

An Approach To Quantitative Non-Functional Requirements In Software Development

Andrew J. Ryan
School of Information Technology and Engineering
George Mason University
MS 5D3 Fairfax, VA 22030
{ajryan@gmu.edu}

Abstract. Non-functional requirements are commonly called the qualitative aspects of a system -- testability, mobility, and scalability, to name a few. However, when taking a holistic view of a system, non-functional requirements take on a quantitative nature. This paper will describe the Requirements Hierarchy Approach (RHA), a quantifiable method to measure and manipulate the effect non-functional requirements have on a system by capturing the utility of functional requirements. Through the use of agent-oriented programming, multi-attribute utility analysis (MAUA), and decision science theory, non-functional attributes (of a system) can be used as containers in an attempt to capture the broadest cross-section of functional requirements, based on stakeholder input. The final result will be an optimal set of requirements that satisfy the stakeholder's needs and are not in opposition to one another.

INTRODUCTION

Requirements drive any systems engineering effort. Most of the focus has been placed in the area of functional requirements. According to Alford, "In nearly every software project which fails to meet performance and cost goals, requirements inadequacies play a major and expensive role in project failure"(Alford 1979). Although written "requirements inadequacies" it should read "functional requirements inadequacies." There has been a tremendous amount of money and effort earmarked to solve this issue. With the emphasis on functional requirements, it is hard to understand why systems engineers still struggle to create a complete set of requirements for a stakeholder. One reason for this has been the inattention to non-functional requirements (NFRs) and the affect they have on fulfilling functional requirements. This is most notable in the omission of non-functional requirements by the two documents which are considered the guidelines for the Engineering of Systems in the United States: Electronics Industry Association (EIA) 632 and the International Standards Organization (ISO) 15228. The choice of this topic is a small step in an attempt to give non-functional requirements a more prominent role in the field of systems engineering.

BACKGROUND

There has been some prior work on non-functional requirements (NFRs) in systems/software engineering. Kenneth Chung presented one of the earlier attempts to capture knowledge in this domain (Chung 1993). His work was later followed up by Myopoulos et al. (Myopoulos 1999). Myopoulos suggested that all requirements be viewed as goals. Each goal would be an umbrella for related functional and non-functional requirements. As these requirements are satisfied or satisfied¹ the system goals are effectively satisfied. Missing from this formula is the impact the stakeholder can have on the requirements elicitation process and the gap that is often present between stakeholder vision and requirements representation.

One of the strong points of the Requirements Hierarchy Approach (RHA) is that it creates a visual tool which illustrates to stakeholders how their requirements relate to each other and the effect this may have on total system performance. Oftentimes, stakeholders have overlapping intentions (not to mention differences in opinion with the developers) and the need for tradeoff becomes necessary. Barber et al. espouses this belief, noting "System stakeholders have varying notions on the purpose of requirements gathering and these differences may lead to communication problems. Further complicating the communication process is the 'requirements gap' that separates the developer from the client" (Barber et al. 1997). The requirements gap is one of the main targets of this paper. Moreover, the Requirements Hierarchy Approach (RHA) can lead to positive results in requirements elaboration, requirements tradeoff, requirements validation, and verification.

NON-FUNCTIONAL REQUIREMENTS EXPLORED

Non-functional requirements are best described as quality-based requirements which can only be realized at the end of the system development process. The overall measurement of a

¹ To get a result that is good enough. Initially coined by Herbert Simon in his 1957 book *Models of Man*.

non-functional requirement is based on the functionality of the system as a whole, and cannot be viewed as independent entities. The ISO has identified a set of quality characteristics (or non-functional requirements) which provides a solid taxonomy for NFRs (see figure 1).

