Exploiting a Goal-Decomposition Technique to Prioritize Non-functional Requirements

M. Daneva[1], M. Kassab[2], M. L. Ponisio[3], R. J. Wieringa[4], O. Ormandjieva[5]
[1,2,3,4]{m.daneva, m.kassab, m.l.ponisio, r.j.wieringa}@utwente.nl
[5]ormandj@cse.concordia.ca

Abstract

Business stakeholders need to have clear and realistic goals if they want to meet commitments in application development. As a consequence, at early stages they prioritize requirements. However, requirements do change. The effect of change forces the stakeholders to balance alternatives and re-prioritize requirements accordingly. In this paper we discuss the problem of priorities to non-functional requirements subjected to change. We, then, propose an approach to help smooth the impact of such changes. Our approach favors the translation of non-operational specifications into operational definitions that can be evaluated once the system is developed. It uses the goal-question-metric method as the major support to decompose non-operational specifications into operational ones. We claim that the effort invested in operationalizing NFRs helps dealing with changing requirements during system development. Based on this transformation and in our experience, we provide guidelines to prioritize volatile non-functional requirements.

1. Introduction

Over the past ten years, the requirements engineering (RE) community has increasingly expanded its adoption and adaptation of goal-oriented approaches to both functional and non-functional requirements (NFRs). Assessing project stakeholders' goals early in the software life cycle was recognized as the major step towards achieving a project scope definition with clearly understood and well-communicated project goals [1,2]. Such an assessment relies on the availability of knowledge on the user-defined requirements and their effort estimates, priorities, as well as their risk. This knowledge enables analysts, managers, and software engineers to identify the most significant requirements from the list of initial defined requirements in the project. For instance, a requirement deemed critical, taking much implementation effort, and posing high risk, may be a good candidate for immediate resourcing.

During RE, software projects essentially share the following context factors:

- Varying requirements' importance: by definition, all requirements are required (i.e. mandatory). However, at certain point of time for some stakeholders, requirements are not all equal in terms of value to them. Project stakeholders are unlikely to agree on which the most important requirements are.
- Limited resources: budget and schedule constraints make it rarely possible to implement all requirements in a given increment.
- Incompatible requirements: some requirements types may be incompatible (e.g. security vs. performance, multi-access vs. security) in the sense that increasing the compliance with one requirement makes it more difficult to achieve the other requirement.
- Subjective prioritization: most prioritization approaches are subjective and influenced by project politics. They also ignore the rationale behind stakeholders' setting their priorities.
- Volatility: requirements that are likely to change (for both anticipated and unanticipated reasons) always result in concomitant changes in the project schedule and budget.

These context factors make the objective prioritization of requirements with various degrees of volatility a critically important part of the RE process in any project. Coupled with these factors is the need to consider NFRs as an integral part of software modeling and development. If the NFR in a project are stated in non-operational specifications, then the effect of change is done, by and large, on an ad-hoc basis, which results in undesired effects such as underestimation of the change impact, and the project team unknowingly striving towards an obsolete goal.

The aim of this research effort is to introduce steps towards operationalization and prioritization of NFRs. We explore the use of goal-oriented solutions to help prevent loosing the links between NFRs and operational parts of the system, especially as priorities change. In the rest of this paper, we first summarize related work in Section 2. In Section 3, we describe our approach. We discuss NFRs priority concerns from a goal-oriented standpoint, present the softgoal and hardgoal modeling approach we adopted, and assess how it fits within the prioritizing process. Section 4 discusses the case of volatile NFRs, which are NFRs whose effects on the remaining NFRs in a project is unknown. We also discuss open issues, provide some guidelines for practitioners to confront the issues, and evaluate our early findings. Finally, Section 5 summarizes our early results.

