The Curse of Copy&Paste — Cloning in Requirements Specifications[*]

Christoph Domann
itestra GmbH
Kaufering, Germany
domann@itestra.de

Elmar Juergens
Institut für Informatik
Technische Universität München, Germany
juergens@in.tum.de

Jonathan Streit
itestra GmbH
Kaufering, Germany
streit@itestra.de

Abstract

Cloning in source code is a well known quality defect that negatively affects software maintenance. In contrast, little is known about cloning in requirements specifications. We present a study on cloning in 11 real-world requirements specifications comprising 2,500 pages. For specification clone detection, an existing code clone detection tool is adapted and its precision analyzed. The study shows that a considerable amount of cloning exists, although the large variation between specifications suggests that some authors manage to avoid cloning. Examples of frequent types of clones are given and the negative consequences of cloning, particulary the obliteration of commonalities and variations, are discussed.

1. Introduction

Cloning in source code has been recognized as a problem for software maintenance by both the research community and practitioners [8]. While the amount of cloned code varies, cloning exists in virtually all larger systems. Cloning negatively impacts maintenance activities. It increases program size and thus effort for size-related activities such as inspections. Moreover, it increases effort required for modification, since a change performed on a piece of code often needs to be performed on its duplicates as well. Furthermore, unintentionally inconsistent changes to cloned code frequently lead to errors [7].

Most of the negative effects of cloning in programs also hold for cloning in requirements specifications. Cloning is considered an obstacle to requirements modifiability [5] and listed, for instance, as a major problem in automotive requirements engineering [11].

To the best of our knowledge, though, cloning in requirements specifications has not been investigated systematically yet. Given the importance of requirements specification quality—errors are known to be costly to correct in later project phases—and the known negative impact cloning has on software engineering activities, we consider this precarious for research and practice.

[*]This work has partially been supported by the German Federal Ministry of Education and Research (BMBF) in the project QuaMoCo (01 IS 08023F and 01 IS 08023B).

Contributions

- Existing clone detection tools can be applied to the detection of cloning in (textual) requirements specifications in a straight-forward fashion.
- Real world requirements specifications contain substantial amounts of cloning, with large variation between specifications.
- Cloning is often used when slightly varying requirements, such as similar use cases, are described. A particularly negative consequence of cloning in this case is that it obliterates the commonalities and variations.

2. Detecting Cloning in Specifications

Terms. A *specification* is interpreted as a sequence of words. A *specification clone* is a (consecutive) substring of the specification with a certain minimal length, appearing at least twice. A *clone group* contains all clones of a specification that have the same content. For analyzing the precision of automated detection, we further distinguish between *relevant* clone groups and *false positives*. Clones of a relevant clone group must convey semantically similar information, and this information must be related to the system described. Examples of false positives are substrings that contain the last words of one and the first words of the subsequent sentence without conveying meaning, headers or fragments of tables as well as substrings consisting of document footers or edit histories.

Detection. To identify cloning in requirements specifications, we adapted existing work from code clone detection [6]. The detection algorithm operates in three phases, as depicted in Fig. 1. Specialized text extractors were implemented for preprocessing. Existing detection and postprocessing components could be reused.

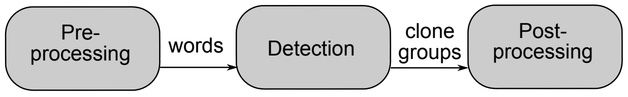

Figure 1. The clone detection phases

- *Preprocessing* extracts text from requirements specification documents. Layout information is removed. Text at the beginning and the end of the document that does not contain requirements information (such as title pages) is discarded. The remaining text is split into a list of words. If a single specification is spread over multiple documents, individual word lists are concatenated into a single global list.

- *Detection* searches for clones by locating all equivalent sublists of the specified minimal length in the word list. This search is performed in linear time and space complexity using a suffix tree.

- *Postprocessing & Output* optionally performs filtering and renders detected clone groups into reports suitable for manual inspection.

3. Study Description

In order to investigate the nature and extent of cloning in requirements specifications, we have performed a study guided by 3 research questions.

3.1. Research questions & study objects

RQ1 *How much duplication do real-world requirements specifications contain?*
To evaluate the impact of specification cloning on software engineering activities, we first need to know if and to which extent it occurs in practice.

RQ2 *How many false positives does the automated detection yield?*
The clone detection tool employed only searches for textual similarity. It does not make certain that the detected clones are relevant clones w. r. t. the above definition. To evaluate the usefulness of automated specification clone detection in practice, we need to analyze its precision.

RQ3 *What kind of information is cloned in requirements specifications?*
To better understand specification cloning, we investigate which parts of requirements specifications are cloned.

We use 11 requirements specifications with altogether 2,500 pages (title pages, document histories and generated tables of content excluded) as study objects. The specifications mostly describe business information systems, and have been written by different teams of authors from several different companies. For non-disclosure reasons, we named the specifications A to K. Overview information on the specifications is depicted in table 1.

