Verification of Non-Functional Requirements Using Formal Semantics

Danielle Gaither
Department of Computer Science and Engineering
University of North Texas
Denton, Texas USA
dcg0063@unt.edu

***Abstract*—Studies have shown that finding defects in a software system is much cheaper in terms of both money and time spent when done during requirements analysis, as opposed to development or testing. To this end, significant advances have been made in the early detection of system defects. However, research on exposing unexpected system behaviors is relatively scant, especially regarding to non-functional requirements (NFRs). We propose a model-driven approach grounded in formal semantics to expose unexpected behaviors in a set of NFRs and propose future work for evaluating the effectiveness of such an approach.**

I. Introduction

Boehm and Basili [1] claim that remedying software defects can be 100 times more expensive during testing than during requirements gathering. Requirements engineering (RE) involves eliciting natural language requirements from system stakeholders and trying to impose a structure on that information [2]. Most approaches to the RE process rely on using the requirements to derive one or more models of some aspect of the system, such as data or behavior.

One such requirements analysis solution is the Causal Component Model (CCM) [3], which attempts to expose unexpected behaviors in a system. Although Foyle and Hooey [4] raised concerns about the lack of accounting for unexpected behaviors in RE tools in 2003, very little work has been done in this area since then. We believe that a requirements analysis tool in this domain should be able to analyze non-functional requirements such as timing and security properties. Many requirements analysis methods either focus solely on functional requirements, or on later stages of system implementation [3].

The remainder of this paper will be structured as follows: first will be discussions of CCM and a survey of related work in the area, including an explanation of how our proposed work improves upon existing approaches. The next section includes a discussion of the approaches and goals intended for the proposed work, including the research questions to be answered. An overview of preliminary and remaining work will follow, as well as a discussion of the merit and impact of the proposed work. The paper will end with a brief conclusion.

II. Background and Related Work

In this section we provide background on the causal component model (CCM), first proposed by Aceituna and Do [3], and provide a survey of related work in the field.

A. The Causal Component Model

The purpose of CCM is to analyze a given set of requirements for the purpose of detecting potential unexpected system behaviors [3]. Further details about the CCM method are described in Section III.

B. Related Work

Many approaches to the RE process are model-based in some form, such as goal-oriented and scenario-based approaches. Goal-oriented approaches consider a system from the perspective of the goals intended to be satisfied by that system [5]. The goals are often expressed in terms of an ontology specific to the domain of the problem at hand [6]. A scenario-based approach considers the system as an aggregation of potential use-cases [7]. Both approaches can be and have been used at varying levels of abstraction to construct system models. However, Carrillo de Gea et al. [8] note that requirements modeling is one of the least supported features among current RE tools. Combining a model-based approach with the detection of unexpected behaviors and the addition of NFRs can therefore contributing to advancing the state of RE as a whole.

Bryant et al. [9] discussed some of the benefits and challenges of formalizing the semantics of domain-specific modeling languages. One benefit is amenability to formal verification methods, which is well-suited to safety- and security-critical systems. Hessel et al. [10] used timed automata via the UPPAAL model checker [11] to generate test cases for a real-time system. Lee and Bryant [12] implemented a two-level grammar using VDM++, the Vienna Development Method meta-language [13].

III. Approaches and Goals

This section will outline the goals of the intended work and the approaches to be studied. The goals of our work are: creating a formal semantics for quantifiable NFRs, and applying the created semantics to a requirements analysis tool. Our approach is explained in Figure 1.

A. Establishing a Formal Semantics for Quantifiable NFRs

The approach for creating the formal semantics will be based on a timed denotational semantics, and the requirements analysis process will use the CCM tool as an example. These

