Quantifying Non-Functional Requirements: A Process Oriented Approach

Raquel Hill[†‡], Jun Wang[‡], Klara Nahrstedt[†]
Department of Computer Science[†], NCSA[‡], University of Illinois Urbana-Champaign
rlhill@uiuc.edu, wangj@ncsa.uiuc.edu, klara@cs.uiuc.edu

Abstract

In this work, we propose a framework for quantifying non-functional requirements (NFRs). This framework uses quality characteristics of the execution domain, application domain and component architectures to refine qualitative requirements into quantifiable ones. Conflicts are resolved during the refinement process and more concrete non-functional requirements are produced.

1. Quantification Enhancements

This work builds upon the process oriented framework presented by Mylopoulos et al [1]. Specifically, we augment the domain analysis process in order to gather specific information regarding the quality characteristics of the execution domain, application domain, architectural domains and algorithmic domains. Additionally, we enhance the goal refinement process by incorporating domain characteristics and quantification nodes. A quantification node is an intermediate node that is used to express how domain characteristics interact with NFRs. The quantification node helps to refine the parent goal, identify conflicts between the parent goal and the domain characteristics, identify conflicts among NFRs, and identify design tradeoffs among multiple satisficing goals.

In the following sections, we describe the information that should be acquired during the domain analysis process and how this information can be used. In addition, we detail how quantification nodes work and some of the relationships that they may be used to express.

2.Acquiring Domain Characteristics

Domain analysis is a term used to describe the systematic activity of identifying, formalizing and classifying the knowledge in a problem domain [2]. It is viewed as an activity that occurs prior to requirements engineering. "While requirements engineering is concerned with analyzing and specifying the problem of developing a software application, domain analysis is concerned with identifying the commonalities between different applications under the same domain" [2].

Our view of domain analysis encompasses those domains which enable or provide supportive function to the application domain. Therefore, we are interested in acquiring quantifiable characteristics of the application domain and related execution, architectural, and algorithmic domains. We use these characteristics to further constrain non-functional requirement goals, thereby producing realistic and achievable non-functional requirements.

During domain analysis, we seek answers to questions such as 'What are the essential performance criteria for this type of application?', 'Where will the application be utilized, and what are the performance features of this environment, i.e. throughput, delay, loss?', and 'What are the performance characteristics of specific algorithms?' Answers to such question would vary based on the domain. For example, domain analysis within the execution domain may produce information regarding processor speed, radio range for wireless devices, battery life for mobile devices, average throughput, delay and loss characteristics of the network, etc. In addition, domain analysis within the application domain will hopefully produce quantifiable characteristics that are inherent to all applications of that type. An example of such an application characteristic is the 150 millisecond one way path propagation delay requirement for IP telephony applications. Studies show that humans tolerate delays in speech of approximately 150 milliseconds. After 150 milliseconds of delay, we begin to talk over or interrupt the speech of the other person. Furthermore, regarding the algorithmic domain, we are interested in the performance of algorithms that may be used to satisfy a specific non-functional requirement. For example, performance specifications of an encryption algorithm may be used to assess the feasibility of employing the encryption algorithm to provide confidentiality and protect the data's integrity.

2.1. Using Domain Characteristics

Domain characteristics may be used in a variety of ways. We use them to quantify satisficing goals and expose possible conflicts between NFR goals.

1. **Quantifying satisficing goals**. Recall that a satisficing goal is a design decision that has been chosen to satisfy a specific non-functional requirement goal. Quantifying a goal involves assigning a particular algorithm to perform the required function. Specific performance characteristics for the algorithm must be available before it can be assigned. After the assignment, the satisficing goal assumes the performance value of its assigned algorithm. When the measurement platform and execution platform differ, then the performance value of the algorithm is estimated.
2. **Conflicts between NFR goals**. Conflicts between NFR goals arise when the characteristics of an algorithm assigned to a satisficing goal violate another requirement.

3. Goal Graph Refinement

We incorporate domain characteristics into the goal graph refinement process by using domain nodes and quantification nodes. Domain nodes store specific domain characteristics, while quantification nodes illustrate the association between two or more domain nodes. Argumentation goals are used to relate the quantification nodes to NFR goals and satisficing goals. Possible relationships include summation, or, maximum, minimum, etc.

1. **Summation.** The value of the quantification node is the sum of the values of each represented domain node.
2. **Or.** The value of the quantification node is equal to the value of one of the specified domains nodes.
3. **Maximum.** The value of the quantification node is the maximum of all specified domain nodes.
4. **Minimum.** The value of the quantification node is the smallest of all specified domain nodes.

Figure 1 illustrates different associations between quantification nodes, domain nodes, NFR goal, and Argumentation goal. Figure 1(a) shows a NFR goal directly depending on a domain node. Figure 1(b) gives an example where one "summation" quantification node is used to capture (sum up) three domain characteristics. Figure 1(c) depicts an example where a NFR goal is refined into a "minimum" quantification node and an argumentation goal. In this case, the NFR goal is satisficed only if the "minimum" value of the quantification node is verified by the argumentation goal. Finally, Figure 1(d) shows a case where one quantification node is associated with another quantification node and two domain nodes.

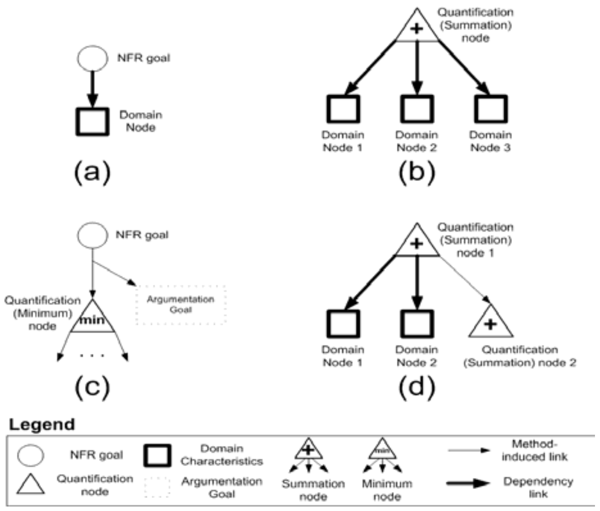

Figure 1 Associations between quantification nodes, domain nodes, NFR goals, and Argumentation goals

4. Conclusion and Future Work

In this paper, we propose to use specific domain information to quantify performance related non-functional requirements. In addition, we introduce domain nodes and quantification nodes to facilitate the goal refinement process, identify tradeoffs and requirement conflicts. We feel that the enhanced refinement process can be used to quantify any non-functional requirement that can be expressed numerically.

In the future, we plan to investigate ways to express the relative performance of algorithms. This work would be most useful when performance statistics have been obtained on obsolete hardware platforms. Additionally, we plan to investigate ways by which attributes of security and other requirements may be quantified.

5. References

[1] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos. *Non-functional Requirements in Software Engineering*. Norwell, Massachusetts: Kluwer Academic Publishers, 2000.

[2] P. Loucopoulos, V. Karakostas. *Systems Requirements Engineering*. Maidenhead, Berkshire, UK: McGraw-Hill, 1995.