Cataloguing Non Functional Requirements as Softgoal Networks

Luiz Marcio Cysneiros – Department of Mathematics and Statistics - Information *Technology Program - York University*
Email: cysneiro@mathstat.yorku.ca

Eric Yu -Faculty of Information Studies
University of Toronto
Email: yu@fis.utoronto.ca

Julio Cesar Sampaio do Prado Leite – Department of Computer Science
University of Toronto on leave from Departamento de Informática PUC- Rio
www.inf.puc-rio.br/~julio

Abstract

Non-Functional Requirements (NFRs) have been frequently neglected or forgotten in software design. In order to treat them as first class citizens it is necessary not only to provide methods and representations, but also to deal with their organization, as to promote future reuse. In this article we offer a preliminary insight on how this knowledge could be organized as catalogues in order to facilitate the sharing and evolution of NFR information. We see that these catalogues could help software designers either if they start from an architectural vision of the system or not.

1. Introduction

Previous work on eliciting and modelling non-functional requirements produced the NFR Framework [3] and an integration strategy to weave functional and non-functional requirements [4]. Both works acknowledge the need for re-using NFR knowledge. Knowledge bases or catalogues are referred in both works as well as in [10] as necessary in order to help the usage of non-functional requirements by requirements engineers. Those catalogues were built not only to store knowledge on NFR elicitation but also to stimulate future reuse of the knowledge stored there. In fact, during two case studies performed by Cysneiros [6] some knowledge embedded in the catalogues presented in [3] was reused. Experience in modeling systems with a goal-oriented approach has shown that these NFR are usually a network of softgoals [3]. Notwithstanding the fact that the idea of catalogues was proposed and to some extent, detailed, we believe there are still several problems that need to be addressed in order to make NFR knowledge available and easy to use. Other works such as [16] and [13] precludes the use of reuse to some NFRs. Most specifically security is deeply dealt with in both works. Although they bring an important contribution to the elicitation process of security requirements they lack a broader approach to NFRs. They also do not stress how to model and deal with the possible conflicts that adopting one strategy for security may bring to another security requirement. Moreover they lack to deal with conflicts between security and other NFRs. It does not require a long effort to see that adopting some strategies to achieve security requirements may conflict with other NFRs such as usability, space and time performance.

All the above mentioned works have one point in common. Retrieving and reusing knowledge on how to achieve certain NFRs and what possible conflicts it might create is still not adequate to facilitate reuse. There should be a way for the requirements engineer to easily address concerns about a specific NFR, how to achieve it and what impacts it may have on the proposed solution space. Section 4 will stress problems related to retrieving and reusing knowledge from catalogues.

This work proposes a way of cataloguing NFRs as softgoal networks building on the faceted schema from [12]. We propose to use a domain-oriented approach using a classification scheme based on a domain-oriented four tuple. We also present a data model to store and retrieve knowledge embedded into NFRs Catalogues. Through the use of this model and the facet classification it is possible to store and retrieve knowledge embedded into these catalogues using traditional data base schemas. We envisage this knowledge to be made available online for public consultation and controlled update

We will briefly introduce the ideas of the NFR framework, and of goal-oriented requirements engineering in Section 2 and 3. We will detail the problems with the existing cataloguing practices in Section 4. Section 5 will present our proposal towards the problems and what we foresee as possible solutions. We conclude with an agenda for further research and development.

2. NFR Framework

As defined in Mylopoulos [11] an NFR can rarely be said to be satisfied, that is, treating NFRs as goals we bring to bear the notion of partial satisfaction. This notion led Hebert Simon [14] to coin the term "satisfice". Goal satisficing suggests that the solution used is expected to satisfy within acceptable limits.

The NFR Framework views NFRs as goals that might conflict among each other. These goals are represented as softgoals to be satisficed. Each softgoal will be decomposed into sub-goals represented by a graph structure inspired by the and/or trees. This process continues until the requirements engineer considers the softgoal satisficed (operationalized), so these goals are understood as operationalizations of the NFR.

An NFR has a *type*, which refers to a particular NFR as for example security or traceability. It also has a subject matter or *topic*. For example, in Figure 1 we can see the NFR type Safety regarding the topic *Room*, meaning that that room should be a safe place regarding illumination aspects, i.e. the room has to have enough light so people do not stumble and fall.

One of the operationalizations that represent part of this NFR satisficing can be seen on the left side of the figure represented by a bold circle, where the information about the minimum illumination in lux that can be used in a room is displayed. The dotted circles represent dynamic operationalizations, opposed to the static one, the bold circle. Static operationalizations are those that call for the use of some data to satisfice the NFR. On the other hand, dynamic operationalizations demand actions to be taken by the software to satisfice the NFR. For example the operationalization Safety [Room.Malfunction.User get informed], represents the requirement for a user to be informed of any malfunction that occurs in the room. The letter S that appears inside each node represents that this sub-goal is Satisficed. The letter P is used for those ones that are Partially satisficed or D for those ones that are Denied. A partially satisficed goal/subgoal means that not all the possible alternatives to satisfice this NFR are being taken here possibly due to conflicts with other NFRs.

