21 Supporting Reverse Engineering Tasks with a Fuzzy Repository Framework

René Witte
Concordia University, Department of Computer Science
Montréal, Québec, Canada
me@rene-witte.net

Ulrike Kölsch
T-Systems International GmbH, Fasanenweg 5, Leinfelden, Germany
koelsch@acm.org

Abstract

Software reverse engineering (RE) is often hindered not by the lack of available data, but by an overabundance of it: the (semi-)automatic analysis of static and dynamic code information, data, and documentation results in a huge heap of often incomparable data. Additionally, the gathered information is typically fraught with various kinds of imperfections, for example conflicting information found in software documentation vs. program code.

Our approach to this problem is twofold: for the management of the diverse RE results we propose the use of a repository, which supports an iterative and incremental discovery process under the aid of a reverse engineer. To deal with imperfections, we propose to enhance the repository model with additional representation and processing capabilities based on fuzzy set theory and fuzzy belief revision.

Keywords: fuzzy reverse engineering, meta model, extension framework, iterative process, knowledge evolution

21.1 Knowledge Acquisition Process in RE

Reverse engineering can be described as a process of knowledge acquisition by proposing, validating, or falsifying hypotheses in order to form an abstract model. Deriving a conceptual model from a given implementation is a task fraught with ambiguity and vagueness because of the impedance mismatch problem.

Due to the large amount of information and the number of colliding hypotheses an automated support for the reverse engineers is highly needed. This automation must be able to keep track of the numerous analysis results, the proposed hypotheses as well as the interdependencies between all gained reverse engineering artifacts. Additionally, an automated support has to be able to detect and propagate information supporting or contradicting specific hypotheses or reverse model entities.

To our understanding the reverse engineering repository is the ideal point to offer such services. The meta model of a repository defines the representation abilities as well as the data manipulation capability of the reverse engineering process using it [1]. In order to improve and automate the RE process we are currently developing a fuzzy extension framework that can deal with imperfection and supports the continuing evolution of a knowledge basis with hypotheses based on formal concepts of fuzzy set theory and belief revision.

21.2 Knowledge Capturing and Representation

The RE repository and its meta model play a central role in establishing the technology that is most needed for dealing with imperfect information and hypotheses during reverse engineering. We developed a fuzzy extension framework that enhances the modeling capacity of RE meta models in such a way that imperfect data can now be handled.

The central requirement for such a model is the ability to handle several kinds of imperfections: *uncertain* and *vague* information, as well as the ability to handle *inconsistencies*. These requirements have also been identified by other research groups, e.g. in [2].

In this section we briefly outline the concept of our fuzzy extension framework for meta models of reverse engineering repositories. For a detailed description please refer to [3].

The formal basis for dealing with the mentioned kinds of imperfections is fuzzy set theory. For our fuzzy RE repository we deploy a fuzzy-set based model that has been developed especially for the use within information systems [4]. It provides the necessary features for supporting the requirements of the RE process, especially the ability to backtrack information and modify a knowledge base through non-monotonic belief revision operators.

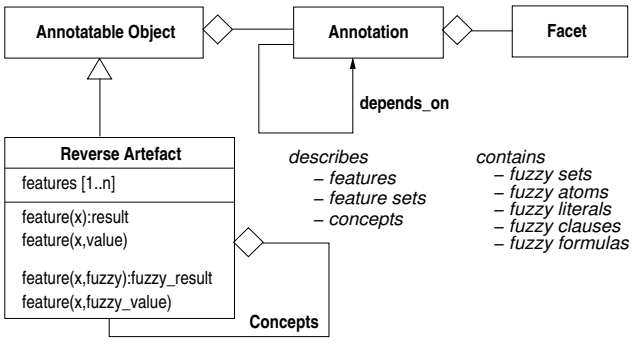

Figure 21.1: The generic fuzzy model

The key idea is to add imperfect information as an orthogonal extension through a frame-like concept, which we call *annotations*. Each object (or part of an object,

like an attribute) holding RE information can thus be *annotated* with container objects referring to one or multiple meta-information objects, called *facets*. Within our model, facets can store imperfect RE information in form of *fuzzy components*, like fuzzy sets, fuzzy literals, fuzzy clauses, or fuzzy formulas.