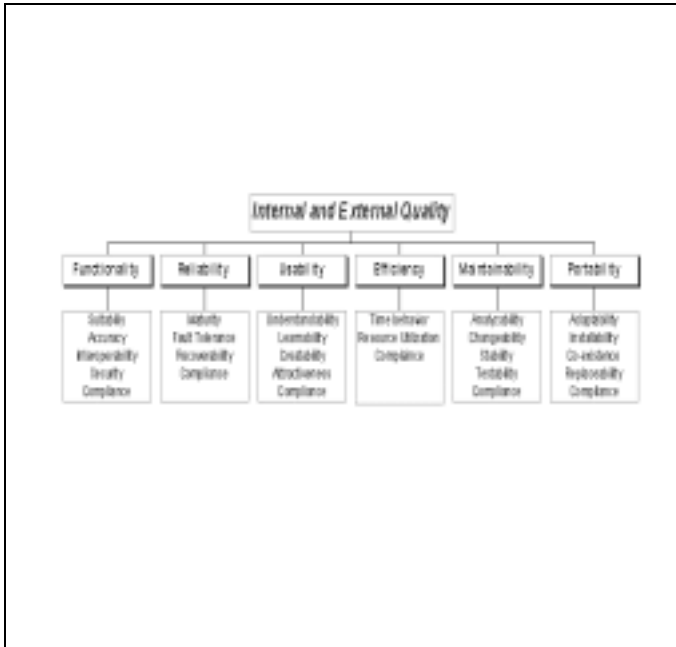


Figure 1. ISO/IEC 9126 Taxonomy of Quality Requirements (ISO/IEC 1991)

Taking this a step further, non-functional requirements can be quantified by the percentage of functional requirements they satisfy. That is, by placing non-functional requirements at the top of the “requirements hierarchy” and allowing their presence to filter down into the functional requirements, a measurable unit can be established. The rest of this paper will be dedicated to explaining this process in greater detail.

ROLE OF NON-FUNCTIONAL REQUIREMENTS IN A SYSTEM

There is much debate about the role non-functional requirements should play in the requirements elicitation and system development process. Systems (and software) engineers fall into two camps when it comes to this question, with the line of demarcation being the “qualitative” nature of non-functional requirements. A qualitative requirement is analogous to an immeasurable requirement, which in any discipline is something to be avoided at all costs. In recent years, there has been a push to make non-functional requirements more quantitative in nature (Chung 1993, Franch 1998, Jacobs 1999). One of the more promising ideas was put forth by a group at the University of Texas (Tran et al. 1999), which includes a CASE tool to view the decision process of their tool.

Stakeholders often describe the system they want built in layman’s terms. Usually citing specifications qualitatively with accompanying verbiage describing how it should happen. For example, one requirement may be: “The system shall allow anonymous login for unrestricted functions.” A second requirement states: “The system shall have protective firewalls to enforce varying levels of access.” Implicitly, the stakeholder is contracting two non-functional requirements: usability and security. For the most part, when viewed independently, non-functional requirements can be addressed routinely. However, there are times (such as the one above) when they present a clash which initiates a ripple affect felt by other requirements – in attempting to satisfy one requirement, the developer is negating another requirement.

Dealing with Requirements Clash. When two or more non-functional requirements directly contradict each other, in terms of fulfillment, we call this a requirements clash. This type of situation often goes unnoticed unless a visual representation, such as the Requirements Hierarchy Approach, is utilized as a tool to model requirements. There are two approaches that can be employed when this situation occurs:

1. Decrease the measure of satisfiability of one or more of the requirements.
2. Use the requirements hierarchy as a basis for requirements trade-off.

The first approach is the preferred method. The power of satisficing goals, comes in the ability to alter the sequence which satisfies a goal. Therefore, if a goal has five sub-goals, of which three are satisfied, one can then reduce the measurement for satisfiability to three with customer approval. By decreasing the impetus placed on satisficing a goal, it also decreases the dependency the requirements have on one other. Doing this can eliminate the clash or at least depreciate it to a point that it is manageable.

The second approach involves meeting with the stakeholder and, based on the requirements and the soft goal tree, to create a list of trade-off requirements. These requirements can then be re-prioritized to alleviate the clash among the requirements.

The hidden danger is that the conflicting nature of the desired functionality (impacted by NFRs) of a system is not realized until late in the development life cycle. By managing these potential conflicts early, the developer protects the success of the project from being compromised.

When bringing a system into being, great measures are taken to appease the stakeholder – to build a trusting relationship and confidence that the completion of the project is in fact a feasible scenario. This action usually appears in concrete form by a developer insisting they can meet all requirements. Taken at face value, there is nothing intrinsically wrong with this, until non-functional requirements are brought into the picture.