2. Related work

The NFR literature suggests a few process-oriented approaches to NFRs [1,2,3,4]. What unites all these authors is the use of techniques to reason about design decisions on the inclusion or exclusion of requirements that will impact the software architecture. Among these literature sources, the NFR framework [4] has been the first to propose the concepts of softgoal and softgoal-satisficing to represent NFRs in the RE context and reason about them. A softgoal is a goal which has no clear-cut definition or criteria to determine whether or not it is satisfied. Goal-satisficing, then, means that the solution used to achieve the goal, is expected to satisfy it within acceptable limits. For example, we never can say that a system is 100% maintainable against future possible changes, but we can build-in enough good maintenance practices to it so that it is considered easy-to-maintain. The NFR framework starts with an initial set of high-level NFRs as NFR softgoals. The NFR softgoals are refined into more specific ones iteratively while establishing interdependencies. These include relationships among softgoals, and relationships between softgoals and interdependencies. Along with these, priorities are identified, operationalizations are considered, tradeoffs are made and design rational is provided. The operation of the framework can be visualized in terms of the incremental and interactive construction, elaboration, analysis and revision of a softgoal interdependency graph (SIG).

For design architects to be able to focus their effort on the most important NFRs, priorities must be identified. The NFR framework suggests that architects: (i) identify those softgoals that are vital to the system's success as *critical*; and (ii) identify softgoals that deal with a significant portion of the organization's workload as *dominant*. In the framework, priority softgoals are identified by an exclamation point (!).

Missing from this approach, however, is (i) the impact that the stakeholders can have on the requirements elicitation process and (ii) the objective reasoning in the decision making process to select from different candidate operationalizations to satisfice NFRs.

Since the publication of the NFR framework, the goal-oriented RE community made, though, a number of attempts to get architecture design goals from requirements to evaluate design alternatives. A few i*-based approaches [5,6] have shown promising in specific project contexts.

Other related work includes the following: in [7] Cysreiros and Leite researched the process to elicit NFRs, analyzed their interdependencies, and traced them to functional conceptual models. They brought extensions of UML conceptual models (namely, Class, Sequence, and Collaboration diagrams) which include a way to express NFRs. The key claim these authors made was that augmenting conceptual models with representations of NFRs can improve the quality of the resulting conceptual models themselves. In [8] Robinson et al put in perspective the metrics-based Root Requirements Analysis technique to confront the requirements interaction problem, which is how to discover, track and resolve conflicting interactions among NFRs. Related work on NFRs prioritization includes [9,11,12,13,14,15]. These sources indicate that if the priority is to measure how much a NFR matter to a stakeholder, then that priority is linked to the value creation, NFR satisfaction (in CBAM) and realization of win conditions (in WinWin), contribution to quality attributes, compliance to legal regulations and contract with customers, support of business values, strategic benefits, probability of the product´s success in its target market. Priority may also reflect business criticality, importance to the customers or users, urgency, importance for the product architecture, or fit to release theme [13,14,15].

While these approaches enable – in a variety of ways, designers to consistently stay in tune with stakeholders, they do not provide answers to questions like how the NFRs are mapped to operationalized elements in the solution space, when to decompose NFR to functional, when not to do it, or how to deal with volatile NFRs. In our solution approach, we suggest a first step towards briging this gap.

3. The solution approach

Our approach rests on five types of sources: (i) the World-Requirements-Goals-Specification-Architecture (WRGSA) reference model [16] that helped us maintain the big picture of how the real-world, NFRs, goals, specifications, and architecture fit together, (ii) the goal-oriented NFR framework [4] which helps us decompose softgoal NFRs into finer operationalizable definitions, (iii) the goal-question-metrics methodology [17] which is the key support to compare possible operationalizations for a specific NFR, (iv) functional size measurement methods [18] which let us quantify these operationalizations, and (v) our own experience in NFRs prioritization.

The solution approach is presented in Figure 1. It shows how the NFR framework and the GQM approach are used in synergy to support the transformation of environment's requirements into goals, system specifications and architecture design options. We deploy a hardgoal extension of the NFR framework to eliciting, document, and analyzing NFRs. We, then, propose the GQM approach as the key support vehicle to find the most meaningful operationalizations suited for a given software development process. The next subsections describe in more detail how our approach is thought to add value in the context of these three RE activities.

