An Approach for Restructuring Text Content

Lerina Aversano, Gerardo Canfora, Giuseppe De Ruvo, Maria Tortorella
Department of Engineering, University of Sannio, Italy
{aversano, canfora, gderuvo, tortorella}@unisannio.it

***Abstract*—Software engineers have successfully used Natural Language Processing for refactoring source code. Conversely, in this paper we investigate the possibility to apply software refactoring techniques to textual content. As a procedural program is composed of functions calling each other, a document can be modeled as content fragments connected each other through links. Inspired by software engineering refactoring strategies, we propose an approach for refactoring wiki content. The approach has been applied to the EMF category of Eclipsepedia with encouraging results.**

***Index Terms*—Refactoring, Reverse Engineering, Reengineering, Wiki, Concept Location, Documentation**

I. Introduction

Software Engineering literature has plenty of methods and techniques that apply Natural Language Processing (NLP) to software artifacts, including requirements [1], design diagrams [2], source code [3], documentation [4], bug reports [5] and developer communications [6]. Many studies focused on traceability recovery, that is linking different types of outputs from the various phases of the software lifecycle [7]. NLP has been used to trace high-level to low-level requirements [8], requirements to design, requirements to source code [9], high-level features to their implementation [10], functional requirements to Java components [4], requirements to test case descriptions [11], requirement changes to the impacted software modules [12], quality consensus to architectural tactics [13], and equivalent requirements across applications [14].

Whilst many researchers have used NPL to support software engineering tasks, in this paper we investigate whether software engineering approaches can be profitably applied to natural language content. More specifically, we focus on wiki content and discuss initial results of adapting a software refactoring method to reorganize the text. As a matter of fact, poorly structured text leads to pages that are hard to navigate and understand, the same way how poorly structured program elements lead to code that is hard to read. The adapted software refactoring method exploits the dominance relation between the nodes of a directed graph [15]; the method proven useful to identify functional [16] and data [17] abstractions.

Our idea stems from the fact that, as a procedural program is composed of functions calling each other, one may imagine a textual document as content fragments connected each other through internal links (i.e. with a target on the same document) or external links (i.e. with a target on another document or resource, e.g. a web page).

This paper is an initial step to extend refactoring concepts into the domain of textual content. There is a recent trend in the refactoring community to apply refactoring to end-user programming tasks [18] [19]; to the best of our knowledge, our work is the first to apply refactoring to a previously unexplored subarea of end-user programming, namely textual content on Wikis.

The remainder of this paper is organized as follows. Section II outlines the proposed approach. Section III shows early results achieved by applying the approach to the EMF category of Eclipsepedia[1]. Finally, Section IV outlines directions for further research.

II. Approach

In this section we outline an approach to refactoring wiki content; the approach is built upon the software refactoring method proposed in references [16] and [17], which exploits the dominance relations on the analysed software system call graph [15].

Our approach comprises three phases: modeling the text, refactoring the model, and reorganizing the text. In the first phase, text is modelled as a directed graph where nodes represent text fragments and edges model the links among the fragments. The meaning of nodes and edges depends on the structure of the document; in the case of wiki, nodes represent *wikiSections* and edges depict links among the sections. In the second phase, the graph is transformed into a tree by using dominance relations. Finally, the content is restructured based on the relations among fragments highlighted by the tree.

Wikis are freely available User Generated (Web) Content. A wiki is composed of *wikiPages*, which can be grouped in various categories on the basis of their topic, called "WikiCategories" (WC). A *wikiPage* (WP) is made up of *wikiSections*, the smallest part of text that can be found in a *wikiPage*. Each *wikiSection* (ws) is identified by its own title.

We model a wiki as a directed Graph, named *WikiGraph* (WG), defined as: $WG = (WS, L)$, where WS is the set of nodes of the graph each representing a *wikiSection*, and L is the set of the edges representing the links between *wikiSections* and *wikiPages*.

In particular, there exist two kinds of links: *intraLink*s (iL) and *InterLink*s (IL). The former indicate links whose destination is in the same *wikiPage*; each section of a page is connected to the subsequent one, through an *intraLink*, to preserve the order existing between *wikiSections*. Instead, an *InterLink* is a link across *wikiPages*.