NON-FUNCTIONAL REQUIREMENTS IGNORED

Late in 1999, NASA's Mars Climate Orbiter and Polar Lander were lost in Mars' atmosphere due to inadequate attention paid to a pair of non-functional requirements: interoperability and testability. In terms of interoperability, the Orbiter was developed by separate companies: Lockheed Martin Aeronautics (main contractor) and the Jet Propulsion Laboratory (JPL). In calculating thrust for the unit, Lockheed Martin scientists used English measurements, while JPL scientists used the metric system. This oversight was only realized after the probe was deployed, a definite cause of the failure.

Secondly, the modules that were developed for the Polar Lander were not as succinct as initially thought. During system testing, one module was excluded (there was a known problem with that module) and tests were run. At a later time, the excluded module was tested independently and final delivery was made. It was not until the system failed in space that that scientists recognized that the omission of the module during system testing (they only ran one full test), affected the functionality of the landing gear. These two mistakes cost the United States taxpayers \$319M.

RHA METHODOLOGY

A non-believer in the importance on NFRs might question their importance by stating that systems have been built without much attention being paid to NFRs. It has been proven that omission of non-functional requirements can be detrimental to the success of a system. Alan Davis states, "Knowing how to specify behavioral requirements for a software system is only half the battle. All applications, from the most trivial to the most complex, have additional requirements that define the overall qualities or attributes to be exhibited by the resulting software system" (Davis 1993). Even if these non-functional requirements are not explicitly defined, it is still up to the developer to create a design which takes these into account. Through the creation of a requirements hierarchy, this step is ingrained into the system development process.

Linking functional requirements to non-functional requirements is not a trivial undertaking. It is worth reiterating that this method is not intended to be an elicitation tool, but a tool which aids in trade-off and understanding of requirements elaboration, tradeoff, validation, and verification. Moreover, many non-functional requirements will have to be implied from functional requirements or reworded to better departmentalize the requirement within the ISO/IEC 9126 taxonomy. It is the opinion of this author that all requirements can be reasonably positioned in the hierarchy represented in Figure 1. Although duplication within a category is allowed, it is encouraged to explore *why* the requirement cannot be placed in one category versus the other. The following section will further build on this idea and the process which goes into creating a quantitative aspect to NFRs. Figure 2 illustrates the recommended cycle when employing this methodology with a stakeholder.

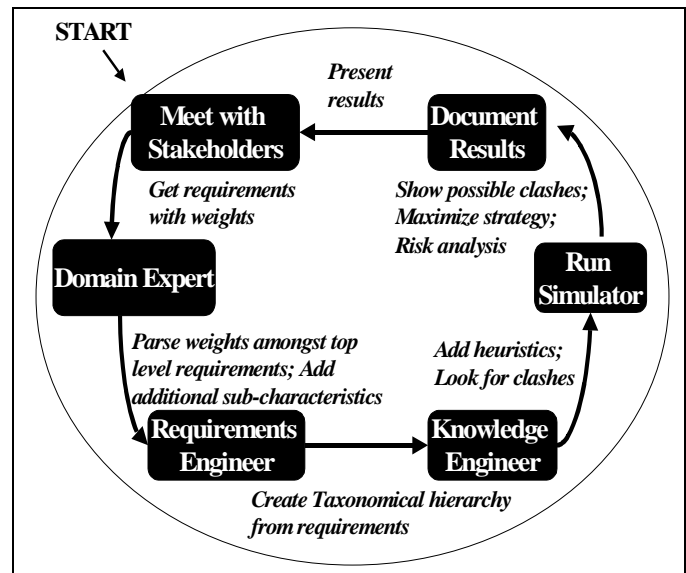


Figure 2. NFR Analysis Cycle

RHA AND MULTI-ATTRIBUTE UTILITY ANALYSIS

Multi-Attribute Utility Assessment (MAUA), first used by Adelman and Donnel (Adelman et al. 1981), was initially used to evaluate knowledge-based systems. MAUA uses scoring and weighting procedures to evaluate the overall utility of a system. Employing the hierarchy presented in Figure 3, it is possible to assign weights to each requirement (on a scale from .01 [wish list] to 1.0 [mandatory]) and traverse upward until a final number is reached. The weight given to each requirement can come either directly from the stakeholder (preferably) or a domain (subject matter) expert.