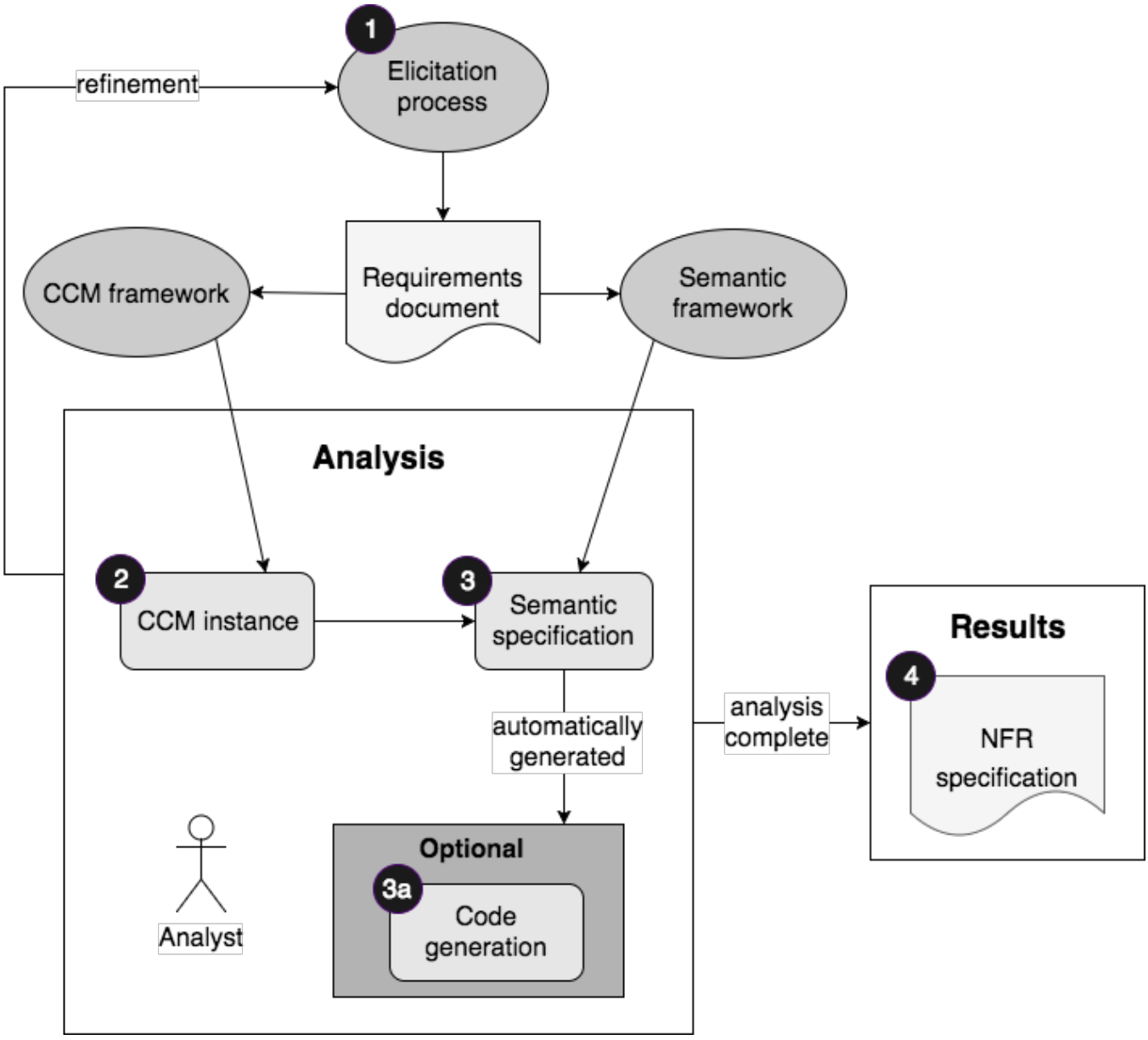

Fig. 1: An overview of the proposed approach.

are represented by the external inputs in the diagram in Figure 1.

The process begins with requirements elicitation (Step 1 in Figure 1). The elicited requirements will then be used to create a CCM instance (Step 2 in Figure 1). This information will serve as an input to create specific semantics for that CCM instance based on the larger semantic specification (Step 3 in Figure 1). Step 3a is optional, involving Coq's code generation capabilities, which can be combined with other code for analysis purposes.

The analysis will attempt to detect unexpected behaviors in NFRs for real-time embedded systems. If unexpected behaviors are found, the process returns to the elicitation phase for clarification. The whole process is repeated until all parties are satisfied with the quality of the CCM instance (Step 4 in Figure 1).

There is some precedent in the literature for creating and using a formal semantics for timed operations. Li [14] used duration calculus to extend the RAISE specification language to support timed operations. Duration calculus is an extension of interval logic [15].

B. Applying Formal Semantics to the Requirements Analysis Process

Since the idea of the CCM approach is to find system defects in the requirements-analysis phase, the lack of an actual system presents particular difficulties with regard to NFRs. An alternative approach is to verify such properties logically

by creating a formal semantic specification and analyzing said specification with currently existing tools.