In the rest of the paper, we focus on *wikiPages* belonging to the same WikiCategory; thus, just *InterLinks* involving

[1] http://wiki.ecplise.org

978-1-4673-3076-3/13 © 2013 IEEE

ICSE 2013, San Francisco, CA, USA
New Ideas and Emerging Results

these pages will be considered. Let WC_y be a generic WikiCategory, the set of *InterLinks*, IL, existing among its *wikiPages* is the set of couples (ws_x, ws_z) such that $ws_x \in WP_x \wedge ws_z \in WP_z \wedge WP_x \in WC_y \wedge WP_z \in WC_y$. An example of *WikiGraph* is shown in Figure 1, where the nodes correspond to *wikiPages*. Figure 2 is a detail of Figure 1 and depicts a fragment of the *WikiGraph*.

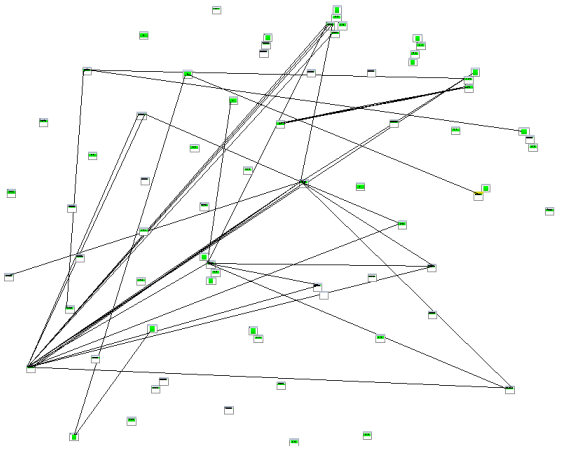

Fig. 1. WikiGraph - EMF category, Eclipsepedia

A *WikiGraph* may comprise several cycles, i.e. strongly connected components. Each cycle represents a set of text fragments that are likely to contribute to defining a concept. Therefore, a cycle constitutes a possible "smell" that suggests refactoring, for instance the aggregation of the involved fragments. We transform the WG into an Acyclic WG (AWG) by collapsing all its cycles, as shown in Figure 3. The AWG is then transformed into a Dominance Tree (DT) [15] by applying the dominance relations.

Fig. 2. WikiGraph - detail of three wikiPages and related wikiSections

In an acyclic graph, a node px dominates another node py if and only if each path from the initial node of the graph to py spans px. There exists a direct dominance relation between px and py if and only if all the nodes that dominate py dominate px, too. Moreover, there exist a strongly direct dominance relation between px and py if and only if px directly dominates py and it is the only node linked to py. px is called the dominator, while py the dominatee.

As an example, Figure 4 shows the dominance tree derived from the *Acyclic WikiGraph* in Figure 3.

A strong direct dominance relation suggests a conceptual dependency. Indeed, if a strong direct dominance relation exists between the *wikiSections* belonging to different *wikiPages*, the content of such *wikiSections* may be nested. *wikiSections* of the referred *wikiPages* are enveloped into the dominant *wikiSection* of the DT, as further shown in Section III.

Therefore, the content of the strongly direct dominated sections can be integrated into the content of dominator one, because conceptual dependencies suggest this action. This may correspond to the definition of subsections nested into the dominator section, with the content of the dominated ones.

Instead, a direct dominance relation means that two or more *wikiPages* are linked with a particular *wikiSection*. In this case, it is not possible to nest the content of the dominated sections because there is not a unique direct dominator. The immediate effect is that such a referred node rises up from the bottom of the AWG to the top of the DT.

In other words, each subtree of the DT represents a level in terms of content. Each strong direct dominance relation may indicate a sub-level of the considered subtree. Each node dominated by only direct dominance relation is considered at the same level of the dominator.

III. Preliminary Results

In this section we apply the approach outlined in the previous section to *Eclipsepedia*, that is the wiki of Eclipse.

From a software perspective, Eclipsepedia is powered by Mediawiki[2]; from a content perspective, it is organized around a set of categories each representing a topic regarding a component or a feature of Eclipse. For instance, the EMF category deals with the Eclipse Modeling Framework, a framework and code generation facility for building Java applications based on simple model definitions.

We focus our investigation on the EMF category, which comprises 72 pages. The WG of such a category is shown in Figure 1. The *wikiSections* of the same *wikiPage* have been grouped in a rectangle. Some *wikiSections* of the WG are not linked to the *wikiSections* of other *wikiPages*. They are related to "islands" in Figure 1. Such "islands" have not been considered in our investigation; indeed, they are related to *wikiPages* that either are linked to other *wikiPages* belonging to another category, or are not linked at all.

