Quantifying Non-Functional Requirements: A Process Oriented Approach

Raquel Hill, Jun Wang
Department of Computer Science/ NCSA
University of Illinois Urbana-Champaign
rlhill@uiuc.edu, wangj@ncsa.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 useful and realistic non-functional requirements are produced.

In addition to providing the framework, we present a case study of how the framework was used to resolve conflicting requirements within a system that provides a secure IP telephony service.

1. Introduction

Users of today's computing and information systems expect these systems to perform their tasks in a timely manner, to provide accurate results consistently, to provide secure transactions, and to be available even when under attack. These enhanced user expectations and the growing security threat to such applications are driving researchers to investigate ways to accurately specify these non-functional requirements (NFRs) and incorporate them into the design of software systems.

Incorporating non-functional requirements into the design process is not a simple task. Researchers face many challenges including accurately refining requirements from abstract goals, formally specifying requirements, incorporating these requirements into models used for specifying functional requirements and resolving design conflicts that arise from merging multiple NFRs.

Researchers have begun to investigate ways by which NFRs can be refined and incorporated into designs, but little has been done within the areas of quantifying NFRs and resolving conflicts among competing NFRs. Conflicts occur because the requirements are usually vaguely specified and individually refined, without consideration to how they may impact the others. Therefore, even after a refinement process, unrealistic requirements may be produced.

Given the growing demand for software that provides real-time performance guarantees and is resilient to denial of service (DoS) attacks, software designers must be able to specify goals that are realistic and achievable. To specify such goals, information regarding the quality characteristics of the execution environment, application domain, architectural domain, and algorithmic domain are needed. This quality information can be used to help designers identify possible conflicts between requirements and propose refinements to requirements that conflict with domain specific quality information. For example, an IP telephony application can withstand an inter-arrival packet delay of 150milliseconds. Let's assume that the execution domain for this application is the Internet, where the average one-way path delay is 75 milliseconds. Now assume that software engineers are designing a secure IP telephony application that has requirements for authentication, confidentiality and integrity. Software Engineers need to know whether the performance overhead of the security algorithms will violate the basic performance requirement. Given this example, qualitative reasoning, may lead to an unknown conclusion. Specific information regarding the performance of encryption algorithms are needed to identify and resolve possible conflicts among requirements. These performance characteristics may also enable engineers to define design tradeoffs.

In this work, we address the problem of quantifying NFRs. Through our quantification process, we resolve design conflicts that arise when trying to satisfy multiple NFRs. In this paper, we focus on performance and security requirements, but we feel that our approach can be applied to a more diverse set of requirements. Our approach builds upon previous process oriented work by incorporating domain specific quality information into the goal refinement process. These domain characteristics are numeric values that describe some feature of the domain.

The remainder of this document is organized in the following manner. In Section 2, we give an overview of the process oriented approach. In section 3, we detail how we enhance this approach. In section 4, we present an example of how the enhancements may be used to quantify non-functional requirements and resolve conflicts. In section 5, we present our conclusion and discuss future work.

2. Process Oriented Approach

Previous research may be either characterized as process-oriented or product oriented. Process oriented techniques aim to integrate non-functional requirements into the design process while product oriented approaches focus on evaluating the end product to determine whether it satisfies the NFRs.

We build upon the process oriented work of Mylopoulos et al [1,2,3]. Their work proposes a framework for refining non-functional requirements and incorporating them into the design process. The approach is qualitative and uses ideas from qualitative reasoning. Requirements are refined through the process of constructing a goal graph. The goal graph contains three mutually exclusive classes of goals: non-functional requirement goals, satisficing goals and argumentation goals. Non-functional requirement goals include such categories as accuracy, security, development costs, operating costs, hardware costs and performance. Satisficing goals include categories of design decisions that may be adopted in order to satisfice one or more non-functional requirements. The term satisfice refers to providing satisfactory designs, not optimal ones [1]. Argumentation goals are formal or informal claims that provide support or counter evidence for a goal or goal refinement.

At the root of the graph is an abstract non-functional requirement. The NFR may be further refined into a set of more concrete NFRs. These refined goals are the children of the root. Satisficing goals may then be proposed as a means for satisfying an NFR. The satisficing goal then becomes a node within the graph. Correlation rules are used to express implicit relationships between individual NFRs and identify conflicts.

