCancellation
(from UseCase)

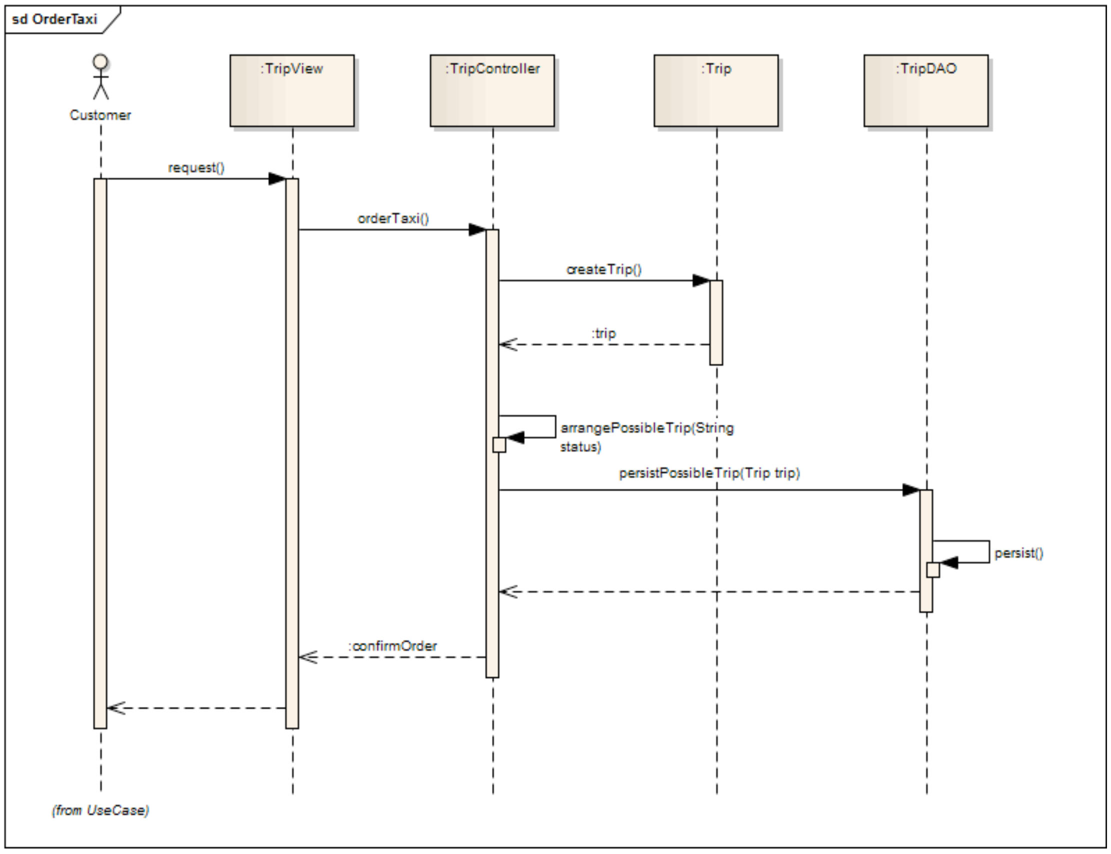

sd CreateAccount
Customer
:CustomerView
:CustomerController
:Customer
:PaymentServiceConnector
:CustomerDAO
request()
createAccount()
dataValidation() : boolean
createCustomer() : Customer
:customer
creditCardVerification() : boolean
check() : boolean
persistCustomer()
persist()
:confirmCreation
(from UseCase)

