Poster: Duplicate Finder Toolkit

Dmitry Luciv
Saint Petersburg State University
LANIT-TERCOM LLC
Saint Petersburg, Russia
d.lutsiv@spbu.ru

Dmitrij Koznov
Saint Petersburg State University
Saint Petersburg, Russia
d.koznov@spbu.ru

George Chernishev
Saint Petersburg State University
Saint Petersburg, Russia
g.chernyshev@spbu.ru

Hamid Abdul Basit
Lahore University of Management Sciences
Lahore, Pakistan
hamidb@lums.edu.pk

Konstantin Romanovsky
Saint Petersburg State University
Saint Petersburg, Russia
k.romanovsky@spbu.ru

Andrey Terekhov
Saint Petersburg State University
Saint Petersburg, Russia
a.terekhov@spbu.ru

ABSTRACT

Software documentation is a significant component of modern software systems. Each year it becomes more and more complicated, just as the software itself. One of the aspects that negatively impact documentation quality is the presence of textual duplicates. Textual duplicates encountered in software documentation are inherently imprecise, i.e. in a single document the same information may be presented many times with different levels of detail and in various contexts. Documentation maintenance is an acute problem, and there is a strong demand for automation tools in this domain.

In this study we present the Duplicate Finder Toolkit, a tool which assists an expert with duplicate maintenance-related tasks. Our tool can facilitate the maintenance process in a number of ways: 1) detection of both exact and near duplicates 2) duplicate visualization via heat maps 3) duplicate analysis — comparison of several duplicate instances, evaluation of their differences, exploration of duplicate context 4) duplicate manipulation and extraction.

CCS CONCEPTS

• **Applied computing** → **Document management and text processing**; • **Software and its engineering** → **Software maintenance tools**; **Documentation**; • **Social and professional topics** → Software maintenance.

KEYWORDS

software documents, near duplicates, software clone detection, meaningful search, copy-paste

ACM Reference Format:
Dmitry Luciv, Dmitrij Koznov, George Chernishev, Hamid Abdul Basit, Konstantin Romanovsky, and Andrey Terekhov. 2018. Poster: Duplicate Finder Toolkit. In *Proceedings of 40th International Conference on Software Engineering Companion (ICSE '18 Companion)*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3183440.3195081

ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3195081
This work was partially supported by RFBR grant No.: 16-01-00304

1 INTRODUCTION

Contemporary software documentation, just like the software, is becoming increasingly more complicated with every passing year.

Therefore, simplification of the documentation maintenance process is a relevant and widely applicable research problem. One of the obstacles for efficient documentation maintenance is the presence of textual duplicates. A textual duplicate is a fragment of text which is repeated throughout the document multiple times, possibly with some degree of variation.

Software documents accumulate a large number of textual duplicates during the development and especially the maintenance stage. Usually, textual duplicates emerge as the result of a copy-paste pattern where a textual fragment is copied and pasted into a different part of the document, possibly in a modified form.

Currently, there is no consensus on whether duplicates in software documentation are desirable or harmful. The drawback of having textual duplicates in a software document is straightforward: multiple fragments have to be updated in case of changes. However, their benefits are more subtle. For example, Parnas [9] identifies two types of software documentation: narrative tutorials and reference documents. The former are to be read from beginning to end and are intended for people with little knowledge on the subject. The latter are used to retrieve specific facts and are designed for people who already have considerable knowledge on the subject. In this case, textual duplicates are desirable because they allow to provide a unified presentation of information.

Proper detection and maintenance of such duplicates can improve documentation quality. Consider an example: "inet daemon can listen on ... port and then transfer the connection to appropriate handler". Suppose that this text fragment appears multiple times in the document with various port numbers in place of "...". In this case, a template can be created with a parameter for the port number.

At the same time, manual approach to the duplicate management problem is infeasible, given the sizes of contemporary documentation. Thus, automatic duplicate management facilities are required. However, contemporary studies [3, 4, 7, 8] address this problem in a rather limited way: they rarely provide duplicate detection mechanisms and do not address near duplicates at all.

In this paper we present the Duplicate Finder Toolkit — a toolkit which allows to detect, explore, visualize and manipulate both exact and near duplicates in software documentation.

2 TOOL

Near Duplicates. Near duplicates are present in any kind of textual content —software documentation, software code, models, etc. Near duplicates are fragments of information which contain a common part —an archetype, and a variative part — deltas [2]. According to the provided definitions, an archetype constitutes a major part of information. Let us define near duplicates more precisely.

Definition 2.1. Text fragments $fr_1, \ldots, fr_M \in D$ form a group of **near duplicates** if:

(1) $\exists \{I_1, \ldots, I_N\}$ — a set of strings (archetype) that are contained in each fr_i in the same order.
(2) $\exists\, 0 < k \leq 1$:

$$\forall j \in \{1, \ldots, M\} : \frac{\sum_{i=1}^{N} |I_i|}{|fr_j|} \geq k$$

Here, k denotes how much common text fr_i and fr_j share, for example, a threshold value of 0.85 was used by Basset [2].

Tool Features. **Duplicate detection.** The tool is able to detect both exact and near duplicates within a single document. Our algorithm is based on the above definition and allows not only to detect near duplicates, but also to reveal parts of text which will become parameters when a template would be created by user. Our tool accepts plain text and DocBook documents. In other cases, the document is converted using the Pandoc utility.

Duplicate visualization. The tool allows to browse found duplicates (see Fig. 1). It visualizes a list of found near duplicate groups, their common part, and duplicate-specific parts. In order to facilitate duplicate analysis, context for each duplicate is also provided.

