Nature Inspired System Analysis

Vasil Tenev
Fraunhofer IESE, Fraunhofer-Platz 1, Kaiserslautern, Germany
Vasil.Tenev@iese.fraunhofer.de

***Abstract*—The process of cloning variants of a system to accommodate increasing customization is often state of the practice where code duplication is caused by the combination of maintenance problems, high customization, and time pressure. This particular situation motivates the research on similarity analysis of system variants. Similarity determination, variability information recovery, and evolution history reconstruction are prime goals in this context. Analogous research problems appear in the bioinformatics. The growing amount of DNA/RNA sequence data requires efficient similarity analysis and proper visualizations. This branch of computer science faces the tasks of simultaneous aligning for multiple genome sequences, and estimating evolutionary correlations in a given set of taxa. Hence, we applied these techniques to analyze a group of related systems from the BSD Unix family as prove of concept towards model-based variant analysis of complex systems.**

***Index Terms*—reverse engineering, software variants, software evolution, visualization.**

I. Introduction and Motivation

The rapid development in the embedded world demands steady growth of customization flexibility for products and services. We exploit methods for better understanding of variability and regaining control over the increasing management effort. At the same time, we want to enable future (unpredicted, but required) functional adaptations by reducing coupling and increasing cohesion. These principles are reflected by variety of strategic reuse practices [1]. Manufactures nowadays attempt to achieve maturity of engineering and higher efficiency similar to the automotive industry by developing a single product line rather than many individual products [2].

Variability evaluation and reuse potential analysis became important topics in this trend. Today's development of complex and extensive systems often takes advantage reusing already existing artifacts. For a group of systems with similar functionality this approach can result in costs reduction and better resource management. Variants of such systems are often developed by adjusting duplicates of existing components that get into their own life cycle, as known as software mitosis [3]. The resulting branches in the system evolution could benefit from reuse techniques. Due to poor documentation and if not done on time, merging these clones can be highly time and resource consuming task. The alternative — renewed forward engineering — also needs a lot of effort and brings high risk levels because of the missing knowledge about the differences across diverse variants. Suitable similarity analysis between such systems can compensate the lack of information and support renovation to a family of system variants. System engineering can take advantage of shared catalog containing mutual artifacts towards a product line approach [4].

As Hutchins [5] also discusses, there are many analogies between the software evolution and the biological evolution, and the research problems seem to be similar. This is motivation to investigate the application of the bioinformatics concepts in the analysis of software variants.

Related work. Many clone detection approaches exist to analyze code inside one or across many systems [6]. Along some refactoring and inspection techniques, reverse engineering also profits from detecting basic similarities between program artifacts, but most of them could hardly be used for direct evaluation of commonalities between multiple systems or analysis on higher abstraction levels.

Algorithms for comparison between system models were proposed as well, e. g. [7]. These approaches analyze abstract representations like UML class diagrams. The introduced solutions are based on heuristic methods with good-in-practise error rates achieved empirically, but only some of them are stochastically proved. All techniques show the importance of model-driven evaluation of large-scale systems. An analysis on higher abstraction level corresponds to the object-oriented development, exploring some of its main characteristics: encapsulation, modularity, polymorphism, and inheritance. As by biological organisms, similarities between variants must be also related to the correlations of these systems. The bioinformatics study these problems and suggest a variety of well-known algorithms with formal proved properties and empirical recognition.

This paper presents work performed in the context of the Fraunhofer Variant Analysis approach — a framework capable to employ various similarity detection algorithms for similarity detection and visualization across set of system variants [4].

II. Approach

Problem Formulation. Similar to UML diagrams, model-based representation of a system is a graph-like structure, where various components are represented by vertices and different types of arrows denote specific structural and logical relations. (A.) To define similarity between two or more systems, we use the typical change events: insertion, deletion, and substitution. In the context of models, computing the similarity between systems then corresponds with a optimal sequence of change events that lead from one system to another. Furthermore, clustering by similarity helps to identify pairs or groups of closely related systems quantitatively. (B.) The recognition of evolutionary relationships is important for reconstruction of the evolution history. This information

