Non-Functional Requirements for Object-Oriented Modeling

Jaime de Melo Sabát Neto*
Julio Cesar Sampaio do Prado Leite**
Luiz Márcio Cysneiros**

Departamento de Informática, PUC-Rio
R. Marquês de São Vicente, 225
22453-900 - Rio de Janeiro, Brasil
e-mail: { jaime, julio, cysneiro }@inf.puc-rio.br

Abstract. Recently, it has been pointed out that the majority of the requirements engineering methods do not take into account non-functional requirements (NFRs) [10][11]. Consequently, we have been experiencing serious problems during the development of software systems, such as cost and schedule overruns. In order to diminish this negligence of NFRs and its consequences, this work proposes a strategy (OONFR) that brings NFRs to object-oriented modeling. The OONFR strategy uses as input a Language Extended Lexicon of the Universe of Discourse (LEL of UofD) and outputs a class diagram with indications of what classes, attributes, operations and relationships are responsible for satisficing[1] NFRs. This strategy consists of the following activities: build the Language Extended Lexicon of Universe of Discourse-NFR (LEL of UofD-NFR), build the scenarios and build the class diagram.

Keywords: Non-Functional Requirements, Language Extended Lexicon, Scenarios, Object-Oriented

1. Introduction

The world increasingly depends on software systems. Several vital functions of our society require software systems, such as telephony, transport and energy supply. Therefore, low quality software systems can endanger human life and cause environmental/economic damage. The quality attributes of software systems were firstly presented by McCall [1] and Boehm [2]. Nowadays, the ISO 9126 standard claims that a high quality software system must have the following attributes [3]: functionality, reliability, usability, efficiency, maintainability, and portability.

To the best of our knowledge, Yeh was the first researcher to bring to light the concept of non-functional requirements (NFRs) [4] and to reveal the importance of software systems' quality attributes (or NFRs) consideration during the initial phases of software development. In the sequel, Roman included the NFR concept into a taxonomy of requirements engineering issues [6], which corroborate the importance of software systems' quality attributes consideration during the definition phase of that systems. The NFR concept was also included into a *curriculum module* named

* Supported by CAPES
** Supported by CNPq
[1] We use the *satisfice* term to indicate that NFRs are satisfied within acceptable limits [8]

Software Requirements [7], which aims to be the basic material of every software requirements course. Thus, NFRs were presented a long time ago and are now part of several requirements engineering syllabus. However, NFRs are less understood than other less critical factors of software development [8]. Consequently, the majority of requirements engineering methods do not take into account NFRs [4][6][9][10][11].

There are several reasons for the negligence of NFRs by the requirements engineering methods, such as great diversity of NFRs, NFRs' dependency on design solutions (e.g. performance), NFRs' subjective nature (e.g. usability), possibility of conflicts among NFRs, and imprecise distinction between NFRs and functional requirements. The consequences of neglecting NFRs are frequently more severe than the consequences of functional requirements' omissions [10]. As consequences of neglecting NFRs, we have some of the well-known problems of the software development, such as cost and schedule overruns, software systems discontinuation (e.g. the London Ambulance Service System [12]), incompleteness of requirements, and dissatisfaction of software systems' clients/users. In addition, we do recognize that NFRs' omission is one of several causes for the software engineering problems quoted above.

Since the majority of requirements engineering methods do not consider NFRs, which leads to serious software development problems, this work proposes a strategy that integrates NFRs into object-oriented modeling. In other words, we propose an object-oriented strategy that gives support to the elicitation and modeling of NFRs and its satisficing strategies (ways of satisficing NFRs that are identified in the literature or software development experiences). In Section 2, we present the concepts required to understand the proposed strategy. In Section 3, we show the activities of the proposed strategy along with examples extracted from a Laboratory Information System (LIS) [20]. In Section 4, we show the main contributions of the present work and some future directions.

2. Basic Concepts

2.1. Non-functional Requirements

Non-functional requirements (NFRs) are quality attributes or constraints of software systems or software system development processes. NFRs can be classified as primary NFRs or specific NFRs [21]. Primary NFRs (e.g. accuracy) have a high level of abstraction and can be decomposed in specific NFRs. Specific NFRs (e.g. value accuracy) have a greater level of detail and show aspects of primary NFRs. In this work, we use the notion of NFR satisficing [8], which indicates that NFRs are satisfied within acceptable limits.

Satisficing strategies (e.g. accuracy auditing) are ways of satisficing NFRs, which are identified in the literature or software system development experiences. In general, satisficing strategies are functional requirements and can also have a negative influence on the satisficing of other NFRs. These negative influences are used to infer conflicts between primary NFRs and between specific NFRs.

2.2. Language Extended Lexicon (LEL of UofD)

Universe of Discourse (UofD) is the general context where the software system should be developed and operated. The Language Extended Lexicon of the UofD describes the language used by the actors and information sources of the UofD. The LEL of UofD (hereafter LEL) is composed by entries, which describe symbols of the language of the UofD, through notions and behavioral responses. These entries can be classified as subject, verb, object, and state. There are heuristics for each entry class that describe what information must be registered on the entries' notions and behavioral responses. During the construction of the LEL, the circularity and minimal vocabulary principles must be followed. The circularity principle prescribes the maximization of the usage of LEL symbols when describing LEL entries. The minimal vocabulary principle prescribes the minimization of the usage of symbols exterior to the LEL when describing LEL entries. As a consequence of the circularity principle, the LEL is a hypertext. LEL entries are the nodes of the hypertext. LEL symbols that appear in the LEL entries are the links of the hypertext. Figure 1 depicts two entries of the LEL of the Laboratory Information System (LIS), which are classified respectively as verb and object.