Figure 3 shows the corresponding AWG. Each node is a *wikiSection*, while the edges are either iL or IL. Figure 4

[2]http://www.mediawiki.org

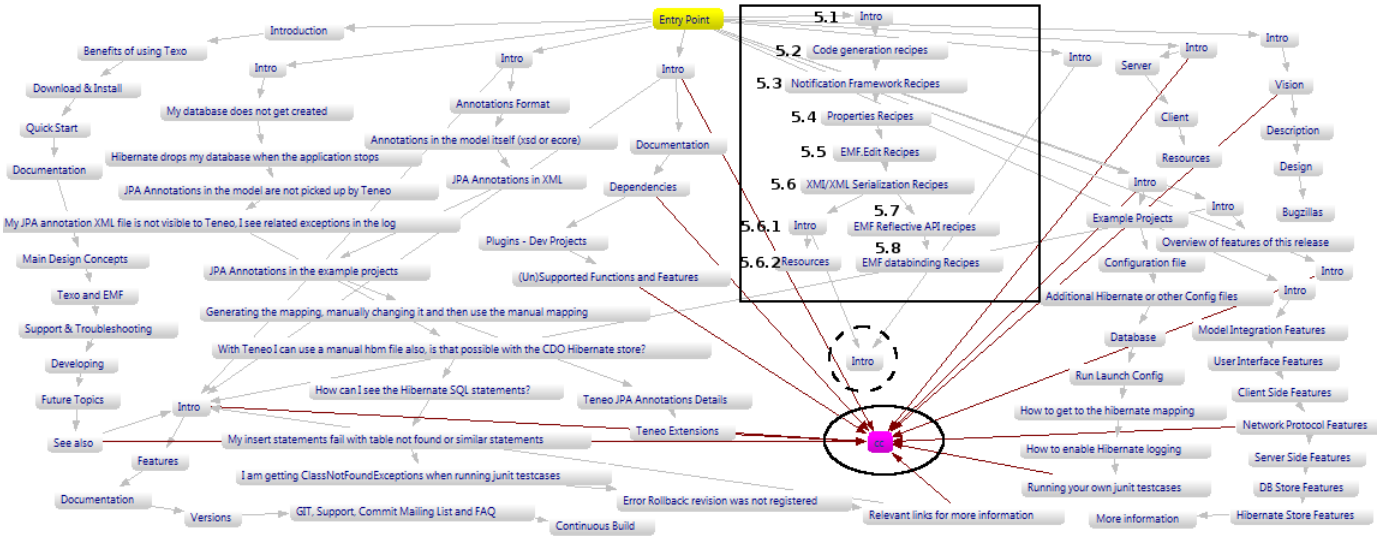

Fig. 3. AWG - EMF category, Eclipsepedia

shows the corresponding DT. The edges depicted with a full line indicate strong direct dominance relations, while the ones depicted with a dashed line indicate only direct dominance relations.

Looking at Figure 3 three "smells" have been highlighted. First, a collapsed cycle - circled with a full line. Second, a node called by more than one node - circled with a dashed line. Third, a set of nodes tied by strong direct dominance relations - surrounded with a rectangle.

A cycle represents a set of circular references among nodes expressing a concept, and suggests refactoring, for instance by aggregating the involved fragments. The analysis of the content associated with the collapsed circle in Figure 3 showed that such fragments come from wikiPages "CDO" and "Net4j". At the beginning of *wikiPage* "Net4j" one can read "The Net4j project is now integrated into the CDO project". This indicates that projects "CDO" and "Net4j" were refactored and integrated, and motivates a need for integrating the related documentation, too, as suggested by the smell. Moreover, such a collapsed cycle is linked with many other nodes as shown in Figure 3. Hence, due to the direct dominance relation the node representing the collapsed cycle rises up from the bottom of the AWG to the top of the DT.

A strong direct dominance relationship between nodes suggest that there are content fragments that participate in the definition multiple concepts. The node circled with a dashed line represents a description of "Ecore" in EMF: the metamodel. The node, which is a target for links from more than one other nodes, floats up in the DT due to the direct dominance relation, as can be seen by comparing the AWG and DT. In this specific case, the node represents a *wikiPage* containing only one *wikiSection*.