The proposed changes to the CCM involve expanding the rule application and analysis phases. A formal semantics will be created for quantifiable NFRs, and the semantics will be used to check the validity of the NFRs. Note that the basic CCM approach does not change. The proposed updates merely broaden its scope. While the change might appear cosmetic, the implementation will require structural change in the CCM tool, as representing timed operations adds some complexity to the implementation [16], [17], [18].

C. Research Questions

The intent is to establish a formal denotational semantics for CCM for the purpose of supporting non-functional requirements, including timing information. The research should answer the following questions:

- RQ1. How can we add formal semantics to a requirements analysis system, such as CCM, to incorporate NFRs?
- RQ2. How does adding formal semantics and timed operations increase the expressiveness of a requirements analysis system?
- RQ3. What properties can be proven about NFRs once a formal semantics is established?

D. Evaluation Methods and Metrics

For a tool or approach to be useful, it generally has to satisfy two main criteria. First, it needs to perform the tasks it claims to perform, and second, it needs to do so in a way that offers some sort of advantage over competing methods. To satisfy the first criterion, we intend to implement the formal semantics we create in a theorem-proving tool such as Coq [19]. This will allow the correctness and completeness of the semantics to be established. To satisfy the second criterion, we expect the main advantage our approach will offer will be time savings compared to similar analysis methods. A human study will be necessary to evaluate the truth of this claim. One possible experiment could involve groups being given the same set of requirements and constructing analysis models using various methods. It is important to have a large enough sample size to control for variations in individual ability.

IV. Completed and Remaining Work

This section will describe preliminary work that has already been performed, to include: formalizing the semantics of a domain-specific modeling language, establishing the link between domain-specific modeling and requirements analysis, and domain analysis with respect to requirements engineering.

A. Establishing a Formal Semantics for a Domain-Specific Modeling Language

Preliminary work has been done in the area of creating and implementing semantics for a domain-specific modeling language. As a proof-of-concept, the MicroRPG modeling language was created to model a simple game. The abstract syntax of MicroRPG was established using a metamodel. A denotational semantics was created for the MicroRPG metamodel and implemented in Haskell [20]. Current work is focused on establishing a denotational semantics based on statecharts that can then be applied to analysis frameworks such as CCM.

B. Summary of Preliminary and Remaining Work

Our preliminary work in this area has shown the potential benefits of providing a formal denotational semantics for a domain-specific model. We have taken the following steps: created a modeling language for a sample domain, created and implemented a denotational semantics for a modeling language, and analyzed appropriate domains for a formal semantics-based requirements analysis tool.

We have recently completed a pilot study that uses statecharts as the basis for a formal semantics. The results of this pilot study have been accepted for publication [21].

The following work remains to be done: establish a larger-scale denotational semantics for NFRs for real-time embedded systems, implement the denotational semantics in a manner consistent with a CCM implementation, use the newly-enhanced CCM to verify timing properties and other quantifiable NFRs for a sample real-time embedded system, and perform empirical studies to evaluate the effectiveness of proposed CCM improvements.

V. Merit and Impact

Since many real-world operations are time-sensitive in some fashion, a robust requirements analysis system needs to be able to support such operations. Adding support for timed operations to CCM will make it more expressive and allow it to be used for more applications than is currently possible. Since functional requirements and nonfunctional requirements cannot be completely separated, it makes sense for a comprehensive requirements analysis solution to handle both. Another benefit is that formal analysis tools can be especially useful for mission-critical applications. Finally, a particular benefit of the denotational semantics-based approach is independence from a particular implementation.

VI. Conclusions and Future Work

We proposed a model-based approach for detecting unexpected behaviors of requirements, particularly non-functional requirements, which have not received the same attention in research as functional requirements have to date.

This work is in its early stages, so there is significant potential for future work. While timing and security requirements are important, there are also things like safety requirements to be considered. Also, we would like to build on our current work to analyze more complex sets of non-functional requirements. Integration with existing requirements analysis tools is also another priority for future work.

REFERENCES

[1] B. Boehm and V. R. Basili, "Software defect reduction top 10 list," *Computer*, vol. 34, no. 1, pp. 135–137, 2001.

