Research in Cloning Beyond Code: A First Roadmap

Elmar Juergens
Technische Universität München, Garching b. München, Germany
juergens@in.tum.de

ABSTRACT

Most research in software cloning has a strong focus on source code. However, cloning occurs in other software artifacts, as well. In this paper, we summarize existing work on cloning in other software artifacts and provide a list of research questions for future work.

Categories and Subject Descriptors

D.2.13 [**Softw. Eng.**]: Reusable Software—*Reuse models*

General Terms

Experimentation, Measurement

Keywords

Clone detection beyond code, roadmap

1. STATE OF THE ART

We briefly summarize work on cloning in code, requirements specifications, models and tests below.

1.1 Code

A lot of work has been done on cloning in source code. Through it, we know how to detect clones in large code bases and have empirical data for many open source and industrial software systems. We understand (some of) its causes, its consequences and its evolution. A comprehensive overview is given by Koschke [6] and by Roy and Cordy [10].

Without doubt, source code is the artifact type that is best researched and understood in terms of cloning. The insights we have gained on cloning in code can guide the investigation of cloning in other artifact types.

1.2 Requirements Specifications

In [5], we analyzed 28 real-world requirements specifications written in natural language (English or German, over 8,500 pages in total). The extent of cloning varied substantially: while some had very little, over 50% of the content of others was duplicated. Figure 1 depicts cloning in 150 use cases from an industrial system. Each rectangle represents a use case, its height corresponding to the length of the use case. Colored stripes depict clones. Just as for code, if a change is made to a requirement clone, it may need to be performed multiple times—impeding maintenance.

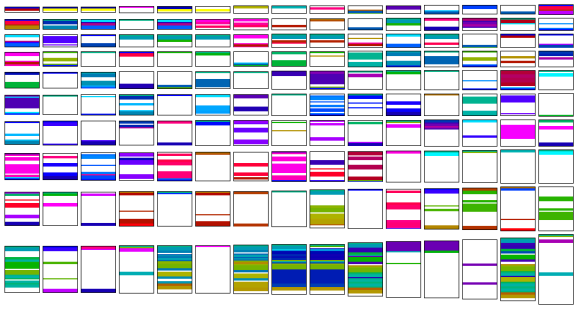

Figure 1: Cloning in Use Cases

1.3 Models

Models can fulfill different roles. On the one hand, they are often employed as specifications or communication artifacts, which developers manufacture into code manually. On the other hand, they can serves as input to code generators, replacing code as the primary development artifact. Cloning has been investigated both for specification (UML) and code generation (MATLAB/Simulink and SCADE) models:

UML. In [7], Liu et al. proposed a suffix-tree based algorithm for clone detection in UML sequence diagrams. They evaluated it on diagrams from two industrial projects, discovering 15% of duplication in the set of 35 sequence diagrams in the first and 8% of duplication in the 15 sequence diagrams of the second project. In [11], Störrle discussed cloning in UML domain models and outlined MQ_{lone} as a tool prototype for its detection.

MATLAB/Simulink. In [1, 2], we presented an approach to detect clones in dataflow models and evaluated it on industrial MATLAB/Simulink models from MAN and BMW. Both case studies analyzed large models (20454 vs. 98251 blocks) and uncovered substantial amounts of cloning (37% in the MAN study). In [9] and [8], Pham et al. and Nguyen et al. present further clone detection algorithms for Matlab/Simulink models. They report discovery of clones in models freely available from MATLAB Central.

IWSC 2011 May 23, 2011, Waikiki, Hawaii, USA

SCADE. In [4], Huhn and Scharff independently extended our clone detector ConQAT [2] to work on SCADE models. They presented results from two models from rail automation industries research projects. The first model comprised 634 nodes, containing 44.5% of cloning. The second model comprised 1264 nodes, of which 53.5% were cloned.

1.4 Test Cases

For manual system tests, test cases are often in the form of natural language documents that describe tester actions and expected system outcomes. We have analyzed 167 system test cases from an industrial business information system, all written in English. We discovered a clone coverage of 54% and over 1000 individual clones. Manual inspection revealed frequent duplication of sequences of interaction steps. Some of the interactions, specifying both the test input and the expected system reaction and state, occurred over 50 times.

1.5 Summary

Cloning is not limited to source code: we have seen evidence for cloning in models, requirements specifications and test cases. Cloning in these artifacts—just as in code—can impede engineering activities. However, our knowledge on cloning in artifacts other than code is still small. Future work, as outlined below, is required.