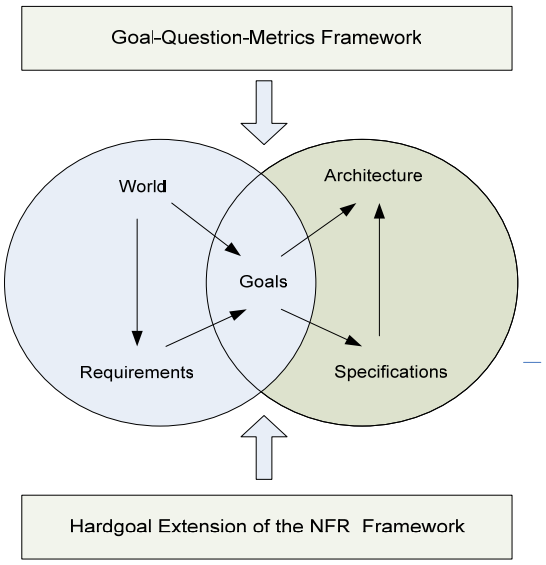

Figure 1. The solution approach.

3.1. Stepwise prioritization of NFR and NFR-volatility

To arrive at a complete and consistent definition of the NFRs, RE teams typically go through iterative NFRs prioritization. It has been the authors' experience that the first step towards initial prioritization is to ask stakeholders, while still eliciting requirements, to group those NFRs they think are critical and thus are likely to proceed with their implementation first. It is very important to notice that this initial prioritization is done without any influence from the developers involved in the downstream project activities. When requirements negotiation meetings happen, we observe that a second step towards prioritizing NFRs is executed. This is when stakeholders acknowledge the presence of conflicting requirements during software execution and when developers' input may be sought after to provide early insights into how conflicting NFRs may impact the downstream project activities. To support this step, we propose to link pairs of requirements with the right sign that indicates their positive, negative or neutral interaction. In what follows, we use the symbols "+", "-" and "" to represent positive, negative, and neutral interaction during software execution, respectively. Our proposal rests on the observation that during software execution, when the executable version of the software is running, hardly any NFRs manifest in isolation. Typically, the provision of one NFR may affect the level of provision of another. We refer to this mutual dependency as non-orthogonality. Given this assumption, we propose a function M to map each pair of the identified NFRs to the values "+", "-" or "". The lack of knowledge on the interaction between a pair of requirements NFR_i, NFR_j is indicated with "?":

$M(NFR_i, NFR_j) \in \{"+", "-", "","?"\}$

We defined the following rules for assigning these values to the pairs of NFRs:

1. The value "-" is assigned to a pair of NFRs originating from the set of NFRs that contribute negatively at the same functionality. This means that one NFR in the pair has a negative (damage) effect on the other at the same functionality. The assignment is based on the experts' judgment of the developers. This is a case of a conflict between NFRs.

2. The value "+" is assigned to a pair of NFRs originating from the set of NFRs that contribute positively if they meet at the same functionality. This means that one NFR in the pair has a positive (constructive) effect on the other. The assignment is based on the experts' judgment of the developers.

3. The value "" is assigned to a pair of NFRs among the ones in the set of NFRs that do not interact. This assignment is based on the experts' judgment of the developers.

4. The value "?" is assigned to a pair of NFRs when the type of their influence is unknown. This in general

makes the NFRs volatile as variations in their influences to other NFRs are expected and would lead to instability of the solution and possibly undesirable impact on other elements.

As a common approach, conflicts among NFRs (those NFRs interacting between each other with "-" during the execution) can be resolved by attributing weights to the interacting NFRs at each user-recognizable piece of functionality. The values are given according to the importance each NFR has from the viewpoint of the stakeholders on a particular functionality. For example, security could be of higher importance than availability at functionality "x" and of less importance at functionality "y". A scale can be built to map the numerical value of weight to the importance. For example,

- *Very important* takes values in the interval]0,8..1,0]
- *Important* takes values in the interval]0,5..0,8]
- *Average* takes values in the interval]0,3 .. 0,5]
- *Not so important* takes values in the interval]0,1..0,3]
- *Do not care* takes values in the interval [0.. 0,1]

These values of *very important*, *important*, *not so important*, and *do not care*, do help stakeholders in attributing priorities to conflicting NFRs. Then, the conflict mentioned above should not be too difficult to resolve, as the weights express priorities.