[2] R. Dömges, S. Jacobs, M. Jarke, H. W. Nissen, K. Pohl, N. Maiden, A. Sutcliffe, C. Taylor, D. Till, P. Constantopoulos, G. Spanoudakis, Y. Vassiliou, G. Grosz, V. Plihon, C. Rolland, J. R. Schmitt, S. Schwer, S. Si-Said, C. Souveyet, J. Bubenko, R. Gustas, P. Holm, P. Johannesson, J. Ljungberg, and B. Wangler, "Defining visions in context: Models, processes and tools for requirements engineering," *Information Systems*, vol. 21, no. 6, pp. 515–547, 1996.

[3] D. Aceituna and H. Do, "Exposing the Susceptibility of Off-Nominal Behaviors in Reactive System Requirements," in *Proceedings of the IEEE International Requirements Engineering Conference*, vol. 23. IEEE, 2015, pp. 136–145.

[4] D. Foyle and B. Hooey, "Improving Evaluation and System Design Through the Use of Off-Nominal Testing: A Methodology for Scenario Development," in *of the Twelfth International Symposium on Aviation Psychology*, 2003.

[5] A. van Lamsweerde, "Goal-oriented requirements engineering: a guided tour," *Proceedings Fifth IEEE International Symposium on Requirements Engineering*, pp. 249–262, 2001.

[6] E. Yu and J. Mylopoulos, "Why Goal-Oriented Requirements Engineering," in *Proceedings of the 4th International Workshop on Requirements Engineering*, 1998, pp. 15–22.

[7] A. G. Sutcliffe, "Supporting scenario-based requirements engineering," *IEEE Transactions on Software Engineering*, vol. 24, no. 12, pp. 1072–1088, 1998.

[8] J. M. Carrillo De Gea, J. Nicolás, J. L. Fernández Alemán, A. Toval, C. Ebert, and A. Vizcaíno, "Requirements engineering tools: Capabilities, survey and assessment," *Information and Software Technology*, vol. 54, no. 10, pp. 1142–1157, 2012.

[9] B. R. Bryant, J. Gray, M. Mernik, P. J. Clarke, R. B. France, and G. Karsai, "Challenges and directions in formalizing the semantics of modeling languages," *Computer Science and Information Systems*, vol. 8, no. 2, pp. 225–253, 2011.

[10] A. Hessel, K. G. Larsen, M. Mikucionis, B. Nielsen, P. Pettersson, and A. Skou, "Testing real-time systems using UPPAAL," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4949 LNCS, 2008, pp. 77–117.

[11] G. Behrmann, A. David, and K. Larsen, *A Tutorial on Uppaal*. Springer-Verlag Berlin Heidelberg, 2004, vol. 3185.

[12] B.-s. Lee and B. R. Bryant, "Automated conversion from requirements documentation to an object-oriented formal specification language"," *Proceedings of the 2002 ACM symposium on Applied computing*, pp. 932–936, 2002.

[13] E. Dürr and J. van Katwijk, "VDM++, a formal specification language for object-oriented designs," in *Computer Systems and Software Engineering*. The Hague, Netherlands: IEEE, 1992, pp. 214 –219.

[14] L. Li, "Towards a denotational semantics of timed RSL using Duration Calculus," *Journal of Computer Science and Technology*, vol. 16, no. 1, pp. 64–76, 2001.

[15] M. R. Hansen and Z. Chaochen, "Duration Calculus : Logical Foundations," *Formal Aspects of Computing - Springer*, vol. 9, no. 3, pp. 283–330, 1997.

[16] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, 1994.

[17] J. Bengtsson and W. Yi, "Timed automata: Semantics, algorithms and tools," *Lecture Notes in Computer Science*, vol. 3098, no. 316, pp. 87–124, 2004.

[18] Ø. Haugen, K. E. Husa, R. K. Runde, and K. Stølen, "Why timed sequence diagrams require three-event semantics," in *International Workshop on Scenarios: Models, Transformations and Tools*, vol. 3466, 2005, pp. 1–25.

[19] INRIA, "The Coq proof assistant," 2016. [Online]. Available: https://coq.inria.fr

[20] D. Gaither and B. R. Bryant, "Toward Denotational Semantics of Domain-Specific Modeling Languages for Automated Code Generation," in *Presented at the International Workshop on the Globalization of Modeling Languages*, Miami, 2013.

[21] D. Gaither, H. Do, and B. R. Bryant, "Toward detection of abnormal behaviors in timing and security requirements," in *24th Asia-Pacific Software Engineering Conference (to be published)*, 2017.