3.2. Study design & procedure

For RQ1, we compute several cloning measures that are also applied to quantify cloning on source code. We use the approach described in Sec. 2 with a minimal clone length of 50 words[1]. *Clone coverage* denotes the part of a specification that is covered by cloning. It can be interpreted as the probability that an arbitrarily chosen specification sentence is cloned at least once. *Number of clone groups and clones* denotes how many different logical specification fragments have been copied and how often they occur. The *cardinality* of a clone group is the number of clones it contains.

For RQ2, detected clone groups are assessed manually according to the above definition of relevant clone groups. To keep manual effort within feasible bounds, assessment for specification A was limited to a random sample of 25% of the detected clone groups. All clone groups were rated for the specifications B-K. From the assessment results, *precision* is determined as the ratio of the number of relevant clone groups to the number of all rated clone groups.

For RQ3, we use a qualitative rather than quantitative approach. We manually inspect specification clones and analyze and classify their content. Inspection is limited to samples of the detected specification clones in order to reduce manual effort.

3.3. Results & Discussion

RQ1 and RQ2. Results are depicted in table 1. Clone coverage varies widely, from two specifications (E, J), in which not a single clone of the required length was found, to specification H containing nearly two thirds of duplicated content, yielding an average of 14.9%. No correlation between size of the specification and cloning was found; specifications A and F even have more than one clone per page. The average cardinality is 3.5, average precision of the detection tool according to our manual assessment is 87%.

[1]This threshold was found to provide a good balance between precision and recall during precursory experiments.

Table 1. Extent of specification cloning

spec.	*pages*	*clone cov.*	*clone groups*	*clones*	*precision*
A	517	23.9%	132	524	100%
B	1,013	1.6%	27	60	67%
C	133	15.1%	21	50	95%
D	241	3.4%	21	64	52%
E	185	0.0%	0	0	n/a
F	42	40.3%	24	79	96%
G	85	0.9%	1	2	100%
H	163	65.3%	20	101	100%
I	53	1.3%	1	2	100%
J	28	0.0%	0	0	n/a
K	39	12.1%	8	17	75%

With a minimum clone length of 20 instead of 50 words (detailed results not shown), average clone coverage increases to 26.1% (ranging from 2.7% to 76.1%). However, average precision drops to 57%[2].

RQ3. It appears from the results that most clones are indeed created through copy&paste and do not occur coincidentally. The following further observations were made:

- Long clones: in several specifications, similar use cases have been cloned entirely. Examples include variations of one use case with country-specific differences or creation and modification of certain data items. The longest clone consists of nearly 6 consecutive pages of identical text.
- Clone groups of high cardinality: parts of the system or environment description occur very often in multiple copies. An extreme case is a clone of 61 words that occurs 41 times in specification A. A longer version that contains 114 words occurs 13 times.
- Other frequent patterns: pre- and post conditions often appear identical or with slight variations in a large number of use case descriptions.

Discussion. We conclude from these results that a considerable amount of requirements cloning occurs in practice. However, the fact that even some of the larger specifications analyzed contain very few clones indicates that cloning can be avoided. This seems logical, considering that natural language allows any required abstraction, while programming language limitations sometimes make code clones less easily avoidable. We find the precision of the automated analysis at minimal clone length 50 quite acceptable for practical use. Improvements should be possible in the—currently very simple—preprocessing to reduce the number of false positives with a length of less than 50 words, and to subsequently extend the automated detection to smaller clones without compromising precision.

[2]Determined on a random sample of 20 clone classes per specification.

3.4. Threats to Validity

In the presented study, we have not investigated false negatives, *i. e.*, the amount of duplication contained in a specification not identified by the automatic detector. The reason is the difficulty of clearly defining the characteristics of such clones (having a semantical relation but little syntactical commonality) as well as the effort required to find them manually. The figures on detected clones are thus only a lower bound. While the investigation of detection recall remains important future work, our limited knowledge about it does not affect the validity of the detected clones.

The minimal clone length threshold can have a strong impact on detection precision and recall. We chose a conservative minimal specification clone length of 50 words to keep the false positive rate low. Smaller clones occur and probably could be reliably detected with improvements in preprocessing. This, however, does not invalidate the presented results for clone length 50. Precision rating for specification A was limited to a random sample of 25% of the detected clones. While potentially less accurate, sampling is commonly applied to keep rating effort feasible and has been argued to produce reliable results [1].

Finally, our study only comprises 11 requirements specifications. Future work on additional requirements specifications would foster the generalizability of our results.

4. Consequences of specification cloning

Specification size increase. Requirements specifications are read during various phases of software development, *e. g.*, inspection during early quality assurance, implementation and testing. The effort involved in reading a specification depends, among other factors, on its size. Specification cloning increases specification size, and thus the cost of the software project, without adding value. The average blow up, *i. e.*, the size increase compared to a specification where all cloning has been perfectly removed, is 16% for the analyzed specifications[3]. Size increase similarly impedes other tasks, *e. g.*, the reformatting or printing of the specifications.

