Poster: A Taxonomy of how Method Stereotypes Change

Michael John Decker
Department of Compute Science
Bowling Green State University
Bowling Green, OH, USA
mdecke@bgsu.edu

Christian D. Newman
Department of Software Engineering
Rochester Institute of Technology
Rochester, NY, USA
cnewman@se.rit.edu

Natalia Dragan
Department of Management and Information Systems,
Kent State University, Kent, OH, USA
ndragan@kent.edu

Michael L. Collard
Department of Computer Science
The University of Akron
Akron, OH, USA
collard@uakron.edu

Jonathan I. Maletic
Department of Computer Science
Kent State University
Kent, OH, USA
jmaletic@kent.edu

Nicholas A. Kraft
ABB Corporate Research
Raleigh, NC, USA
nicholas.a.kraft@us.abb.com

ABSTRACT

The role of a well-designed method should not change frequently or significantly over its lifetime. As such, changes to the role of a method can be an indicator of design improvement or degradation. To measure this, we use method stereotypes. Method stereotypes provide a high-level description of a method's behavior and role; giving insight into how a method interacts with its environment and carries out tasks. When a method's stereotype changes, so has its role. This work presents a taxonomy of how method stereotypes change and why the categories of changes are significant.

CCS CONCEPTS

• **Software and its engineering → Software creation and management**.

KEYWORDS

Software evolution, method stereotypes, software change

1 INTRODUCTION

A software system is typically in a constant state of evolution over its lifetime. At points in the lifetime of a system, its design may degrade or be broken by this constant change. This eventually requires developers to spend time redesigning parts (or the entirety) of the system. Code smells are one indicator of poor design or design degradation. Thus, there has been research conducted to automatically identify certain code smells with the goal of warning users when a system's design is potentially degrading [1-3]. The work presented here takes a similar approach. That is, we are trying to understand what types of changes to a system are potentially hazardous to the system's design. Here, we are specifically interested in how individual methods change over time. While a change to an individual method will not significantly degrade a system's design, changes to large sets of methods can. Hence, we propose a taxonomy based on a relatively simple abstraction that provides solid clues that a change is a potential problem.

The abstraction we are using is the idea of *method stereotypes* [4]. Method stereotypes present a powerful abstraction of the role and behavior of a method within the context of the class to which it belongs and how it is used within the system. A number of researchers have leveraged stereotypes for various applications [5-7]. Two simple examples of stereotypes are *accessor* (aka getter) and *mutator* (aka setter). A complete description of method stereotypes can be found in [4].

The goal of our change taxonomy is to provide an understanding of which types of changes from one stereotype to another stereotype are potentially hazardous. Intuitively, a method should not change drastically from its original intent. If there is a drastic change in its role and/or behavior this may be an indication of a problem. However, clearly systems do evolve and the role of methods sometimes must evolve in a positive manner. There may also be changes that fix poor designs and these types of changes may be reflected in changes in the stereotype of a method. This work, to the best of our knowledge, is the first to investigate changes in method stereotype and their consequences.

2 Taxonomy of Changes in Method Stereotype

As stereotypes describe a method's behavior and role at a high level, we are able to use transitions in stereotype to theorize about the consequences of such changes. We define a stereotype *transition* as a change in stereotype caused by changes made to a method. Stereotype transitions can be indicators of design improvement or degradation. For example, a change that causes a C++ method to transition from being a *non-const get* to a *get* is an example of a positive

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3194998

transition, as it forbids modification of object state by the callee and allows const objects to use the method. Thus, it improves design by restricting undefined behavior and makes the system easier to maintain. If the opposite occurs, a transition in stereotype from a *get* method to a *non-const get* method, we have an example of a potentially negative change that indicates a degradation in the design of the system (i.e., code smell). Note, the addition of a *non-const get* method is sometimes necessary in C++ and therefore valid. However, a change which replaces a *const* accessor (which is necessary for *const* objects) with a *non-const get* method indicates that development is loosening restrictions on previously forbidden behavior, in addition to limiting the objects in which the method can be invoked. This type of change must be highlighted and reviewed, hence we call it negative.

Table 1 contains our taxonomy on stereotype transitions. This taxonomy is not meant to be complete, but to highlight important transitions, i.e., those that have a well-defined consequence. Each classification in the taxonomy has a transition type (a name for that classification of transitions), a stereotype category (which indicate what category of method stereotypes [4] is included), a description of the transition type and what specific stereotype transitions are a part of that transition type, and lastly notes about the significance of such transition and whether such transitions are generally positive, negative, or neutral.