2. RESEARCH DIRECTIONS

Detection. Most existing detection algorithms operate on sequences, trees or graphs. Non-code artifacts can be represented as such, as well. However, properties of the artifact influence how well a specific algorithm works. For example, detection algorithms for data flow models ignore layout information. They thus cannot directly be applied to Max/MSP models [3], where spatial layout carries semantic meaning. Which existing detection algorithms work how well for which artifact types?

Causes. Some causes of code cloning probably apply to other artifacts as well, while others must be expected to differ. For example, insufficient expressiveness of abstraction mechanisms, which is frequently given as a reason for cloning in code, also causes cloning in MATLAB/Simulink models [2]. However, no weak abstraction mechanism constrains the creation of natural language documents. Since understanding its causes is necessary for control, we need to characterize and quantify causes of cloning across artifacts.

Extent and Evolution. No two artifact types have the same forces (stakeholders, languages, tools, ...) that drive their creation and evolution (and thus of their clones). Results valid for one might thus not apply to another. We need empirical studies to help us understand the extent and evolution of cloning for each artifact type.

Infection. Does cloning in one software artifact type affect cloning in others? In [5], we have seen indication that cloning in requirements specifications can cause redundancy in the implementation: both as code cloning and as independent redevelopment of similar functionality. Was this a peculiarity of the analyzed systems, or can we observe this in general? Are there further infection paths, such as between requirements and models or test cases?

Economic Trade-offs. There is no single best treatment of clones. While some might best be removed, others are more beneficial to keep but track; yet others can be ignored entirely. Even worse, the factors influencing clone management decisions must be expected to vary across artifact types. We thus need a better understanding of the economic trade-offs of clone management alternatives for all artifacts.

Management Tools. Recent work has suggested clone management tools for code to help developers cope with existing duplication. How could analog tool support look for requirements specifications, models or test cases?

Further Artifact Types. The list of software artifacts treated in this paper is far from complete. How about cloning in the others? In process models, system architectures, configuration files, feature models, schemas, ...?

3. ARTIFACT REPOSITORY

Clone detection research has benefited from the availability of open source code. While code is publicly available, requirements specifications, models and test cases are less so. We need a repository of real-world software artifacts of all types to facilitate research into cloning in them. Such a repository could also contain cloning benchmark data.

4. REFERENCES

[1] F. Deissenboeck, B. Hummel, E. Juergens, M. Pfaehler, and B. Schaetz. Model clone detection in practice. In *Proc. of IWSC '10*, 2010.

[2] F. Deissenboeck, B. Hummel, E. Juergens, B. Schaetz, S. Wagner, J.-F. Girard, and S. Teuchert. Clone detection in automotive model-based development. In *Proc. of ICSE '08*, 2008.

[3] N. Gold, J. Krinke, M. Harman, and D. Binkley. Issues in Clone Classification for Dataflow Languages. *Proc. of IWSC '10*, 2010.

[4] M. Huhn and D. Scharff. Some observations on scade model clones. In *Proc. of MBEES '10*, 2010.

[5] E. Juergens, F. Deissenboeck, M. Feilkas, B. Hummel, B. Schaetz, S. Wagner, C. Domann, and J. Streit. Can clone detection support quality assessments of requirements specifications? In *Proc. of ICSE '10*.

[6] R. Koschke. Survey of research on software clones. In *Duplication, Redundancy, and Similarity in Software*. Dagstuhl Seminar Proceedings, 2007.

[7] H. Liu, Z. Ma, L. Zhang, and W. Shao. Detecting duplications in sequence diagrams based on suffix trees. In *Proc. of APSEC '06*, 2006.

[8] H. Nguyen, T. Nguyen, N. Pham, J. Al-Kofahi, and T. Nguyen. Accurate and efficient structural characteristic feature extraction for clone detection. *Proc. of FASE '09*, 2009.

[9] N. Pham, H. Nguyen, T. Nguyen, J. Al-Kofahi, and T. Nguyen. Complete and accurate clone detection in graph-based models. In *Proc. of ICSE '09*, 2009.

[10] C. K. Roy and J. R. Cordy. A survey on software clone detection research. Technical Report 541, Queen's University at Kingston, 2007.

[11] H. Störrle. Towards clone detection in uml domain models. In *Proc. of ECSA'10, Companion Volume*, ECSA '10, 2010.