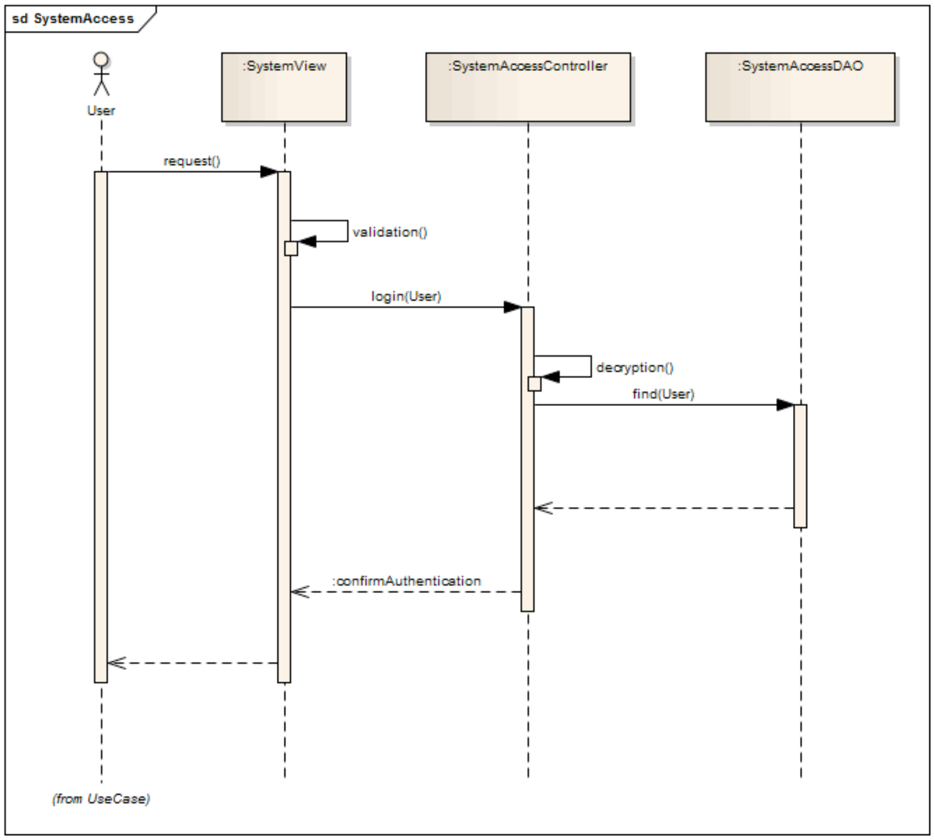

sd DeleteAccount
Customer
:CustomerView
:CustomerController
:CustomerDAO
request()
deleteAccount()
find(Customer)
:customer
changeStatus(Customer)
update(Customer)
:confirmCancellation
(from UseCase)


sd FinishTrip
Vehicle
:EquipmentServiceConnector
:VehicleController
:TripController
:PaymentServiceConnector
:TripDAO
request()
finishTrip(Trip)
processTrip(Trip)
calculateTripTotal(Trip)
makePayment(Trip)
charge(PaymentMethod)
updateTripStatus(Trip)
:tripInfo
:tripInfo
:tripInfo
(from UseCase)