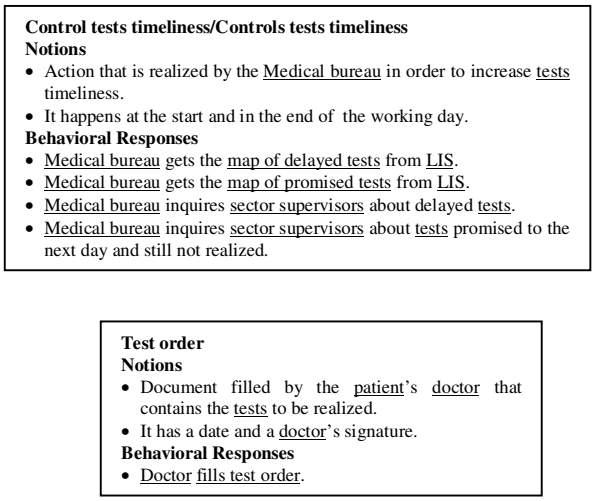


Figure 1. Entries *control tests timeliness* and *test order* of a LEL of UofD

2.3. NFRs' Language Extended Lexicon / NFRs' Knowledge Base

The NFRs' Language Extended Lexicon (NFR LEL) register the language, associated with NFRs, that should be used by requirements and software engineers. The symbols of this language are primary NFRs, secondary NFRs, and satisficing strategies. As the LEL, the NFR LEL is composed of entries, must follow the principles of circularity and minimal vocabulary, and can be presented as a hypertext. However, the NFR LEL entries are classified as primary NFR, secondary NFR, and satisficing strategy. There are templates (Figure 2) for each entry class that prescribe what to register on the entries' notions and behavioral responses. Since these LEL

entries describe NFRs and satisficing strategies, it can be viewed as an NFRs' knowledge base and, therefore, is an important support to the elicitation of NFRs and satisficing strategies.

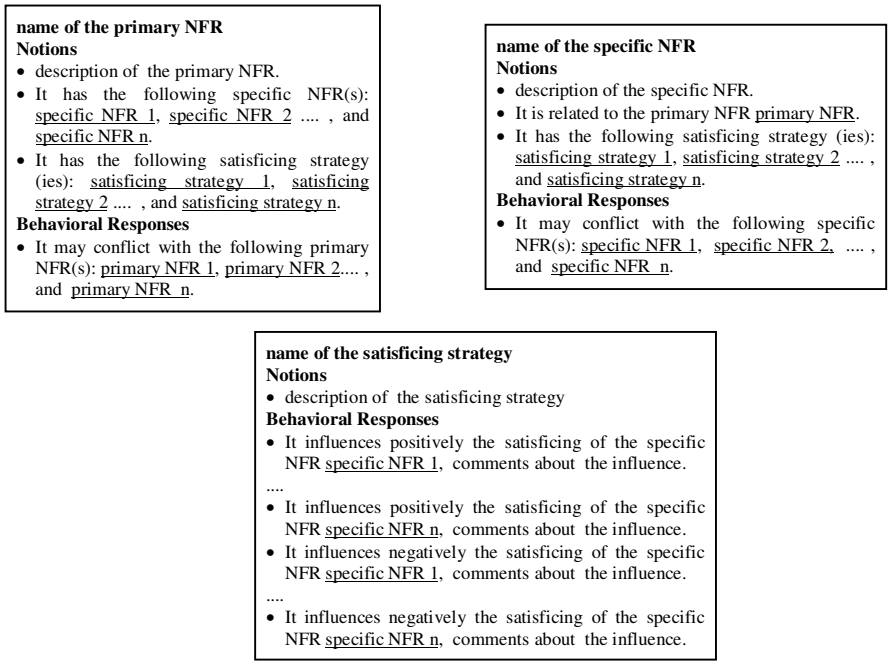


Figure 2. Templates for NFR LEL entries

2.4. Scenarios

In this work, we use the scenario model proposed by Leite [16]. The main points of this proposal are the following: scenarios describe situations in the macrosystem; scenarios evolve during software development; scenarios are naturally linked to the LEL; and scenarios are described in natural language. Each scenario has title, goal, context, actors, resources and episodes. Resources and episodes may have constraints. Episode constraints record the satisficing strategies specialized by episodes and the specific NFRs satisficed by episodes. Resource constraints record the specific NFRs related with resources. In Figure 3, we present the scenario *Medical bureau controls tests timeliness*, which occurs at the LIS.

Title: Medical bureau controls tests timeliness.
Goal: Increase the tests timeliness.
Context: At the start and at the end of the working day.
Actors: medical bureau.
Resources: test **Constraint:** must have timely accuracy; map of delayed tests; map of promised tests.
Episodes
Medical bureau gets the map of delayed tests from the LIS.
Constraint: test must have timely accuracy, using accuracy auditing as satisficing strategy
Medical bureau gets the map of promised tests from the LIS.
Constraint: test must have timely accuracy, using accuracy auditing as satisficing strategy
Medical bureau inquires sector supervisors about delayed tests.
Medical bureau inquires sector supervisors about tests promised to the next day and still not realized

Figure 3. Scenario *Medical bureau controls tests timeliness*

3. OONFR Strategy

The OONFR strategy uses, as input, a LEL and outputs a conceptual class diagram with indications of what classes, attributes, operations, and relationships are responsible for satisficing NFRs. In other words, the class diagram has signals of what classes, attributes, operations, and relationships specialize satisficing strategies and satisfice NFRs. Since the class diagram is conceptual, it has only semantic classes, which are identified within the problem space/UofD [13]. The OONFR strategy consists of the following activities: build the Language Extended Lexicon of the Universe of Discourse-NFR (LEL of UofD-NFR), build the scenarios and build the class diagram. Figure 4 portrays an SADT diagram for the OONFR strategy.

