1 EPOSee: A Tool For Visualizing Software Evolution Patterns

Michael Burch
Saarland University
michael@cs.uni-sb.de

Stephan Diehl
Catholic University Eichstätt-Ingolstadt
diehl@acm.org

Peter Weißgerber
Catholic University Eichstätt-Ingolstadt
peter.weissgerber@ku-eichstaett.de

Software archives contain historical information about the development process of a software system. Using data mining techniques patterns can be extracted from these archives. In this paper we present our tool EPOSee [1] that allows to interactively explore these patterns. For this particular application domain, it extends standard visualization techniques for association rules [1] and sequence rules [2] to also show the hierarchical order of items. Clusters and outliers in the resulting visualizations provide interesting insights into the relation between the temporal development of the system and its static structure.

1.1 Introduction

During the life time of a software system many versions will be produced. Analyzing the source code of these versions, as well as documentation and other meta-information can reveal regularities and anomalies in the development process of the system at hand.

Industrial, as well as open source projects keep track of versions and changes using configuration management systems [3] like RCS and CVS. Other tools keep track of additional information, e.g. bug databases or e-mails. The information stored by a configuration management system and related tools is called a software archive. The software archive provides the history of a software system.

Previously we have used data mining to extract association rules from such archives to characterize the development process [4] or to support programmers [5]. In this paper we discuss the visualization techniques that we implemented to analyze association and sequence rules (which in the following we call software-evolution patterns) and show some kinds of insights that can be gained about the evolution of a software system by visualizing these patterns. To interactively explore the mining rules extracted from software archives we developed EPOSee (see Figure 1.1) which provides the following visualizations:

- Visualization of Association Rules
  - Pixelmap (overview, context)
  - 3D Bar Chart (of selected rules, focus)
- Visualization of Sequence Rules
  - Parallel Coordinate View (overview, context)
  - Decision Tree (overview, context)
  - 3D Branch View (of selected rules, focus)
  - Rule Detail Window (of selected rule, focus)
- Histogram (distribution of confidence and support)

In addition, rules can be filtered according to their support and confidence, searched for keywords, and various schemes can be used for color-coding. All visualizations shown in this paper have been produced with EPOSee.

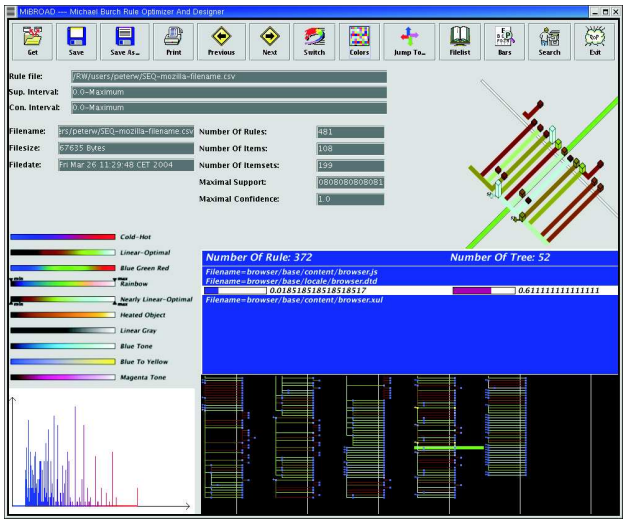

Figure 1.1: EPOSee

The items in the software-evolution patterns are software artifacts like files, classes, or methods for the case of software archives. In the visualization we use a total order derived from a hierarchy stemming from the application domain, e.g. methods are contained in classes, classes are

[1] Evolution Patterns of Software

contained in files, files are contained in directories, and directories are contained in other directories.

1.2 Visualizing Association Rules

To detect relations between items we first look at how often two items have been changed together, i.e. how often have they been checked into the software archive at the same time. As a result, we obtain a table $S : I \times I \to N$ of *change counts* where I is the set of items.

This table can be read as follows: Item i and item j have been changed together $S_{i,j}$ times. We call the matrix S the *support matrix* as it indicates how much evidence is there for each dependency. In particular, $S_{i,i}$ is the total number of times item i was changed.

Next we compute the strength of each dependency, i.e. the number of changes of a pair of items relative to the number of changes of a single item. As a result we get the *confidence matrix* C: $C_{i,j} = \frac{S_{i,j}}{S_{i,i}}$. Given a support matrix S, we can easily compute C by dividing every row by its element on the diagonal. In contrast to S, C is not symmetric.