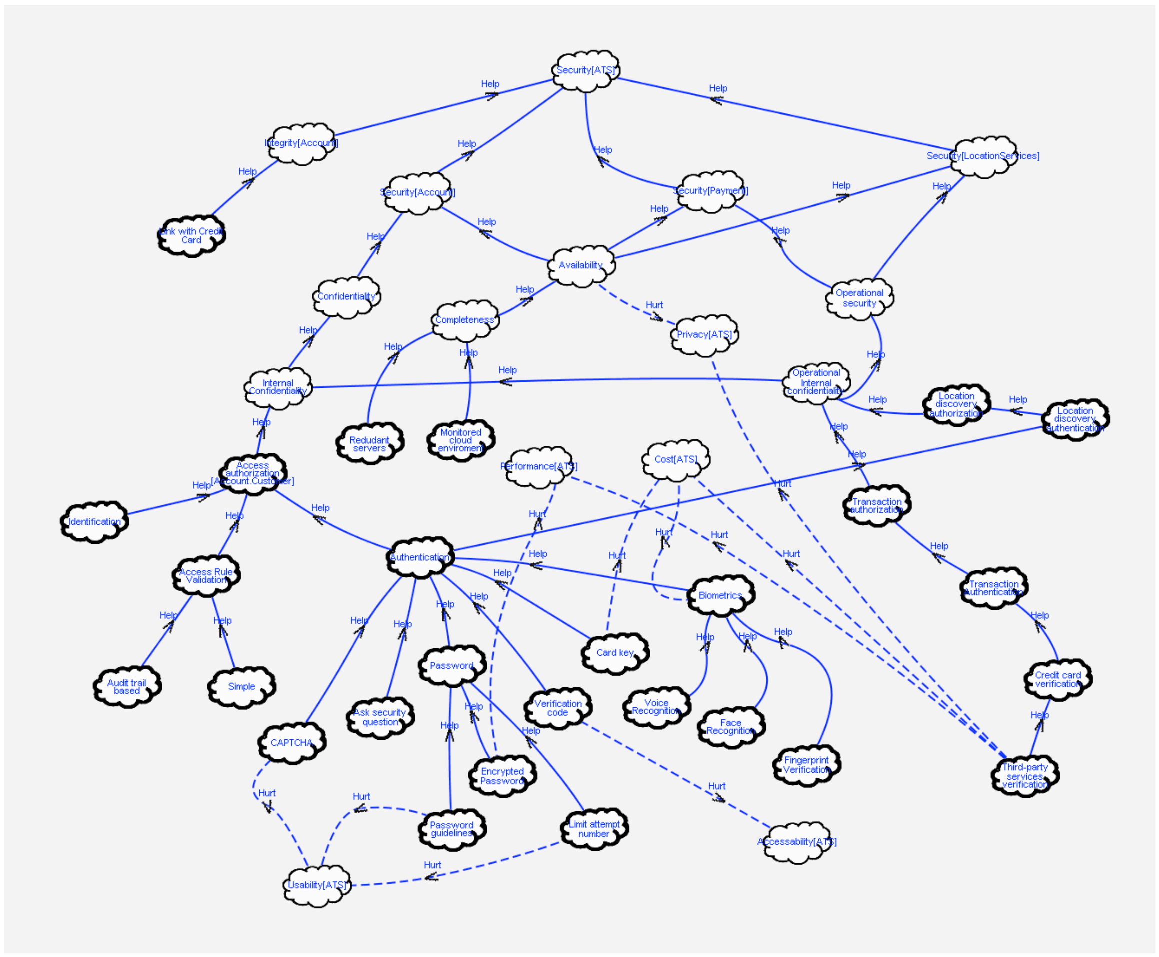

sd GenerateReports
Service Administrator
:ServiceAdministratorView
:ServiceAdministratorController
:TripController
:TripDAO
:ReportDAO
:Report
request()
generateReport()
generateTripReport()
find()
:tripList
:tripList
createReport()
:report
mergeReportWithTripList()
persistReport(Report r)
persist()
:confirmCreation
:showReport
(from UseCase)

