Behavioral Grime: The Identification, Classification, Evaluation, and Prediction of Behavioral Design Decay in Design Patterns

Derek Reimanis, Ph.D. Comprehensive Exam

College of Engineering
Gianforte School of Computing

Friday, October 13th, 2017
2:00 p.m. – 3:00 p.m.
Barnard Hall 347

Abstract: Design patterns embody recurring and proper solutions to common problems in the software engineering domain. Pure design pattern implementations are found to be of high quality, meaning they are highly maintainable, testable, understandable, and reusable micro-architectures within the design of a software project. However, research has found that design pattern instances tend to evolve away from their initial pure implementations due to poor design choices implemented as a result of any one or combination of three sources during a pattern's evolution; (1) a developer being unfamiliar with a design pattern, (2) pressure from management expecting speedy implementations of features to meet market gaps, and (3) pattern quality being neglected, thus encouraging poor quality in future versions of that pattern's instance. *Pattern grime*, which is defined as unintended artifacts within a pattern instance that manifest during a pattern's evolution, has been found to be detrimental to the quality aspects of a design pattern, to the extent that it detracts from the benefits of using the design pattern in the first place. Previous work regarding design pattern evolution has only considered the structural aspects of pattern grime, or the elements of that pattern that include class members, such as methods and attributes, and relationships between elements of that pattern.

This work seeks to extend the body of knowledge surrounding design pattern evolution by capturing the behavioral features of design pattern grime, or the artifacts that appear while the design pattern instance is being executed by the software application. Completing this work involves a four-step plan; (1) the identification of behavioral grime in design pattern instances, (2) the classification of these behavioral grime artifacts, (3) the evaluation of behavioral grime on the software's quality while considering the trade-offs between developing good quality software vs. rushing software to market, and finally (4) the identification of indicators that seek to predict the presence of behavioral grime in future versions of a pattern instance. Included in this plan across all steps is an evaluation of the relationship between behavioral grime and structural grime, as well as case studies performed across both open source and commercial software applications. The completion of this plan will provide a significant and meaningful contribution to the field, which will constitute the Doctor of Philosophy in Computer Science degree for the author.