3. Quantification Enhancements

This work builds upon the process oriented framework presented by Mylopoulos et al [1,2,3]. 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 used to express how domain characteristics interact with NFRs. The quantification node either helps to refine the parent goal, identify conflicts between the parent goal and the domain characteristics, identify conflicts among NFRs, or identify design tradeoffs among multiple satisficing goals.

In the following sections, we describe what information 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.

3.1 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 [7]. 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” [7].

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.

3.2 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.

3.3 Goal Graph Refinement

Goal graph refinement follows the process oriented approach defined by Mylopoulos et al [1,2,3]. We assume that the NFR goals have been acquired during the requirements elicitation process. Therefore we focus only on refining those goals and incorporating domain characteristics into the refinement process.

We incorporate domain characteristics 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.

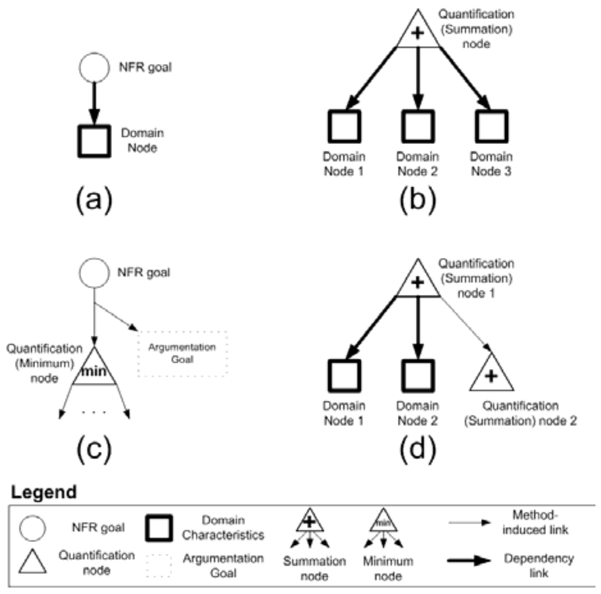


Figure 1 Associations between quantification nodes, domain nodes, NFR goals, and Argumentation goals

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.

4. Refinement Example

We will use the IP telephony application as an example to elaborate how the quantification enhancements are added to the goal graph and how the refinement is conducted. This example showcases dependency relationships between satisficing goals and domain nodes. In addition, it illustrates a quantification node that expresses a summation relationship between domain nodes.

4.1 Secure IP Telephony

Software engineers are designing a secure IP telephony application that has additional requirements for confidentiality (including both authentication and data encryption), integrity, and availability. To meet these additional security requirements, some security algorithms have to be used, and extra performance overhead is introduced. The designers must assess whether the extra performance overhead of the security algorithms will violate the basic performance requirement. Therefore, there may be a conflict between the basic performance requirement and the additional security requirements.

Recall that an IP telephony application should ensure a basic performance requirement that the inter-arrival packet delay is less than 150ms. This requirement is our application domain characteristic.

In addition, let's assume that the execution domain for this application is the Internet (WAN), where the average one-way propagation delay is 75 ms. .

Given this example, the designers must determine whether the extra performance overhead of the security algorithms will violate the application domain's performance requirement. Specific information regarding the performance of security algorithms is needed to identify and resolve the possible requirements conflict.

Note that in this example, we assume that we have already obtained all requirements during the elicitation process.

Figure 2 shows a detailed goal graph for the secure IP telephony example, with our quantification enhancement. Specifically, two non-functional requirements are refined: *Performance*[connection] and *Security*[connection]. *Performance*[connection] is decomposed into two goals: *Bandwidth*[connection] and *Delay*[connection], while *Security*[connection] is decomposed into *Confidentiality*[connection], *Integrity*[connection], and *Availability*[connection]. Two satisficing goals, *Encryption*[connection] and *Authentication*[connection] are proposed to further refine *Confidentiality*[connection].

We use bold squares to represent the domain characteristics. In Figure 2, we also identify three such domain characteristics: *Encryption-Algorithms*[connection], *Authentication-Methods*[connection], and *Propagation-Delay*[connection], which will be further broken down in Figure 3. The satisficing goal, *Encryption*[connection] depends on the domain node *Encryption-Algorithms*[connection], while the satisficing goal, *Authentication*[connection] depends on the domain node, *Authentication-Methods*[connection].

Now, we use two **quantification nodes**, *Processing-Delay*[connection] and *Total-Delay*[connection] (shown as triangles in the figure), to capture the quantifiable characteristic of delay for a connection. Since we are interested in summation of delays, both quantification nodes are marked by "+" signs. By using the quantification nodes, we clearly describe the relationship between the total delay, propagation delay, processing delay, and the delays of security algorithms. The overall goal to choose encryption algorithms and authentication methods to meet the security requirement, while keeping the total delay within the required delay bound.