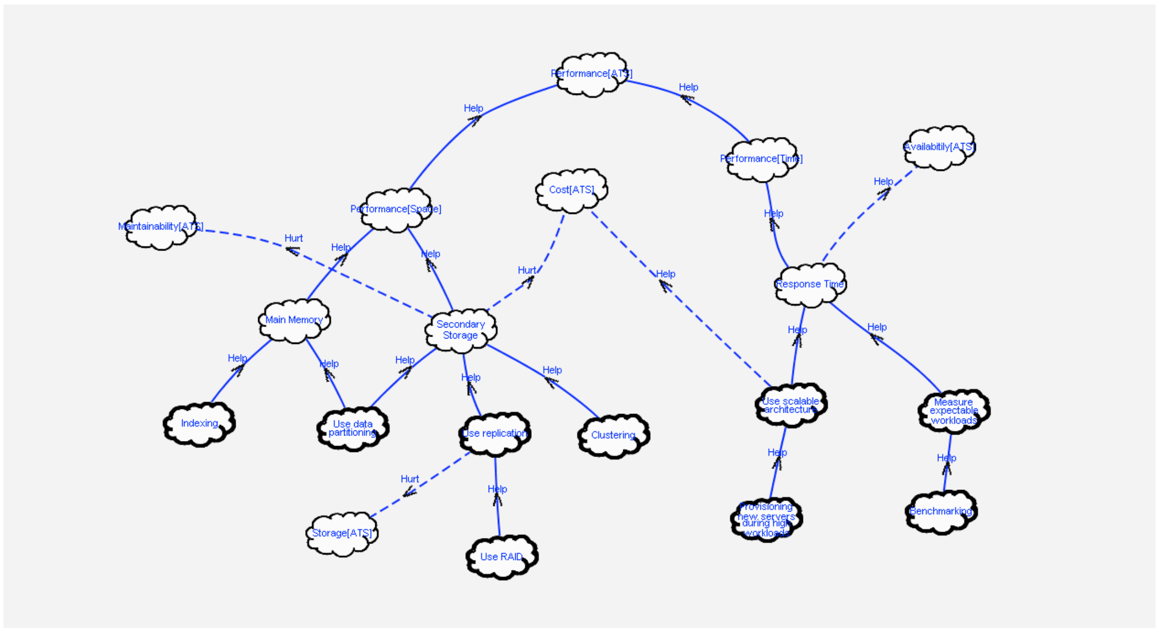

sd ListAvaiableVehicles
Service Administrator
:ServiceAdministratorView
:ServiceAdministratorController
:VehicleController
:VehicleDAO
:EquipmentServiceConnector
request()
listAvailableVehicles()
findAvailableVehicles()
find()
:availableVehicles
retrieveVehicleLocation(List<Vehicle>)
retrieveLocation(List<Vehicle>)
:availableVehiclesList
:availableVehiclesList
:availableVehiclesList
(from UseCase)