Direct dominance relationships individuate concepts that span across multiple nodes. The nodes surrounded with a rectangle represent the *wikiSections* of *wikiPages* "EMF/Recipes" and "Edapt". Such *wikiSections* are linked by a strong direct dominance relation. Figure 5 shows a structure for the content of the involved nodes; the numbers attached to the text fragments are the same of *wikiSections* in Figure 5. The figure shows a structure of the content of the involved nodes. A strong dominance relation means a conceptual dependency. In particular, *wikiSections* "Intro" and "Resources" of *wikiPage* "Edapt" are nested into "5.6 XMI/XML Serialization Recipes". Indeed, *wikiSection* 5.6 deals with a "data migration problem" and Edapt, that is a framework for migrating EMF models, better clarifies that concept. It is like the *wikiSection*

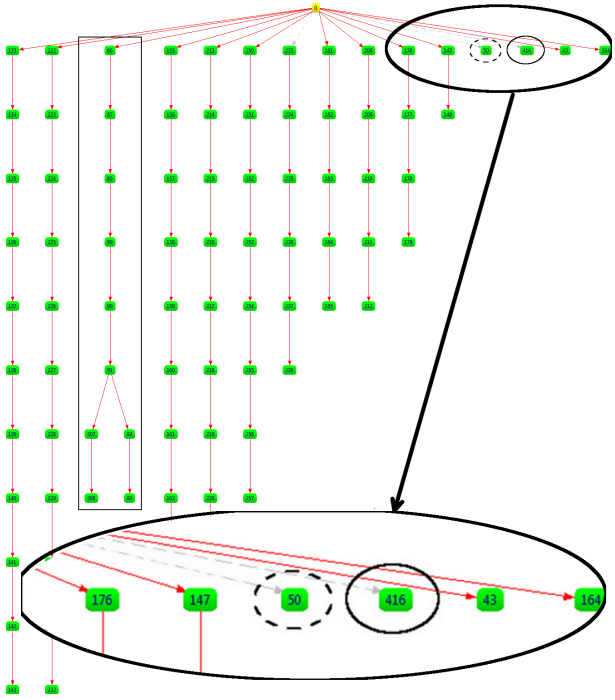

Fig. 4. Dominance Tree - EMF category, Eclipsepedia

numbered 5.6 needs the *wikiSections* of *wikiPage* "Edapt" and, thus, we suggest to incorporate the contents of "Edapt" into the *wikiSection* "XMI/XML Serialization Recipes". For this reason, we added a level of nesting between the involved contents resulting in a "flattened" page (Fig. 5).

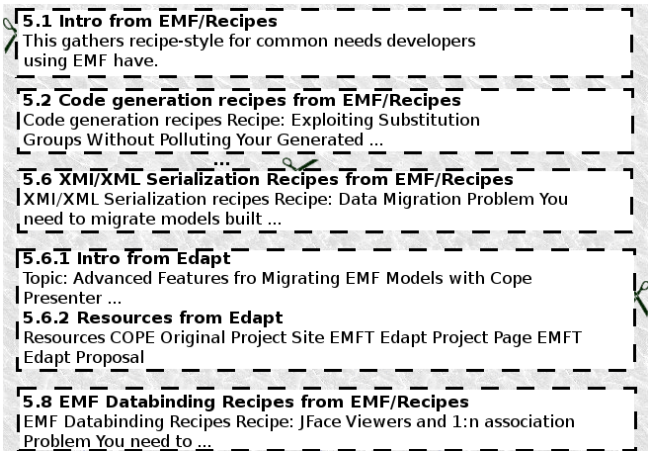

Fig. 5. Text reorganization

IV. Conclusions and Further Work

In this paper we explore the possibility to apply software refactoring techniques to textual content, as NLP techniques have been profitably applied to software artifacts.

In particular, we have shown how a software refactoring method based on graph transformations and dominance relations [16] [17] can be adapted to reorganize wiki content. Specifically, we have analysed the EMF category of Eclipsepedia, achieving encouraging preliminary results. Although we used wikis, the approach is generic and can be applied to other kinds of text, provided that a convenient segmentation/representation is built.

This paper is a initial step towards the definition of methods to help refactoring textual content, wikis in particular, and much work remains to do. Many code refactoring approaches use catalogues of behavior-preserving transformation rules. Opdyke [20] defines individual refactoring operations as: precondition, a composition of elementary behavior preserving code transformations, and post-conditions guaranteed by the refactoring operations. An open area of investigation is to understand to what extend these ideas apply to text refactoring, and how rules and pre- and post-conditions could be defined and formalized for text. Similarly, a catalogue of "smells" and their rationale need to be defined for text. The paper has discussed a few examples (circular references among nodes, nodes that participate in multiple concepts, and concepts that span across multiple nodes) but much work remains to do. Metrics to assess the benefit of refactoring needs also to be defined. The quality of user-generated content varies drastically from excellent to abuse and spam [21], and this makes the definition of metrics a challenging task. Of course, empirical studies are needed to assess the prevalence of smells, their actual effect on the accessibility and readability wiki content, and the effectiveness of refactoring.