Note that Figure 2 shows a snapshot of the goal graph right before the expansion process, and we have omitted the detailed refinements of the non-functional goals of *Bandwidth*, *Integrity*, and *Availability*.

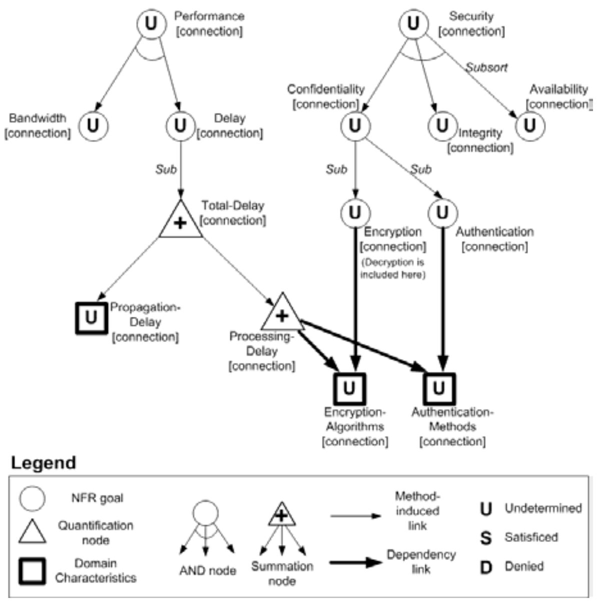
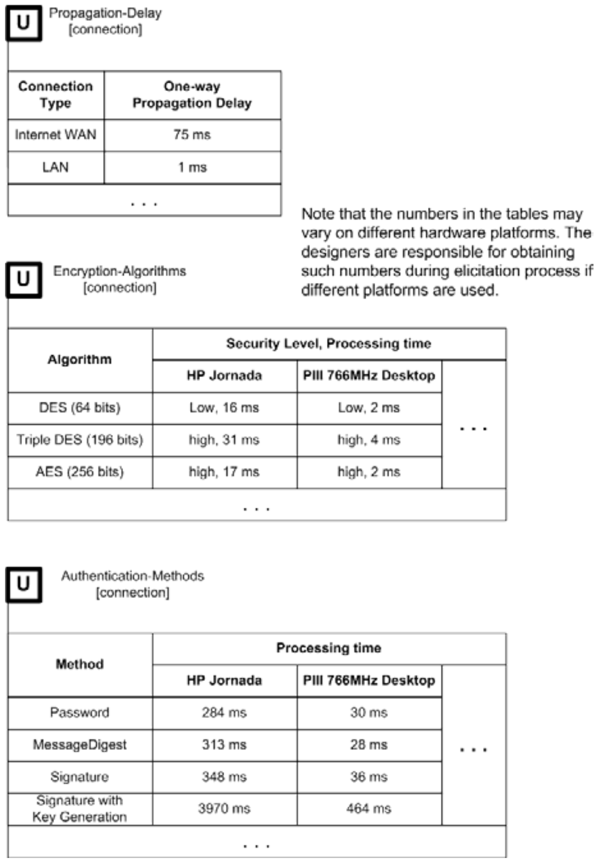


Figure 2 Goal graph of the IP telephony example



U Propagation-Delay [connection]

Connection Type	One-way Propagation Delay
Internet WAN	75 ms
LAN	1 ms
. . .	

Note that the numbers in the tables may vary on different hardware platforms. The designers are responsible for obtaining such numbers during elicitation process if different platforms are used.

U Encryption-Algorithms [connection]

Algorithm	Security Level, Processing time		
	HP Jornada	PIII 766MHz Desktop	
DES (64 bits)	Low, 16 ms	Low, 2 ms	. . .
Triple DES (196 bits)	high, 31 ms	high, 4 ms	
AES (256 bits)	high, 17 ms	high, 2 ms	
. . .			

U Authentication-Methods [connection]

Method	Processing time		
	HP Jornada	PIII 766MHz Desktop	
Password	284 ms	30 ms	. . .
MessageDigest	313 ms	28 ms	
Signature	348 ms	36 ms	
Signature with Key Generation	3970 ms	464 ms	
. . .			