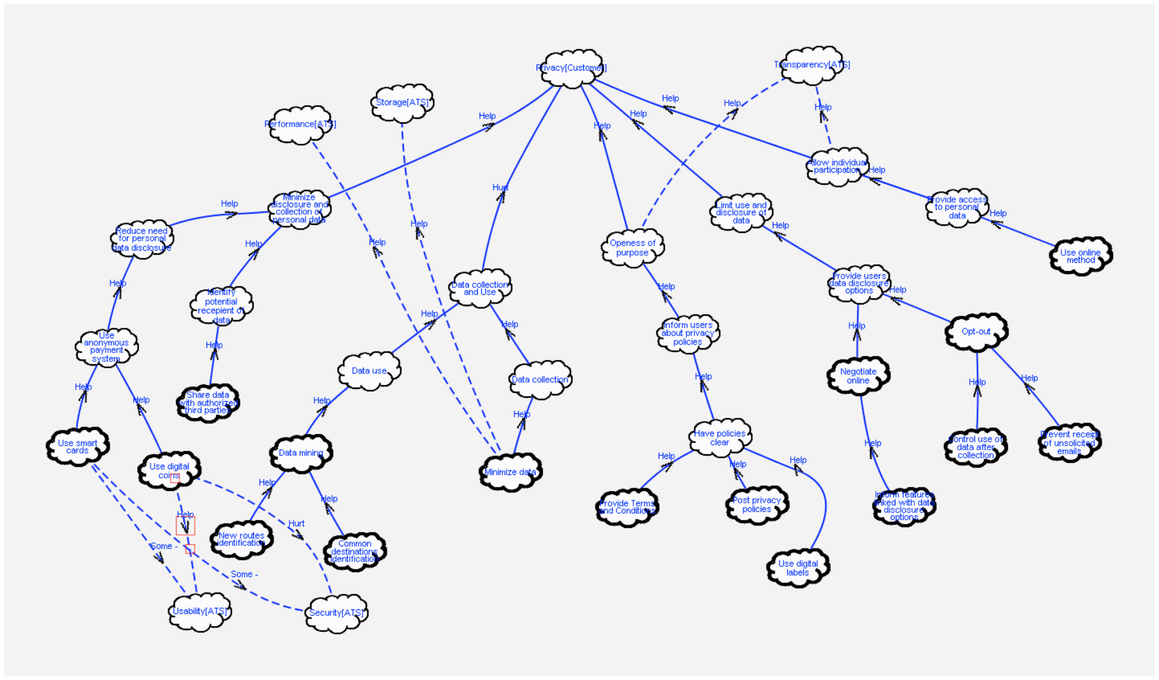

sd OrderTaxi
Customer
:TripView
:TripController
:Trip
:TripDAO
request()
orderTaxi()
createTrip()
:trip
arrangePossibleTrip(String status)
persistPossibleTrip(Trip trip)
persist()
:confirmOrder
(from UseCase)

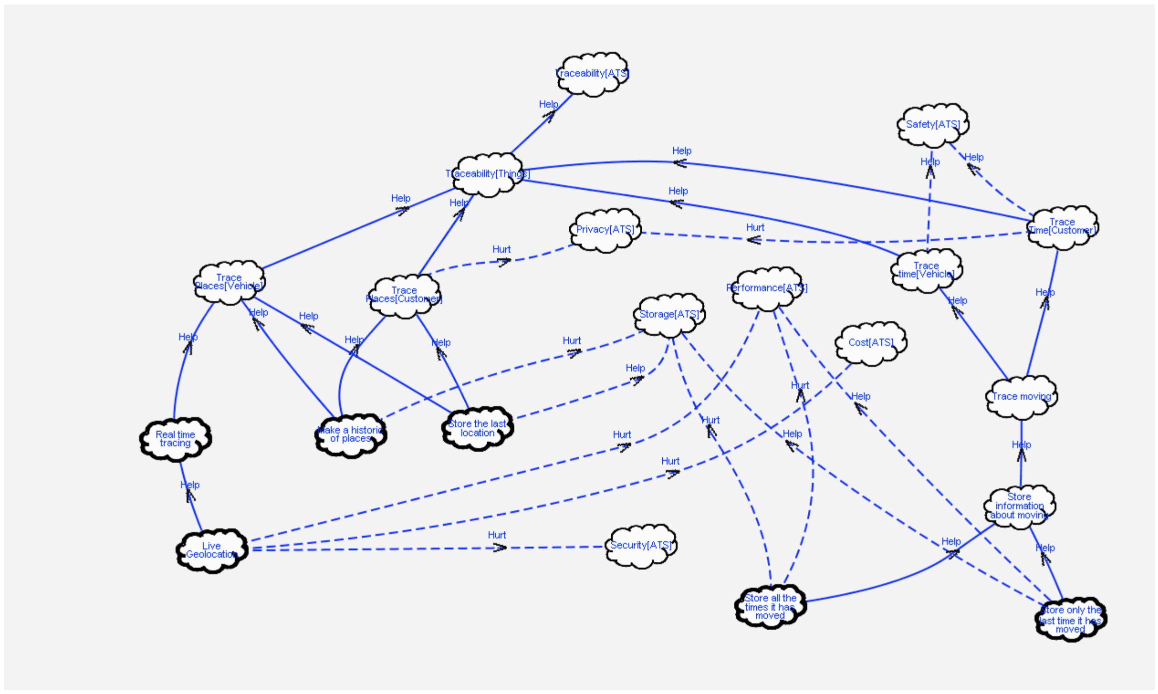

sd SystemAccess
User
:SystemView
:SystemAccessController
:SystemAccessDAO
request()
validation()
login(User)
decryption()
find(User)
:confirmAuthentication
(from UseCase)