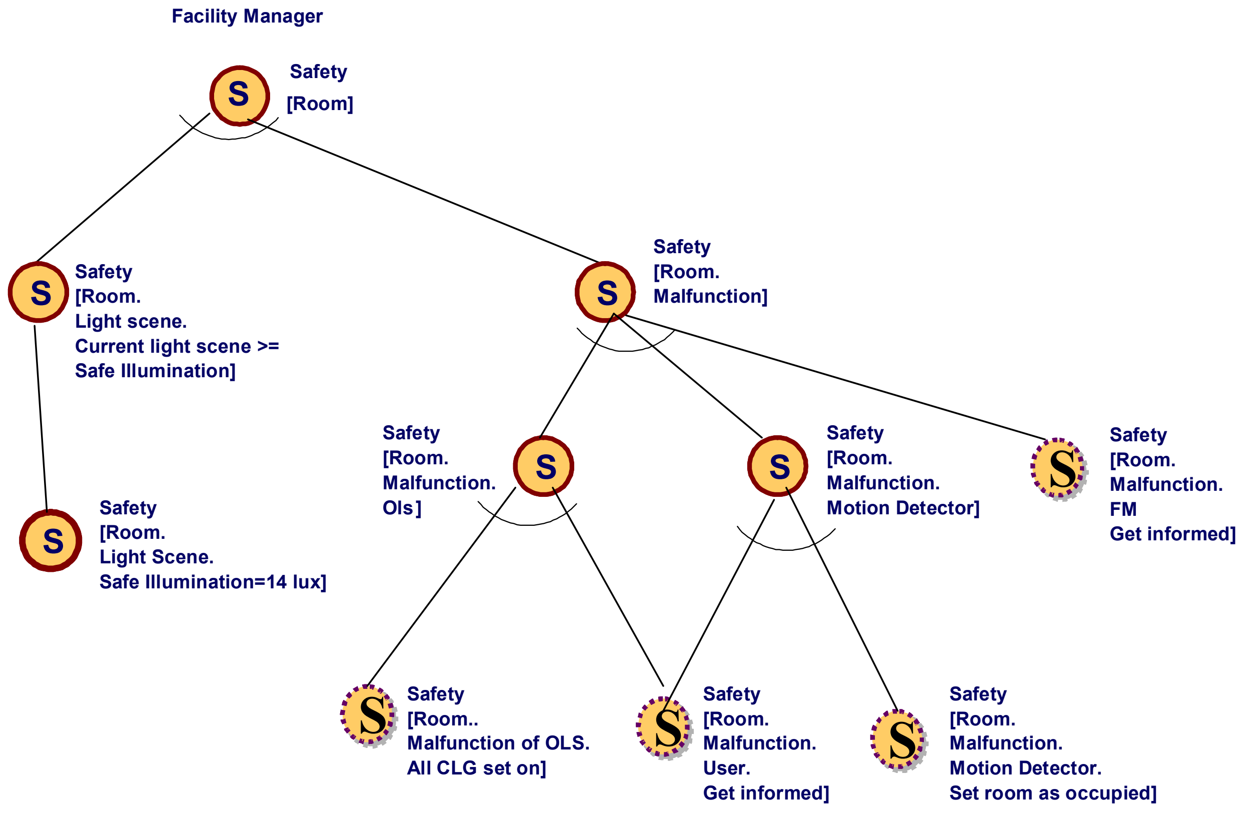
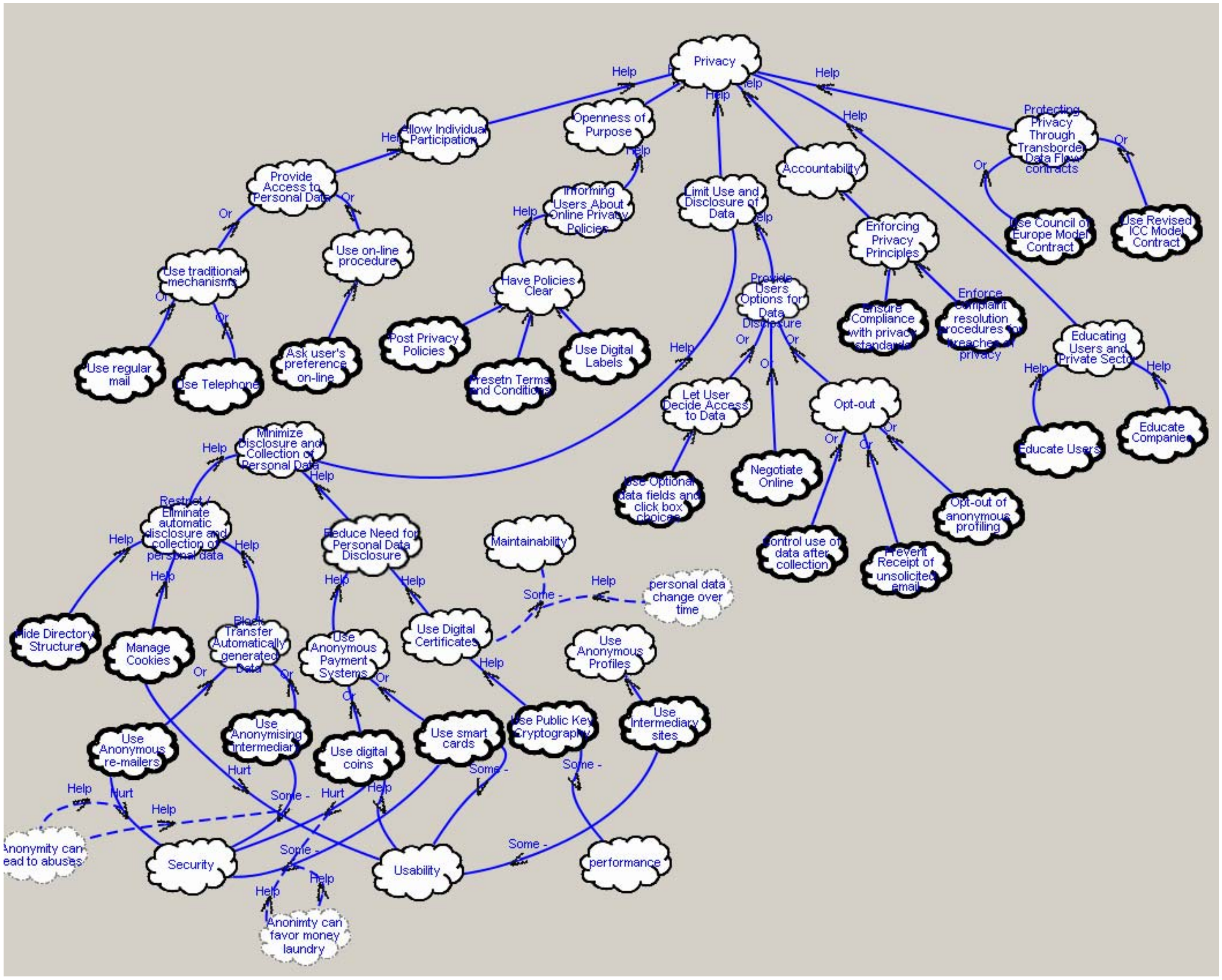