On other hand, we observe that the approach of attributing a weight of significance to NFRs in order to identify dominance is not always applicable. In complex systems, such as concurrent systems, two or more NFRs may affect the same functionality with changing priorities with respect to the execution of the behavior of some component (e.g. method body), so assigning a hard-coded prioritization will not follow the correct semantics. For example, we may have a case with synchronization "sync" and scheduling "sched" whereby <sync, sched> method body<sched, sync> [6]. If authentication is introduced in the system, then priorities also change: <authentication, sync, sched>method-body<sched, sync, authentication>. In addition, this approach of conflict resolution requires a major involvement of stakeholders. This makes it costly and dependent on stakeholder's availability. Moreover, in contrast to developers, business stakeholders are not interested in such system concerns and they may not have the necessary expertise to feel comfortable to get involved in these matters. They would merely want their requirements implemented.

For the purpose of this research, we keep our focus on identifying the NFRs of predicted variation or "volatility" in the conflict resolution process. Dealing with the volatile NFRs would require a deeper understanding of their impact on other NFR, which is further explored by applying the GQM goal-decomposition technique as described in section 3.3.

3.2. NFR Framework: softgoals vs. hardgoals

In this section we focus on how we extended the modeling notation of the NFR framework so that we explicitly include hardgoal NFRs. This was done to confront the general tendency to treat NFRs as softgoals, which is known to add ambiguity to the requirements specifications [7,27,28]. For example, the response time in a user interface is typically soft, whereas response time requirements in real-time systems can be hard. In such a case, our extension to the taxonomy of the NFR framework would alleviate this problem; it would allow architects to identify those NFRs that need to be stated in a clear-cut manner. For example, a performance requirement maybe specified as "The system shall respond within 3 seconds". This NFR describes an objective criterion for testing the quality of the service to be delivered. Hence, as a third step towards prioritizing NFRs, we propose an extension of the NFR framework and its softgoal notation by using the two elements shown in Figure 2.

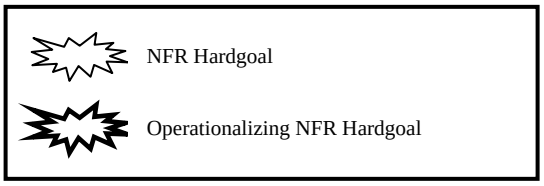

Figure 2: Elements added for the extended NFR framework.

To illustrate this extension, suppose the stakeholders state their interest in a good system performance in a more restricted way: "*Transfer online client investment orders for account manager's approval with good performance and the response time should be within 3 seconds.*" This statement is a hardgoal NFR concerned with the quality constraints of the system under development, and, as such, it needs to be absolutely satisfied rather than satisficed. Stakeholders may also ask for an architecture constraint to be imposed on the system, either as an independent requirement or as an operationalization for a stated goal; e.g. "*Database indexing should be applied on the columns used most*".

The graph in Figure 3 shows a performance softgoal with the new condition on response time and the imposed architecture constraint. The main use of hardgoals is to cope with prioritization and resource limitation. Suppose design architects are given a large

number of goals. An important question is, then, if they should put equal amount of effort into meeting each of them. The intuitive answer to this question is that architects most likely should not, especially if they only have a limited amount of time available. Instead, they would want to prioritize the goals and spend more time on goals of high priority.

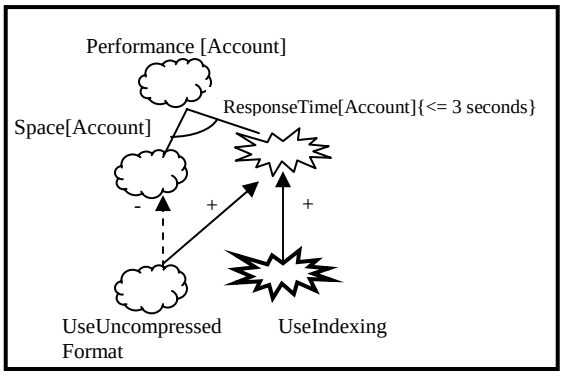

Figure 3: Employing the hardgoal concept in the NFR framework.