Obliteration of commonalities and variation. Cloning has another negative impact on readability. Readers can often benefit from being pointed to similarities between requirements. In particular, developers who design and implement the specified system will be eager to exploit them

[3]Please refer to the documentation at `www.conqat.org` for details on the redundancy-free size computation algorithm.

in order to reuse code. For instance, the use cases for creating and modifying a data item (see 3.3) would probably be based on the same GUI code. With identical content repeated several times, readers are forced to compare the clone instances and search for potential differences manually. They might easily overlook a hidden divergence and wrongly assume that two requirements are identical.

Multiple modifications and probability of errors. When information is duplicated in a specification, one single requirements change might entail modifications in several places of the specification, thus increasing modification effort. Even worse, consistency of the requirements is jeopardized, as modifications may be performed in some places and forgotten in other parts. Both effects are well known from code cloning [7].

5. Related Work

Requirements quality assessment. Cloning is a quality aspect of requirements specifications. From our experience, *manual inspection* (usually guided by a set of criteria such as [5]) is the most common method for requirements quality assessment in practice. As it requires human action, it does introduce subjectiveness and causes high expenses. Various *metrics on text documents* that can be calculated automatically have been derived from linguistics (*e. g.*, [13]). However, many of them are limited to the—grammatically rather simple—English language. Specialized, structured representations of requirements allow the calculation of more *specialized metrics* [2]. They can, however, not be applied in the frequent case of textual documents.

Similarity detection. Numerous approaches to software clone detection have been proposed [8, 10]. They have been applied to source code [8, 10] and models [4]. Substantial investigation of the consequences of cloning has established its negative impact on maintenance activities [7, 8, 10]. To the best of our knowledge, this work is the first to report on requirements specification cloning.

Algorithms for similarity detection in documents have been developed in several other fields of research, though. Clustering algorithms for document retrieval, such as [12], search for documents about issues similar to those of a reference document. However, the criteria are broader than those for clone detection: instead of cloned text, they require similar occurrence of thematic keywords. Plagiarism detection algorithms, like [3, 9], have similar criteria as clone detection but a different starting point: they search for clones between one given document and a very large set of other documents, while we consider clones within a document and optionally a few neighboring documents.

6. Conclusion

We have shown in this paper that a considerable degree of cloning exists in real-world requirements specifications and how clones can be detected efficiently using an existing tool. Future research will be dedicated to improving precision and recall of automated detection and to quantifying the actual impact of requirements cloning on a software project. From our inspections, we suspect that cloning often stems from insufficient abstraction skills on the part of the authors, a misunderstanding of concepts such as use cases and overly strict adherence to templates. We therefore consider high clone rates, beside the cost increase they cause, an important warning signal on requirements quality.

Acknowledgments The authors want to thank F. Deissenboeck, B. Hummel, D. Méndez Fernández, B. Penzenstadler, S. Wagner and B. Schaetz for helpful comments.

References

[1] S. Bellon, R. Koschke, G. Antoniol, J. Krinke, and E. Merlo. Comparison and evaluation of clone detection tools. *IEEE Trans. Softw. Eng.*, 33(9):577–591, 2007.

[2] B. Bernárdez, A. Durán, and M. Genero. Empirical evaluation and review of a metrics-based approach for use case verification. *Journal of Research and Practice in Information Technology*, 36(4):247–258, 2004.

[3] F. Culwin and T. Lancaster. A review of electronic services for plagiarism detection in student submissions. In *8th Annual Conf. on the Teaching of Computing*, 2000.

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

[5] IEEE. Recommended practice for software requirements specifications. Standard 830-1993, IEEE, 1993.

[6] E. Juergens, F. Deissenboeck, and B. Hummel. Clonedetective – a workbench for clone detection research. In *ICSE'09*, 2009.

[7] E. Juergens, F. Deissenboeck, B. Hummel, and S. Wagner. Do code clones matter? In *ICSE'09*, 2009.

[8] R. Koschke. Survey of research on software clones. In *Duplication, Redundancy, and Similarity in Software*, 2006.

[9] C. Lyon, R. Barrett, and J. Malcolm. A theoretical basis to the automated detection of copying between texts, and its practical implementation in the ferret plagiarism and collusion detector. In *Plagiarism: Prevention, Practice and Policies Conference*, 2004.

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

[11] M. Weber and J. Weisbrod. Requirements engineering in automotive development – experiences and challenges. In *Int. Conf. on Requirements Engineering*, 2002.

[12] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang. Clustering user queries of a search engine. In *Intl. conf. on WWW*, 2001.

[13] W. M. Wilson, L. H. Rosenberg, and L. E. Hyatt. Automated analysis of requirement specifications. In *ICSE'97*, 1997.