Facility Manager
S
Safety [Room]
S
Safety [Room. Light scene. Current light scene >= Safe Illumination]
S
Safety [Room. Malfunction]
S
Safety [Room. Light Scene. Safe Illumination=14 lux]
S
Safety [Room. Malfunction. Ols]
S
Safety [Room. Malfunction. Motion Detector]
S
Safety [Room. Malfunction. FM Get informed]
S
Safety [Room.. Malfunction of OLS. All CLG set on]
S
Safety [Room. Malfunction. User. Get informed]
S
Safety [Room. Malfunction. Motion Detector. Set room as occupied]

Figure 1: An NFR graph

3. A Representation for Softgoals Networks

As mentioned before, we will focus on the idea of representing NFRs using catalogues that accumulates knowledge from one or more researchers and practitioners on different NFRs. These catalogues are not static, on the contrary, the idea is that they represent the knowledge acquired up to now on a certain NFR and should be faced as an "eternal" ongoing project. This means that, as time goes on, more and more requirements engineers will use these catalogues and they will probably face unforeseen situations and should therefore update these catalogues with new alternatives to satisfice the NFRs being used. Thus, these catalogues represent a snap shot of the knowledge about a NFR at a specific point in time.

Figure 2, for example, represents the knowledge we have gathered up to now on the Privacy NFR [18]. In this figure we can see, for example, that Privacy can be refined into Limit Use and Disclosure of Data, which is further decomposed into Minimize Disclosure and Collection of Personal Data and later decomposed, among other options, into Reduce Need for Personal Data. To satisfice the latter, we may find three options: Use Anonymous Payment, Use Digital Certificates and Use Anonymous Profile. These options can be used alone or together to achieve different needs for privacy. Notice that to Use Digital Certificates while contributing to Privacy will eventually hurt Maintainability since personal data change over time. The use of Public Key Cryptography can implement Digital Certificates but may also have a negative impact on Performance. Note that, although we represent many layers of decomposition, the requirements engineer can choose to focus directly on the operationalizations if he feels comfortable enough to that. The layers of refinements aim only to reproduce the rationale involved in achieving each operationalization in order to give the engineer some idea on how one got to the operationalizations



Privacy
Help
Allow Individual Participation
Openness of Purpose
Accountability
Protecting Privacy Through Transborder Data Flow contracts
Or
Use Council of Europe Model Contract
Use Revised ICC Model Contract
Provide Access to Personal Data
Informing Users About Online Privacy Policies
Limit Use and Disclosure of Data
Enforcing Privacy Principles
Ensure Compliance with privacy standards
Enforce complaint resolution procedures for breaches of privacy
Use traditional mechanisms
Use on-line procedure
Have Policies Clear
Provide Users Options for Data Disclosure
Educating Users and Private Sector
Use regular mail
Use Telephone
Ask user's preference on-line
Post Privacy Policies
Presetn Terms and Conditions
Use Digital Labels
Let User Decide Access to Data
Opt-out
Educate Users
Educate Companies
Minimize Disclosure and Collection of Personal Data
Use Optional data fields and click box choices
Negotiate Online
Opt-out of anonymous profiling
Control use of data after collection
Prevent Receipt of unsolicited email
Restrict / Eliminate automatic disclosure and collection of personal data
Reduce Need for Personal Data Disclosure
Maintainability
Some -
personal data change over time
Hide Directory Structure
Manage Cookies
Block Transfer Automatically generated Data
Use Anonymous Payment Systems
Use Digital Certificates
Use Anonymous Profiles
Use Anonymous re-mailers
Use Anonymising intermediary
Use digital coins
Use smart cards
Use Public Key Cryptography
Use Intermediary sites
Hurt
Anonymity can lead to abuses
Security
Usability
performance
Anonimty can favor money laundry

Figure 2: A Privacy network

Figure 3 shows a catalogue for Traceability [7]. We can see in this figure that Traceability is first decomposed into Traceability for Processes and Things. Processes will tackle concerns about being able to reconstruct all the steps of a Process such as furnishing a piece of equipment. When we furnish a piece of equipment usually several steps are involved in the process. If some piece of equipment (e.g. a laptop computer) shows problems when tested before shipping, the manufacturer might want to be able to trace all the steps involved in furnishing that laptop to trace the cause of the problem.