References

[1] H. Sultanov and J. H. Hayes, "Application of Swarm Techniques to Requirements Engineering: Requirements Tracing," *2010 18th IEEE International Requirements Engineering Conference*, pp. 211–220, Sep. 2010.

[2] Z. Soh, Z. Sharafi, B. V. den Plas, G. C. Porras, Y.-G. Guéhéneuc, and G. Antoniol, "Professional status and expertise for uml class diagram comprehension: An empirical study," in *ICPC*, 2012, pp. 163–172.

[3] S. Abebe, S. Haiduc, P. Tonella, and A. Marcus, "The effect of lexicon bad smells on concept location in source code," in *Source Code Analysis and Manipulation (SCAM), 2011 11th IEEE International Working Conference on*, sept. 2011, pp. 125 –134.

[4] G. Antoniol, G. Canfora, G. Casazza, A. D. Lucia, and E. Merlo, "Recovering traceability links between code and documentation," *IEEE Trans. Software Eng.*, vol. 28, no. 10, pp. 970–983, 2002.

[5] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun, "An approach to detecting duplicate bug reports using natural language and execution information," in *Proceedings of the 30th international conference on Software engineering*, ser. ICSE '08. New York, NY, USA: ACM, 2008, pp. 461–470.

[6] S. Panichella, J. Aponte, M. D. Penta, A. Marcus, and G. Canfora, "Mining source code descriptions from developer communications," in *ICPC*, 2012, pp. 63–72.

[7] D. Binkley and D. Lawrie, "Information retrieval applications in software maintenance and evolution," *Encyclopedia of Software Engineering*, 2009.

[8] A. De Lucia, R. Oliveto, and G. Tortora, "Assessing ir-based traceability recovery tools through controlled experiments," *Empirical Software Engineering*, vol. 14, pp. 57–92, 2009.

[9] N. Ali, Y.-G. Gueheneuc, and G. Antoniol, "Requirements Traceability for Object Oriented Systems by Partitioning Source Code," *2011 18th Working Conference on Reverse Engineering*, pp. 45–54, Oct. 2011.

[10] D. Poshyvanyk, Y.-G. Guéhéneuc, A. Marcus, G. Antoniol, and V. Rajlich, "Feature location using probabilistic ranking of methods based on execution scenarios and information retrieval," *IEEE Trans. Software Eng.*, vol. 33, no. 6, pp. 420–432, 2007.

[11] E. J. Uusitalo, M. Komssi, M. Kauppinen, and A. M. Davis, "Linking requirements and testing in practice," in *RE*, 2008, pp. 265–270.

[12] G. Antoniol, G. Canfora, G. Casazza, and A. D. Lucia, "Identifying the starting impact set of a maintenance request: A case study," in *CSMR*, 2000, pp. 227–230.

[13] M. Mirakhorli, Y. Shin, J. Cleland-Huang, and M. Cinar, "A tactic-centric approach for automating traceability of quality concerns," in *2012 34th International Conference on Software Engineering (ICSE)*. Ieee, Jun. 2012, pp. 639–649.

[14] D. Falessi, G. Cantone, and G. Canfora, "Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques," *IEEE Trans. Software Eng.*, vol. 39, no. 1, pp. 18–44, 2013.

[15] R. E. Tarjan, "Depth-first search and linear graph algorithms," *SIAM J. Comput.*, vol. 1, no. 2, pp. 146–160, 1972.

[16] A. Cimitile and G. Visaggio, "Software salvaging and the call dominance tree," *Journal of Systems and Software*, vol. 28, no. 2, pp. 117–127, 1995.

[17] G. Canfora, A. Cimitile, M. Tortorella, and M. Munro, "A precise method for identifying reusable abstract data types in code," in *Software Maintenance, 1994. Proceedings., International Conference on*, sep 1994, pp. 404 –413.

[18] K. T. Stolee and S. Elbaum, "Refactoring pipe-like mashups for end-user programmers," in *Proceedings of the 33rd International Conference on Software Engineering*. ACM, 2011, pp. 81–90.

[19] S. Badame and D. Dig, "Refactoring meets spreadsheet formulas," in *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*, sept. 2012, pp. 399 –409.

[20] W. F. Opdyke, "Refactoring: A program restructuring aid in designing object-oriented application frameworks," Ph.D. dissertation, University of Illinois at Urbana-Champaign, 1992.

[21] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne, "Finding high-quality content in social media," in *Proceedings of the international conference on Web search and web data mining*. ACM, 2008, pp. 183–194.