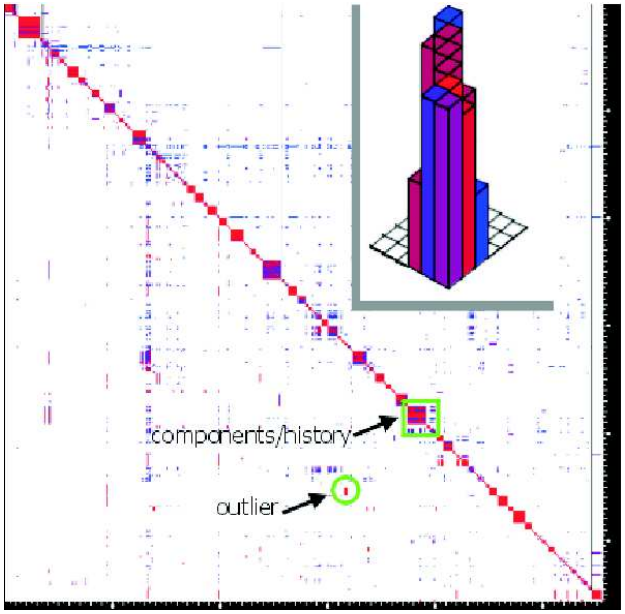

Figure 1.2: Pixelmap of the confidence matrix of MOZILLA

The pixelmap in Figure 1.2 shows the associations of the files in the `/browser` subdirectory of the CVS archive of the MOZILLA project. As the files are ordered hierarchically one can see that files which are next to each other, i.e. those that are in the same part of the hierarchy, are stronger related than others. Thus clusters typically extend along the diagonal of the pixelmap and very much correspond to the hierarchical structure of the system.

Software developers are mainly interested in the outliers, i.e. those pixels representing couplings between files in different directories. This kind of coupling, we call it evolutionary coupling, is based on the simultaneous changes of files rather than on one referencing the other. Outliers can be a sign of aspects orthogonal to the system hierarchy, but also a sign of a bad system architecture. In other words, if in the pixel map we do not find rectangular areas nicely aligned along the diagonal, then it might be a good idea to restructure the system.

The 3D bar chart shown at the corner of the pixelmap is a zoom of any part of the map and illustrates both support (height of the bar) and confidence (color of the bar) at the same time.

1.3 Visualizing Sequence Rules

Next, we would like to know in what temporal order changes typically occur. To this end we compute and visualize sequence rules. Both the antecedent and the consequent of a sequence rule are sequences of items. This gives the antecedent and consequent a time component.

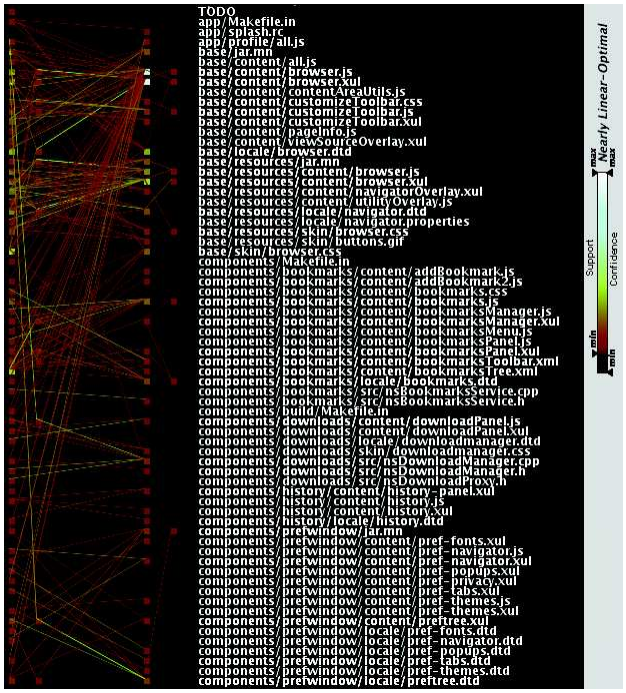

Figure 1.3: Parallel Coordinate View of MOZILLA

For example the sequence rule $a_1 \to a_2 \to a_3 \Rightarrow b_1 \to b_2$ means that if a_1 is changed before or at the same time as a_2 and a_2 before or at the same time as a_3, then it is likely that some time later b_1 and simultaneously or later b_2 will be changed. Figure 1.3 shows a parallel coordinate view of the `/browser` directory. Every sequence rule is displayed by connecting the node in the n-th column representing the n-th item in the sequence with the node in the $n+1$-th column representing the $n+1$-th item. The color of the nodes indicates the weighted sum of the support values of the subsequences ending at this node of all rules which share this node, while the color of the edges indicates the weighted sum of the confidences. As the nodes are ordered with respect to the hierarchical order of the items, we see multiple clusters consisting of many edges which only relate items in the same subdirectory. We also see that the files `base/content/browser.js` and `base/content/browser.xul` are related in a very

interesting way to almost all Javascript respectively XUL files: they are often changed after one of these other files has been changed.

In contrast to the parallel coordinate view in which one edge can belong to multiple rules, the decision tree visualization (see the bottom right corner of Figure 1.1) allows to have a deeper look at the *single* rules. Due to the color coding it is easily possible to find strong rules. Furthermore, one can see the structure of the rules, e.g. the length of the antecedents and consequents of the rule set, or the number of consequents for one given antecedent.

Bibliography

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*. 1993.

[2] R. Agrawal and R. Srikant. Mining sequential patterns. In *Eleventh International Conference on Data Engineering*.

[3] R. Conradi and B. Westfechtel. Version Models for Software Configuration Management. *ACM Computing Surveys*, 30(2), 1998.

[4] T. Zimmermann, S. Diehl, and A. Zeller. How history justifies system architecture (or not). In *Proc. International Workshop on Principles of Software Evolution (IWPSE 2003)*, 2003.

[5] T. Zimmermann, P. Weißgerber, S. Diehl, and A. Zeller. Mining version histories to guide software changes. In *Proceedings of International Conference on Software Engineering ICSE 2004*, 2004.