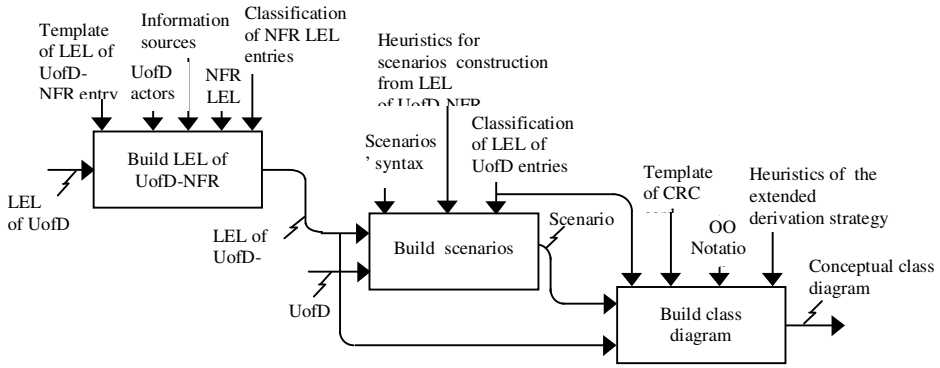


Figure 4. SADT diagram for the OONFR strategy

3.1. Building the LEL of UofD-NFR

In this activity, the Language Extended Lexicon of UofD-NFR (LEL of UofD-NFR) is built from the LEL. The LEL of UofD-NFR is a LEL that has indications of what entries, notions and behavioral responses specialize satisficing strategies and satisfice primary and specific NFRs. These NFRs and satisficing strategies are described by NFR LEL entries, which are included in the LEL of UofD-NFR. The entries, notions, and behavioral responses that specialize satisficing strategies and

satisfice NFRs may already be in the LEL or may be elicited during the LEL of UofD-NFR construction.

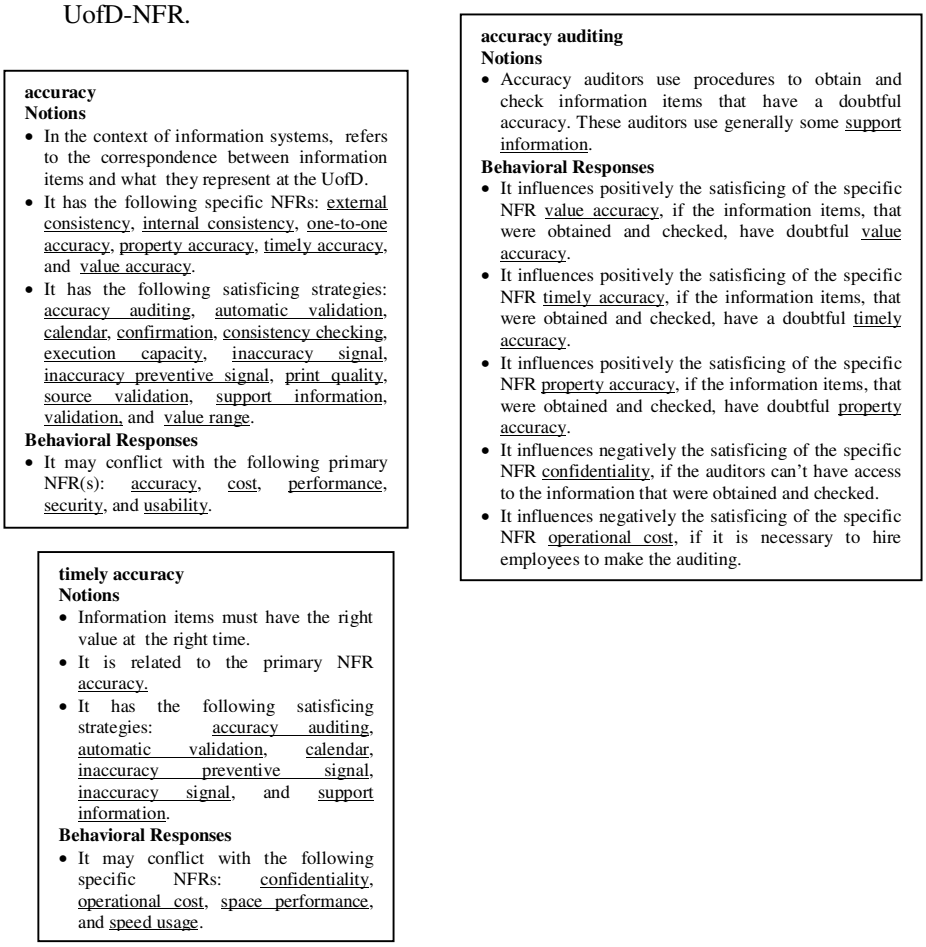
The LEL of UofD-NFR construction process is composed by the following steps: verify notions that specialize satisficing strategies and satisfice NFRs, verify behavioral responses that specialize satisficing strategies and satisfice NFRs, and elicit new entries, notions and behavioral responses. We present below the LEL of UofD-NFR construction process along with examples extracted from the LIS. The underlined words are symbols of the LEL of UofD-NFR.

1. Obtain all the NFR LEL entries that are classified as satisficing strategy, primary NFR, and specific NFR.
   Example: During the verification of the NFR LEL, we obtained the entries that are classified as satisficing strategy, such as *accuracy auditing* (Figure 5), *alarm*, *authentication*, *confirmation*, *consistency checking*, *identification*, *inaccuracy signal*, *support information*, *validation*, and *value range*. We also got the entries that are classified as primary NFR, such as *accuracy* (Figure 5), *cost*, *performance*, *security*, and *usability*. Finally, we obtained the entries that are classified as specific NFR, such as *availability*, *confidentiality*, *development cost*, *learning facility*, *operational cost*, *property accuracy*, *space performance*, *speed usage*, *timely accuracy* (Figure 5), and *value accuracy*.