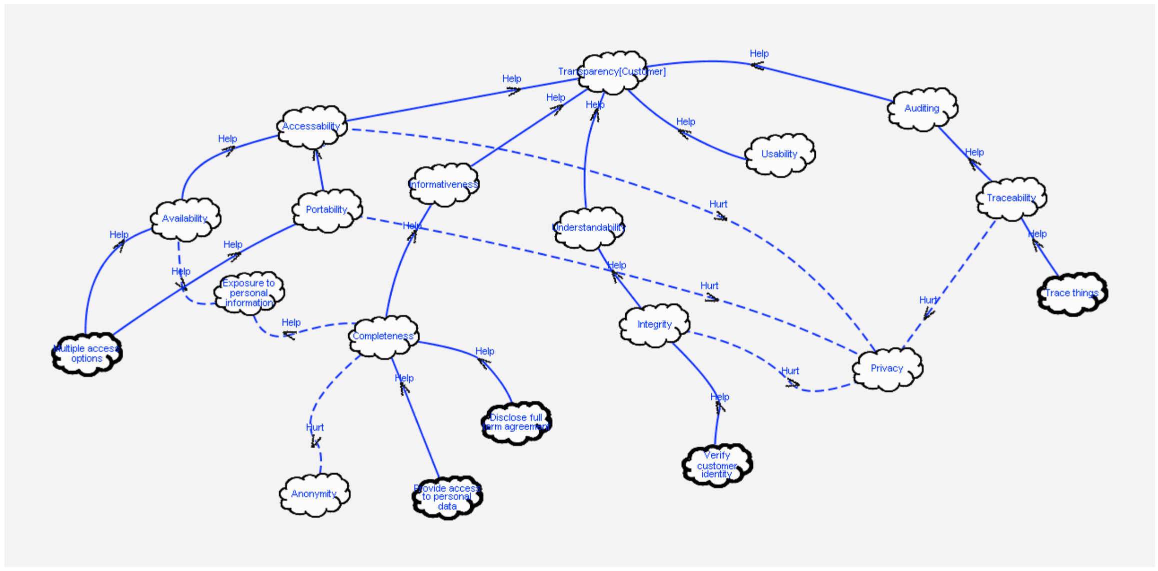
# Appendix B – Authoritative Control

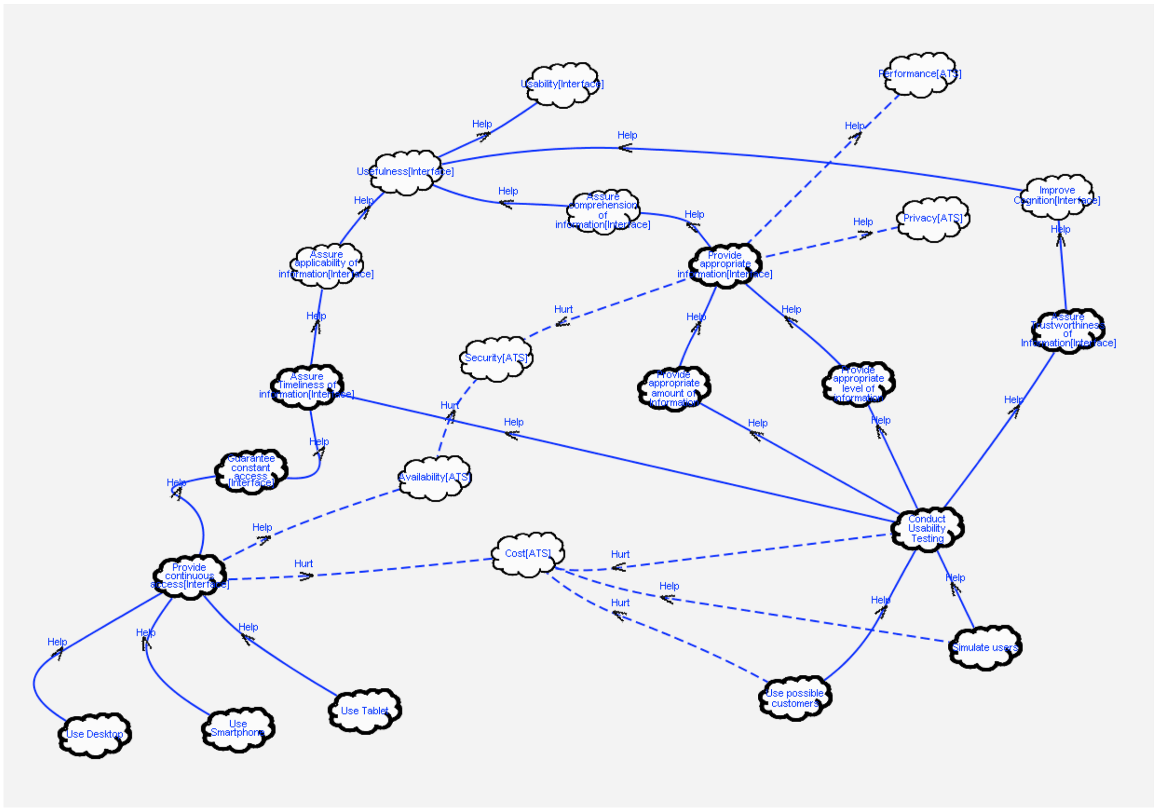



Performance[ATS]
Performance[Time]
Availabilty[ATS]
Cost[ATS]
Performance[Space]
Maintainability[ATS]
Main Memory
Secondary Storage
Response Time
Indexing
Use data partitioning
Use replication
Clustering
Use scalable architecture
Measure expectable workloads
Storage[ATS]
Use RAID
Provisioning new servers during high workloads
Benchmarking
Help
Hurt


Privacy[Customer]
Transparency[ATS]
Storage[ATS]
Performance[ATS]
Minimize disclosure and collection of personal data
Reduce need for personal data disclosure
Identify potential recepient of data
Data collection and Use
Openess of purpose
Limit use and disclosure of data
Allow individual participation
Provide access to personal data
Use online method
Provide users data disclosure options
Opt-out
Use anonymous payment system
Share data with authorized third parties
Data use
Data collection
Inform users about privacy policies
Negotiate online
Use smart cards
Use digital coins
Data mining
Minimize data
Have policies clear
Control use of data after collection
Prevent receipt of unsolicited emails
New routes identification
Common destinations identification
Provide Terms and Conditions
Post privacy policies
Inform features linked with data disclosure options
Use digital labels
Usability[ATS]
Security[ATS]
Help
Hurt
Some -


Traceability[ATS]
Safety[ATS]
Traceability[Things]