Tracing Things may involve many different options. Here, the term Things might be applied, but not necessarily restricted, to people, objects and information. One might want to trace the places some object is or have been located. One might also want to trace what changes were made to an object or to a piece of information in order to assure Security and Reliability. Another option is that one might want to trace times for an object or information. For example one might want to know when an object was moved from one place to another, or simply when one object was changed. Finally it is also possible that we may want to keep trace of whole-part relationships expressing for example that an object was aggregated to another or one was split from another. Each of these options can be refined to different possibilities for operationalizations. For example, Trace Change might be refined into Have Changes Traced. This will call for the need to store information about when an object or information was changed. For example, in a hospital I may want to trace every change made to a patient record so I can precisely follow the treatment prescribed to a patient and the associated pathology. On the other hand, when storing schedule for Nurses shift, I may want to keep the last used schedule just in case one nurse fails to show up and I have to ask another nurse to do a double shift. Knowing the last schedule may help on such a decision but further storage of previous schedules would not be necessary. Thus, operationalizing this goal can be done in two different ways. You can Store all states for a giving Thing or you can simply Store Last State. Notice that to Store all states *hurts* (contribute negatively) both time and space performance, but will also help (contribute positively) security and reliability. Tradeoffs should be

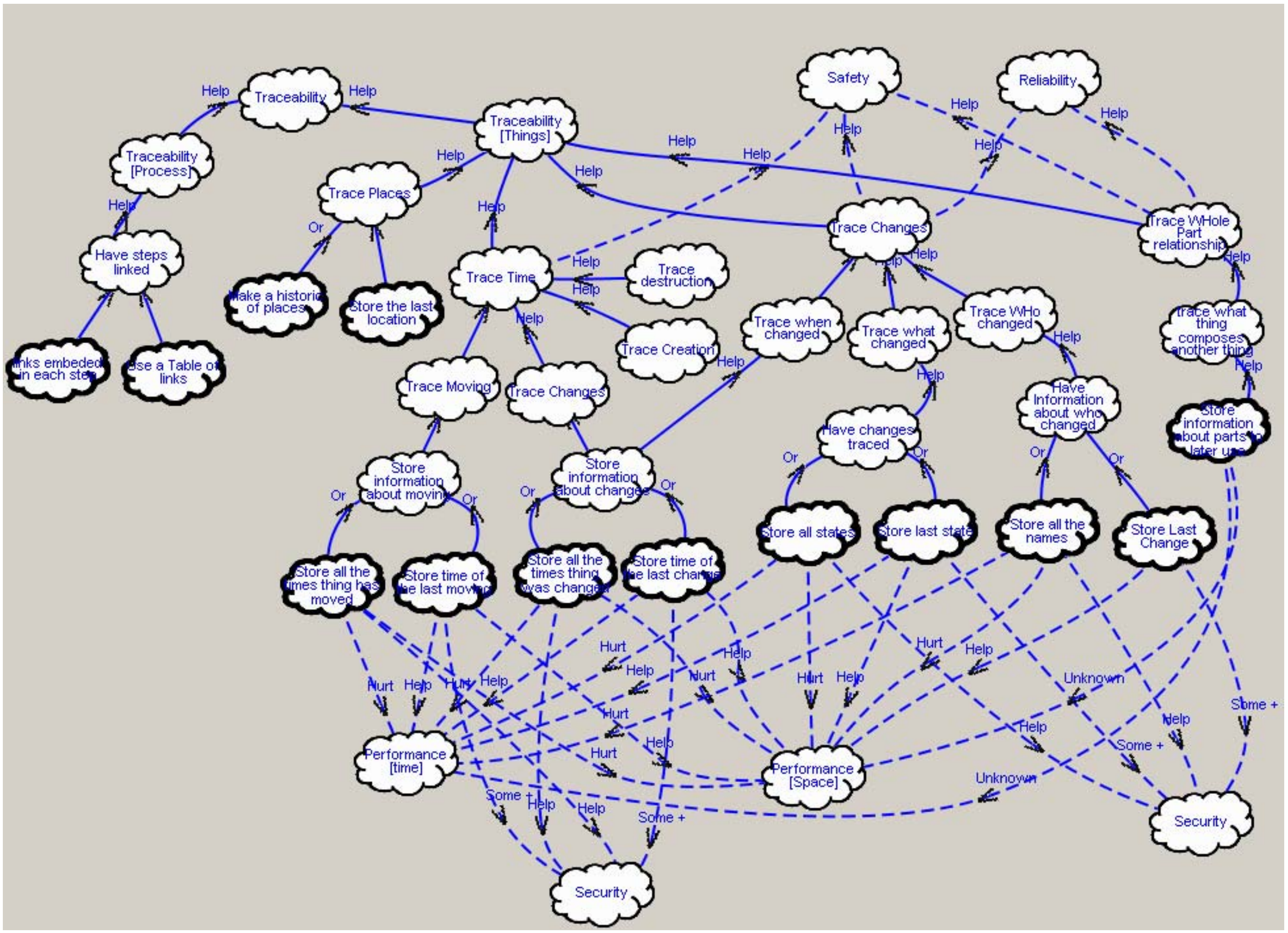


Figure 3: A Traceability network

made to evaluate which alternative would prevail for the case being studied.