For each LEL entry, do:

2. Verify if the LEL entry specializes a satisficing strategy and satisfices NFRs through the usage of NFR LEL entries. If the LEL entry specializes a satisficing strategy and satisfices NFRs, include new notions for the entry that register these information. In addition, add new behavioral responses for the entry that register information about NFRs conflicts.
   Example: After the verification of the NFR LEL entries that are classified as satisficing strategy, we realized that the *control tests timeliness* LEL entry (Figure 1) specializes the entry *accuracy auditing* of the NFR LEL (Figure 5). As a result, we added the following notion for this entry: *It is a specialization of the satisficing strategy* <u>*accuracy auditing*</u>. The entry *accuracy auditing* of the NFR LEL was included in the LEL of UofD-NFR. After checking the behavioral responses of the *accuracy auditing* entry, we identified the specific NFRs that are positively influenced by this satisficing strategy: *property accuracy*, *timely accuracy*, and *value accuracy*. We chose the timely accuracy specific NFR as the one that the *control tests timeliness* entry aims to satisfice. During the verification of the NFR LEL entry that describes the *timely accuracy* specific NFR (Figure 5), we identified the *accuracy* NFR as its primary NFR. Therefore, we have identified the NFRs that this entry aims to satisfice. Afterwards, we registered these information on the *control tests timeliness* entry through the addition of the following notions: *it aims to satisfice the following primary NFR:* <u>*accuracy*</u> and *it aims to satisfice the following specific NFR:* <u>*timely accuracy*</u>. The *accuracy* and *timely accuracy* NFR LEL entries were included in the LEL of UofD-NFR. In the sequel, after the verification of the behavioral responses of the accuracy auditing NFR LEL entry, we identified the specific NFRs that were negatively influenced by this satisficing strategy: *confidentiality* and *operational cost*. After the verification of the NFR LEL entries that describe these specific NFRs, we

identified the *security* and *cost* NFRs as its primary NFRs. Thus, we have identified the NFRs with which the *control tests timeliness* entry may conflict. Next, we registered these information on the *control tests timeliness* entry through the inclusion of the following behavioral responses: *It may conflict with the following primary NFRs: cost and security* and *It may conflict with the following specific NFRs: confidentiality and operational cost*. The *confidentiality*, *cost*, *operational cost*, and *security* NFR LEL entries were included in the LEL of UofD-NFR. Figure 7 depicts the entry *control tests timeliness* of the LEL of UofD-NFR.



accuracy
Notions
- In the context of information systems, refers to the correspondence between information items and what they represent at the UofD.
- It has the following specific NFRs: external consistency, internal consistency, one-to-one accuracy, property accuracy, timely accuracy, and value accuracy.
- It has the following satisficing strategies: accuracy auditing, automatic validation, calendar, confirmation, consistency checking, execution capacity, inaccuracy signal, inaccuracy preventive signal, print quality, source validation, support information, validation, and value range.

Behavioral Responses
- It may conflict with the following primary NFR(s): accuracy, cost, performance, security, and usability.

accuracy auditing
Notions
- Accuracy auditors use procedures to obtain and check information items that have a doubtful accuracy. These auditors use generally some support information.

Behavioral Responses
- It influences positively the satisficing of the specific NFR value accuracy, if the information items, that were obtained and checked, have doubtful value accuracy.
- It influences positively the satisficing of the specific NFR timely accuracy, if the information items, that were obtained and checked, have a doubtful timely accuracy.
- It influences positively the satisficing of the specific NFR property accuracy, if the information items, that were obtained and checked, have doubtful property accuracy.
- It influences negatively the satisficing of the specific NFR confidentiality, if the auditors can't have access to the information that were obtained and checked.
- It influences negatively the satisficing of the specific NFR operational cost, if it is necessary to hire employees to make the auditing.

timely accuracy
Notions
- Information items must have the right value at the right time.
- It is related to the primary NFR accuracy.
- It has the following satisficing strategies: accuracy auditing, automatic validation, calendar, inaccuracy preventive signal, inaccuracy signal, and support information.

Behavioral Responses
- It may conflict with the following specific NFRs: confidentiality, operational cost, space performance, and speed usage.

Figure 5. Entries *accuracy*, *timely accuracy*, and *accuracy auditing* of the NFR LEL

For each notion of the LEL entry, do:

3. Verify if the notion specializes a satisficing strategy and satisfices NFRs through the use of NFR LEL entries. If the notion specializes a satisficing strategy and satisfices NFRs, register these information at the end of the notion, enclosed by parenthesis.

Example: After the verification of the NFR LEL entries that are classified as satisficing strategy, we realized that the notion *it has a minimum and maximum value* of the *test* entry of the LEL specializes the *value range* satisficing strategy. In the sequel, after the verification of the behavioral responses of the range value NFR LEL entry, we identified that this satisficing strategy influences positively the following specific NFRs: *low error rate*, *property accuracy*, and *value accuracy*. Since the tests' results must be correct, we chose the value accuracy specific NFR as the one that the notion aims to satisfice. Finally, we modified this notion to indicate the satisficing strategy that it specializes and the specific NFR that it satisfices: *it has a minimum and maximum value (value range satisficing value accuracy)*. The *test* entry of the LEL of UofD-NFR is depicted in Figure 6.

Test/Tests
Notions
- Process that determines enzyme values or presence of abnormal elements.
- It has a minimum and maximum value. (value range satisficing value accuracy)
- It may need complementary data types. (support information satisficing value accuracy)
- It has electronic signature range. (support information satisficing value accuracy)
- It has edition security range. (support information satisficing value accuracy)
- It may have normality values. (support information satisficing value accuracy)
- It may be critical.
- It has result.
- It may have repetition mark.

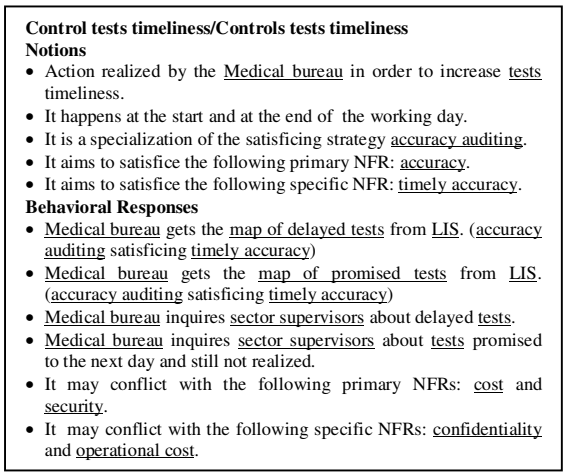
Behavioral Responses
- Employee manually makes test.
- Employee programs mono-directional analyzer to make tests.
- LIS programs bi-directional analyzer to make tests.
- Employee repeats test.