The final number to be evaluated is for the top-level NFRs. Functionality, reliability, usability, efficiency, maintainability, and portability need to be assessed in relation to each other. That is, the entire system is deemed to be (at) 100% when *all* requirements are satisfied. This number is the summation of the lower six NFRs. It is up to the domain expert to allocate priority across the six categories, based on the project requirements. Once this has been done, the process now switches to the requirements engineer to fill in the skeleton.

The process of linking functional requirements to quality-based attributes is a very objective task. The requirements engineer should work closely with the domain (subject matter) expert in doing this. It will be shown later in this paper (via experiment) that agreement (in a hierarchy) among multiple parties is possible even without communication. The ability to speak with experts while creating the hierarchy will only strengthen the reliability of the hierarchy.

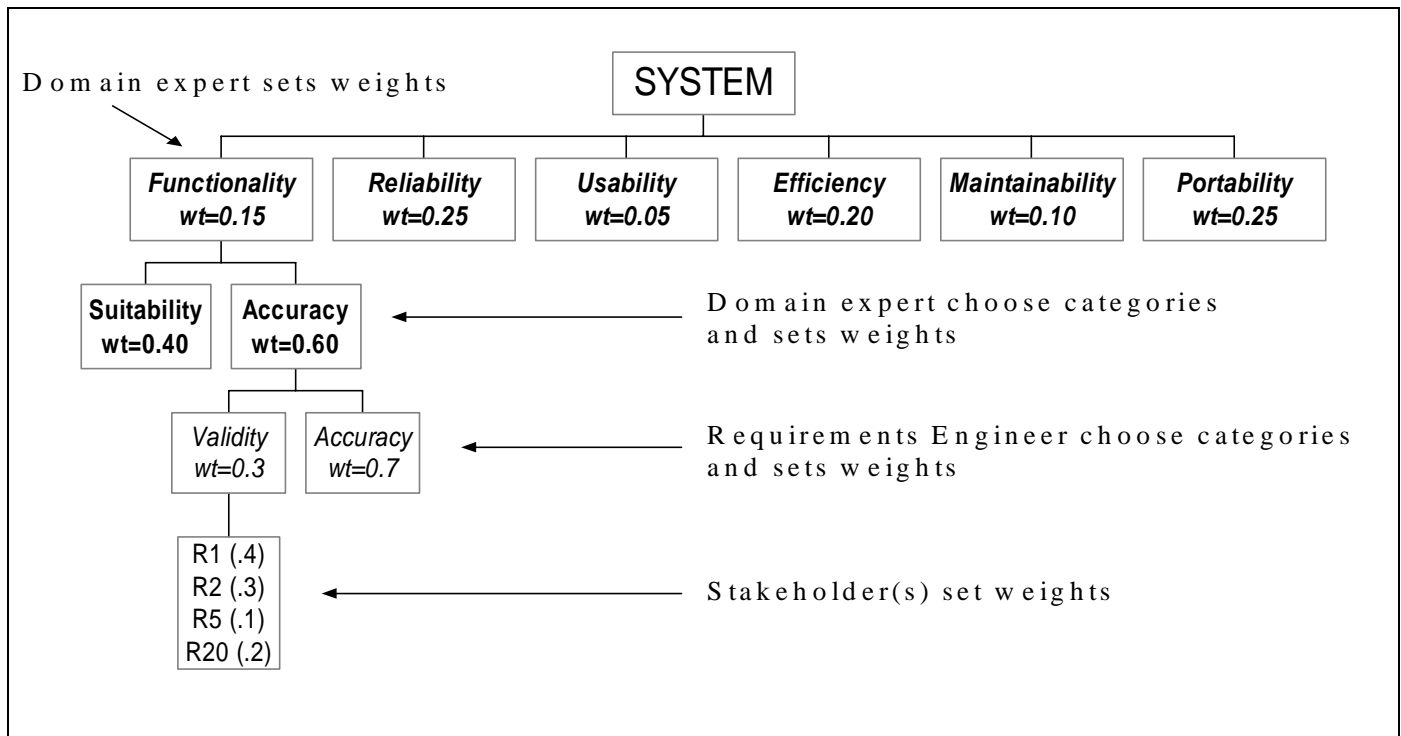


Figure 3. Partial Requirements Hierarchy

QUANTITATIVE ANALYSIS OF RESULTS

It would be impractical to believe a static representation of non-functional requirements can solve any problems a requirements engineer may encounter. In using the RHA, subject matter experts and experienced requirements/knowledge engineers are invaluable. Without a rigid taxonomy of NFRs to act as a baseline, familiarity with this approach becomes essential.