Figure 3 Example domain characteristics breakdown

Figure 3 lists algorithm options for specific security domains [8]. This information will be used to assign specific algorithms to satisficing goals and ultimately satisfy a subset of the requirements.

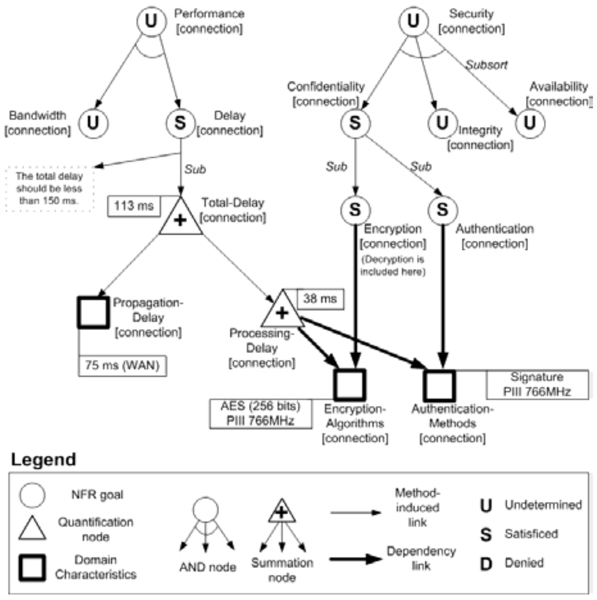


Figure 4 Goal graph of the IP telephony example, after expansion process

During the goal graph expansion process, the effect of each design decision is propagated from offspring to parents via *evaluation* procedure (which labels nodes as *satisficed* (S), *denied* (D), *undetermined* (U), etc.) [2]. Figure 4 shows a snapshot of the goal graph after the *Delay* and *Confidentiality* non-functional goals are evaluated. In this case, we choose PIII 766MHz desktop PCs as the hardware platform. We use 256-bit-key AES algorithm as the encryption algorithm and use signature as the authentication method. According to the domain characteristics shown in Figure 3, AES algorithm takes 2 ms and the signature method takes 36 ms on the chosen hardware platform. Therefore, the processing delay will be 38 ms. Since the typical propagation delay of the Internet is 75 ms, the total delay (summation of the processing delay and the propagation delay) will be 113 ms, which satisfies the "*less than 150 ms*" delay requirement. Hence, the *Delay* non-functional goal in Figure 4 has been marked as "S". Likewise, assuming that the AES algorithm and the signature method can reach the required security level, we mark the *Confidentiality* goal as "S", too.

The secure IP telephony example clearly shows that our quantification enhancements are able to help software engineers to identify and resolve possible requirement conflicts in a more precise way, and to define design tradeoffs.

5. Conclusion and Future Work

In this paper we have shown that specific domain information can be used to quantify performance related non-functional requirements. Quantification of these requirements facilitates goal refinement, helps to identify conflicts and design tradeoffs. In addition to performance requirements, this framework 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 security and other requirements may be numerically represented.

6. References

[1] J. Mylopoulos, L. Chung, and B. Nixon, "Representing and Using Non-Functional Requirements: A Process-Oriented Approach", *IEEE Transactions on Software Engineering*, 18(6), June 1992, pp. 483-497.

[2] L. Chung and B. Nixon, "Dealing with Non-Functional Requirements: Three Experimental Studies of a Process-Oriented Approach", *Proc. 17th International Conference on Software Engineering*, Seattle, Washington, April 1995, pp. 24-28.

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

[4] L. Liu, E. Yu, and J. Mylopoulos, "Security and Privacy Requirements Analysis within a Social Setting", *Proc.11th IEEE International Requirements Engineering Conference (RE'03)*, Monterey Bay, California, September 2003, pp. 151-161.

[5] R. C. Linger, N. R. Mead, and H. F. Lipson, "Requirements Definition for Survivable Network Systems", *Proceedings of the International Conference on Requirements Engineering*, Colorado Springs, CO, April 1998.

[6] A. Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour", *Proceedings of the IEEE International Symposium on Requirements Engineering*, Toronto, August 2001, 249-263.

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

[8] Chui Sian Ong, "Quality of Protection for Multimedia Applications in Ubiquitous Environments", *Master Thesis*, University of Illinois at Urbana-Champaign, 2003.

[9] J. Peters and W. Pedrycz. *Software Engineering, An Engineering Approach*. John Wiley and Sons, Inc., 2000.

[10] I. Sommerville. *Sofware Engineering*, 6th ed. Addison-Wesley, 2001.