Figure 6. Entry *test* of the LEL of UofD-NFR

For each behavioral response of the LEL entry, do:

4. Verify if the behavioral response specializes a satisficing strategy and satisfices NFRs through the use of NFR LEL entries. If the behavioral response specializes a satisficing strategy and satisfices NFRs, register these information at the end of the behavioral response, enclosed by parenthesis.
   Example: After the verification of the NFR LEL entries, we realized that the behavioral response *Medical bureau gets the map of delayed tests from the LIS* that belongs to the *control tests timeliness* LEL entry (Figure 1) specializes the *accuracy auditing* satisficing strategy. In the sequel, after the verification of the behavioral responses of the *accuracy auditing* NFR LEL entry, we identified the specific NFRs that this strategy influences positively: *property accuracy*, *timely accuracy*, and *value accuracy*. Since the tests must have the right value at the right time, we chose the timely accuracy specific NFR as the one that the behavioral response aims to satisfice. Finally, we modified the behavioral response to indicate the satisficing strategy that it specializes and the specific NFR that it satisfices: *Medical bureau gets the map of delayed tests from the LIS (accuracy*

auditing satisficing timely accuracy). The entry *control tests timeliness* of the LEL of UofD-NFR is depicted in Figure 7.



Control tests timeliness/Controls tests timeliness
Notions
- Action realized by the Medical bureau in order to increase tests timeliness.
- It happens at the start and at the end of the working day.
- It is a specialization of the satisficing strategy accuracy auditing.
- It aims to satisfice the following primary NFR: accuracy.
- It aims to satisfice the following specific NFR: timely accuracy.

Behavioral Responses
- Medical bureau gets the map of delayed tests from LIS. (accuracy auditing satisficing timely accuracy)
- Medical bureau gets the map of promised tests from LIS. (accuracy auditing satisficing timely accuracy)
- Medical bureau inquires sector supervisors about delayed tests.
- Medical bureau inquires sector supervisors about tests promised to the next day and still not realized.
- It may conflict with the following primary NFRs: cost and security.
- It may conflict with the following specific NFRs: confidentiality and operational cost.

Figure 7. Entry *control tests timeliness* of the LEL of UofD-NFR

5. Identify new entries, notions and behavioral responses through the use of NFR LEL entries. At the previous steps, we verify if these entries, notions and behavioral responses specialize satisficing strategies and satisfice NFRs.
   Example: After the verification of the NFR LEL entries that are classified as primary NFR, we realized that the *accuracy* entry was relevant to the *test order* entry of the LEL (Figure 1). During the verification of the notions of the *accuracy* entry, we obtained its specific NFRs, such as *property accuracy*, *timely accuracy*, and *value accuracy*. We considered that these three specific NFRs are important to the *test order* LEL entry. During the verification of the notions of these specific NFR entries, we obtained its satisficing strategies, such as *automatic validation* and *validation.* Through the use of the *validation* NFR LEL entry, we identified the *check test order* entry of the LEL. Through the usage of the *automatic validation* NFR LEL entry, we identified the *check test order automatically* entry. We also identified the following behavioral responses of the *test order* entry: *attendant checks test order* and *LIS checks test order automatically*.

3.2. Building the Scenarios

The activity of building scenarios is an extension of the scenarios construction process proposed by Hadad [15]. This extension consists of the usage of the LEL of UofD-NFR as the process input and of the addition of heuristics to elicit episode and resource constraints. The following steps compose the scenarios construction process: *identify actors of the UofD*, *identify candidate scenarios*, and *describe candidate scenarios*. We present below the scenarios construction process in tandem with examples extracted from the LIS. The underlined words are symbols of the LEL of UofD-NFR.

Step 1: Identify actors of the UofD. Take the LEL of UofD-NFR entries that are classified as subject, which represent the actors of the UofD.
Example: After the verification of the LEL of UofD-NFR of the LIS, we identified the actors of the UofD, such as: *analyzer*, *attendant*, *doctor*, *LIS*, *medical bureau*, *patient*, and *supervisor*.

Step 2: Identify candidate scenarios. Take the behavioral responses of the LEL of UofD-NFR entries that describe the actors of the UofD. Next, eliminate equal behavioral responses. The behavioral responses obtained are the titles of the candidate scenarios.
Example: After the verification of the behavioral responses of the *medical bureau* entry of the LEL of UofD-NFR, we identified some candidate scenarios, such as *Medical bureau controls tests timeliness* and *Medical bureau revises patient's report*.

Step 3: Describe candidate scenarios.
For each candidate scenario identified at step 2, do:
If the scenario title has an LEL of UofD-NFR entry that is classified as verb, then:
Example: We present below the description of the candidate scenario *Medical bureau controls tests timeliness* (Figure 3).

1. Take the LEL of UofD-NFR entry that is classified as verb and is present at the scenario title.
   Example: We took the *controls tests timeliness* entry (Figure 7).
2. Define the scenario goal through the verification of the scenario title and through the checking of the notions of the LEL of UofD-NFR entry obtained at step 1.
   Example: After the verification of the first notion of the *controls tests timeliness* entry of the LEL of UofD-NFR (Figure 7), we defined the following goal for the scenario: *increase the tests timeliness*.
3. Define the scenario context through the verification of the notions of the LEL of UofD-NFR entry obtained at step 1. Check also if exist a precedence order among the behavioral response that originated the scenario and other behavioral responses. If this precedence order exists, then register the behavioral response, which precedes the scenario, on the scenario context.
   Example: After the verification of the second notion of the *controls tests timeliness* entry of the LEL of UofD-NFR (Figure 7), we defined the following context for the scenario: *at the start and at the end of the working day*.
4. Define the scenario episodes from the behavioral responses of the LEL of UofD-NFR entry obtained at step 1. If the behavioral response from which the episode was defined specializes a satisficing strategy and satisfices NFRs, then the episode specializes the same satisficing strategy and satisfices the same NFRs. These information are registered on the episode constraint, which must obey the following syntax: ***Constraint:*** *1{resource}n must have 1{SpecificNFR}n, using satisficing strategy that the episode specializes as satisficing strategy*.
   Example: From the behavioral response *Medical bureau gets the map of delayed tests from LIS (accuracy auditing satisficing timely accuracy)* of the entry *controls tests timeliness* (Figure 7), we defined the following episode for the scenario *Medical bureau controls tests timeliness*: *Medical bureau gets the map*

of delayed tests from LIS. Since that behavioral response specializes the *accuracy auditing* satisficing strategy and satisfices the *timely accuracy* specific NFR, we defined the following constraint for this episode: ***Constraint:*** *test must have timely accuracy, using accuracy auditing as satisficing strategy*.