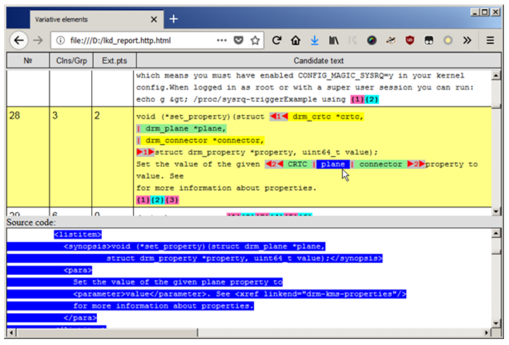
Duplicate manipulation. Our tool allows to modify found duplicates by shifting borders of each found duplicate instance. The duplicates which have been deemed meaningful by an expert can be "accepted" and extracted: e.g. marked and moved outside of the source document. Manual approach is essential, since human judgment is required to discern an actual duplicate from duplicate candidates.

The tool also supports refactoring of DocBook documents: the user can mark a near duplicate group to create a template. This template would contain formal parameters that are extracted from the near duplicate. Then, in each occurrence of the duplicate, the template would be used with formal parameters substituted with the actual ones taken from the text. During the refactoring process, the tool maintains correctness of both text and markup, thus preserving modified document as a valid DocBook document.

Tool Modes. Our tool has two modes: **Automatic mode**, which is intended for a light-weight evaluation of duplicates in the document. Automatic mode provides an overview of all duplicates in the document. See Fig. 1 for an example of the output of this mode.

Interactive mode, which is intended for the discovery of all meaningful near duplicates in the documentation file. Due to the complex nature of near duplicates, this mode requires a human expert and implies a significant amount of work. At the same time, interactive mode provides some automatic processing capabilities: the near duplicate candidate heat map and the near duplicate search.

Using heat map, an expert can select the document part which is heavily populated with near duplicate candidates. At the next step, the expert selects the text fragment that they deem suitable



№	Clns/Grp	Ext.pts	Candidate text
			which means you must have enabled CONFIG_MAGIC_SYSRQ=y in your kernel config.When logged in as root or with a super user session you can run: echo g &gt; /proc/sysrq-triggerExample using {1}{2}
28	3	2	void (*set_property)(struct ◀1◀ drm_crtc *crtc, | drm_plane *plane, | drm_connector *connector, ▶1▶struct drm_property *property, uint64_t value); Set the value of the given ◀2◀ CRTC | plane | connector ▶2▶property to value. See for more information about properties. {1}{2}{3}

```
        <listitem>
          <synopsis>void (*set_property)(struct drm_plane *plane,
                    struct drm_property *property, uint64_t value);</synopsis>
          <para>
            Set the value of the given plane property to
            <parameter>value</parameter>. See <xref linkend="drm-kms-properties"/>
            for more information about properties.
          </para>
```

Figure 1: Browsing: found near duplicates

(relevant) to be a near duplicate. Then, they select the threshold of similarity and run pattern matching. Afterwards, they can analyze the presented near duplicates, which are essentially textual fragments similar to the selected one. Next, like in the automatic mode, the user can manipulate, refactor, and extract a near duplicate. Having finished, the user can recalculate heat map and start to work with the next duplicate.

Tool Architecture. Exact duplicate detection is performed using a software clone detection tool, Clone Miner [1]. Next, the algorithm [6] that uses exact duplicates to construct near ones is invoked. Constructed duplicate candidates are visualized using the Near Duplicate Browser, which is also capable of duplicate analysis (see Fig. 1).

These duplicates can be used to refactor the original document if the source is a DocBook document. This will allow to refactor different duplicate groups into reusable text fragments, similarly to the way described in the introduction. For this purpose, we use the DocBook extension described in [5].

3 CONCLUSION

In this paper we have presented the Duplicate Finder Toolkit — a tool for detection, exploration, and analysis of duplicates in software documentation.

REFERENCES

[1] H.A. Basit, S.J. Puglisi, W.F. Smyth, A. Turpin, and S. Jarzabek. 2007. Efficient Token Based Clone Detection with Flexible Tokenization. In *ESEC-FSE companion '07*. 513–516.
[2] Paul G. Bassett. 1997. *Framing Software Reuse: Lessons from the Real World.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA.
[3] Michihiro Horie and Shigeru Chiba. 2010. Tool Support for Crosscutting Concerns of API Documentation. In *Proc of AOSD '10*. 97–108.
[4] E. Juergens, F. Deissenboeck, M. Feilkas, B. Hummel, B. Schaetz, S. Wagner, C. Domann, and J. Streit. 2010. Can clone detection support quality assessments of requirements specifications?. In *proc of ICSE'10*, Vol. 2. 79–88.
[5] D.V. Koznov and K.Y. Romanovsky. 2008. A method for software product lines documentation development. *Programming and Computer Software* 34, 4 (8 2008), 216–224. https://doi.org/10.1134/S0361768808040051
[6] D. V. Luciv, D. V. Koznov, G. A. Chernishev, and A. N. Terekhov. 2017. Detecting Near Duplicates in Software Documentation. *ArXiv e-prints* (Nov. 2017). arXiv:cs.SE/1711.04705
[7] Milan Nosál' and Jaroslav Porubän. 2014. Reusable software documentation with phrase annotations. *CEJCS* 4, 4 (2014), 242–258.
[8] M. A. Oumaziz, A. Charpentier, J. Falleri, and X. Blanc. 2017. *Documentation Reuse: Hot or Not? An Empirical Study.* Springer, Cham, 12–27.
[9] David Lorge Parnas. 2011. *Precise Documentation: The Key to Better Software.* Springer Berlin Heidelberg, Berlin, Heidelberg, 125–148.