Figure 4 shows a decomposition of time performance soft-goal. It can be seen that to achieve higher performance, both hardware and software improvements are useful, and therefore the *help* relationship. The figure, adapted from [17], depicts performance under the perspective of software refactoring. In this work the authors were mainly focused on the software improvements, since software improvement may make the programs fit better to the hardware platform. They present some hardware alternatives such as faster CPU, multiprocessors, faster memory, larger cache with higher associativity. These alternatives are reflected by the refactoring techniques aiming to exploit the potentials. Hence, most soft-goals for software refactoring have corresponding hardware constraints as well. In Figure 4 we can se for example, that we may need more CPU to implement parallelism, which in its turn, can be improved by loop partitioning transformations. As for the softgoal of FewerCachesMisses, enlarging cache size often resolve capacity cache misses while loop transformations like tiling and fusion helps to shorten the stack reuse distances so as to avoid capacity misses for a given cache size. Thus, in most cases, without considering the cost of hardware or the cost of software optimization, they are both positive alternatives to reach the performance softgoal.

4. Problems in Cataloguing NFR as Softgoals Networks

The NFR framework [3] has proposed that NFR knowledge be organized into three different catalogues: type catalogue, method catalogue and correlation catalogue. On the other hand, [5] proposes the use of a softgoal network indexed by a list of types (non-functional requirements). Nonetheless, as we move to a widespread use of goal driven requirements engineering [12], we face problems with the existing proposals.

First, a catalogue of types may come up to be a large taxonomy tree or a list, and as such does not provide much organization to allow for retrieval and reuse. A catalogue of methods stores possible refinements of a given NFR either by decomposition, operationalization or argumentation. Although an overall typology was defined, there is a lack of a more elaborated organization. The catalogue of correlations was conceived as a way of relating given operationalizations for different non-functional requirements (types) according to a fixed classification of possible contributions, that is, in what way an operationalization would impact non-functional requirements. These catalogues were proposed without any organizational support towards facilitating reuse, i.e.,