5. The scenario actors are LEL of UofD-NFR entries that are present at the scenario episodes, are classified as subject and execute actions in the scenario.
   Example: After the verification of the episodes of the scenario *Medical bureau controls tests timeliness* (Figure 3), we identified the *medical bureau* actor.
6. The scenario resources are LEL of UofD-NFR entries that are present at the scenario episodes and are classified as object. The constraints of episodes must be verified in order to identify constraints of resources, which must obey the following syntax: ***Constraint:*** *must have 1{SpecificNFR}n*.
   Example: After the verification of the episodes of the scenario *Medical bureau controls tests timeliness* (Figure 3), we identified the following resources: *test*, *map of delayed tests*, and *map of promised tests.* After the verification of the episodes restrictions of this scenario, we elicited the following constraint for the *test* resource: ***Constraint:*** *must have timely accuracy*.

3.3. Building the Class Diagram

The activity of building class diagram is an extension of the derivation strategy proposed by Leite [17] and Leonardi [18]. This extension consists of the usage, as input, of the LEL of UofD-NFR along with the scenarios built from this LEL of UofD-NFR and of the modification of the derivation strategy original steps in order to consider the information about NFRs and satisficing strategies that are present at that models (LEL of UofD-NFR and scenarios). The class diagram construction process consists of the following steps: *identify primary classes and their responsibilities*, *identify secondary classes and their responsibilities*, *refine responsibilities and collaborations*, and *build conceptual class diagram*. Primary classes are active entities of UofD, such as persons and organizations. Secondary classes are passive entities of UofD (generally, they are data repositories). We present below the class diagram construction process along with examples extracted from the LIS. The underlined words are symbols of the LEL of UofD-NFR.

Step 1: Identify primary classes and their responsibilities. The actors of the scenarios are primary classes. Eliminate the classes that are redundant. From the behavioral responses of the LEL of UofD-NFR entry that describes the class, we define the class responsibilities. If the behavioral response specializes a satisficing strategy and satisfices NFRs, the responsibility specializes the same satisficing strategy and satisfices the same NFRs. These information are registered at the end of the responsibility, enclosed by parenthesis.
Example: During the verification of the actors of the scenario *Medical bureau controls tests timeliness* (Figure 3), we identified the medical bureau primary class. During the verification of the behavioral responses of the *medical bureau* entry of the LEL of UofD-NFR, we elicited the following responsibilities of the medical bureau class: *revise patient's report (validation satisficing value accuracy)* and *control tests timeliness (accuracy auditing satisficing timely accuracy)*. The first responsibility

specializes the *validation* satisficing strategy and satisfices the *value accuracy* specific NFR. The second responsibility specializes the *accuracy auditing* satisficing strategy and satisfices the *timely accuracy* specific NFR.

Step 2: Identify secondary classes and their responsibilities. The union of the resources of scenarios and the LEL of UofD-NFR entries, that are classified as objects and are present at primary classes' responsibilities, contains all the secondary classes. Eliminate the classes that are redundant or are attributes of other classes. The secondary classes' responsibilities are generally of the form *register information* and *supply information*. The information, registered and supplied by secondary classes, is elicited from the notions of the LEL of UofD-NFR entries that describe these classes.
Example: During the verification of the resources of the scenario *Medical bureau controls tests timeliness* (Figure 3), we identified the following secondary classes: *test*, *map of delayed tests*, and *map of promised tests.* During the verification of the notions of the *test* entry of the LEL of UofD-NFR, we identified the information that the test class register and supply: *minimum value*, *maximum value*, *complementary data types*, *electronic signature range*, *edition security range*, *normality values*, *criticality*, *result*, and *repetition mark*. Thus, the responsibilities of the *test* class are to register and supply these information.

Step 3: Refine responsibilities and collaborations. Build a CRC card for each class. If the LEL of UofD-NFR entry, that describes the class, specializes a satisficing strategy and satisfices NFRs, the CRC card specializes the same satisficing strategy and satisfices the same NFRs. Thus, register these information on the justification part of the CRC card, following the syntax: *Is a specialization of the satisficing strategy that the class specializes satisficing strategy and satisfices the following specific NFRs: 1{SpecificNFR}n.* Place the responsibilities, that were identified previously, in the responsibilities part of the CRC card. Verify the notions of the LEL of UofD-NFR entry that describes the class in order to identify generalization, specialization and aggregation relationships. In the sequel, register these relationships' information on the *superclass, subclasses, and parts* slots of the CRC card. If the CRC card describes a primary class, identify additional responsibilities for this class through the verification of the episodes of the scenarios in which the class participates as an actor. If the episode has a constraint that register the satisficing strategy that it specializes and the NFRs that is satisfices, the responsibility that was obtained from this episode specializes the same satisficing strategy and satisfices the same NFRs. Thus, register this information at the end of the responsibility, enclosed by parenthesis. For each responsibility of a class, verify the classes with which this class needs to cooperate to fulfill the responsibility. These classes must be placed at the collaborations slot of the CRC card.
Example: We built a CRC card (Figure 8) for the *medical bureau* class. The responsibilities identified previously were placed in the responsibilities slot of the CRC card. From the first episode of the scenario *Medical bureau controls tests timeliness* (Figure 3), we identified the following responsibility for the medical bureau class: *get the map of delayed tests from LIS. (accuracy auditing satisficing timely accuracy).* This responsibility specializes the *accuracy auditing* satisficing strategy and satisfices the *timely accuracy* specific NFR. To fulfill the responsibility

sign report, the medical bureau class needs to cooperate with the *report* class. To carry out the responsibility *control tests timeliness*, the medical bureau class necessitates the cooperation of the following classes: *test*, *report*, *LIS*, *map of delayed tests*, *map of promised tests*, and *supervisor*. All these classes were placed at the collaborations slot of the *medical bureau* CRC card.