While only experience can provide the requirements engineer with the intuition to analyze the results, the following is a prescribed routine, once the hierarchy has been created:

1. Comparison of each of the top level nodes (functionality, reliability, usability, efficiency, maintainability, portability)
 - a. A low score does not necessarily mean product failure. Portability may not be of relative importance to a stakeholder if the system is an electrical heating system for a skyscraper; however, if it were a solar powered generator, portability would become of prime importance to capture as much of the sun as possible.
2. Check for uniqueness among requirements:
 - a. It is the responsibility of the requirements engineer to parse out requirements in an unbiased manner, regardless of duplication.

An inexperienced requirements engineer may want to limit their hierarchy to the 28 sub-NFRs in the ISO/IEC model, while a seasoned requirements engineer can be more detailed and have more categories.

- b. While there is nothing intrinsically wrong with requirements appearing in more than one category, on a case-by-case instance it may be necessary to reevaluate the requirement in terms of scope and fulfillment.
 - i. To avoid bias, if possible, have more than one knowledgeable engineer do this (include something about sensitivity analysis).
 - ii. Another approach can be splitting requirements into sub-parts, therefore capturing individual functionality in each respective sub-requirement.
3. Take a heuristic view of the requirements breakdown and make suggestions to the stakeholder where deemed appropriate.

Having the entire team sit in on this evaluation process is always a good idea so all disciplines and viewpoints are present to provide perspective.

PRESENTATION OF RESULTS TO STAKEHOLDERS

The Requirements Hierarchy Approach (RHA) lends itself more to prototype-driven development efforts (spiral, incremental, evolutionary) than iterative approaches (i.e. waterfall). But, it can be looped and used in a waterfall process too. Regardless of the method used, there will be a point in time when the stakeholder will reenter the picture. This meeting will give birth to opportunities for tradeoff talks, issues of satisficing versus satisfying, and can serve as a baseline for initial user acceptance. Stakeholders will be impressed by the simplicity of the requirements hierarchy and will be able to match their project goals against the aspiration levels listed for each requirement. Figure 4 demonstrates the actions a stakeholder can take after the first iteration of the cycle.

the second part, selected respondents were then asked to categorize each requirement using the first tier (the top six) of the ISO 9126 framework. The users were given official ISO definitions to fully understand the intended meaning of each category. The results of the survey were then queried by profession, gender, and education to determine trends (if any).

In the first part of the experiment, respondents collectively identified the same requirements as more important than others. Among those receiving the highest scores were: “The system shall operate on a universally accepted platform” and “Users shall receive 24 hour technical support.” Low importance were given to requirements dealing with the inclusion of a voice recognition interface and the usage of Commercial off the Shelf (COTS) solutions where possible. In addition to the average of each relative importance, the variance was also calculated as a measure of

uncertainty among the total. Requirements with high/low relative importance but high variance are further examined. One interesting discovery was that the entire scale (0-10) was not used by many respondents, especially those with non-technical backgrounds. This idea is not a new one as math/science disciplines use the total scale much more than their liberal arts/social science counterparts. Understanding the stakeholder is an

important aspect of using the RHA.