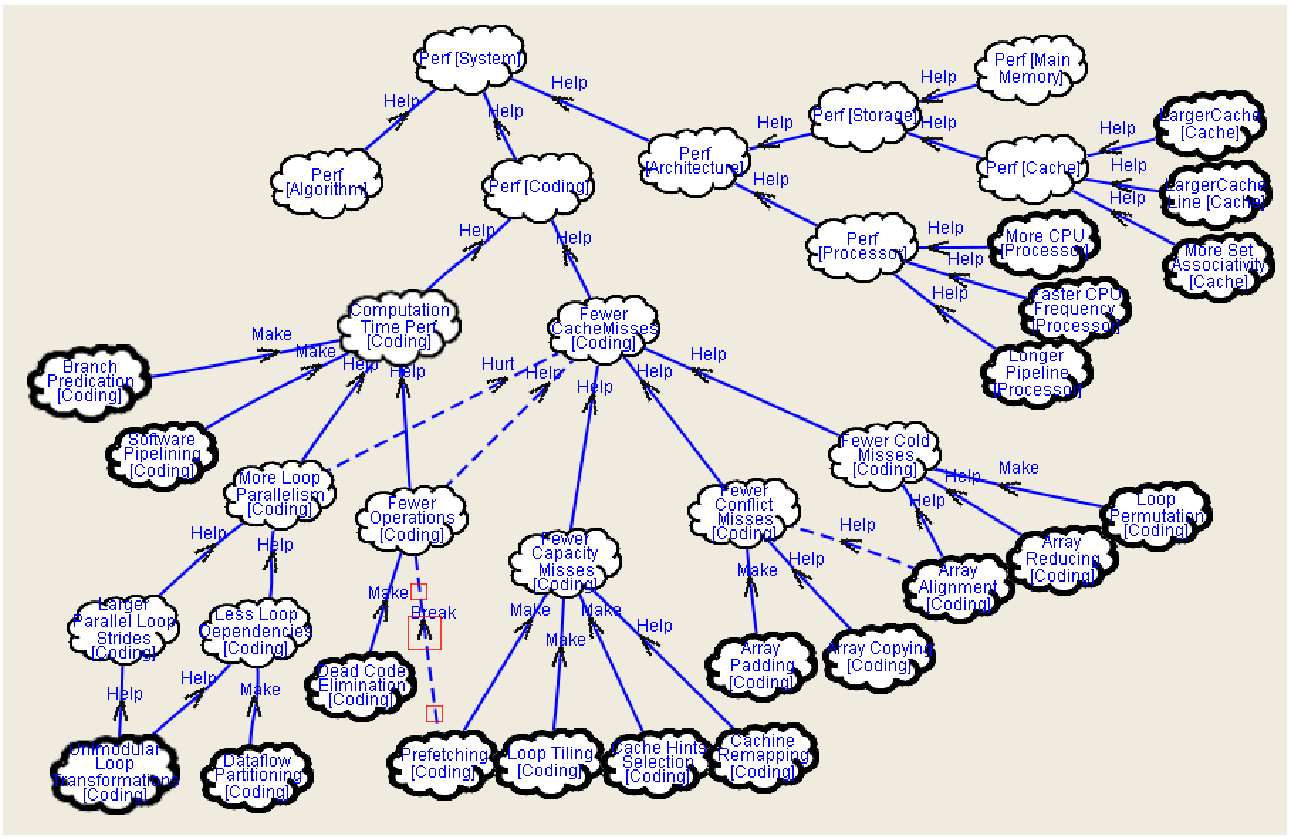

Perf [System]
Perf [Algorithm]
Perf [Coding]
Perf [Architecture]
Perf [Storage]
Perf [Main Memory]
Perf [Cache]
LargerCache [Cache]
LargerCache Line [Cache]
More Set Associativity [Cache]
Perf [Processor]
More CPU [Processor]
Faster CPU Frequency [Processor]
Longer Pipeline [Processor]
Computation Time Perf [Coding]
Fewer CacheMisses [Coding]
Branch Predication [Coding]
Software Pipelining [Coding]
More Loop Parallelism [Coding]
Fewer Operations [Coding]
Fewer Capacity Misses [Coding]
Fewer Conflict Misses [Coding]
Fewer Cold Misses [Coding]
Loop Permutation [Coding]
Array Reducing [Coding]
Array Alignment [Coding]
Array Padding [Coding]
Array Copying [Coding]
Larger Parallel Loop Strides [Coding]
Less Loop Dependencies [Coding]
Dead Code Elimination [Coding]
Unimodular Loop Transformations [Coding]
Dataflow Partitioning [Coding]
Prefetching [Coding]
Loop Tiling [Coding]
Cache Hints Selection [Coding]
Cachine Remapping [Coding]
Help
Make
Hurt
Break

Figure 4: A Performance network

with no policy for classification, indexing and retrieval. The same observation applies to the lists proposed in [Cysneiros 01].

However, as we presented in Section 3, the knowledge of types, methods and correlations are being presented as a network of softgoals by users of the goal based requirements[1]. As such, the catalogues are embedded in a single network, which is usually a chunk of knowledge. One interesting aspect we observed is that those chunks are domain driven. That is, they are not a single big complex network, but have been put together by domain, by type and by subject matter (*topic*). Some domains may be divided into sub-domains where NFRs will be instantiated differently for each sub domain. For other domains one can expect to find common understanding with few or no variations throughout the domain, for instance we believe that the Traceability network, Figure 3, is applicable to a set of domains beyond production lines or medical laboratories. Although this was not a written policy for the NFR framework, commonsense drove those chunks to be domain oriented, which is a starting point in organizing these knowledge.

5. Issues and Solutions on Cataloguing NFR

Looking at the problem of cataloguing NFRs we observed the similarity with the problem of reusing components out of a library, a classical problem in software reuse. Software reuse [Prieto-Díaz 91] has borrowed from the knowledge of information sciences the concept of facet classification. The justification given by Prieto-Díaz for the advantages of a faceted schema over the Library of Congress or the Dewey Decimal is that it provides higher accuracy and flexibility. The two other approaches are enumerative, since all possible classes need to be pre-defined. A faceted schema reduces the need for keywords, but because the facets need to be named, a standard vocabulary needs to be adopted for the facets attributes.

If we organize the NFR knowledge following a facet classification we will certainly improve the previous proposals, as seen on Section 4. However, there are some issues that are not trivial to deal with, one of them is vocabulary and the other is evolution.

If we take for instance the proposal set forward by Prieto-Díaz we would have a given entry in the catalogue being described by list of descriptors corresponding to a set of facets. A facet term matrix describes for each facet a list of terms (controlled vocabulary). These terms are organized in a thesaurus with a list of synonyms. A facet schema is applied to specific domains. So, for example, in the Unix tools domain we will have the following facets: **by action**, **by object**, **by data structure** and by **system**. For instance, the by action facet contains terms such as: get, put, update, append, make among others. In the other hand the facet data-structure contains the terms: buffer, tree, table, file, archive. A given component B, of the Unix tool domain, may be described as: B:=(get, number, buffer, line-editor). Using the flexible structure for classification and with the help of a thesaurus we may find this component even if we require a component that reads numbers from cache in a string line oriented editor.

We propose to organize the catalogues of softgoal networks using the facets: **type**, **list of related-types, list of operationalizations, and topic** under specific domains. Take for instance the Figure 3 in Section 3, it would have to be divided into two networks and would be classified as:

T1:=(traceability,{}, {links embedded in each step, use a table of links}, process) and

T2:=(traceability, {safety, performance, security, reliability}, {make a historic of places, store the last location, store all the times thing has moved, store time of last move, store all the times thing was changed, …}, things)

As to the vocabulary aspect, we believe that it is reasonable to think that we do have a considerable well defined vocabulary related to the general naming of NFRs types. The literature has a reasonable agreement on the general naming of those types, see for instance [1], [2], [15]. So we can assume that the *type* taxonomy is based on a controlled vocabulary. However, we do not see the same applying to operationalizations. It may happen in the future. As the knowledge of using NFR catalogues grow, we may try to standardize the vocabulary for operationalizations, but we do not see it feasible to use it today.