Our approach acknowledges this situation and postulates that NFR hardgoals are of higher priority than NFR softgoals. We take this standpoint mainly because hardgoals are to be satisfied rather than satisficed. In other words, goals to be satisfied are more important than goals to be satisficed. The failure in satisfying the stated conditions breaks the hard goal.

3.3. The Goal-Question-Metric approach to operationalizing NFRs

For architects to be able to use hardgoals for NFR prioritization purposes, they need a deeper understanding of the context in which the system is to function. We propose to use the GQM [17] approach to (ii) achieve enough knowledge on NFRs modeled as softgoals and, subsequently, (ii) state them in operational hardgoal terms. If we can see GQM as a problem-solving process where the goals are decomposed into quantifiable indicators of NFRs, then we should be able - in the context of requirements, to decompose non-operational NFRs into operational ones through questions while focusing on project-specific goals. Furthermore, the GQM technique would also help eliciting volatile NFRs, which might otherwise become a source of uncertainly and risk in the development process. In our approach, the NFR framework is used to identify the most likely value ("+", "-"or "") of interaction of a given volatile NFR with the NFRs already incorporated in the solution. At this level, we can apply some reasoning in order to assign the most likely value to a volatile NFR which would optimize the conflict resolution for all NFRs. The GQM is further applied to elicit the volatile NFR until enough knowledge on it is gained and its contributions to the NFR framework are identified. We also have to provide a clearly defined substitution property which has to be fulfilled when the nature of the volatile NFR is better explored and this NFR is operationalized, for instance:

- if a volatile NFR has been assigned "+" in its relation with NFR_i, then the GQM outcome has to be a contribution "+";
- if a volatile NFR has been assigned "-" in its relation with NFR_i, then the GQM outcome has to be a contribution "-";
- if a volatile NFR has been assigned "" in its relation with NFR_i, then the GQM outcome has to be a contribution " ".

4. Discussion

Although the definition of our approach is at its early stage and, therefore, does not pretend to be complete, we analyzed - in terms of open issues, each of the three steps towards a better NFRs prioritization process. For each issue, we put together a list of existing solution elements that are worth considering when further elaborating of our approach. We ended up with a set of early guidelines for software staff to use to smooth the impact of the issues.

We also did an early evaluation in an attempt to better understand certain aspects of our approach. The next subsections focus on our open issues, our initial set of guidelines, and our evaluation.

4.1. Issues

Our analysis yielded three open issues that require special attention and further research efforts:

1. When to decompose NFR to FR? The literature on software measurements [19,21] suggests NFR be first decomposed to FR. Then a functional size measurement method (like classic Function Point Analysis [20] or COSMIC-FFP [18]) takes as its input those FR that result from the NFR decomposition and, then yields as output the contribution of the NFR to the project size and, ultimately, to the effort estimated to build the system. Functional size metrics practitioners assume that it always makes sense to decompose all NFRs to FRs. However, an alternative viewpoint in the software metrics literature [20], assumes that in each project there is always a portion of NFR which can not be decomposed to FR when sizing. These specific NFR are seen as criteria to make architecture design

decisions. Therefore, instead of decomposing them to FR, it makes sense to treat them as context factors which are expected to introduce uncertainty to the estimation process [20]. To the best of our knowledge, neither the RE community not the software measurement community has come with a list of criteria about when decomposition of NFRs to FRs is a good thing to do and when it is not.

2. How to deal with those NFR which can be decomposed into FRs up to a certain level? Certainly, the majority of the NFRs in a project should and can be decomposed to FR, but the level to which this is possible may vary [22].

3. How to deal with NFRs which should/could not be decomposed to FRs? Recent experiences in functional size measurement [20] suggest NFRs that are not decomposable be seen as architecture design choices. As such, they have their influence on an early project cost estimate. However, very few guidelines exist on how to account for this type of NFRs when estimating project effort, so that the estimates are more realistic.

Based on our own experience and the studied NFRs literature sources [1,2,3,7,8,9,22,23], we suggest the following guidelines as first steps to confront the three issues identified:

- Develop consensus among stakeholders regarding the priorities of NFR.
- Identify NFR that deal with significant portions of the system under development
- Divide and conquer (each piece should have a less NFR as possible).
- Identify stakeholders goals that are most important to the success of the system and the NFRs vital to these goals.
- Apply GQM to NFRs identified as softgoals or volatile.
- Postpone details concerning technology as much as possible.
- Document those hardgoals which are architecture design options.