Trace Time[Customer]
Privacy[ATS]
Trace time[Vehicle]
Trace Places[Vehicle]
Trace Places[Customer]
Performance[ATS]
Storage[ATS]
Cost[ATS]
Trace moving
Real time tracing
Make a historic of places
Store the last location
Store information about moving
Live Geolocation
Security[ATS]
Store all the times it has moved
Store only the last time it has moved
Help
Hurt


Transparency[Customer]
Auditing
Accessability
Usability
Informativeness
Availability
Portability
Understandability
Traceability
Exposure to personal information
Trace things
Multiple access options
Completeness
Integrity
Privacy
Disclose full term agreement
Verify customer identity
Anonymity
Provide access to personal data
Help
Hurt


Usability[Interface]
Performance[ATS]
Help
Help
Help
Usefulness[Interface]
Improve Cognition[Interface]
Help
Help
Assure comprehension of information[Interface]
Help
Help
Privacy[ATS]
Help
Assure applicability of information[Interface]
Provide appropriate information[Interface]
Help
Hurt
Help
Assure Trustworthiness of Information[Interface]
Security[ATS]
Help
Assure Timeliness of information[Interface]
Provide appropriate amount of information
Provide appropriate level of information
Help
Hurt
Help
Help
Help
Help
Guarantee constant access [Interface]
Availability[ATS]
Help
Help
Conduct Usability Testing
Cost[ATS]
Hurt
Hurt
Provide continuous access[Interface]
Help
Help
Hurt
Help
Help
Help
Help
Simulate users
Use possible customers
Use Tablet
Use Desktop
Use Smartphone