3 Conclusions and Future Work

A taxonomy of method stereotypes transitions is presented along with consequences (i.e., positive, negative, or neutral) associated with such changes. For future work, we will perform an empirical study to ascertain how stable (resilient to change) a method's stereotype is and what are the most prevalent transitions. In addition, a manual investigation will be performed to validate the consequences of method stereotype transitions as presented in the taxonomy.

REFERENCES

[1] Marinescu, R., "Detection Strategies: Metrics-based Rules for Detecting Design Flaws", in Proceedings of 20th IEEE International Conference on Software Maintenance, Chicago, IL, USA, 2004, pp. 350-359.

[2] Moha, N., Gueheneuc, Y.-G., Duchien, L., and Meur, A.-F. L., "DECOR: A Method for the Specification and Detection of Code and Design Smells", *IEEE Transactions on Software Engineering*, vol. 36, no. 1, 2010, pp. 20-36.

[3] Fontana, F. A., Zanoni, M., Marino, A., and Mantyla, M. V., "Code Smell Detection: Towards a Machine Learning-based Approach", in Proceedings of 29th IEEE International Conference on Software Maintenance (ICSM), Eindhoven, Netherlands, 2013, pp. 396-399.

[4] Dragan, N., Collard, M. L., and Maletic, J. I., "Reverse engineering method stereotypes", in Proceedings of 22nd IEEE International Conference on Software Maintenance (ICSM'06), Philadelphia, Pennsylvania, USA, 2006, pp. 24-34.

[5] Andras, P., Pakhira, A., Moreno, L., and Marcus, A., "A measure to assess the behavior of method stereotypes in object-oriented software", in Proceedings of 2013 4th International Workshop on Emerging Trends in Software Metrics (WETSoM), 21-21 May 2013 2013, pp. 7-13.

[6] Canfora, G. and Cerulo, L., "Impact analysis by mining software and change request repositories", in Proceedings of Software Metrics, 2005. 11th IEEE International Symposium, 1-1 Sept. 2005 2005, pp. 9 pp.-29.

[7] Linares-Vásquez, M., Cortés-Coy, L. F., Aponte, J., and Poshyvanyk, D., "ChangeScribe: A Tool for Automatically Generating Commit Messages", in Proceedings of 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, 16-24 May 2015 2015, pp. 709-712.

Table 1. Taxonomy of important stereotype transitions. The transition-type column labels each classification in the taxonomy, the stereotype-category column indicates what category of method stereotypes is included in the classification, the description-column is a description of the transition-type which specifies the associated method stereotype transitions, and the notes-column contains details about such transitions.

Transition Type	Stereotype Category	Description	Notes
Move to/from Unclassified	**Unclassified**	Method transitioned to/from Unclassified	• Methods that cannot be classified lack a clear abstraction. • To unclassified (negative) • From unclassified (positive)
Move to/from Non-const get	**Structural Accessor**	Method transitioned between non-const get and another Accessor.	• From Accessor to non-const get breaks ability use on constants/degrades design (negative). • From non-const get to Accessor increases information hiding. Method most likely should have always been Accessor (positive).
Add Collaborational	**Collaborational**	A method adds a Collaborational stereotype or transitions from another category to a Collaborational.	• Addition of Collaborational indicates increased dependency to other object(s).
Remove Collaborational		A method removes a Collaborational or transitions from a Collaborational to one of another category.	• Removal of Collaborational indicates decreased dependency to other object(s).
Add Degenerate	**Degenerate**	A method adds a Degenerate stereotype or transitions from another category to a Degenerate.	• Method's functionality has been diminished. • Indicates method does not have enough responsibility and consider removal.
Remove Degenerate		A method removes a Degenerate stereotype or transitions from a Degenerate stereotype to one of another category.	• Degenerate methods do not provide enough functionality. • Removal is generally a positive. • Indicates addition of more functionality or increased complexity.
Cross	**Multiple**	A change from Structural Accessor, Structural	• Massive change to function behavior.

Stereotype Boundaries	**Categories**	Mutator, and Creational to a different one of those categories.	• Change is suspicious and should be investigated.
Add/Remove/ Replace Categories		Method that has multiple stereotypes from at least two of the following: Structural Accessor, Structural Mutator, and Creational.	• Method has too much responsibility. • Presence is possible code smell. • Transition that adds additional method stereotypes indicates degrade in design. • Transition that removes stereotypes indicates a design improvement. • When categories are replaced with others indicates poor design.