4.2. Early evaluation

The preliminary research we carried out brought us to two early findings. First, in classic GQM, it is not easy to determine goals [17]. When GQM is used as complementing the NFR framework, this is not difficult, because we explicitly formulate the NFRs as hardgoal or softgoal statements. In addition, such goal statements can be relatively easily derived from the original stakeholders' requirements stated in natural language. Our experiences seem similar to experiences by other authors [17,24].

Second, linking the NFR framework with GQM allows the metrics in GQM to serve as measurable and changeable variables for NFR operationalization. This is an essential prerequisite for understanding and – eventually, gaining control over volatile requirements.

Third, we consider the guidelines we formulated as solutions to the uncovered issues as preliminary. This list of guidelines is only the beginning of an ongoing effort to develop better prioritization process for volatile NFR. Although the guidelines are not validated in case study settings, they sound intuitive and worth further investigation. Currently, we are planning a case-study-driven research effort in companies' sites, as part of a research project [26] that aims at improving the linkages between RE and architecture design.

Fourth, we started assessing the question if our approach is capable of dealing with any type of NFR. Clearly, because we favor the use of hardgoals, we can expect that our approach will be more effective when prioritizing those NFRs which effectively demand actions to be performed by the system, and therefore affects the architecture design choices. For example, a NFR as maintainability is not easily operationalized as part of RE, but rather will be traced back to what architecture options were chosen. Our solution can document NFRs like maintainability, but because these NFRs are not operationalizable, they are not dealt with in our prioritization approach. In contrast to this, NFRs such as availability, safety, performance, accuracy, frequently demand the design to be carefully analyzed and evaluated in order to satisfice these NFRs [7]. So, we think, that it is more likely that these NFRs will be the ones which our solution approach will fit the most.

5. Conclusions

Getting the right software project scope in a volatile environment is one of the earliest project activities, and the one that has the greatest potential to cause serious problems if it is done wrongly. In this paper, we proposed solutions to improving some aspects of the NFRs prioritization process. Our solution uses an extension of the NFR framework and its integration with the GQM approach to operationalizing NFRs. We also identified open issues related to the goal decomposition of NFRs. These set up research directions for our future research efforts, namely (i) extending the COSMIC FFP method to include NFRs, (ii) integrating refined hardgoal concepts into the original NFR framework, and (iii) exploring solutions for resolving NFRs conflicts by relating them to architecture design alternatives that have attendant risks, uncertainties, and budget implications.

6. Acknowledgement

This research has been supported by the Netherlands Organization for Scientific Research (NWO), under the Quality-Driven Requirements Engineering and Architectural Design (QuadREAD) project, and by Centre for Telematics and Information Technology, the Netherlands, under the COSMOS project.