medical bureau	
Responsibilities • revise patient's report (validation satisficing value accuracy) • analyze tests' results of the report regarding the normality values. (validation satisficing value accuracy) • analyze tests' results of the report regarding complementary data. (validation satisficing value accuracy) • sign report. • control tests timeliness (accuracy auditing satisficing timely accuracy) • get the map of delayed tests from LIS. (accuracy auditing satisficing timely accuracy) • get the map of promised tests from LIS. (accuracy auditing satisficing timely accuracy) • inquire sector supervisors about delayed tests. • inquire sector supervisors about tests promised to the next day and still not realized.	**Collaborations** • test • report • LIS • map of delayed tests • map of promised tests • patient • supervisor

Figure 8. medical bureau CRC card

Step 4: Build a Conceptual Class Diagram.

4.1. Identify attributes and operations. For each CRC card, represent a class according to whatever object-oriented notation you use. If the CRC card specializes a satisficing strategy and satisfices NFRs, include the prefix NR_ in the name of the class. Inspect the notions of the LEL of UofD-NFR entry that describes the class and the responsibilities of the class in order to identify the class attributes. If the notion/responsibility specializes a satisficing strategy and satisfices NFRs, add the prefix NR_ to the name of the attribute. From the responsibilities of the class, which are registered on the CRC card, define the class operations. If the responsibility specializes a satisficing strategy and satisfices NFRs, add the prefix NR_ to the name of the operation.

medical bureau
NR_ReviseReport(report) NR_AnalyseResultsRegardingNormalityValues(test, report,normality values) NR_AnalyseResultsRegardingComplementaryData(test,report,complementary data) SignReport(report) NR_ControlTestsTimeliness() NR_GetDelayedTestsMap() NR_GetPromisedTestsMap() AskSupervisorAboutDelayedTests(supervisor, tests) AskSupervisorAboutPromisedTests(supervisor, tests)

Figure 9. medical bureau class

Example 1: We represented a class (Figure 9) for the medical bureau CRC card (Figure 8), according to UML[19]. From the responsibility *control tests timeliness (accuracy auditing satisficing timely accuracy)* of the medical bureau class, we defined the operation *NR_ControlTestsTimeliness()*. Since that responsibility specializes the *accuracy auditing* satisficing strategy and satisfices the *timely accuracy* specific NFR, we added the prefix NR_ to the name of this operation. We did not identify any attribute for this class.

4.2. Identify relationships. For each CRC card, represent a class according to whatever object-oriented notation and without showing attributes and operations. If the CRC card specializes a satisficing strategy and satisfices NFRs, add the prefix NR_ to the class name. For each primary class, verify the collaborations slot of its CRC card in order to identify the classes with which the primary class collaborates. Represent associations for each of these collaborations. Verify the responsibilities of the CRC card and the notions of the LEL of UofD-NFR, that describe the primary class, in order to acquire the name of the associations. If the notion/responsibility specializes a satisficing strategy and satisfices NFRs, include the prefix NR_ in the association name. The generalization, specialization and aggregation relationships are represented, using the information registered on the *superclass*, *subclasses*, and *parts* slots of CRC cards.

Example 1: We represented classes for each CRC card of the LIS according to UML [19], such as *LIS*, *map*, *RN_map of delayed tests*, *RN_map of promised tests*, *medical bureau*, *patient*, *report*, *supervisor*, and *test*. The prefix NR_ was added to the name of the classes that specialize satisficing strategies and satisfice NFRs. For example, the class *RN_map of delayed tests* specializes the *support information* satisficing strategy and satisfices the *value accuracy* specific NFR.

Example 2: After the verification of the *medical bureau* CRC card, we obtained the classes with which the *medical bureau* primary class needs to collaborate to fulfill its responsibilities: *test*, *report*, *LIS*, *map of delayed tests*, *map of promised tests*, *patient*, and *supervisor*. For each collaboration, we represented an association. For instance, we represented the association *NR_revises* for the collaboration between the *medical bureau* and *report* classes. The name of this association was obtained from the responsibility *revise patient's report (validation satisficing value accuracy)* of the *medical bureau* CRC card. Since this responsibility specializes a satisficing strategy and satisfices a NFR, we added the prefix NR_ to the name of that association. Figure 10 depicts the part of the class diagram of the LIS that was built from the *medical bureau* primary class.

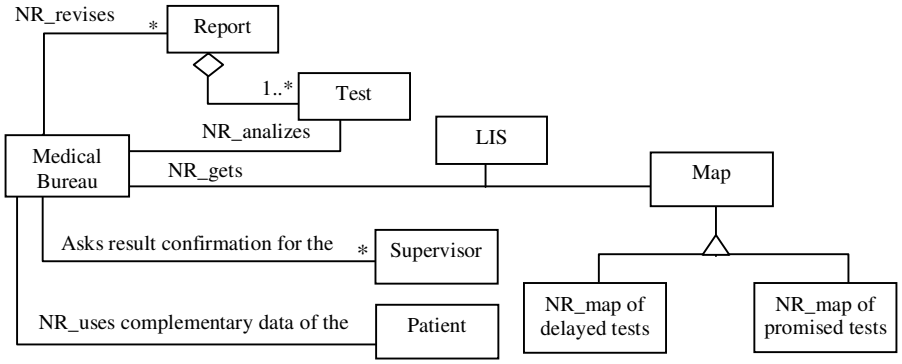


Figure 10. Part of the class diagram of the LIS

4. Conclusion

Since the majority of the requirements engineering methods do not take into account NFRs [4][6][9][10][11], we believe that the main contribution of our work is the support to the elicitation and modeling of NFRs and its satisficing strategies during the construction of LEL, scenarios, CRC cards, and class diagrams. In addition, despite the impossibility of complete requirements, the models built through the OONFR strategy are more complete than others constructed by processes or methods that do not consider NFRs.