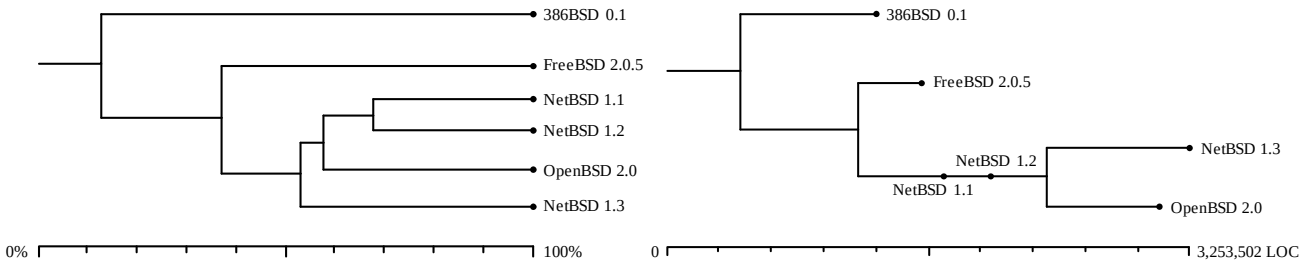

Figure 1. Dendrogram (left) and cladogram (right) constructed for the example BSD systems. The evolutionary relationships are reconstructed correctly and fully automatically, without using any knowledge about the history of the systems [8].

reveals whether the potentially unknown input systems are variants developed in parallel, or rather consecutive versions. The appropriate visualization techniques necessary to present the results to a domain analyst for further evaluation are discussed. Due to space limitations, for more details and formal definitions we refer to [8, 9].

A. Similarity between Multiple Systems

In [9] we defined an approach for a simultaneous alignment on the structure of multiple software systems by extending the existing algorithms for DNA aligning. An approximation algorithm for pairwise aligning of software systems is proposed and combined with an iterative variant of Gusfield's Center-Star method to construct a multiple alignment. This proved that there exists a polynomial approximation algorithm. We then analyzed the source code of six systems from the BSD Unix family in Figure 1 (containing from 5,404 to 12,608 code files). Although we were unable to verify this huge data set, for a smaller example (`sys/kern` folder with 55 to 72 files) our algorithm has 94% precision and 94% recall.

For visualizing the quantitative information about pairwise similarity between the systems we used a dendrogram (Figure 1). The classical WPGMA algorithm was used to generate the weighted average clustering.

B. Evolutionary Relationships

Using similarity data about related systems we showed in [8] that cladogram can be used to present the reconstruction of evolutionary relationships in software mitosis (Figure 1). The length of tree branches is proportional to the amount of common code. Classical algorithms assume a perfect cladistic history: once a feature appears in an ancestor, it is present in all of its successors. However, for software variants this is often not the case, since code parts might be exchanged between the variants after they already split up. Hence, we can only reconstruct a *probable* evolution history, which can differ from the actual history if the code of historically near systems is strongly dissimilar (e.g. due to a large-scale modification), or the code of historically distant systems is similar (e.g. due to cross-system cloning). We implemented an algorithm for constructing a Hasse diagram to reflect a non-perfect cladistic history. The diagram is reduced to a tree by ignoring all edges having length less than a defined threshold. In most cases, the cladogram tree can be constructed unambiguously. If this is not possible, we construct an extended tree where the possible evolution alternatives are presented as parallel tree branches.

III. Conclusions

We presented an application of bioinformatics concepts, i. e. multiple alignments, dendrograms, and cladograms, in an analysis of software variants. As following steps, we are planing to apply these results to model-based representations of embedded systems to enable a model-based variant analysis for self-awarded smart adaptable systems.

References

[1] K. Pohl, G. Böckle, and F. Van Der Linden, *Software product line engineering: foundations, principles, and techniques*. Springer-Verlag New York Inc, 2005.

[2] C. Krueger, "Easing the transition to software mass customization," in *Software Product-Family Engineering* (F. van der Linden, ed.), vol. 2290 of *Lecture Notes in Computer Science*, pp. 178–184, Springer Berlin / Heidelberg, 2002.

[3] D. Faust and C. Verhoef, "Software product line migration and deployment," *Software: Practice and Experience*, vol. 33, no. 10, pp. 933–955, 2003.

[4] S. Duszynski, *Analyzing Similarity of Cloned Software Variants using Hierarchical Set Models*. PhD thesis, Technical University of Kaiserslautern, 2015.

[5] D. Hutchins, "A biologist's view of software evolution," in *RAM-SE'05-ECOOP'05*, pp. 95–105, 2005.

[6] C. K. Roy, J. R. Cordy, and R. Koschke, "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach," *Science of Computer Programming*, vol. 74, no. 7, pp. 470 – 495, 2009. Special Issue on Program Comprehension (ICPC 2008).

[7] M. Girschick, "Difference detection and visualization in UML class diagrams," Technical Report TUD-CS-2006-5, Technical University of Darmstadt, 2006.

[8] V. Tenev and S. Duszynski, "Applying bioinformatics in the analysis of software variants," in *2012 20th IEEE International Conference on Program Comprehension (ICPC)*, pp. 259–260, June 2012.

[9] V. Tenev, "Directed Coloured Multigraph Alignments for Variant Analysis of Software Systems," Bachelor's thesis, Technical University of Kaiserslautern, December 2011.