8. References

1. Boehm B., H. Hoh, "Identifying Quality-Requirement Conflicts IEEE Software pp. 25-36, Mar. 1996
2. Robertson, S., J. Robertson, Mastering the Requirements Process, Addison-Wesley, 1999.
3. T.G. Kirner and A.M. Davis, "Nonfunctional Requirements of Real-Time Systems," Advances in Computers, vol. 42, pp. 1-38, 1996.
4. Chung, L., B. A. Nixon, E. Yu, and J. Mylopoulos, Nonfunctional Requirements in Software Engineering, Kluwer Academic Publishing, 2000.
5. Yu, Y. J.C. Sampaio do Prado Leite, and J. Mylopoulos, From Goals to Aspects: Discovering aspects from Requirements Goal Models, IEEE Joint Int. Conf. on Requirements Engineering, pp.33–42, 2004.
6. Hui, B., S. Liaskos, and J. Mylopoulos, Requirements Analysis for Customizable Software: A Goals-skills-preferences Framework, Requirements Engineering Conference, pp.117–126, Sept. 2003.
7. Cysneiros, L.M., J.C.S. do Prado Leite, Non-functional Requirements: from Elicitation to Conceptual Models, IEEE Trans. On Soft Eng, 30(5), May, 2004, p.328-350.
8. Robinson, W., S. Pawlowski, and V. Volkov, Requirements Interaction Management, ACM Comput. Surv., 35 (2), pp.132–190, 2003.
9. Ryan A., An Approach to Quantitative Non-Functional Requirements in Software Development, Proc. the 34th Annual Government Electronics and Information Association Conference, 2000.
10. Kazman, R., In H.P., Chen H.-M., From Requirements Negotiation to Software Architecture Decisions, Information and Software Technology 47 (2005), pp. 511-520.
11. Kazman, R., Asundi, J., Klein, M.: Quantifying the Cost and Benefits of Architectural Decisions. In: Proc. Int. Conf. Software Eng. (2001) 297-306
12. Azar, J., R. K. Smith, D. Cordes, Value Oriented Prioritization, IEEE Software, Jan, 2006.
13. Lehtola, L., M. Kauppinen, S. Kujala, Requirements Prioritization Challenges in Practice. Proc. of 5th Int´l Conf. On Product Focused Software Process Improvement (PROFES), Kansai Science City, Japan, April 2004, pp.497-508.
14. Berander, P., A. Andrews, Requirements Prioritization, in: A. Aurum, C. Wohlin (Eds.): Engineering and Managing Software Requirements, Springer, Berlin, Heidelberg, 2005, pp. 69-94.
15. Davis A., The Art of Requirements Triage. IEEE Computer, 36 (3), March, 2003, pp 42 – 49.
16. Yamamoto, S., H. Kaiya, K. Cox, S. Bleistein, Goal Oriented Requirements Engineering: Trends and Issues, IEICE Trans on Inf. & Syst, E89(11), Nov 2006, pp. 2701-2711.
17. Basili V., G. Caldiera, and H. D. Rombach. "Goal Question Metric Paradigm,". In: Encyclopedia of Software Engineering, ed. J. J. Marciniak. New York: John Wiley & Sons, 1994.pp. 528-532.
18. Abran, A., Desharnais, J.-M., Oligny, S., St-Pierre, D. and Symons, C., COSMIC FFP – Measurement Manual (COSMIC implementation guide to ISO/IEC 19761:2003), École de technologie supérieure – Université du Québec, Montréal, Canada, 2003, URL: http://www.gelog.etsmtl.ca/cosmic-ffp/manual.jsp
19. Fenton N.E, S. L. Pfleeger, Software Metrics: A Rigorous and Practical Approach, PWS Publishing, 2nd edition, revised printing, 1998.
20. Pfleeger, S. L., F. Wu, R. Lewis, Software Cost Estimation and Sizing Methods,: Issues and Guidelines, RAND Corporation, 2005.
21. FISMA, Experience Situation Analysis, Finnish Software Metrics Association, 2001, http://www.fisma.fi/wp-content/uploads/2006/09/fisma_situation_analysis_method_nd21.pdf
22. Mylopoulos, J., Goal-oriented Requirements Engineering, Keynote at the 14th IEEE International Conference on Requirements Engineering, IEEE Computer Society Press, 2006.
23. Lamsweerde, A., R. Darimont, and P. Massonet, "Goal Directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learnt. Proc. 2nd IEEE Int. Symp. on Requirements Eng. pp. 194-203.
24. Mylopoulos, J., Chung, L., Nixon, B., Representing and Using Nonfunctional Requirements: A process Oriented Approach. IEEE Trans. S.E. 18, 6(1992) 483-497.
25. Haruhiko Kaiya H., Osada A., Kaijiri K., Identifying Stakeholders and Their Preferences about NFR by Comparing Use Case Diagrams of Several Existing Systems, Prod. Of the International Conference on Requirements Engineering, 2004 (RE04), pp.112-121.
26. http://quadread.ewi.utwente.nl/
27. Rosa, N. S., Cunha, P. R. F., Justo, G. R. R.: Process NFL: A language for Describing Non-Functional Properties. Proc. 35th HICSS, IEEE Press (2002)
28. Kavakli E., Loucopoulos P., Goal Driven Requirements Engineering: Evaluation of Current Methods, Proc. of EMMSAD'03, LNCS, Springer.