Improving Software Quality from the Requirements Specification

Joshua Eckroth
The Ohio State University
eckroth@cse.ohio-state.edu

Guy-Alain Amoussou
Humboldt State University
amoussou@humboldt.edu

ABSTRACT

The first stage of software development, functional requirements specification, is considered the most important stage in the software lifecycle. Requirements constructed in this stage affect all other stages of the lifecycle, and thus affect software quality. We provide a method for determining how functional requirements affect software quality. To do so, we utilize a functional modeling framework that includes a controlled language for requirements specification and assess software qualities. Then we apply an information entropy metric to measure the significance of each software requirement. Using this method the designer can identify which requirements, when implemented, will most affect software quality.

Categories and Subject Descriptors

D.2.1 [**Software engineering**]: Requirements; D.2.8 [**Software engineering**]: Metrics; D.2.9 [**Software engineering**]: Management—*Software quality assurance*

General Terms

Design, Measurement

Keywords

Requirements engineering, functional requirements, hypernyms, requirements metrics, software quality, information theory, information entropy

1. REQUIREMENTS FRAMEWORK

Requirements are typically expressed in natural language [8, 9] and often divided into two categories: functional and non-functional. Non-functional requirements are "cognitive requirements", or statements of human desires, such as "user login must be efficient and simple," while functional requirements only state the required functionality: "the system must allow users to log in" [6]. The manner in which functional requirements are specified, however, may possess ambiguities [5] and misrepresentations [7].

Instead of natural language, we employ a framework which dictates a simple format, "actions over objects," for specifying functional requirements. In particular, we focus on phrases with the syntax "[verb] [object]," with an active voice and transitive verbs, such as "request user credentials."

The Common Functional Modeling Framework (CFMF) [4, 3] defines functional requirements in the syntax described above, as actions over objects. The semantics of the CFMF involve the action and object choice. The objects will be project-specific, and as general as possible while still being distinct. The actions are chosen from a dictionary of "functional primitives," which are transitive verbs that are more general than other verbs which a requirements engineer may use. The functional primitive is a hypernym for other, more specific verbs.

2. REQUIREMENTS METRIC

A benefit for writing functional requirements in the Common Functional Modeling Framework is the relative ease of developing requirements metrics which further inform designers.

We utilize information entropy to determine the significance, in terms of information content, of specific functional requirements in a requirements document written using the Common Functional Modeling Framework. Previous work [1] has shown Shannon's formulation of information entropy [10], rather than other, more generalized forms [2], is most meaningful when measuring software systems. Since we position our information entropy metric as a requirements metric, the entities we are measuring are functional requirements. Each functional requirement has a measurable amount of information content, as do the modules composed of these functional requirements.

This metric helps designers quantitatively determine the impact functional requirements have on the functionality of the product, or meeting the stakeholders' needs. However, our goal is to improve software quality. To do so, we establish relationships between functional primitives and specific software qualities, such as *Secure*, *Maintainable*, *Portable*, and so on. These relationships help identify how software requirements impact software quality.

3. SOFTWARE QUALITY

When developing software, software quality should be addressed as early as possible. We empirically develop rela-

SOD'07 Science of Design Symposium 2007, Humboldt State University, Arcata, California, USA

tionships between functional primitives and software quality that provide knowledge about which functional primitives, and hence functional requirements, are and will be important to quality. The information entropy metric provides an additional aid by quantifying which functional requirements most impact software quality.

Before and during development, the functional requirements document should continually be checked against the functional primitive–software quality relationships so that, when designing, implementing, and testing functional requirements, the related software qualities are considered. Functional requirements with high information content should also be paid particularly close attention, as the qualities that their functional primitives impact will be correspondingly highly impacted.

4. CONCLUSION

Our goal is to enable a designer to improve the final product by addressing quality issues in the functional specification of the system, since this first set of requirements significantly affects the entire system engineering cycle. The process we outline in our research integrates quality concerns into the establishment of functional requirements.

5. ACKNOWLEDGMENTS

This research was funded by the National Science Foundation, Grant CCF-0453491, in the context of Research Experience for Undergraduates–Role Models in Science at Humboldt State University, Arcata, California, USA.

6. REFERENCES

[1] S. Abd-El-Hafiz. Entropies as measures of software information. In *17th IEEE International Conference on Software Maintenance (ICSM'01)*, 2001.

[2] J. Aczél and Z. Daróczy. *On Measures of Information and their Characterization*. Academic Press, 1975.

[3] G.-A. Amoussou. *Thèse présentée pour l'obtention du grade de Docteur de l'UTC*. PhD thesis, Universitè de Technologie, Compiègne, 1999.

[4] G.-A. Amoussou and S. Rohmer. Functional modeling: A survey for common framework design. In *Modeling and Simulation (ESM'2002)*, 2002.

[5] D. Berry, E. Kamsties, and M. Krieger. From contract drafting to software specification: Linguistic sources of ambiguity–a handbook. Online: http://se.uwaterloo.ca/~dberry/handbook/ambiguityHandbook.pdf, 2003.

[6] M. Felici, M.-A. Sujan, and M. Wimmer. Integration of functional, cognitive and quality requirements: A railways case study. *Information and Software Technology*, 42:993–1000, 2000.

[7] V. Gervasi and D. Zowghi. Reasoning about inconsistencies in natural language requirements. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 14(3):277–330, 2005.

[8] L. Mich, M. Franch, and P. N. Inverardi. Market research for requirements analysis using linguistic tools. *Requirements Engineering*, 9(1):40–56, 2004.

[9] C. Neill and P. Laplante. Requirements engineering: The state of the practice. *IEEE Software*, 20(6):40–45, 2003.

[10] C. Shannon. A mathematical theory of communication. *SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.