The annotation model offers the possibility to fuzzify individually all artifacts of a meta model, e.g. objects, relationships, features, and feature sets. As shown in Figure 21.1, every artifact inherits from the generic meta-object ANNOTATABLE OBJECT. Through inheritance all properties and operations of the fuzzy model now become available within the extended meta model.

21.3 Knowledge Consolidation and Hypothesis Support

The fuzzy enhanced repository model already allows us to store more detailed information for each RE artifact. Information from different sources (e.g., different analysis tools) can be stored together and, provided the tools have been adapted to the fuzzy format, automatically compared.

But this is only the first step in providing a semantically richer repository framework: we also need to support different computations based on the fuzzy data structures. For example, although the repository can store conflicting information about an artifact, such conflicts are not automatically recognized and resolved. But the sheer amount of data makes it impractical to simply rely on the reverse engineer to detect and resolve such conflicts. Instead, we want to support the RE process by defining higher-level methods that allow to specify relationships and interdependencies between artifacts. Gathered data will then automatically be compared with already existing information.

Finding and automatically resolving conflicts is achieved by executing *fuzzy belief revision* operators [5] on the gathered RE information. The idea of belief revision is to maintain a consistent knowledge base by removing (revising) existing information that cause a conflict with any added new information. By using fuzzified versions of these operators, we can allow for a varying *degree* of inconsistency within a repository: most often the information found by various tools in different sources will not match exactly due to inherent uncertainties in the used data or applied methods. We can even adjust the allowed degree of inconsistency over time: at the beginning of a RE process, where there is not much known about a system, it is permissible to start with a low degree of internal consistency, then gradually increasing it over time and finally converging to a single consistent view of the examined system.

To also allow such belief revision operators across *different* concepts and artifacts, we enhanced the structural model outlined in the last section with the concept of *fuzzy dependency graphs*. Dependency graphs show how the RE artifacts within a repository are connected to each other, they externalize knowledge about a system from a RE engineer by storing the dependencies between the artifacts in the repository. Annotations attached to RE artifacts become nodes in this graph, and directed edges represent dependencies between artifacts. An example for a dependency would be the relationship between the *type* of a program variable and its *usage* within the application. An additional data dictionary holds transformation functions that show precisely how a change to the information at one node affects the dependent nodes.

By using the dependency graph, information from one RE step is not simply stored in the repository, it can now trigger a graph revision operation that propagates changes throughout the repository. The performing RE engineer can adjust the required degree of consistency as outlined above and also select preferences on the available data, which will influence which information are removed in case of a conflict.

More about the theoretical foundations of this work can be found in [3, 4].

21.4 Conclusions and Future Work

Our approach deals with two important problems within the reverse engineering domain: the integration of heterogeneous results through a RE repository and the (semi-) automatic support of result consolidation and hypothesis support through fuzzy belief networks.

The core idea of our work is the acknowledgment that imprecision, inconsistency, and vagueness are unavoidable elements in reverse engineering. Instead of ignoring such imperfections, we provide tailored support for them based on the established theoretical foundation of fuzzy set theory. This allows us to deal with imperfect information explicitly, adding new capabilities to the RE domain.

With the theoretical and technical foundations in place, we are currently preparing experiments on several real-world examples that will combine information obtained through automated analysis of source code with a domain model gathered by text mining the program's documentation and specification through natural language analysis.

Bibliography

[1] Jean-Marie Favre, Mike Godfrey, and Andreas Winter, editors. *Preproceedings of the 1st International Workshop on Meta-Models and Schemas for Reverse Engineering*, Victoria, British Columbia, Canada, November 2003.

[2] Jens H. Jahnke and Andrew Walenstein. Reverse Engineering Tools as Media for Imperfect Knowledge. In *Proc. of the 7th WCRE*. IEEE Computer Society Press, 2000.

[3] Ulrike Kölsch and René Witte. Fuzzy Extensions for Reverse Engineering Repository Models. In *Proc. of the 10th WCRE*. IEEE Computer Society Press, 2003.

[4] René Witte. *Architektur von Fuzzy-Informationssystemen*. BoD, Norderstedt, Germany, 2002. ISBN 3-8311-4149-5.

[5] René Witte. Fuzzy Belief Revision. In *9th Intl. Workshop on Non-Monotonic Reasoning*, pages 311–320, Toulouse, France, April 19–21 2002. `http://rene-witte.net`.