The second part of the experiment tested the respondents’ ability to place each requirement into their non-functional requirement categories. This exercise is important, as it forces the stakeholder to anticipate how their requirements will affect the system. Stakeholders will often agree to a requirement in principle without realizing that it has a broad definition and may affect various functions of the system. This is especially common when the stakeholders represent various departments within the organization. Respondents were allowed to choose two categories: a first and a second choice. The first choice was given ten (10) points and the second five (5) points. The results were then measured against a silver standard that was established prior to conducting the survey. Respondents successfully matched the silver standard in 16 of

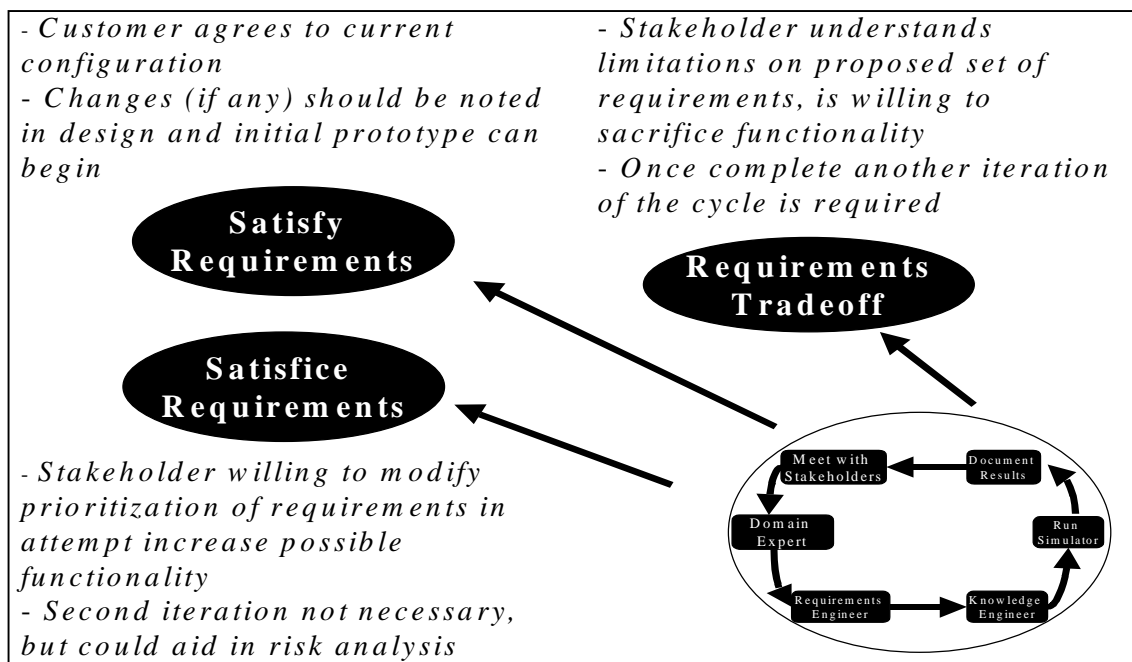


Figure 4. Stakeholder Options after first iteration

THE REQUIREMENTS HIERARCHY APPROACH IN PRACTICE

To demonstrate the effectiveness of the RHA, a two-part experiment was performed via a web-based questionnaire. The first part consisted of twenty-five requirements which were taken from a System Requirements Specification (SRS) used in a graduate level course at George Mason University. The participants were given the system’s Statement of Work (SOW) and then asked to rate the relative importance (on an 11 point scale) of each requirement with respect to the SOW. In

21 (76%) requirements. This illustrates that many of the requirements are broadly understood and only those other requirements need to be reviewed. Moreover, if face-to-face communication were possible, this percentage would be even greater.

With the experiment complete, the final part was to map the results. For the sake of exhibiting the full potential of the RHA, each category has been broken down into its second tier. Furthermore, under *usability*, the sub-category marked *understandability* has been branched to show an alternate configuration (note: only one can be chosen). The overall system score shows that trade-off is necessary if full functionality is to be achieved. Finally, the bold on R15 and R24 indicate requirements clash (see Appendix A for full figure).

CONCLUSION

Benefits of the Requirements Hierarchy Approach. In using the Requirements Hierarchy Approach (RHA), clear gains can be found in the areas of requirements elaboration, tradeoff, and requirements validation and verification. Involving the stakeholder in the initial concept phase helps clarify their vision of the system. Furthermore, categorizing requirements and integrating the level of importance stakeholders identified creates a synergistic effect and sets the table for requirements tradeoff and preliminary acceptance metrics. A complimentary benefit of establishing acceptance levels early in the life cycle is decreasing the risk involved in building a system the stakeholders asked for, but not what they wanted. More eloquently put, oftentimes developers appropriate resources to functionality that a user may care little about. In using the RHA, developers can be more in tune with the stakeholder's aspiration level as it pertains to specific functionality.