Regarding *topic*, we propose to use this facet, using terms that are domain oriented. As such, the topic would be derived from the application vocabulary. In fact the use of the Language Extended Lexicon (LEL) [9] in integrating NFR and functional models has shown that *topic* does have a controlled vocabulary [Cysneiros 02].

We adopt a domain oriented approach, where a classification schema would be defined by the following four tuple:

<*Domain*, Facet **type**, Facet **list of related types**, Facet **topic**>

[1] It is a fact, well pointed out by the reviewers, that the focus here is not producing softgoals for reuse. However, it is also a fact that without the proper infrastructure it is not possible to build for reuse. We see our proposal as a starting point on producing the indexing infrastructure that is needed.

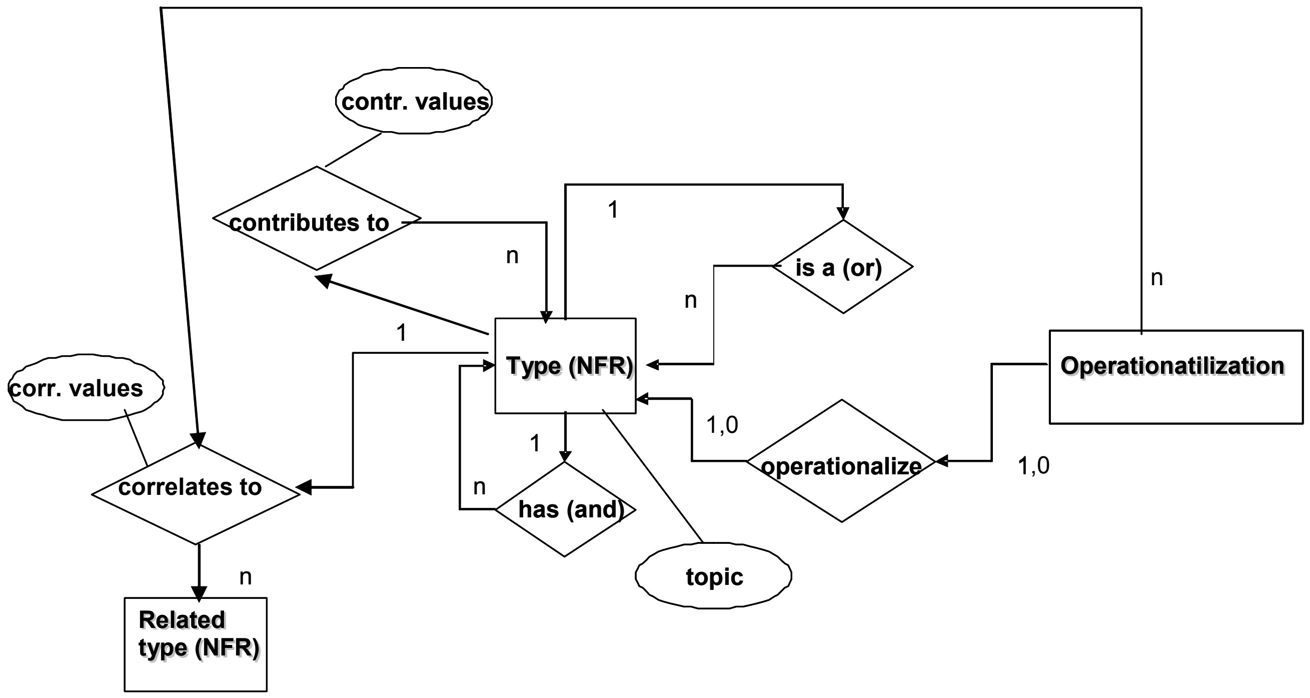


Figure 5: the Data Model for the NFR Catalogue

The vocabulary to support *type* is the linearization of the published type taxonomies. The list of related types , of course, uses the same vocabulary. Topic can be brought out from the LEL of the application. As such, the topic has already a thesaurus associated with it, since the LEL does handle synonyms. It would be then a reasonable task to build thesaurus or lexicons for *types*.
Another important issue is how we would map the networks into a data structure. We describe a data structure by means of an entity relationship model (Figure 5) to address this concern.

Given this model and a facet classification, it is possible to store this model under traditional data base schemas, and thus allowing a series of queries over the catalogue . The attribute c*ontr. values* refer to the scale of contribution for instance (+, ++, -, --, unknown,….). So, each entry in the catalogue will be a chunk or a softgoal network. A unique primary key would be based on the four tuple above. This way, each tuple instantiation points to a given specific softgoal network. The data model represents the internal structure of each softgoal network. For instance: the entire Figure 3 could be a catalogued chunk, or, as above, it could be divided in two chunks (process and things) or could even be more detailed, like for instance, Trace Places.

We use the idea of a general database, where each chunk, following the overall data model of Figure 5, would be classified by domain and will have the attribute "context". Context will be additional information on the applicability of the given chunk as well as special pre-conditions that must hold for its application.