The NFR LEL is an important technique that gives support to the elicitation of NFRs and satisficing strategies. It has knowledge about NFRs and satisficing strategies that is used to verify what NFRs and satisficing strategies are relevant at the UofD. Moreover, through the usage of the NFR LEL, we elicit LEL entries that would rarely be identified without its use. During the execution of the OONFR strategy, these new entries result in the elicitation of new scenarios, primary classes, secondary classes, responsibilities, attributes, and operations.

In the KAOS approach [23], goals are used to justify and explain the existence of objects (agents, events, entities, and relationships). In this work, the identification of satisficing strategies specializations and NFRs satisficing is in some form a justification and explanation of the existence of models' components through the use of NFRs. For example, we achieve the justification and explanation of the existence of classes, attributes, operations, and relationships through the use of NFRs.

We intend to extend the OONFR strategy in order to permit the construction of other object-oriented models (e.g. collaboration diagram) with the same support to the elicitation and modeling of NFRs and satisficing strategies. Based on the ideas presented here and in [22], we also aim to extend the UML[19] in order to support the modeling of NFRs and satisficing strategies. Finally, we intend to investigate the LEL of UofD-NFR evolution and its consequences on the models built from this LEL.

5. Bibliography

[1] McCall, J.A. et. al. *Factors in Software Quality*. Vol. 1, 2 e 3, AD/A-049-014/015/055, Nat'l Tech. Information Service, Springfield, Va.,1977.

[2] Boehm, B. *Characteristics of Software Quality*. North Holland Press, 1978.

[3] ISO 9126 *Information Technology – Software Product Evaluation – Quality Characteristics and Guidelines for their Use*, International Organization for Standardization, Geneva, 1992.

[4] Yeh, R. et. al. *Software Requirements: New Directions and Perspectives*. Handbook of Software Engineering, 1984, pp. 519-543.

[5] Macedo, N.A.M. e Leite, J.C.S.P. *Integrando Requisitos Não Funcionais aos Requisitos Baseados em Ações Concretas*. 2º Workshop Ibero-americano de Ingeniería de Requisitos y Ambientes Software (IDEAS´99), San José, Costa Rica, 1999.

[6] Roman, G.-C. *A Taxonomy of Current Issues in Requirements Engineering*. IEEE Computer, 18(4), 1985, pp.14-22.

[7] Brackett, J.W. *Software Requirements*. SEI-CM-19-1.2, Software Engineering Institute, January 1990.

[8] Chung, K.L. *Representing and Using Non-functional Requirements: A Process-oriented approach.* Ph.D. Thesis, Department of Computer Science, University of Toronto, 1993.

[9] Landes, D. and Studer, R.. *The Treatment of Non-Functional Requirements in MIKE.* LNCS 989 (Software Engineering:ESEC'95), Springer-Verlag, 1995, pp. 294-306

[10] Ebert, C.. *Dealing with Nonfunctional in Large Software Systems.* Annals of Software Engineering, 3, 1997, pp. 367-395.

[11] Kotonya, G. and Sommerville, I. *Requirements Engineering: Processes and Techniques.* John Willey & Sons, 1998.

[12] Breitman, K. K., Leite J.C.S.P. and Finkelstein A. *The World's Stage: A Survey on Requirements Engineering Using a Real-Life Case Study*. Journal of the Brazilian Computer Society, vol. 6, n.1, July 1999.

[13] Monarchi, D. et.al. *A Research Typology for Object-Oriented Analysis and Design.* Communications of the ACM, 35, September 1992, pp. 35-47.

[14] Franco, A. P. M. *Métodos e Representação de Suporte à Aquisição de Linguagens de Aplicação.* Dissertação de Mestrado, Departamento de Informática, PUC-Rio, 1992.

[15] Hadad, G. et. al. *Construcción de Escenarios a partir del Léxico Extendido del Lenguage*. JAIIO'97, Argentina, 1997, pp. 65-77

[16] Leite, J.C.S.P. et.al. *Enhancing a Requirements Baseline with Scenarios*. Requirements Engineering, 2(4):184-198, 1997.

[17] Leite, J.C.S.P. e Breitman, K. K. Tutorial: Utilizando Cenários para o Desenvolvimento de Sistemas Orientados a Objetos. Simpósio Brasileiro de Engenharia de Software, Brasil, 1997.

[18] Leonardi, C. et. al. *Una Estrategia de Análisis Orientada a Objetos basada en Escenarios.* Actas II Jornadas de Ingeniaria de Software JIS97, España, Set. 1997.

[19] *UML Document Set*. http://www.rational.com/uml

[20] Cysneiros, L.M. and Leite, J.C.S.P. *Integrating Non-Functional Requirements into Data Modeling*. Fourth IEEE International Symposium on Requirements Engineering, 1999.

[21] Sabát Neto, J. de M. *Integrando Requisitos Não Funcionais à Modelagem Orientada a Objetos.* Dissertação de Mestrado, Departamento de Informática, PUC-Rio, 2000.

[22] Cysneiros, L. M., Sabát Neto, J. de M. e Leite, J. C. S.P. *Integrando Requisitos Não Funcionais na Modelagem Orientada a Objetos*. Anais do Workshop de Engenharia de Requisitos (WER´99), Buenos Aires, Argentina, 1999. pp. 117-128

[23] Dardenne, A.. van Lamsweerde A, Fickas, S.. *Goal Directed Requirements Acquisition*. Science of Computer Programming, Vol. 20 pp: 3-50, Apr. 1993.