Another benefit in using the RHA is requirements management/project monitoring. With requirements listed categorically, a project manager is able to track progress back to the hierarchy. Although developers would not be writing code against the hierarchy, it creates a measurable correlation between the functionality they are working on and the underlying hierarchy in which these requirements are a part.

Areas for Future Study. The methodology explained in this paper is ripe for further study. The main area of particular interest is automation. Knowledge-based systems are now entering their fourth generation – reflecting the usage of autonomous agents to capture data. Moreover, the field of knowledge engineering is just beginning to appreciate the power of agents. Organized correctly, there is great promise for a system that uses agents as the cognitive engine which “assists” in mapping the requirements and then performs analysis on the hierarchy that was created. In time, through refinement, the RHA could ultimately be viewed as an objective measure (which would be helpful for requirements verification and validation) as opposed to a subjective one.

Finally, adopting an agreed upon taxonomy for NFRs is of prime importance. It is unrealistic to expect designers and developers to incorporate an entity they cannot readily identify. While taxonomies aim to be inclusive of the entire set of entities in question, a one or two level taxonomy would suffice initially as according to Chung et al. (Chung et al. 2000), there are over 161 identifiable NFRs. Once an endorsed taxonomy is in place, researchers and practitioners would have the ability to operate around this standard, as opposed to operating from a de facto standard.

REFERENCES

- Adelman, Leonard; Donnell, M.L., *Evaluating decision support systems: A General framework and case study*. In S.J. Andriole (Ed.), *Microcomputer Decision Support Systems: Design, Implementation, and Evaluation* (pp285-310). Wellesley, MA: QED Information Science 1986.
- Alford, M and Lawson, J., “Software Requirements Engineering Methodology (Development)”, U.S.A.F. Rome Air Development Center RADC-TR-79-168, 1979.
- Barber, K. Suzanne, et al., “The Systems Engineering Process Activities (SEPA – Supporting Early Requirements Analysis and Integration Prior to Implementation Design)” *The Laboratory for Intelligent Processes and Systems The University of Texas at Austin* 1997.
- Chung, K. L., “Representing and Using Non-functional Requirements for Information System Development: A Process Oriented Approach”, Ph.D. Thesis, also Tech. Rpt. DKBS-TR-93-1, Department of Computer Science, University of Toronto, June 1993.
- Chung, Lawrence et al., *Non-Functional Requirements in Software Engineering* Boston: Kluwer Academic Press 2000.
- Davis, Alan. *Software Requirements: Objects, Functions, & States*, New Jersey: Prentice Hall, 1993.
- Franch, Xavier, “Systematic Formulation of Non-Functional Characteristics of Software” *Proceedings of the Third IEEE International Conference on Requirements Engineering* 1998.

- ISO/IEC Information Technology – “Software Product evaluation – Quality Characteristics and guidelines for their Use” ISO/IEC 9126 1991 (E).
- Jacobs, Stephan, “Introducing Measurable Quality Requirements: A Case Study” *Fourth IEEE International Symposium on Requirements Engineering* 1999.
- Myopoulos, John, et al., “From Object-Oriented to Goal Requirements” *Transactions of the ACM* January 1999.
- Tran, Quan and Chung, Lawrence, “NFR-Assistant: Tool Support for Achieving Quality” *IEEE Symposium on Application-Specific Systems and Software Engineering & Technology* 1999.

BIOGRAPHY

Andrew J. Ryan was raised in the Bronx, New York. He holds a Bachelors degree in Computer Science from Binghamton University, a Master’s degree in Systems Engineering and is currently working on a PhD in Information Technology – both at George Mason University. His interests include agent programming, knowledge based systems, and decision science theory. He is currently employed at Metron Inc. as a software analyst and also teaches at George Mason University. He is a member of IEEE, ACM, NSBE, INCOSE, and Who’s Who of International Professionals.

Appendix A – Requirements Hierarchy for Proposed System