6. Conclusion

It is important to remember that our proposal towards reuse of NFR is not supposed to be complete. We are proposing a catalogue, organized by facets that could be stored using a general database for future retrieval of needed knowledge. We foresee this catalogue as being used by researchers and practitioners on a collaborative manner. We are not proposing to integrate the catalogue with other software artifacts at the moment. So, we did not explore other organization possibilities, like, for instance: patterns or frameworks [8].

However, this is a first model that structures and prepares an organization for storing softgoals networks, and as such, there are some open issues related to the proposal. First, although the experience in reuse tell us that catalogues must be domain oriented to be effective, does the fact of dealing with cross-cutting knowledge (non-functional) points out for a general classification and not a domain oriented one?

Second, what is the appropriated granularity for the basic units of organization? Using softgoal networks as the basic retrieval entity, we are dealing with coarser chunks of knowledge, since a network may embody a refinement method and correlations among softgoals and operationalizations. Dealing with a coarser granularity is a result of adopting a domain oriented approach. However, we have to better evaluate our proposal; not only to address the question of granularity within a more coarse approach, but also address the question regarding domain oriented reuse in the context of NFR.

It is our aim to create an on-line catalogue of softgoal networks that will work as an open source catalogue. Using the basic organization outlined above we will pursue an implementation, with a version system, where researchers and practitioners could retrieve softgoals networks as well as contribute to the catalogue or with other networks or with comments or questions about the existing ones. As this catalogue grows our knowledge about it will grow, such that we could evolve our first organization schema. We also will have the opportunity to monitor the use of operationalizations, as to study the feasibility of constructing an operationalization thesaurus.

Acknowledgements. This work was partially supported by Capes and NSERC grants

7. Bibliography

[1] Boehm, B., Brown, J. R., Lipow, M., "*Quantitative Evaluation of Software Quality*". ICSE, 1976, IEEE Computer Society Press, 1976, pp. 592-605.

[2] Boehm, Barry e In, Hoh. "*Identifying Quality-Requirement Conflicts*". IEEE Software, March 1996, pp. 25-35

[3] Chung, L., Nixon, B., Yu, E. and Mylopoulos,J. "Non-Functional Requirements in Software Engineering" Kluwer Academic Publishers 2000.

[4] Cysneiros, L.M. and Leite, J.C.S.P. *"Integrating Non-Functional Requirements into data model"* 4th International Symposium on Requirements Engineering – Ireland June 1999.

[5] Cysneiros,L.M., Leite, J.C.S.P. and Neto, J.S.M. *"A Framework for Integrating Non-Functional Requirements into Conceptual Models*" Requirements Engineering Journal — Vol 6 , Issue 2 Apr. 2001, pp:97-115.

[6] Cysneiros,L.M. and Leite, J.C.S.P. *"Using UML to Reflect Non-Functional Requirements"* Proceedings of the 11th CASCON, IBM Canada, Toronto Nov 2001 pp:202-216

[7] Cysneiros, L.M. and Yu,Eric "Non-Functional Requirements Elicitation" to appear in Perspective in Software Requirements Kluwer Academics Publishers 2003.

[8] Fiorini, S. T, Leite, J.C.S.P, and Lucena, C.J.P., Process Reuse Architecture, in Proceedings of CAiSE 2001, LNCS 2068, Springer-Verlag Berlin, pp: 284-298, 2001.

[9] Leite J.C.S.P. and Franco, A.P.M. *"A Strategy for Conceptual Model Acquisition "* in Proceedings of the First IEEE International Symposium on Requirements Engineering, SanDiego, Ca, IEEE Computer Society Press, pp 243-246 1993.

[10] L. Liu, E. Yu, J. Mylopoulos "Analyzing Security Requirements as Relationships Among Strategic Actors" 2nd Symposium on Requirements Engineering for Information Security (SREIS'02). Raleigh, North Carolina, October 16, 2002.

[11] Mylopoulos,J. Chung, L., Yu, E. and Nixon, B., *"Representing and Using Non-functional Requirements: A Process-Oriented Approach",* IEEE Trans. on Software Eng, 18(6), pp:483-497, June 1992.

[12] Prieto-Diaz, R., *Implementing Facet Classification for Software Reuse*, Communications of the ACM, vol. 34, n. 5, pp: 88-97, 1991.

[13] Sindre,G., Firesmith,D.G., Opdahl,A.L. "A Reuse-BAsed Approach to Determining Security Requirements" In Proc. 9th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03), Klagenfurt/Velden, Austria, 16-17 Jun 2003.

[14] Simon,H.A. "The Sciences of the Artificial" , 3rd edition MIT Press, 1969

[15] Sommerville, I. Software Engineering, 6th edition (http://www.software-engin.com/)(Chapter 5, slides 17 and 21), Addison-Wesley, 2000.

[16] Toval, A., Nicolas,J., Moros,B., Garcia,F. "Requirements Reuse for Improving Information Systems Security: A practitioner Approach" Requirements Eng.. Journal Vol 6 pp206-219, 2002 Springer-Verlag

[17] Yijun Yu, John Mylopoulos, Lin Liu, Erik D'Hollander, Kristof Beyls, "Software refactoring guided by multiple soft-goals", in preparation.

[18] Yu, E. and Cysneiros, L.M. "Designing for Privacy in a Multi-Agent World" In: R. Falcone, S. Barber, L. Korba and M. Singh (Eds) "Trust, Reputation and Security: Theories and Practice": Springer-Verlag, LNAI 2631, May 2003.